Decomposing Temporal Equilibrium Strategy for Coordinated Distributed Multi-Agent Reinforcement Learning

Chenyang Zhu[1]*, **Wen Si**[1], **Jinyu Zhu**[1], **Zhihao Jiang**[2]

[1] School of Computer Science and Aritificial Intelligence, Changzhou University
[2] School of Information Science and Technology, ShanghaiTech University
zcy@cczu.edu.cn,jiangzhh@shanghaitech.edu.cn

Abstract

The increasing demands for system complexity and robustness have prompted the integration of temporal logic into Multi-Agent Reinforcement Learning (MARL) to address tasks with non-Markovian properties. However, incorporating non-Markovian properties introduces additional computational complexities, as agents are required to integrate historical data into their decision-making process. Also, optimizing strategies within a multi-agent environment presents significant challenges due to the exponential growth of the state space with the number of agents. In this study, we introduce an innovative hierarchical MARL framework that synthesizes temporal equilibrium strategies through parity games and subsequently encodes them as individual reward machines for MARL coordination. More specifically, we reduce the strategy synthesis problem into an emptiness problem concerning parity games with optimized states and transitions. Following this synthesis step, the temporal equilibrium strategy is decomposed into individual reward machines for decentralized MARL. Theoretical proofs are provided to verify the consistency of the Nash equilibrium between the parallel composition of decomposed strategies and the original strategy. Empirical evidence confirms the efficacy of the proposed synthesis technique, showcasing its ability to reduce state space compared to the state-of-the-art tool. Furthermore, our study highlights the superior performance of the distributed MARL paradigm over centralized approaches when deploying decomposed strategies.

Introduction

A multi-agent system represents a decentralized computational framework in which a collection of independent agents engage within a shared environment, collaborating or competing rationally to optimize task completion and effectively achieve specified objectives. Multi-Agent Reinforcement Learning (MARL) leverages cooperative or adversarial interactions to enhance the problem-solving capabilities of such systems, aiming to attain desired goals efficiently. This approach has been widely applied to real-world applications such as autonomous driving (Yuan et al. 2023) and cooperative robotics (Omidshafiei et al. 2019).

*Corresponding Author

The increasing complexity and robustness demands of multi-agent systems necessitate alignment with predefined temporal patterns to maintain desired temporal properties over time. Coordination and communication among agents require collaborative strategies to achieve collective objectives within designated temporal constraints (Keroglou and Dimarogonas 2020). In stochastic and dynamic environments, agents may encounter challenges demanding long-term planning and temporal reasoning. Nonetheless, traditional MARL algorithms face significant hurdles when dealing with non-Markovian properties, requiring agents to retain historical information. Incorporating non-Markovian aspects escalates state and action space dimensions, augmenting computational complexity (Hammond et al. 2021). This incorporation also introduces delayed rewards, impacting multi-agent communication and coordination.

Linear Temporal Logic (LTL) is a formalism for expressing system requirements, encompassing safety and liveness criteria (Pnueli 1977). Integrating LTL to specify high-level objectives for multi-agent systems gives agents a clear understanding of desired behavior. This integration assists agents in learning strategies aligning with these objectives. Synthesizing LTL strategies as high-level directives guides agent collaboration while adhering to temporal constraints. LTL also offers a formalism to articulate requirements, guiding agents toward short-term and long-term goals. However, verifying temporal logic properties in multi-agent systems where agents act rationally to achieve individual goals remains challenging.

This paper focuses on multi-agent systems in which agent goals and requirements are represented as LTL formulae using the Simple Reactive Modules Language (SRML) (van der Hoek, Lomuscio, and Wooldridge 2006). We deploy MARL to learn strategies that satisfy these specifications. Specifically, we propose a novel hierarchical MARL framework named Multi-Agent Temporal Equilibrium Analysis (MATEA), which synthesizes temporal equilibrium strategies through parity games, encoding them as individual reward machines for MARL coordination. The process involves transforming temporal goals into parity games and subsequently using deterministic Streett automata for high-level temporal strategy synthesis. This synthesized strategy is then decentralized into reward machines for MARL training. The paper presents theoreti-

cal proofs ensuring the invariance of Nash equilibrium between the parallel composition of the decomposed automaton and the original automaton. Empirical results validate the synthesis method's efficacy, highlighting its ability to reduce state space compared to existing techniques. Additionally, our study demonstrates the superiority of distributed MARL over centralized paradigms when implementing decomposed strategies.

Related Work

Temporal Equilibrium Analysis Several tools are available for checking temporal logic equilibrium properties within multi-agent systems. PRALINE, for instance, utilizes the alternating Büchi automata approach to determine Nash equilibrium within concurrent game structures (Brenguier 2013). MCMAS, on the other hand, employs Strategy Logic with memoryless strategies to verify the existence of a Nash equilibrium in multi-agent systems (Čermák et al. 2014). Employing statistical model checking, UPPAAL-SMC approximates Nash equilibria in multi-agent games (David et al. 2015). With a focus on CTL goals, EAGLE verifies if a given strategy profile constitutes a Nash equilibrium (Toumi, Gutierrez, and Wooldridge 2015). PRISM-games primarily engage in strategy synthesis for turn-based stochastic games (Kwiatkowska et al. 2020). EVE, meanwhile, computes a bisimulation-invariant Nash equilibrium for multi-agent systems by addressing the parity game derived from the multi-agent LTL game (Gutierrez et al. 2018). However, the computational complexity of these tools is 2EXPTIME-complete (Gutierrez et al. 2020), making it less suitable for large state spaces.

In comparison, our work yields two key contributions. Firstly, our tool significantly reduces the state space when compared to EVE, thereby enhancing strategy synthesis performance. Additionally, our framework facilitates the integration of MARL with synthesized strategies from temporal equilibrium analysis, allowing for learning within larger state and action spaces within complex environments.

Integrating MARL with LTL In recent years, integrating LTL as prior knowledge into reinforcement learning has garnered attention for tackling complex tasks (Ding et al. 2014). This involves constructing non-Markovian reward functions derived from LTL specifications. Commonly employed methods transform LTL formulae into automata, which are then integrated with the system's Markov Decision Process (MDP) through the product operation. Commonly used automaton types include Deterministic Rabin Automata (DRA), Limit-Deterministic Büchi Automata (LDBA), and Deterministic Finite Automata (DFA).

Approaches utilizing DRA and LDBA often design rewards based on the reachability probability of the resulting product MDP (Gao et al. 2019; Wolff, Topcu, and Murray 2012; Hasanbeig et al. 2019). However, DRA-based strategies may lead to suboptimal outcomes and underestimate specification satisfaction probabilities (Hahn et al. 2019). Similarly, LDBA methods design rewards around repeated reachability of states satisfying Büchi conditions, which can result in sparse rewards. While some approaches transform multi-agent LTL specifications into LDBA and employ function approximation to ensure correctness and convergence (Toro Icarte et al. 2022), they are limited to system-level specifications, not individual agents.

Additionally, certain approaches employ DFA to construct reward machines, enabling precise reward specification and function structure depiction (Toro Icarte et al. 2022). Neary et al. advocate the decomposition of reward machines into individual units for decentralized training in MARL, demonstrating faster learning compared to centralized paradigms (Neary et al. 2021). However, their method imposes strict constraints on verifying bisimilarity between the original reward machine and parallel compositions of decomposed counterparts, limiting real-world applicability. In contrast, our proposed approach operates without such limitations, allowing us to verify the preservation of temporal equilibrium through forward simulation.

Preliminaries

Multi-agent Reinforcement Learning

In MARL, multiple agents learn to make decisions by simultaneously interacting with the environment and each other. The MARL problem can be formalized as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), which can be used to model a team of cooperative agents in a stochastic environment (Oliehoek and Amato 2016). A modified version of Dec-POMDP is shown in Definition 1.

Definition 1 (Dec-POMDP) *A Dec-POMDP is a tuple* $\mathcal{M} = \langle N, S, s_0, (O_i, A_i)_{i \in N}, T, R \rangle$*, where:*

- $N = \{1, \ldots, n\}$ *is a set of agents, where* n *denotes the number of agents;*
- S *is a set of states, with the initial state* $s_0 \in S$*;*
- *Each* (O_i, A_i) *is a set of observation-action pair for agent* $i \in N$*, where* O_i *denotes the partial observation of* S *and* A_i *denotes the action set taken by agent* i*;*
- $T \in S \times A_1 \times \cdots \times A_n \to Dist(S)$ *denotes the transition probability distribution;*
- $R \in S \times A_1 \times \times A_n \to \mathbb{R}$ *denotes the reward function that gives the immediate reward for the agents taking the action, where* $\mathbb{R}$ *is the real number.*

In the context of MARL, Q-learning is a well-known value-based method where each agent aims to learn a value function $\mathcal{Q}_i \in S \times A_i \to \mathbb{R}$ that gives the expected future reward for taking actions in a certain state (Clifton and Laber 2020). The agents in decentralized Q-learning update their policies based on their individual experiences independently (Arslan and Yuksel 2017). Given the reward $R_i \in S \times A_i \to \mathbb{R}$, the Q-values of agent i is updated with Equation (1). Here $\alpha \in [0, 1]$ denotes the learning rate and γ denotes the discount factor.

$$\begin{aligned} \mathcal{Q}_i(s, a_i) \leftarrow & \mathcal{Q}_i(s, a_i) + \alpha(R_i(s, a_i) \\ & + \gamma max_{a_i'} \mathcal{Q}_i(s', a_i') - \mathcal{Q}_i(s, a_i)) \end{aligned} \tag{1}$$

Multi-agent LTL Game

Pnueli introduced Linear Temporal Logic (LTL) as a formalism for specifying properties of discrete systems, enriching propositional logic with a suite of temporal operators such as the 'eventually' operator ($\Diamond$), the 'always' operator ($\Box$), the 'until' operator ($\mathcal{U}$), and the 'next' operator ($\bigcirc$). Given the set of atomic propositions as AP, the syntax of LTL can be presented as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid \varphi_1 \mathcal{U} \varphi_2 \mid \bigcirc \varphi_2 \mid \Box\varphi \mid \Diamond\varphi$$

Here $p \in AP$ is the atomic proposition. LTL formulas can be interpreted over the infinite sequence of states $\pi \in S^\omega$ with a labeling function $\mathcal{L} \in S \to 2^{AP}$. Basically, the infinite sequence of propositional interpretations $\mathcal{L}(\pi)$ satisfies φ if $(\mathcal{L}(\pi), 0) \models \varphi$. Given the trace $\mathcal{L}(\pi)$ and a temporal index $i \in \mathbb{N}$, the following inductive rules are applied on the structure of φ to show that $(\mathcal{L}(\pi), i) \models \varphi$, which verifies that the trace $\mathcal{L}(\pi)$ satisfies the LTL formula φ at index i:

- $(\mathcal{L}(\pi), i) \models p$ iff $p \in \mathcal{L}(\pi_i)$
- $(\mathcal{L}(\pi), i) \models \neg\varphi$ iff $(\mathcal{L}(\pi), i) \not\models \varphi$
- $(\mathcal{L}(\pi), i) \models \varphi_1 \mathcal{U} \varphi_2$ iff $\exists k \cdot k \geq i, (\mathcal{L}(\pi), k) \models \varphi_2 \wedge \forall j \cdot i \leq j < k, (\mathcal{L}(\pi), j) \models \varphi_1$
- $(\mathcal{L}(\pi), i) \models \bigcirc \varphi$ iff $(\mathcal{L}(\pi), i+1) \models \varphi$
- $(\mathcal{L}(\pi), i) \models \Diamond \varphi$ iff $\exists k \cdot k \geq i, (\mathcal{L}(\pi), k) \models \varphi$
- $(\mathcal{L}(\pi), i) \models \Box \varphi$ iff $\forall k \cdot k \geq i, (\mathcal{L}(\pi), k) \models \varphi$

In the later notations, we write $\mathcal{L}(\pi) \models \varphi$ if $(\mathcal{L}(\pi), i) \models \varphi$ to denote $\mathcal{L}(\pi)$ satisfies φ.

In the context of LTL and Dec-POMDP, a multi-agent LTL game can be defined as Definition 2 by introducing the labeling function $\mathcal{L}$, the goal φ_i of each agent and the goal of the system. By playing the game, each agent would have a strategy σ_i with a format of a Non-deterministic Büchi Automaton on Words (NBW) $\mathcal{A} = \langle AP, Q, q^0, \xi, \mathcal{F} \rangle$. Here, Q represents a collection of internal states. Notably, the internal states are presumed to be encoded with the current state $s \in S$ by the labeling function $\mathcal{L}$. q^0 is the initial state, $\mathcal{F} \subseteq Q$ is the set of accepting state, which requires that states in $\mathcal{F}$ should be visited infinitely often. $\xi \in Q \times AP \to Q$ is the transition function between states with corresponding actions. We use $\vec{\sigma}$ to denote the team strategy of the agents and $\Pi(\vec{\sigma})$ to denote the collection of sequences $\pi(\vec{\sigma})$.

Definition 2 (Multi-agent LTL Game) *A multi-agent LTL game is defined as a tuple: $\mathcal{G}_\mathcal{M} = \langle N, \tilde{\mathcal{M}}, \mathcal{L}, (\varphi_i)_{i \in N}, \Phi \rangle$, where:*

- *$\tilde{\mathcal{M}} = \langle \tilde{S}, \tilde{s_0}, (\tilde{A}_i)_{i \in N}, \tilde{T} \rangle$;*
- *$\tilde{S} \subseteq S$ is the abstract state set and $\tilde{s_0} \in \tilde{S}$ is the initial abstract state;*
- *$\tilde{A}_i \subseteq A_i$ is the abstract action set;*
- *$\tilde{T} \in \tilde{S} \times \tilde{A}_1 \times \cdots \times \tilde{A}_n \to \tilde{S}$ is the deterministic transition function;*
- *$\mathcal{L} \in S \to 2^{AP}$ is the labeling function that maps the states in $\mathcal{M}$ to 2^{AP};*
- *φ_i is the goal of each agent i presented as an LTL formula over AP;*
- *Φ is the system goal presented as an LTL formula over AP.*

We initially establish the preference relation between strategies based on the multi-agent LTL game. Specifically, we define that an agent favors strategy $\vec{\sigma}$ over another strategy $\vec{\sigma}'$, denoted as $\vec{\sigma} \succeq_i \vec{\sigma}'$ in Equation (2). This formulation demonstrates that the set of sequences generated by the strategy $\pi(\vec{\sigma})$ satisfying the property φ_i is inclusively contained within the set of sequences generated by $\pi(\vec{\sigma}')$ satisfying φ_i. In simpler terms, for every sequence adhering to φ_i produced by $\pi(\vec{\sigma})$, there exists at least one corresponding sequence in $\pi(\vec{\sigma}')$, yet the strategy $\vec{\sigma}$ is subject to additional constraints.

$$\vec{\sigma} \succeq_i \vec{\sigma}' \triangleq \Pi(\pi(\vec{\sigma}) \models \varphi_i) \subset \Pi(\pi(\vec{\sigma}') \models \varphi_i) \qquad (2)$$

Then on the basis of Nash equilibrium (Gutierrez et al. 2018), we can define the set of Nash equilibria of $\mathcal{G}_\mathcal{M}$ as $NE(\mathcal{G}_\mathcal{M})$ in Definition 3. If there exists Nash equilibrium for the multi-agent LTL game, then $NE(\mathcal{G}_\mathcal{M}) \neq \varnothing$.

Definition 3 (Set of Nash Equilibria)

$$NE(\mathcal{G}_\mathcal{M}) = \{\vec{\sigma} \mid \forall i \cdot i \in N, \vec{\sigma} \succeq_i (\vec{\sigma}_{-i}, \sigma_i')\}$$

Here $\vec{\sigma}_{-i}$ denotes the collection of strategies without strategy of agent i and $(\vec{\sigma}_{-i}, \sigma_i')$ denotes that the strategy of agent i in $\vec{\sigma}$ is replaced by some other strategy σ_i'.

Forward simulation has been used in formal verification to establish a relation between two systems such that one system can take a particular transition and the simulating system can take a corresponding transition without violating the relationship between their states. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automaton. $\mathcal{A}_1$ is forward simulated by $\mathcal{A}_2$, denoted as $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$, if there exists a binary relation $J \in Q_1 \leftrightarrow Q_2$ such that:

- $(q_1^0, q_2^0) \in J$;
- $\forall (q_1, q_2) \in J$, if $q_1' \in \xi_1(q_1, a)$, then there exists q_2' such that $(q_1', q_2') \in J$ and $q_2' \in \xi_2(q_2, a)$;

Bisimulation is stricter, ensuring a bidirectional relationship: $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_1$.

Multi-agent Temporal Equilibrium Analysis

The design of reward functions in MARL poses challenges due to the dynamic nature of environments and the dimensionality of state and action spaces. We first use an example to illustrate the challenges.

Motivation Example In this scenario, three agents traverse a grid world with randomly generating task locations. They aim to collaborate in reaching distinct task locations to achieve the system's final goal. The setup involves seven task locations labeled as a, b, c, d, e, f, and g, assigned to three agents: ma, mb, and mc. The task specifications encompass the following:

- Agent ma is tasked with reaching locations a, b, and e, while Agent mb is assigned to locations c, d, and f. Agent mc is responsible for reaching location g;
- The initial positions are specified: ma is positioned at a, mb is located at c, and mc is not present at g initially;

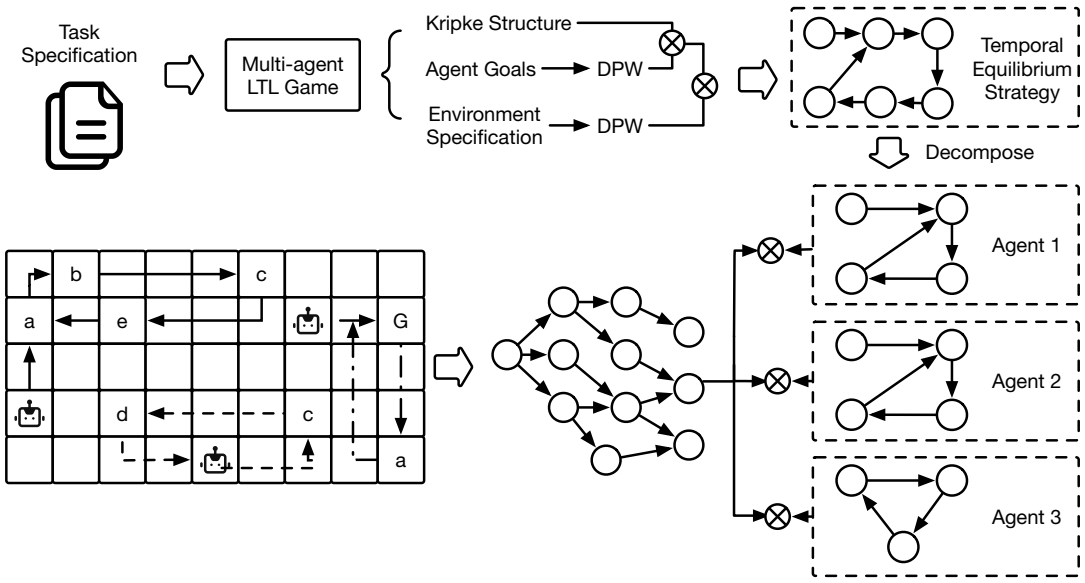


Figure 1: Overall Framework of Decomposing Temporal Equilibrium Strategy for Coordinated Distributed Multi-Agent Reinforcement Learning

- Agents' movements are interlinked: ma can move from a to b only if mb reaches c first. Subsequently, ma can proceed from b to e only when mb reaches d, and ma can return from e to a provided mb reaches f;
- Similarly, mb can progress from c to d when ma reaches b. mb can advance from d to f if ma attains e, and mb can transition from f back to c with the condition that ma reaches e;
- Agent mc can access location g when ma reaches b and mc reaches d. Conversely, mc must leave g when ma reaches e and mc reaches f.

The goals of the agents and the environment are listed as follows:

- Agent ma is required to visit location e infinitely often;
- Agent mb must recurrently visit location d;
- Agent mc is tasked with periodically visiting location g;
- The specific locations e, d, and g are mandated to be visited infinitely often.

Illustrated by the presented example, different agents operate under distinct task specifications within a large state space. Using tools like EVE and PRISM-games to tackle the temporal equilibrium strategy synthesis is under 2EXPTIME-complete complexity. Moreover, the LTL specifications involve non-Markovian properties, requiring agents to incorporate past states and actions to accomplish periodic tasks, which also brings challenges in precisely formulating reward functions. Further complexity arises as agents' strategies evolve during training, contributing to the non-stationarity of the environments.

In this study, we introduce a novel hierarchical framework named MATEA, depicted in Figure 1, to effectively address these challenges. Initially, we transform agent goals and environmental specifications, expressed through Linear Temporal Logic (LTL) formulas, into Deterministic Parity Words (DPW). Next, we reduce the synthesis of the temporal equilibrium strategies problem to check the emptiness problem of the intersection of diverse DPWs and the Kripke structure of the multi-agent LTL game. Following deriving these synthesized strategies, we decompose them into individual strategies based on each agent's local events. Subsequently, the Dec-POMDP is extended with these reward machines for decentralized training. However, the parallel composition of the decomposed strategies might not be bisimilarly equivalent to the original strategy. So the Nash equilibrium may not be preserved after the decomposition. In the following sections, we focus on the theoretical foundations for strategy synthesis and prove that Nash equilibrium is preserved for the parallel composition of the decomposed strategies. Also, we use experiments to show that training with decomposed strategies is more efficient than training with the original strategy.

Temporal Equilibrium Analysis

LTL synthesizing problem is 2EXPTIME-complete (Pnueli and Rosner 1989). In this work, we incorporate several techniques to improve the temporal equilibrium analysis for the mutli-agent games. Algorithm 1 shows the procedure of temporal equilibrium analysis for multi-agents.

Given an LTL formula φ, an NBW $\mathcal{A}$ can be constructed so that some strategy σ is accepted by $\mathcal{A}$ and satisfies φ. And NBW with n states can be translated to Deterministic Parity Automata (DPW) with $n^{\mathcal{O}(n)}$ states and parity index $\mathcal{O}(n)$ (Safra and Vardi 1989). In the parity condition, each priority value corresponds to a set of states that must be visited infinitely often. The DPW sets the parity condition $\mathcal{F}$ as a partition $\{F_1, \ldots, F_n\}$ of Q, where each F_i is a set of states with priority i that must be visited infinitely often. The parity condition can be interpreted as a Streett condition. For each priority value F_i in the parity condition, a

Algorithm 1: Temporal Equilibrium Analysis

Input: $\mathcal{G}_{\mathcal{M}}$
Output: Team strategy $\vec{\sigma}$

$\forall i \in N, \mathcal{A}_i = LTL2DPW(\varphi_i)$;
$\mathcal{A}_\Phi = LTL2DPW(\Phi)$;
σ_a = $\tilde{\mathcal{M}} \bigotimes \times_{i \in N} \mathcal{A}_i$ based on Definition 4;
σ_a = Simplify(σ_a);
if σ_a exists **then**
    σ_b = $\sigma_a \bigotimes_{streett} \mathcal{A}_\Phi$;
    σ_b = Simplify(σ_b);
end if
if σ_b exists **then**
    return σ_b;
else
    return Nash Equilibrium does not exist;
end if

pair of sets (E_i, C_i) can be created for the Streett condition, where $E_i = F_{2i+1}$ specifies the states with the same priority value, and $C_i = \bigcup_{j \leq i} F_{2j}$ includes states with lower priority values. $m = n/2$ pairs of (E_i, C_i) are used as acceptance condition of DPW. This formulation ensures that if a state from the set E_i is visited infinitely often, then there must also be a state from the set C_i that is visited infinitely often, which mirrors the requirements of the parity condition. The emptiness of DPW of Streett condition can be solved in polynomial time (Perrin and Pin 2004). Here we first use Theorem 1 to demonstrate the invariance of the Nash equilibrium set $\mathcal{G}_{\mathcal{M}}$ during the transformation process from LTL to DPW.

Theorem 1 *The set of Nash equilibria $NE(\mathcal{G}_{\mathcal{M}})$ remains unchanged when the individual LTL specifications $(\varphi_i)i \in N$ and the global specification Φ are transformed to DPWs.*

Proof: Based on the Definition, given the strategy $\vec{\sigma}$, $\pi(\vec{\sigma}) \models \varphi$ **iff** $\vec{\sigma}$ is accepted by the constructed NBW $\mathcal{A}_\varphi$. We proceed by employing a proof by contradiction. We assume the existence of a strategy vector $\vec{\sigma}$ that belongs to the set of Nash equilibria $NE(\mathcal{G}_{\mathcal{M}})$, but post transformation, it does not qualify as a Nash equilibrium. In simpler terms, there is a player i with an altered strategy $\sigma' i$ such that the outcome $(\vec{\sigma}{-}i, \sigma_i') \succeq_i \vec{\sigma}$. This leads to the validity of Equation (3).

$$\Pi(\pi(\vec{\sigma}) \models \varphi_i) \subset \Pi(\pi((\vec{\sigma}_{-i}, \sigma_i')) \models \varphi_i), \forall i \in N \qquad (3)$$

However, if σ_i' is accepted by the constructed NBW, then Equation (4) also holds true, This, in turn, results in a contradiction with Equation (3). The ensemble of Nash equilibria remains unchanged following the transformation.

$$\Pi(\pi((\vec{\sigma}_{-i}, \sigma_i')) \models \varphi_i) \subset \Pi(\pi(\vec{\sigma}) \models \varphi_i) \qquad (4)$$

□

As outlined in Algorithm 1, our approach begins by converting the individual agent specifications $(\varphi_i)i \in N$ and the environment specification Φ into DPWs. Subsequently, we execute a product operation between the game structure $\tilde{\mathcal{M}}$ and the individual automata $\bigotimes \times_{i \in N} \mathcal{A}_i$, following Definition 4, resulting in the synthesized strategy σ_a. The strategy σ_a undergoes a simplification process to eliminate strongly connected components and self-loops using the $Simplify$ function. Afterward, a check is conducted to ascertain the existence of σ_a. If it exists, a Streett product is then performed between σ_a and the automaton $\mathcal{A}_\Phi$, yielding the strategy σ_b. Like before, σ_b is subjected to the simplification process to remove strongly connected components and self-loops using the $Simplify$ function. If σ_b exists, it is returned as the team strategy for the temporal equilibrium analysis. On the contrary, if σ_b does not exist, this indicates the absence of a Nash Equilibrium in the multi-agent LTL game.

Definition 4 (Product Operation) *Given the underlying game structure $\tilde{M}$ of $\mathcal{G}_{\mathcal{M}}$, let $\mathcal{A}_i$ be the DPW of agent i's goal φ_i, then $\mathcal{A}' = \tilde{\mathcal{M}} \bigotimes \times_{i \in N} \mathcal{A}_i$ is defined as a tuple $\langle AP, Q', q'^0, \xi', \mathcal{F} \rangle$, where:*

- *$Q' = \tilde{S} \times \times_{i \in N} Q_i \setminus \mathcal{D}$, where $D \subseteq \tilde{S} \times \times_{i \in N} Q_i$ contains the set of tuples $(s, q_1, \ldots, q_n)$ that are duplicated;*
- *$q'^0 = (s_0, q_1^0, \ldots, q_n^0)$;*
- *$\xi' \in \tilde{S} \times \times_{i \in N} Q_i \times AP \to \tilde{S} \times \times_{i \in N} Q_i$. ξ' is defined only when the tuple $(s, q_1, \ldots, q_n) \in \tilde{S} \times \times_{i \in N} Q_i \setminus \mathcal{D}$ and $\tilde{T}(s)$ and $(\xi(p_i))_{i \in N}$ are defined;*
- *For each Streett acceptance condition pairs (E_i', C_i'), E_i' is the set of states from Q' that correspond to E_i in $(\mathcal{A}_i)_{i \in N}$. C_i' is the set of states from Q' that correspond to the union of C_j for $j \leq i$ in $(\mathcal{A}_i)_{i \in N}$.*

Complexity Analysis In terms of complexity analysis, the conversion of LTL formula φ into an NBW $\mathcal{A}_\varphi$ yields a size of $2^{|\varphi|}$. Subsequently, the state space of the DPW is determined to be $2^{2^{|\varphi|}}$, characterized by a parity index of $2^{|\varphi|}$. As a result, the total number of states encapsulating the product of the underlying game, along with the specifications of agent goals and environment goals, is given by $\mid \tilde{S} \mid \cdot 2^{2^{|\varphi_1|}} \cdot \ldots \cdot 2^{2^{|\varphi_n|}} \cdot 2^{2^{|\Phi|}}$. Notably, the computational task of verifying the emptiness of the DPW, subject to Streett conditions, requires polynomial complexity concerning the state space of the automaton, as highlighted by (Perrin and Pin 2004). Thus, the overall computational complexity for Nash equilibrium analysis emerges as $\mid \tilde{S} \mid \cdot 2^{2^{|\varphi_1|}} \cdot \ldots \cdot 2^{2^{|\varphi_n|}} \cdot 2^{2^{|\Phi|}}$. Removing duplicated states and eliminating strongly connected components and self-loops clearly demonstrates a substantial reduction in the computational complexity associated with tasks related to temporal equilibrium analysis.

Decomposing Temporal Equilibrium Strategy

The synthesized strategy is structured as an automaton $\vec{\sigma} = \langle AP, Q, q^0, \xi, \mathcal{F} \rangle$, which can be used for constructing reward machines to guide the reinforcement learning process (Camacho et al. 2019). Considering the advantages of decentralized MARL in terms of scalability and communication efficiency, we propose a method to decompose the synthesized strategy into sub-strategies for individual agents.

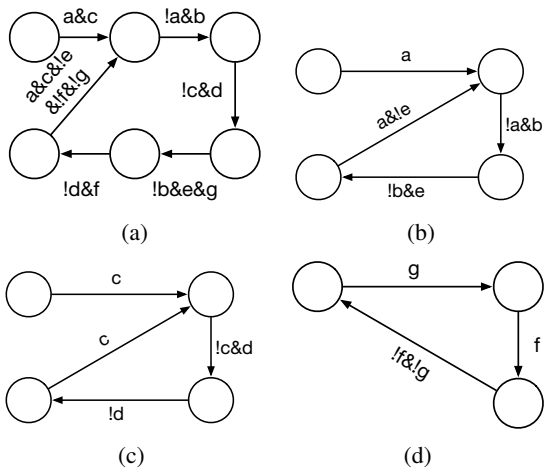


Figure 2: Temporal equilibrium strategy derived from MATEA tool and its individual strategies for each agent: (a) overall temporal equilibrium strategy, (b) individual strategy for agent ma, (c) individual strategy for agent mb, (d) individual strategy for agent mc

This decomposition is based on the local event set $\mathcal{H}_i$ specific to each agent i. To achieve this, we introduce a mapping function $\delta_i \in q \to \hat{Q}$ through Equation (5), aimed at associating a state $q \in Q$ with a set of states $\hat{Q} \subseteq Q$ that are not transitioned from their corresponding local events.

$$\delta_i(q) = \{q' \in Q \mid q' = \xi(q,a), \forall q \in Q \wedge a \in AP \backslash \mathcal{H}\} \quad (5)$$

Figure 2 shows the temporal equilibrium strategy synthesized from MATEA tool with its decomposed strategies for each agent for the motivation example. The synthesized team strategy mandates a synchronized sequence of actions, wherein agents ma and mb must reach locations a and c, respectively, to transition to the subsequent state. Subsequently, ma advances from a to b, followed by mb proceeding from c to d. Upon ma reaching b, mb gains the liberty to shift from d to f, and ma concurrently traverses from b to e. Additionally, agent mc traverses from its present position to g, completing a well-coordinated coordination. Subsequent steps entail ma and mb returning to a and c respectively, effectively initiating an iterative cycle. This cyclic progression ensures that the corresponding locations are visited infinitely often.

The decomposed strategies only require each agent to move to different locations based on its local event set. Consider the instance of agent ma: it initiates by progressing from a to b, subsequently transitioning from b to e. Ultimately, it navigates from e back to a, thereby perpetuating the recurring cycle.

Then we construct the individual strategy automaton from the synthesized strategy for MARL in Definition 5.

Definition 5 (Individual Strategy Automaton) *Given the synthesized strategy $\vec{\sigma}$ and the local event set $\mathcal{H}_i$ for each agent, the individual strategy is defined as $\sigma_i = \langle \mathcal{H}_i, Q_i, q_i^0, \xi_i, \mathcal{F}_i \rangle$, where:*

- $Q_i = \{\delta_i(q) \mid q \in Q\}$, $q_i^0 \in \delta_i(q)$;
- $\xi_i \in Q_i \times \mathcal{H}_i \to Q_i$, $q_i' = \xi_e(q_i, a)$ *is defined* ***iff*** $q_i \in \delta(q), q_i' \in \delta(q')$ *and* $q' = \xi(q,a)$;
- $\mathcal{F}_i = \{\delta_i(q) \mid q \in \mathcal{F}\}$;

The strategy $\vec{\sigma}$ represents the Nash equilibrium, ensuring that the runs of the automaton satisfy both the agents' and the environment's goals. Nonetheless, the parallel composition of the decomposed individual strategies may not be bisimilar equivalent to the initial automaton (Karimadini and Lin 2011). Hence, we need to verify the persistence of the Nash equilibrium following the decomposition and parallel composition.

There have been theories to prove that the automaton is forward simulated by the parallel composition of its decomposed automaton based on local event sets (Zhu et al. 2023). Based on the result, we have $\vec{\sigma} \sqsubseteq ||_{i=1}^n \sigma_i$. Then we construct Theorem 2 to prove that if $\vec{\sigma}$ is one of the Nash equilibrium strategies of $\mathcal{G}_\mathcal{M}$, then $||_{i=1}^n \sigma_i$ also preserves the Nash equilibrium of $\mathcal{G}_\mathcal{M}$.

Theorem 2 *If $\vec{\sigma} \in NE(\mathcal{G}_\mathcal{M})$, then $||_{i=1}^n \sigma_i \in NE(\mathcal{G}_\mathcal{M})$.*

Proof: Based on the fact that $\vec{\sigma}$ is forward simulated by $||_{i=1}^n \sigma_i$, then for any run of $\pi(||_{i=1}^n \sigma_i)$, there exists a corresponding run $\pi(\vec{\sigma})$. So we can get Equation (6) holds.

$$\Pi(\pi(||_{i=1}^n \sigma_i)) \subseteq \Pi(\pi(\vec{\sigma})) \quad (6)$$

Based on the definition, if $\vec{\sigma} \in NE(\mathcal{G}_\mathcal{M})$, then:

$$\Pi(\pi(\vec{\sigma}) \models \varphi_i) \subset \Pi(\pi((\vec{\sigma}_{-i}, \sigma_i')) \models \varphi_i), \forall i \in N$$

Then we can get Equation (7), which can be used to prove $||_{i=1}^n \sigma_i \in NE(\mathcal{G}_\mathcal{M})$.

$$\Pi(\pi(||_{i=1}^n \sigma_i)) \subseteq \Pi(\pi(\vec{\sigma})) \subset \Pi(\pi((\vec{\sigma}_{-i}, \sigma_i')) \models \varphi_i), \forall i \in N \quad (7)$$

□

Subsequently, we formulate reward machines based on these individual strategies to guide MARL within the context of Dec-POMDPs. As outlined in Definition 6, a reward machine primarily introduces the reward function ξ_i^r to the strategy automaton. The function ξ_i^r allocates a real-value reward r exclusively when the subsequent state corresponds to one of the acceptance states. Conversely, a reward of 0 is assigned in other cases.

Definition 6 (Individual Reward Machine) *Given the individual strategy automaton σ_i, the individual reward machine $\mathcal{N}_i$ can be defined as a tuple $\langle \sigma_i, \xi_i^r \rangle$, where $\xi_i^r \in Q_i \times \mathcal{H}_i \to \mathbb{R}$ defines a reward function. $\xi_i^r(q,a) = r$ if $\xi_i(q,a) \in \mathcal{F}_i$, otherwise $\xi_i^r(q,a) = 0$.*

The Dec-POMDP is subsequently extended by incorporating individual reward machines denoted as $\mathcal{N}_i$, to reward the participating agents. The extension process is elucidated using Definition 7, which expounds on the augmented Dec-POMDP. We used the labeling function $\lambda_i \in S \times A_i \times S \to \mathcal{H}_i$ to abstract the state transitions to the higher level of abstract strategies. In cases where the tuple $(q_i, \lambda_i(s, a_i, s')) \in \mathrm{dom}(\xi_i)$, the automaton progresses from state q_i to $\xi_i(q_i, \lambda_i(s, a_i, s'))$, concurrently acquiring the corresponding reward $\xi_i^r(q_i, \lambda_i(s, a_i, s'))$. Conversely, if the condition is not met, the reward remains 0.

Definition 7 (Extend Dec-POMDP with Reward Machine) *Given the Dec-POMDP $\mathcal{M}$, the labeling function $\mathcal{L}$ and the individual reward machines $(\mathcal{N}_i)_{i\in N}$, the extended Dec-POMDP for agent i can be defined as $\mathcal{M}_{\mathcal{N}_i} = \langle N, \hat{S}, \hat{s}_0, (O_i, A_i)_{i\in N}, \hat{T}, \hat{R}\rangle$, where:*

- $\hat{S} = S \times Q_i$;
- $\hat{s}_0 = s_0 \times q_0^i$;
- $\hat{T}(\langle s', q_i'\rangle \mid \langle s, q_i\rangle, a_1, \ldots, a_n) = T(s' \mid s, a_1, \ldots, a_n)$;
- $\hat{R}(\langle s, q_i\rangle, a_1, \ldots, a_n, \langle s', q_i'\rangle) = \xi_i^r(q_i, \lambda_i(s, a_i, s'))$;

Experiments

In this section, we mainly evaluated the performance temporal equilibrium analysis and the sample efficiency of the agents that uses temporal equilibrium for MARL under the decentralized and centralized paradigm.

Experiment Setup

We first use our tool to generate temporal equilibrium strategies and their corresponding decomposed counterparts from the multi-agent LTL game, as specified using the high-level description language SRML. To evaluate our approach, we constructed six distinct scenarios, including varying agent counts and settings. Specifically, The "coop(*)" tasks keep the same settings as the motivation example yet exhibit variations in agent quantities. The "gossip(3)" scenario, derived from the work in (Hammond et al. 2021), emulates a networking protocol mirroring the information diffusion patterns observed in social networks. In this context, multiple replica managers periodically exchange gossip messages to ensure the up-to-date status of system data. During their operational phase, each replica manager can persist in its current servicing mode or transition to a gossiping mode. Should a replica manager be in the gossiping mode while another manager simultaneously engages in such activity, the former reverts to the servicing mode in the subsequent step. This is designed to support the bounded nature of information exchange during the gossip process. The fundamental goal for each replica manager is to engage in gossip interactions perpetually. The system requires that, among the three replica managers, at least two must remain in gossiping mode infinitely often. The "ring-based(3)" scenario, as adopted from work in (Gutierrez, Harrenstein, and Wooldridge 2017), represents a ring-based mutual exclusion model. All agents are organized in a cyclic arrangement within this scenario, and a token is passed along this cycle. Agents possessing the token are thereby granted access to the critical section. Agents collectively aspire to attain continuous entry into the critical section, while the overarching systemic goal focuses on the first agent to enter the critical section infinitely often.

We conducted experiments to record the number of states and state transitions, as well as the time used for temporal equilibrium analysis, with results being compared against those obtained using the EVE tool to check the existence of Nash equilibrium (Gutierrez et al. 2018). These experiments are conducted on a PC with an Intel i9-12900KF GPU clocked at 6.7GHz, accompanied by 64 GB of RAM, and operating on the Ubuntu 18.04 platform.

Exp(num of agents)	EVE			MATEA		
	state	trans	time(ms)	state	trans	time(ms)
coop(2)	10	11	1.16	5	6	0.86
coop(3)	8	9	1.08	5	6	0.96
coop(5)	352	705	40.45	10	16	6.81
coop(7)	1092	3841	449.69	14	22	42.03
gossip(3)	36	146	7.21	9	32	2.83
ring-based (3)	136	269	13.13	8	13	2.96

Table 1: Comparison the Performance of Temporal Equilibrium Analysis between EVE and MATEA

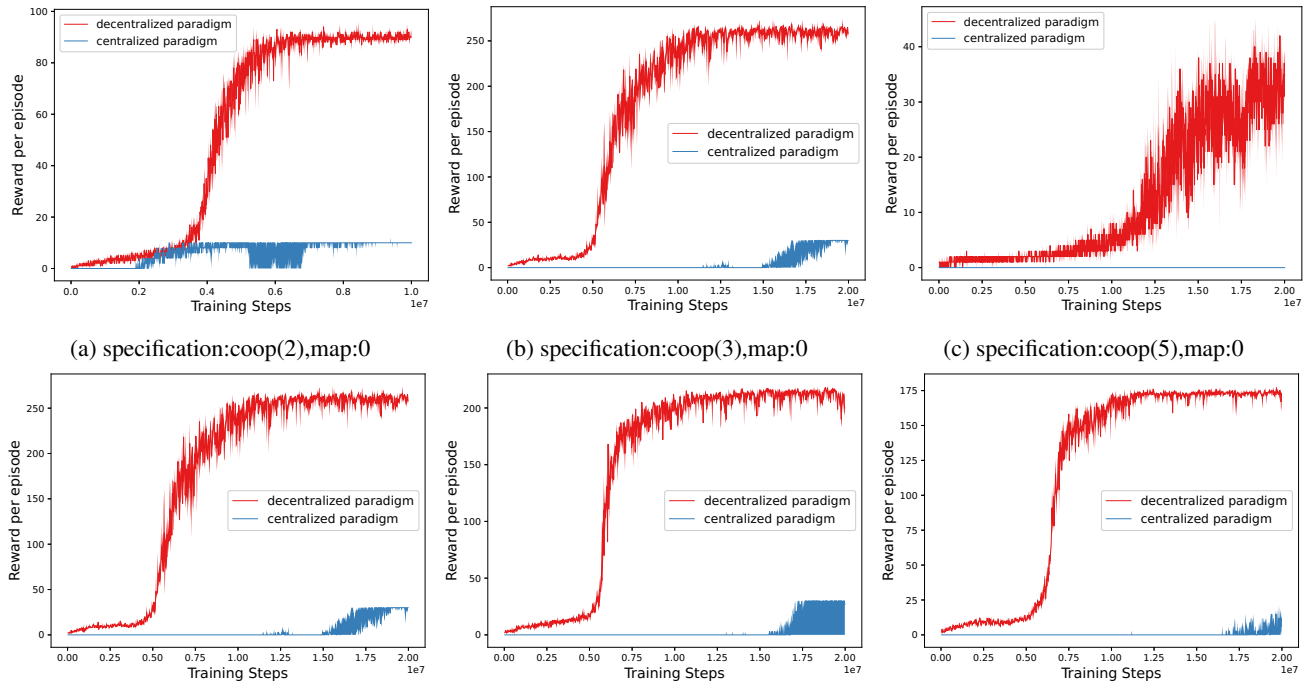
The synthesized strategies are then used for MARL scenarios, instantiated within the OpenAI Gym environment (Brockman et al. 2016), and reproducing the 40×40 grid-world task settings outlined in the motivational example. We generate three distinct maps adorned with randomly positioned task locations to impart diversity. Subsequently, we employ the "coop(2)," "coop(3)," and "coop(5)" specifications to derive temporal equilibrium strategies, steering multi-agent learning processes across these maps. Our focus lies in comparing two distinct learning paradigms: centralized MARL Q-learning, which adheres to the team strategy, and decentralized MARL Q-learning, which operates based on decomposed strategies.

Our experimental setup set the learning rate α as 0.1 and the discount factor as 0.9. For each experiment, five independent trials were conducted to ensure the robustness and reliability of the results. The "coop(2)" experiment was trained with $1 * 10^4$ episodes, each with 1000 steps. Moreover, the "coop(3)" and "coop(5)" experiments were trained with $2 * 10^4$ episodes, each episode with 1000 steps.

Discussions

Table 1 provides a comprehensive comparison between the EVE and MATEA tools in various scenarios, each involving a different number of agents. Notably, MATEA consistently surpasses EVE in terms of state space reduction and the count of state transitions required for the synthesized strategy. This distinction is particularly evident in cooperative tasks with more agents, where MATEA achieves a remarkable reduction in both state space and state transitions compared to EVE. Additionally, the efficiency of MATEA is noteworthy, as it significantly reduces the analysis time across all scenarios. Notably, in the "coop(7)" task, MATEA requires merely 42.03 ms for analysis, compared with 449.69 ms of EVE. This performance enhancement positions MATEA as an efficient solution for real-world applications, particularly when dealing with complex specifications and a substantial number of agents.

Figure 3 provides a comparative analysis of learning curves within the centralized and decentralized paradigms across various maps and specifications. The progressive increase in the accumulated reward per episode over training steps in Figures 3a to 3c illustrates the consistent



(a) specification:coop(2),map:0 (b) specification:coop(3),map:0 (c) specification:coop(5),map:0

(d) specification:coop(3),map:0 (e) specification:coop(3),map:1 (f) specification:coop(3),map:2

Figure 3: Analyzing learning curves in centralized vs decentralized systems across diverse maps and specifications with constant hyper-parameters: experiments with α=0.1, γ=0.9 conducted in five independent trials. Row one (a, b, c) focuses on varied specifications and agent numbers on a single map; row two (d, e, f) examines a consistent specification across multiple maps.

performance enhancement achieved by the decentralized paradigm. Furthermore, it is apparent that the decentralized approach consistently outperforms the centralized counterpart, evident from both the learning curves and the accumulated reward per episode.

For instance, considering the "coop(2)" case, the decentralized paradigm attains a steady accumulated reward of 90, while the centralized paradigm only reaches 10. Similarly, in the "coop(3)" scenario, multi-agents within the decentralized paradigm start to learn around $5 * 10^6$ steps, whereas the centralized approach initiates learning around $1.5 * 10^7$ steps. Moreover, the decentralized paradigm achieves an accumulated reward of 250, whereas the centralized counterpart only reaches 30. In the case of "coop(5)," the decentralized paradigm continues learning despite the complicated agent requirements, while the centralized approach struggles due to the extensive requirements of different agents.

Figures 3d through 3f consistently apply the same specification across diverse maps. Evidently, there is an impact on sample efficiency due to the spatial proximity of task-related locales on the map. In instances where destinations such as d and e are in close proximity, the learning process for agents becomes more facile, whereas distance correlates with increased complexity in learning. Regardless of the variability in final steady-state rewards across different maps, the decentralized paradigm consistently outperforms its centralized counterpart. This correspondence underscores the augmented efficiency and resilience of the decentralized approach, spotlighting its applicability across a variety of settings and environments.

Conclusion and Future Work

This paper presents a novel hierarchical MARL framework, MATEA, which synthesizes temporal equilibrium strategies through parity games. The results of our comparative analysis underline MATEA's capability to reduce state and action spaces across diverse multi-agent tasks, ultimately enhancing the efficiency of temporal equilibrium analysis. Furthermore, the experimental evaluations conducted under both centralized and decentralized paradigms reveal the benefits of decomposing temporal equilibrium strategies for decentralized training.

Our current work only employs Q-learning within discrete state and action spaces. In forthcoming research, more advanced models such as DQN and DDPG could be used to accommodate continuous state and action spaces, further broadening the applicability of our approach.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No.62202067), Natural Science Foundation of the Higher Education Institutions of Jiangsu Province (No. 22KJB520012) and Postgraduate Research and Practice Innovation Project of Jiangsu Province (No. SJCX231485).

References

Arslan, G.; and Yuksel, S. 2017. Decentralized Q-Learning for Stochastic Teams and Games. *IEEE Trans. Autom. Control*, 62(4): 1545–1558.

Brenguier, R. 2013. PRALINE: A tool for computing Nash equilibria in concurrent games. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*, 890–895. Springer.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Camacho, A.; Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2107–2116. PMLR, International Joint Conferences on Artificial Intelligence Organization.

Clifton, J.; and Laber, E. 2020. Q-Learning: Theory and Applications. *Annual Review of Statistics and Its Application*, 7(1): 279–301.

David, A.; Jensen, P. G.; Larsen, K. G.; Mikučionis, M.; and Taankvist, J. H. 2015. Uppaal stratego. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems(TACAS)*, 206–211. Springer.

Ding, X.; Smith, S. L.; Belta, C.; and Rus, D. 2014. Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *IEEE Trans. Autom. Control*, 59(5): 1244–1257.

Gao, Q.; Hajinezhad, D.; Zhang, Y.; Kantaros, Y.; and Zavlanos, M. M. 2019. Reduced variance deep reinforcement learning with temporal logic specifications. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 237–248. ACM.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2017. From model checking to equilibrium checking: Reactive modules for rational verification. *Artif. Intell.*, 248: 123–157.

Gutierrez, J.; Najib, M.; Perelli, G.; and Wooldridge, M. 2018. EVE: A tool for temporal equilibrium analysis. In *Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16*, 551–557. Springer.

Gutierrez, J.; Najib, M.; Perelli, G.; and Wooldridge, M. 2020. Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. *Artif. Intell.*, 287: 103353.

Hahn, E. M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; and Wojtczak, D. 2019. Limit reachability for model-free reinforcement learning of ω-regular objectives. In *Proceedings of the Fifth International Workshop on Symbolic-Numeric methods for Reasoning about CPS and IoT*, 395–412. Springer, ACM.

Hammond, L.; Abate, A.; Gutierrez, J.; and Wooldridge, M. 2021. Multi-Agent Reinforcement Learning with Temporal Logic Specifications. *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems(AAMAS)*, 583–592.

Hasanbeig, M.; Kantaros, Y.; Abate, A.; Kroening, D.; Pappas, G. J.; and Lee, I. 2019. Reinforcement Learning for Temporal Logic Control Synthesis with Probabilistic Satisfaction Guarantees. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 5338–5343. IEEE, IEEE.

Karimadini, M.; and Lin, H. 2011. Guaranteed global performance through local coordinations. *Automatica*, 47(5): 890–898.

Keroglou, C.; and Dimarogonas, D. V. 2020. Communication Policies in Heterogeneous Multi-Agent Systems in Partially Known Environments under Temporal Logic Specifications. *IFAC-PapersOnLine*, 53(2): 2081–2086.

Kwiatkowska, M.; Norman, G.; Parker, D.; and Santos, G. 2020. PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *International Conference on Computer Aided Verification(CAV)*, 475–487. Springer.

Neary, C.; Xu, Z.; Wu, B.; and Topcu, U. 2021. Reward Machines for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, 934–942. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Oliehoek, F. A.; and Amato, C. 2016. *A Concise Introduction to Decentralized POMDPs*, volume 1. Springer International Publishing. ISBN 9783319289274, 9783319289298.

Omidshafiei, S.; Kim, D.-K.; Liu, M.; Tesauro, G.; Riemer, M.; Amato, C.; Campbell, M.; and How, J. P. 2019. Learning to teach in cooperative multiagent reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 6128–6136.

Perrin, D.; and Pin, J.-E. 2004. *Infinite Words - Automata, Semigroups, Logic and Games*. Elsevier. ISBN 9780125321112.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57. IEEE, IEEE.

Pnueli, A.; and Rosner, R. 1989. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '89*, 179–190. ACM Press.

Safra, S.; and Vardi, M. Y. 1989. On -automata and temporal logic. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, 127–137.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2022. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *J. Artif. Intell. Res.*, 73: 173–208.

Toumi, A.; Gutierrez, J.; and Wooldridge, M. 2015. A tool for the automated verification of Nash equilibria in concurrent games. In *International Colloquium on Theoretical Aspects of Computing(ICTAC)*, 583–594. Springer.

van der Hoek, W.; Lomuscio, A.; and Wooldridge, M. 2006. On the complexity of practical ATL model checking. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 201–208. ACM.

Čermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2014. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings 26*, 525–532. Springer.

Wolff, E. M.; Topcu, U.; and Murray, R. M. 2012. Robust control of uncertain Markov Decision Processes with temporal logic specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 3372–3379. IEEE, IEEE.

Yuan, T.; Chung, H.-M.; Yuan, J.; and Fu, X. 2023. DACOM: Learning Delay-Aware Communication for Multi-Agent Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11763–11771.

Zhu, C.; Zhu, J.; Cai, Y.; and Wang, F. 2023. Decomposing Synthesized Strategies for Reactive Multi-agent Reinforcement Learning. In *International Symposium on Theoretical Aspects of Software Engineering*, 59–76. Springer.