Enhancing Strategy Logic with Procedural Rationality

Ruiqi Jin[1], **Shuyi Li**[2], **Yongmei Liu**[1*]

[1]Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]School of Mathematics (Zhuhai), Sun Yat-sen University, Zhuhai 519082, China
{jinrq7, lishy277}@mail2.sysu.edu.cn, ymliu@mail.sysu.edu.cn

Abstract

ATL and Strategy Logic (SL) are important languages for representation and reasoning about strategic abilities of coalitions in multi-agent systems. In analyzing strategies of agents in multi-agent systems, an important concept to consider is rationality. Strategy Logic can express rationality concepts such as Nash Equilibrium (NE). Recently, there has been work on logics for joint abilities incorporating rationality concepts based on iterated elimination of dominated strategies (IEDS). Each of NE and IEDS has its strengths and limitations. However, when the payoff is binary, e.g., whether a goal is satisfied, IEDS has more distinguishing power than NE. In this work, we propose Strategy Logic with IEDS (SL_{IEDS}), an extension of Strategy Logic with an IEDS operator, where we can reason about rational strategies that survive IEDS. We prove that SL_{IEDS} is strictly more expressive than SL. Finally, we prove that model checking memoryless SL_{IEDS} is EXPTIME-complete.

Introduction

Logics for strategic abilities in multi-agent systems is an active research field in knowledge representation and reasoning. One of the fundamental works proposed is Alternating-time Temporal Logic (ATL/ATL*) (Alur, Henzinger, and Kupferman 2002), where formula $\langle\langle A \rangle\rangle \psi$ specifies that coalition A has a collective strategy to achieve temporal goal ψ, where ψ is a Linear Temporal Logic (LTL) formula. Strategy Logic (SL) (Chatterjee, Henzinger, and Piterman 2010; Mogavero et al. 2014) extends ATL* with explicit quantification on strategies, e.g., the formula $\exists x.\forall y.\exists z.(1, x)(2, y)(3, z)\psi$ means that there exists a strategy of agent 1 s.t. for all strategies of agent 2 there exists a strategy of agent 3 that can achieve the goal ψ. This provides more expressive power than ATL*, as in the latter the only allowed quantification alterations are $\exists\forall$ or $\forall\exists$. This makes SL very expressive in representing strategic abilities.

In analyzing strategies of agents in multi-agent systems, an important concept to consider is rationality. A rational agent chooses the best action to achieve her goal given her knowledge or beliefs about the world and the other agents. However, ATL ignores the issue of rationality. For example, two cars have a joint strategy to avoid collision by both staying still, but this joint strategy is not rational given their goals to reach the destinations. In game theory (Osborne and Rubinstein 1994), there are two common concepts of rationality: Nash Equilibrium (NE) and Iterative Elimination of Dominated Strategies (IEDS). NE represents outcome rationality, while IEDS represents procedural rationality. In general, each of NE and IEDS has its strengths and limitations. For example, NE doesn't specify how to arrive at an equilibrium, while IEDS fails in games without dominated strategies. However, when it comes to the situation where the payoff is binary, e.g., whether a goal is satisfied, IEDS has more distinguishing power than NE. To give a simple example, consider two agents trying to cooperate and achieve a goal: Both agents 1 and 2 have two strategies a and b, and (a, a) is the only joint strategy that cannot achieve the goal. Then any joint strategy other than (a, a) is a NE, but only (b, b) is preferred by IEDS. The properties and algorithmic complexities of IEDS are thoroughly studied (Berwanger 2007; Pauly 2016). It is well known that SL can express NE; however, SL cannot express IEDS due to its procedural property. There is other literature on procedural or bounded rationality (Simon 1955; Russell and Wefald 1991; Gigerenzer, Todd, and the ABC Research Group 2000), which consider bounded or procedural rationality as how the agents choose strategies under informational or computational limits.

Nonetheless, rational strategic reasoning has received considerable attention. Works along this line can be put into two groups. The first group is strategy verification and synthesis. Wooldridge et al. (2016) introduces rational verification, concerning whether a given temporal logic formula is satisfied in some or all equilibrium computations of a multi-agent system, where each agent has a goal specified with a temporal logic formula. Kupferman, Perelli, and Vardi (2016) thoroughly investigates rational synthesis under different rationality concepts in cooperative and non-cooperative settings. Aminof et al. (2021) explores best-effort synthesis, which synthesizes non-dominated strategies to achieve LTL goals. Gutierrez et al. (2021) investigates the problem of verification of strict ϵ Nash equilibria, where agents have both an LTL goal and an additional goal to minimize cost. Gutierrez et al. (2023) considers rational verification of mean-payoff games and provides improved

*Corresponding author

complexity results. Recently, Hyland et al. (2024) studies rational verification problem in a setting where agents have quantitative probabilistic goals. The second group is strategic logics, mostly extending ATL. Bulling, Jamroga, and Dix (2008) extends ATL with multiple operators, which enable reasoning about agents that only play strategies satisfying certain rational properties, including Nash Equilibria and non-dominated strategies. Gutierrez, Harrenstein, and Wooldridge (2014; 2017) proposes a logic containing operator $[NE]\psi$, meaning ψ holds on all NE computations. Lorini (2016) proposes a modal logic for interactive epistemology to reason about game-theoretic solution concepts in normal form games. Huang and Ruan (2017) extends ATL with modalities $\mathtt{CE}_G^{\bowtie w}\psi$, meaning group G has collective strategies in the form of correlated equilibria with utilitarian value $\bowtie w$. Liu et al. (2020) proposes a modal logic JAADL for joint abilities by extending ATL* with an operator $(A)_\psi^\infty\varphi$, meaning φ holds after IEDS w.r.t. group A and goal ψ. Li, Lorini, and Mittelmann (2025) extends coalition logic and ATL with minimal rationality modalities $\langle\langle C\rangle\rangle^{rat}\psi$ for existence of non-dominated strategies to achieve ψ.

In this paper, we enhance Strategy Logic with procedural rationality, and propose Strategy Logic with IEDS ($\mathrm{SL}_{\mathtt{IEDS}}$). We choose IEDS because it is a profoundly important and highly influential notion of procedural rationality. Following JAADL, we extend SL with operators for elimination of dominated strategies (EDS) $[\pi]\varphi$ and IEDS $[\pi]^\infty\varphi$, which means φ holds after EDS and IEDS, respectively, w.r.t. goals specified with π. Different from JAADL and similar to rational verification (Wooldridge et al. 2016), different agents can have different goals, represented using formulas, which allows us to reason about non-cooperative settings. We prove that adding only the EDS operator to SL does not increase its expressiveness. We also prove that adding the IEDS operator to SL makes it strictly more expressive. To this end, we construct two classes of models such that there is a $\mathrm{SL}_{\mathtt{IEDS}}$ formula to distinguish between them, but no SL formula can do so. Note that our definition of $\mathrm{SL}_{\mathtt{IEDS}}$ and expressiveness results apply to both memoryless and memoryful cases. Finally, we prove that model-checking memoryless $\mathrm{SL}_{\mathtt{IEDS}}$ is EXPTIME-complete. This complexity is higher than model-checking memoryless SL, which is PSPACE-complete (Čermák et al. 2018). To prove the non-trivial EXPTIME-hardness result, we first show that IEDS on payoff matrices succinctly represented with circuits is EXPTIME-hard, and then reduce it to model-checking memoryless $\mathrm{SL}_{\mathtt{IEDS}}$.

Preliminaries

In this section, we introduce SL, and define the concept of strategy space, which is needed for defining our logic.

Let AP be a finite non-empty set of atomic propositions, Ac a finite non-empty set of actions, and Ag a finite non-empty set of agents.

Definition 1. A concurrent game structure (CGS) is a tuple $\mathcal{G} = \langle W, L, P, \tau, w^0\rangle$, where

- W is a finite non-empty set of states; $w^0 \in W$ is the initial state; $L : W \to 2^{AP}$ is a labeling function; For each agent i, $P_i : W \to 2^{Ac}$ specifies a non-empty set of its available actions at each state;
- A decision at state w is a function mapping each agent i to an action from $P_i(w)$. The state transition function τ maps a state w and a decision d at w to a new state.

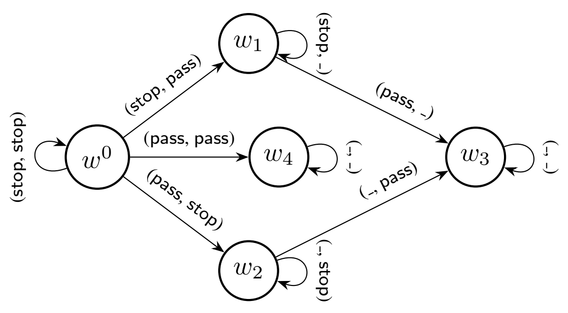

Figure 1: The intersection formalized as a CGS.

Example 1 (Crossing road (Li, Lorini, and Mittelmann 2025))**.** Two vehicles, vehicle 1 and vehicle 2, approch an intersection and both want to go straight. Vehicles can choose to pass or stop. If one of the vehicles chooses to pass, while the other one chooses to stop, then the moving vehicle can cross the road safely and win. However, if both of the vehicles deside to pass at the same time, then the two vehicles will $crash$. In the CGS, atomic proposition win_i denotes that i reached its goal, $crash$ denotes that the two vehicle crash. Aside from initial state w^0 where both vehicles don't cross the intersection, we have four states: w_1, where 1 stops; w_2, where 2 stops; w_3, where both pass successfully; and w_4, where both pass at the same time and crashed. We formalize the CGS shown in Figure 1, where:

- $Ag = \{1, 2\}$, $Ac = \{\mathsf{pass}, \mathsf{stop}\}$, and $AP = \{win_1, win_2, crash\}$; $W = \{w^0, w_1, w_2, w_3, w_4\}$;
- $L(w^0) = \varnothing$, $L(w_1) = \{win_2\}$, $L(w_2) = \{win_1\}$, $L(w_3) = \{win_1, win_2\}$, $L(w_4) = \{crash\}$;
- $P_i(w) = Ac$ for $i = 1, 2$ and all $w \in W$;
- τ is defined as in Figure 1.

Definition 2. A history h in a CGS $\mathcal{G}$ is a finite state sequence $w_0 w_1 \dots w_n$. We let $h[i]$ be the ith state w_i on the history, and $last(h)$ be w_n.

Definition 3. A computation λ in a CGS $\mathcal{G}$ is an infinite state sequence $w_0 w_1 \dots$. We let $\lambda[i]$ be the ith state w_i on the computation.

Definition 4 (Strategies)**.** A memoryless strategy on a CGS $\mathcal{G}$ is a function $\sigma : W \to Ac$. A memoryful strategy on a CGS $\mathcal{G}$ is a function $\sigma : W^* \to Ac$. We denote the set of all memoryless strategies Str^r, the set of all memoryful strategies Str^R. We use Str to range over Str^r and Str^R.

Example 2 (Strategies in the crossing road game)**.** Consider the CGS in Example 1. There are in general two kinds of memoryless strategies that may lead to winning:

- Strategy σ_p, where $\sigma_p(w^0) = \mathsf{pass}$, *i.e.*, to pass first.
- Strategy σ_s, where $\sigma_s(w^0) = \mathsf{stop}$, $\sigma_s(w_1) = \sigma_s(w_2) = \mathsf{pass}$, *i.e.*, to stop before the intersection and wait after the other vehicle passes.

If both agents adopt the same strategy σ_p, it will result in a crash. Both agents adopting σ_s will not lead in to a crash, but a traffic obstruction: no one can win as the CGS is stuck in w^0. They will safely pass the intersection if one chooses σ_p, and the other σ_s.

In the case of memoryful strategies, both vehicles stopping at w^0 does not inevitably lead to a traffic obstruction, as agents can pass after some amount of time. Still, a crash may happen if the two agents pass at the same time.

We define the notion of executable strategies, and introduce the concept of strategy spaces.

Definition 5 (Executablility)**.** A memoryless strategy σ is executable for an agent i, if for all $w \in W$ we have $\sigma(w) \in P_i(w)$. A memoryful strategy is executable for an agent i, if for all $h \in W^*$ we have $\sigma(h) \in P_i(last(h))$. We denote the set of all memoryless (resp. memoryful) executable strategies of agent i as Str_i^r (resp. Str_i^R).

Definition 6. A strategy space Σ on a CGS is a function that maps each agent i to a subset of Str_i. The full strategy space Σ_f on a CGS maps each agent i to Str_i.

We now define the notion of an agent strategy assignment, which assigns a set of agents their executable strategies.

Definition 7. An agent strategy assignment α is a partial function from Ag to Str. An agent strategy assignment is defined on a set $A \subseteq Ag$ of agents, if $i \in A$ iff $\alpha(i) \neq \bot$. An agent strategy assignment α is restricted to a strategy space Σ if for all i s.t. $\alpha(i) \neq \bot$, we have $\alpha(i) \in \Sigma(i)$. Given an agent strategy assignment α, an agent i, and $\sigma \in Str \cup \{\bot\}$, $\alpha[i \mapsto \sigma]$ denotes the assignment that maps i to σ and is otherwise equal to α.

We denote the set $Ag - \{i\}$ as $-i$, and $Ag - A$ as $-A$.

Definition 8 (Outcome)**.** A state w and an agent strategy assignment α defined on Ag determine a unique computation. We call this computation the outcome of w and α, and denote it as $out(w, \alpha)$.

We introduce SL as in (Mogavero et al. 2014). The definition is slightly modified, in which we use two seperate assignments to assign agents and strategy variables. Let StV be a countable non-empty set of strategy variables. We give the syntax of SL.

Definition 9 (SL syntax)**.** We define the syntax of SL formula φ as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2 \mid \exists s.\varphi \mid (i, s)\varphi,$$

where $p \in AP$, $i \in Ag$, and $s \in StV$.

Intuitively, strategy quantification $\exists s$ means "there exists a strategy s", while agent binding (i, s) means "bind agent i to strategy s". We use $\top$ for $true$ and $\bot$ for $false$. Standard abbreviations including $\mathbf{F}$ and $\mathbf{G}$ are defined as usual.

Definition 10. The set of free agents and variables of an SL formula φ, denoted $\mathsf{free}(\varphi)$, is defined inductively as follows (omitting cases of $\neg$ and $\wedge$):

- $\mathsf{free}(p) = \varnothing$; $\mathsf{free}(\mathbf{X}\varphi) = Ag \cup \mathsf{free}(\varphi)$;
- $\mathsf{free}(\varphi_1 \mathbf{U} \varphi_2) = Ag \cup \mathsf{free}(\varphi_1) \cup \mathsf{free}(\varphi_2)$;
- $\mathsf{free}(\exists s.\varphi) = \mathsf{free}(\varphi) - \{s\}$;
- $\mathsf{free}((i, s)\varphi) = (\mathsf{free}(\varphi) \cup \{s\}) - \{i\}$ if $i \in \mathsf{free}(\varphi)$;
- $\mathsf{free}((i, s)\varphi) = \mathsf{free}(\varphi)$ if $i \notin \mathsf{free}(\varphi)$;

A formula φ is a sentence if $\mathsf{free}(\varphi) = \varnothing$.

Definition 11. A variable strategy assignment $\chi : StV \to Str$ maps each strategy variable to a strategy. For a variable strategy assignment χ, a strategy variable s and a strategy σ, $\chi[s \mapsto \sigma]$ is the variable strategy assignment that maps s to σ and is otherwise equal to χ.

Definition 12 (SL Semantics)**.** Given a CGS $\mathcal{G}$, a state w, a variable strategy assignment χ, an agent strategy assignment α defined on Ag, we define the semantics of SL formulas inductively as follows (omitting cases of $\neg$ and $\wedge$):

- $\mathcal{G}, w \models_{\chi,\alpha} p$ if $p \in L(w)$.
- $\mathcal{G}, w \models_{\chi,\alpha} \exists s.\varphi$ if there exists a strategy $\sigma \in Str$ s.t. $\mathcal{G}, w \models_{\chi[s \mapsto \sigma],\alpha} \varphi$.
- $\mathcal{G}, w \models_{\chi,\alpha} (i, s)\varphi$ if $\chi(s) \in Str_i$, and we have $\mathcal{G}, w \models_{\chi,\alpha[i \mapsto \chi(s)]} \varphi$.
- $\mathcal{G}, w \models_{\chi,\alpha} \mathbf{X}\varphi$ if $\mathcal{G}, out(w, \alpha)[1] \models_{\chi,\alpha} \varphi$.
- $\mathcal{G}, w \models_{\chi,\alpha} \varphi_1 \mathbf{U} \varphi_2$ if there exists $k \in \mathbb{N}$ s.t. $\mathcal{G}, out(w, \alpha)[k] \models_{\chi,\alpha} \varphi_2$ and for all $i < k$, we have $\mathcal{G}, out(w, \alpha)[i] \models_{\chi,\alpha} \varphi_1$.

For a sentence φ we omit χ and α and write $\mathcal{G}, w \models \varphi$.

Given agents $i_1, \ldots, i_n$ and their respective goals $\varphi_1, \ldots, \varphi_n$, SL can express the existence of deterministic Nash equilibra (Mogavero et al. 2014), with the formula $\exists s_1 \ldots \exists s_n.(i_1, s_1) \ldots (i_n, s_n)\varphi_{nd}$, where φ_{nd} states that all agents have no intention to deviate from the joint strategy represented by $(s_1, \ldots, s_n)$: φ_{nd} is the conjunction of formulas $(\exists y.(i_k, y)\varphi_k) \to \varphi_k$ for $k = 1, \ldots, n$. We denote this formula with $\exists NE(\varphi_1, \ldots, \varphi_n)$.

Syntax and Semantics of $\mathrm{SL}_{\mathtt{IEDS}}$

In this section, we propose Strategy Logic with IEDS ($\mathrm{SL}_{\mathtt{IEDS}}$). We introduce its syntax and semantics, compare it to related logics, and analyze valid formulas in the logic.

JAADL extends ATL* with operators $(A)_\psi \varphi$ and $(A)_\psi^\infty \varphi$, meaning φ holds after EDS and IEDS, respectively, w.r.t. group A and goal ψ. Inspired by JAADL, we extend SL with strategy elimination and iterated elimination operators, written as $[\pi]$ and $[\pi]^\infty$ respectively. Different from JAADL and similar to rational verification (Wooldridge et al. 2016), different agents can have different goals, specified with the goal expression π. This allows us to reason about non-cooperative settings.

Definition 13 ($\mathrm{SL}_{\mathtt{IEDS}}$ Syntax)**.** We define the syntax of $\mathrm{SL}_{\mathtt{IEDS}}$ formula φ and goal expression π as follows:

$$\begin{gathered}\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2 \mid \\ \exists s.\varphi \mid (i, s)\varphi \mid [\pi]\varphi \mid [\pi]^\infty \varphi, \\ \pi ::= \varepsilon \mid \pi, (i : \varphi),\end{gathered}$$

where $p \in AP$, $i \in Ag$, and $s \in StV$.

$\mathrm{SL}_{\mathtt{IEDS}}$ formulas φ have an SL-like syntax, with temporal modalities $\mathbf{X}$ and $\mathbf{U}$, strategy quantifier $\exists s$ and agent binding operator (i, s). The two new operators are the operator $[\pi]$

for elimination of dominated strategies (EDS) and the operator $[\pi]^\infty$ for IEDS. The goal expression π in these operators denotes the goal of each agent, which is used to define the strategy dominance relation for the agent. For example, the pair $(i : \varphi)$ means agent i holds the goal φ.

We also define abbreviations for the elimination operators: $[\pi]^2\varphi := [\pi][\pi]\varphi$, and similarly for $[\pi]^k\varphi$. We also write π in abbreviation: We omit pairs like $(i : \top)$, and combine agents with the same goal into a coalition. For example, with $Ag = \{1, 2, 3, 4\}$, we write $(1 : \varphi_1), (2 : \varphi_1), (3 : \varphi_2), (4 : \top)$ as $(\{1, 2\} : \varphi_1), (3 : \varphi_2)$ or $(1, 2 : \varphi_1), (3 : \varphi_2)$.

Similarly to SL, we can define the set of free agents and variables in a formula.

Definition 14. The set of free agents and variables of $\mathrm{SL}_{\mathtt{IEDS}}$ formulas is defined as follows: $\mathsf{free}([\pi]\varphi) = \mathsf{free}(\varphi)$; $\mathsf{free}([\pi]^\infty\varphi) = \mathsf{free}(\varphi)$; and for other types of formulas, the definitions are the same as in SL. An $\mathrm{SL}_{\mathtt{IEDS}}$ formula φ is a sentence if $\mathsf{free}(\varphi) = \varnothing$.

In order to define the semantics of $\mathrm{SL}_{\mathtt{IEDS}}$, we need to interpret the goal expression π. We begin so by defining goal assignments.

Definition 15. A goal assignment goal maps each agent i to an $\mathrm{SL}_{\mathtt{IEDS}}$ formula φ. The formula $\mathsf{goal}(i)$ is called agent i's goal. For a goal assignment goal and an $\mathrm{SL}_{\mathtt{IEDS}}$ formula φ, $\mathsf{goal}[i \mapsto \varphi]$ is the goal assignment that maps agent i to φ and is otherwise equal to goal. The default goal assignment goal_0 maps each agent to $\top$.

The goal assignment maps each agent to her goal. Intuitively, every goal expression π denotes a goal assignment. We interpret a goal expression π as a goal assignment in the following way:

Definition 16. The goal expression π is evaluated inductively as follows: $[\![\varepsilon]\!] = \mathsf{goal}_0$; $[\![\pi_1, (i : \varphi)]\!] = [\![\pi_1]\!][i \mapsto \varphi]$.

Finally, we define semantics of $\mathrm{SL}_{\mathtt{IEDS}}$. The interpretation of $\mathrm{SL}_{\mathtt{IEDS}}$ formulas is defined w.r.t. a strategy space. We define two strategy space reduction operators $R_{\pi,w,\chi}(\Sigma)$ and $R^\infty_{\pi,w,\chi}(\Sigma)$, which mean the reduction of Σ via EDS and IEDS, respectively. The definitions of the interpretation of formulas and the reduction operators are mutually inductive. Thus the following definition is long, consisting of 3 parts.

Definition 17 ($\mathrm{SL}_{\mathtt{IEDS}}$ Semantics)**.** We define the interpretation of $\mathrm{SL}_{\mathtt{IEDS}}$ formulas and the strategy space reduction operators R and R^∞ mutually inductively as follows:

(1) Given a CGS $\mathcal{G}$, a state w, a strategy space Σ, a variable strategy assignment χ, an agent strategy assignment α defined on Ag, the interpretation of $\mathrm{SL}_{\mathtt{IEDS}}$ formulas is inductively defined as follows (omitting cases of $\neg$ and $\wedge$):

- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} p$ if $p \in L(w)$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} \exists s.\varphi$ if there exists a strategy $\sigma \in Str$ s.t. $\mathcal{G}, w, \Sigma \vDash_{\chi[s\mapsto\sigma],\alpha} \varphi$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} (i, s)\varphi$ if $\chi(s) \in \Sigma(i)$, and we have $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha[i\mapsto\chi(s)]} \varphi$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} [\pi]\varphi$ if $\mathcal{G}, w, R_{\pi,w,\chi}(\Sigma) \vDash_{\chi,\alpha} \varphi$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} [\pi]^\infty\varphi$ if $\mathcal{G}, w, R^\infty_{\pi,w,\chi}(\Sigma) \vDash_{\chi,\alpha} \varphi$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} \mathbf{X}\varphi$ if $\mathcal{G}, out(w, \alpha)[1], \Sigma \vDash_{\chi,\alpha} \varphi$.
- $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha} \varphi_1\mathbf{U}\varphi_2$ if there exists $k \in \mathbb{N}$ s.t. $\mathcal{G}, out(w, \alpha)[k], \Sigma \vDash_{\chi,\alpha} \varphi_2$ and for all $i < k$, we have $\mathcal{G}, out(w, \alpha)[i], \Sigma \vDash_{\chi,\alpha} \varphi_1$.

For a sentence φ we omit χ and α and write $\mathcal{G}, w, \Sigma \vDash \varphi$.

(2) We now define the domination relation between strategies. We begin with the notion of the compatible set M of a strategy. Given a goal expression π, an agent i, a state w, a variable assignment χ, a strategy space Σ, and $\sigma \in \Sigma(i)$, the compatible set of σ w.r.t. π, i, w, χ, Σ, written $M_{\pi,i,w,\chi,\Sigma}(\sigma)$, is the set of agent strategy assignments defined on $-i$ and restricted to Σ that can work with σ to satisfy agent i's goal $[\![\pi]\!](i)$ at w w.r.t. χ, i.e., for all agent strategy assignments α defined on $-i$ and restricted to Σ, $\alpha \in M_{\pi,i,w,\chi,\Sigma}(\sigma)$ iff $\mathcal{G}, w, \Sigma \vDash_{\chi,\alpha[i\mapsto\sigma]} [\![\pi]\!](i)$.

For strategies $\sigma, \sigma' \in \Sigma(i)$, we write $\sigma \geq_{\pi,i,w,\chi,\Sigma} \sigma'$ if $M_{\pi,i,w,\chi,\Sigma}(\sigma) \supseteq M_{\pi,i,w,\chi,\Sigma}(\sigma')$. Additionally, we write $\sigma >_{\pi,i,w,\chi,\Sigma} \sigma'$ if $M_{\pi,i,w,\chi,\Sigma}(\sigma) \supset M_{\pi,i,w,\chi,\Sigma}(\sigma')$, and we say σ dominates σ' in Σ w.r.t. π, i, w, χ.

(3) The reduction of a strategy space Σ w.r.t. goal π, state w, and variable assignment χ, written $R_{\pi,w,\chi}(\Sigma)$, is defined as follows: For all strategy $\sigma \in \Sigma(i)$, $\sigma \in R_{\pi,w,\chi}(\Sigma)(i)$ iff there is no $\sigma' \in \Sigma(i)$ s.t. $\sigma' >_{\pi,i,w,\chi,\Sigma} \sigma$. Note that if agent i has goal $\top$, then no strategy of i dominates another, thus $R_{\pi,w,\chi}(\Sigma)(i) = \Sigma(i)$. For $k \geq 2$, we define $R^k_{\pi,w,\chi}(\Sigma) = R_{\pi,w,\chi}(R^{k-1}_{\pi,w,\chi}(\Sigma))$. Finally, we define the iterated reduction in the following way: For each agent $i \in Ag$, $R^\infty_{\pi,w,\chi}(\Sigma)(i) = \bigcap_{k=0}^\infty R^k_{\pi,w,\chi}(\Sigma)(i)$.

Definition 18 (Payoff matrix)**.** Given a CGS $\mathcal{G}$, a state w, a strategy space Σ, a goal formula φ, a variable strategy assignment χ, the payoff matrix, denoted $P_{\mathcal{G},w,\Sigma,\varphi,\chi}$, is a Boolean matrix with indices labeled by agent strategy assignments α defined on Ag and restricted to Σ such that for all such α, $P_{\mathcal{G},w,\Sigma,\varphi,\chi}(\alpha) = 1$ iff $\mathcal{G}, w \vDash_{\chi,\alpha} \varphi$. We omit the subscripts of P if there is no ambiguity.

Finally, we use the Crossing road example to illustrate the EDS and IEDS operators of $\mathrm{SL}_{\mathtt{IEDS}}$.

Example 3. We focus on the memoryless case, and consider three situations where agents have different goals. The payoff matrices of the three situations are given in Table 1.

First, both agents have the safety goal of never crash, denoted $\varphi^s := \mathbf{G}\neg crash$. Then the formula $[(1, 2 : \varphi^s)]\forall x.\forall y.(1, x)(2, y)\varphi^s$ holds. It says that after a single EDS, any remaining joint strategies achieve the goal. This is because: as can be easily seen from Table 1a, for each agent, the strategy σ_s dominates σ_p, which is thus eliminated.

Second, besides the safety goal, each agent also has the liveness goal of reaching the destination, denoted $\varphi^l_i := \mathbf{F}win_i$ for $i = 1, 2$. Then the formula $[(1 : \varphi^s \wedge \varphi^l_1), (2 : \varphi^s \wedge \varphi^l_2)]\forall x.\forall y.(1, x)(2, y)(\varphi^s \wedge \varphi^l_1 \wedge \varphi^l_2)$ does not hold. As can be seen from Table 1b, no strategy can be eliminated, and if both agents choose σ_s or both choose σ_p, the joint goal cannot be achieved.

Third, suppose that according to the law, vehicle 1 should yield the right of way, *i.e.*, it should let vehicle 2 go first. We modify our language and the CGS to reflect it:

- Let $illegal_1, illegal_2 \in AP$;
- Let $illegal_1 \in L(w_2)$ and $illegal_1 \in L(w_4)$.

(a)

	σ_p	σ_s
σ_p	$0,0$	$1,1$
σ_s	$1,1$	$1,1$

(b)

	σ_p	σ_s
σ_p	$0,0$	$1,1$
σ_s	$1,1$	$0,0$

(c)

	σ_p	σ_s
σ_p	$0,0$	$0,1$
σ_s	$1,1$	$0,0$

Table 1: Payoff matrices of the situations in Example 3. Agent 1 (resp. 2) adopts row (resp. column) strategies.

Here, atom $illegal_i$ represents i breaks the law. We assume the agents have goal $\pi_{legal} := (1 : \varphi^s \wedge \varphi_1^l \wedge \mathbf{G}\neg illegal_1), (2 : \varphi^s \wedge \varphi_2^l \wedge \mathbf{G}\neg illegal_2)$.

From Table 1c, for agent 1, σ_p will be dominated by σ_s and is eliminated; Yet, for agent 2, neither strategy dominates the other. Thus the formula $[\pi_{legal}]\forall x.\forall y.(1,x)(2,y)(\varphi^s \wedge \varphi_1^l \wedge \varphi_2^l)$ does not hold. However, the formula $[\pi_{legal}]^\infty \forall x.\forall y.(1,x)(2,y)(\varphi^s \wedge \varphi_1^l \wedge \varphi_2^l)$ holds. This is because: in the second round of EDS, and agent 2's σ_s is eliminated. With agent 1 does σ_s and agent 2 does σ_p, the joint goal is achieved.

We now compare $\text{SL}_{\texttt{IEDS}}$ to JAADL and ATL with minimal rationality of (Li, Lorini, and Mittelmann 2025). We show that a fragment of JAADL can be translated to $\text{SL}_{\texttt{IEDS}}$. Aside from the EDS and IEDS operators, JAADL introduces action atoms and regular expressions, which are not expressible in SL. We name as JAATL the fragment of JAADL, which is the extension of ATL* with the operators $(A)_\psi\varphi$ and $(A)_\psi^\infty\varphi$. Then JAATL can be translated to $\text{SL}_{\texttt{IEDS}}$ as follows: In the base case, we use the translation from ATL* to SL. In the induction case, we have $tr((A)_\psi\varphi) = [(A : tr(\psi))]tr(\varphi)$, and similarly for the IEDS operator.

Li, Lorini, and Mittelmann (2025) extends coalition logic and ATL with minimal rationality modalities $\langle\langle C\rangle\rangle^{rat}\psi$ for existence of non-dominated strategies to achieve ψ, where the domination relation between strategies is defined according to a total preorder on computations of the CGS. In comparison, our logic has a couple of advantages: First, they only consider EDS, while we consider both EDS and IEDS. Secondly, our definition of domination between strategies is defined via specifying goals for agents, which is in line with works of rational verification (Wooldridge et al. 2016), and we think this way is more natural and easier. Finally, ours allows to reason in the same formula about different domination relations specified via different goal expressions.

Finally, We discuss valid formulas in $\text{SL}_{\texttt{IEDS}}$. Liu et al. (2020) analyzed valid formulas in JAADL (Propositions 1-4). When these valid formulas are in JAATL, their translation to $\text{SL}_{\texttt{IEDS}}$ are valid, too. Additionally, we have:

Proposition 1. *The following formulas are valid:*

- $[\pi]^k(\varphi_1 \to \varphi_2) \to ([\pi]^k\varphi_1 \to [\pi]^k\varphi_2), k \in \mathbb{N} \cup \{\infty\}$.
- $\neg[\pi]^k\varphi \leftrightarrow [\pi]^k\neg\varphi, k \in \mathbb{N} \cup \{\infty\}$.

Moreover, some relations between NE and IEDS can be represented as valid formulas in $\text{SL}_{\texttt{IEDS}}$. An example is the property that if after IEDS all remaining strategy profiles can achieve both agents' goals, then there exists a Nash Equilibrium in the original game, as shown below:

Proposition 2. *Let $Ag = \{1,2\}$, π be $(1 : \varphi_1)$,$(2 : \varphi_2)$, where φ_1, φ_2 are $\text{SL}_{\texttt{IEDS}}$ formulas. Then, $[\pi]^\infty(\varphi_\exists \wedge \varphi_\forall) \to \exists NE(\varphi_1, \varphi_2)$ is valid, where $\varphi_\forall$ is $\forall x_1 \forall x_2\ (1,x_1)(2,x_2)(\varphi_1 \wedge \varphi_2)$, $\varphi_\exists$ is $\exists x_1 \exists x_2(1,x_1)(2,x_2)(\varphi_1 \wedge \varphi_2)$, and $\exists NE(\varphi_1, \varphi_2)$ is the formula expressing the existence of an NE w.r.t. goals φ_1 and φ_2.*

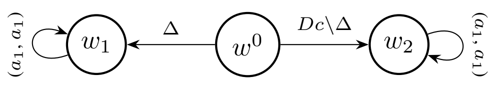

Figure 2: CGSes in $\mathcal{C}_1$ and $\mathcal{C}_2$.

$$\begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & 1 & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \\ & & & & & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \\ & & & 1 & & 1 \end{bmatrix}$$

Figure 3: Payoff matrices of $\mathcal{G}_3^1$(left) and $\mathcal{G}_3^2$(right).

Expressiveness Comparison

In this section, we compare the expressive power of $\text{SL}_{\texttt{IEDS}}$ to its fragments. We prove that SL is as expressive as $\text{SL}_{\texttt{EDS}}$, *i.e.*, SL with only the EDS operator, while $\text{SL}_{\texttt{IEDS}}$ is strictly more expressive than SL. Our proofs apply to both memoryless and memoryful cases.

We first present the classic definition of expressive power.

Definition 19. Let L_1 and L_2 be two logics interpreted over the same classes of models $\mathcal{M}$. We say that L_2 is at least as expressive as L_1, written $L_1 \preceq_e L_2$, if for every formula $\varphi_1 \in L_1$, there exists a formula $\varphi_2 \in L_2$ s.t. for every model $M \in \mathcal{M}$, we have $M \models \varphi_1$ iff $M \models \varphi_2$.

Theorem 3. *SL $=_e$ $\text{SL}_{\texttt{EDS}}$.*

Proof Sketch. To show that $\text{SL}_{\texttt{EDS}}$ can be translated to SL, it suffices to prove that the formula $[\pi]\varphi$ with no EDS operator in its subformulas can be expressed in SL. For each agent binding subformula $(i,x)\varphi_1$ of φ, we write a formula $E_i^\psi(x)$ to denote that x is dominated by another strategies of i, according to i's goal, and rewrite $(i,x)\varphi_1$ as $\neg E_i^\psi(x) \wedge (i,x)\varphi_1$. Finally, we remove the operator $[\pi]$. □

Theorem 4. *SL $\prec_e$ $\text{SL}_{\texttt{IEDS}}$.*

Proof. The main idea of the proof is that we can construct two classes of CGSes $\mathcal{C}_1 = \{\mathcal{G}_m^1 \mid m \geq 3\}$ and $\mathcal{C}_2 = \{\mathcal{G}_m^2 \mid m \geq 3\}$ s.t. there is an $\text{SL}_{\texttt{IEDS}}$ formula to distinguish between $\mathcal{C}_1$ and $\mathcal{C}_2$, but no SL formula can do so.

The CGSes in the two classes all have the same structure as given in Figure 2. $\mathcal{G}_m^1$ and $\mathcal{G}_m^2$ both have $2m$ actions $a_1, \ldots, a_{2m}$, where for both agents, a_1 is the only possible action in w_1 and w_2, and they only differ in the set of decisions Δ that can achieve the goal of getting to the winning

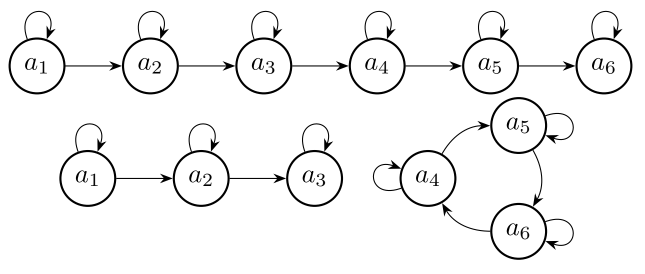

Figure 4: The graph of $G(\mathcal{G}_3^1)$ (up) and $G(\mathcal{G}_3^2)$ (down).

state w_1 (which is the only state labeled with win in both models) from w^0. We note that on such models, what distinguishes a strategy from another is the action to take in w^0. Thus a strategy can be represented by the action taken in w^0. So such a model can be uniquely represented with a payoff matrix. We give the payoff matrices of $\mathcal{G}_3^1$ and $\mathcal{G}_3^2$ in Figure 3. In general, the only difference between the payoff matrices of $\mathcal{G}_m^1$ and $\mathcal{G}_m^2$ is that the 1 at $(m+1, m+1)$ is moved to $(2m, m+1)$. It's clear that in CGSes in $\mathcal{C}_1$, every remaining strategy profile can achieve the goal after IEDS (for both agents, only strategy a_{m+1} is left), and it is not the case in $\mathcal{C}_2$ (for both agents, any strategy a_i with $i \geq m+1$ is left). Thus, the following SL_{IEDS} formula can distinguish $\mathcal{C}_1$ and $\mathcal{C}_2$: $\varphi = [(1,2 : \mathbf{X}win)]^\infty \forall x.\forall y.(1,x)(2,y)\mathbf{X}win$.

To prove that no SL formula can distinguish $\mathcal{C}_1$ and $\mathcal{C}_2$, we translate SL formulas on $\mathcal{C}_1$ and $\mathcal{C}_2$ to FOL (first-order logic) formulas and models in $\mathcal{C}_1$ and $\mathcal{C}_2$ to first-order structures so that the satisfaction relation is maintained. The key information of the CGSes is what decisions make the transition from w^0 to w_1. Thus we introduce a binary predicate $P(x,y)$ for this, and to translate a CGS $\mathcal{G}$ to a first-order structure $G(\mathcal{G})$, we use the set of actions of $\mathcal{G}$ as the domain, and interpret $P(x,y)$ according to $\mathcal{G}$. Figure 4 shows the first-order structures obtained from $\mathcal{G}_3^1$ and $\mathcal{G}_3^2$, where actions in w^0 are viewed as elements, and a directed edge from element a to b means $P(a,b)$ is true. We now show how to translate formulas. Firstly, we note that on models from the two classes, every SL formula can be written without $\mathbf{U}$, as $\varphi_1 \mathbf{U} \varphi_2$ is equivalent to $\varphi_2 \vee (\varphi_1 \wedge \mathbf{X}\varphi_2)$. Thus the key of the translation is how to translate $\mathbf{X}\varphi$. For this purpose, we can prove from a simple induction on φ that for any $w \in \{w_1, w_2\}$, any SL formula φ is equivalent to either $\top$ or $\bot$; we write $w \models \varphi$ in the former case and $w \not\models \varphi$ in the latter. Thus, let x and y be the variables of the strategies assigned to agents 1 and 2 respectively, $\mathbf{X}\varphi$ can be translated as follows:

- $\top$, if $w_1 \models \varphi$ and $w_2 \models \varphi$;
- $P(x,y)$, if $w_1 \models \varphi$ and $w_2 \not\models \varphi$;
- $\neg P(x,y)$, if $w_1 \not\models \varphi$ and $w_2 \models \varphi$;
- $\bot$, if $w_1 \not\models \varphi$ and $w_2 \not\models \varphi$.

We denote this translation as $tr_{x,y}(\mathbf{X}\varphi)$. Based on the given translation of $\mathbf{X}\varphi$, we give the translation from SL formulas to FOL. We first rename the variables s.t. no quantifiers share variables. We use an agent variable assignment $\beta : Ag \to StV$ in the translation to record the strategy variables assigned to agent 1 and 2. The formula $tr(\varphi, \beta)$ is defined inductively as follows:

- $tr(win, \beta) = \bot$, as $win \notin L(s_0)$;
- $tr(\neg\varphi, \beta) = \neg tr(\varphi, \beta)$;
- $tr(\varphi_1 \wedge \varphi_2, \beta) = tr(\varphi_1, \beta) \wedge tr(\varphi_2, \beta)$;
- $tr(\exists s.\varphi, \beta) = \exists s.tr(\varphi, \beta)$;
- $tr((i,s)\varphi, \beta) = tr(\varphi, \beta[i \mapsto s])$;
- $tr(\mathbf{X}\varphi, \beta) = tr_{\beta(1),\beta(2)}(\mathbf{X}\varphi)$

For example, SL formula $\exists x.\exists y.\exists z.(1,x)(2,y)\mathbf{X}p \wedge (1,x)(2,z)\mathbf{X}\neg p$ can be translated to $\exists x.\exists y.\exists z.P(x,y) \wedge \neg P(x,z)$. The correctness of the translation can be proven by a simple induction on the formula if no quantifiers share variables.

Finally, we make use of a result from Libkin (2004) which states that no FOL sentences can distinguish two classes of graphs: one with linear orders of size $2m$, and the other with the union of a linear order of size m and a directed cycle of size m, as exemplified by Figure 4. Therefore, no SL sentences can distinguish the classes $\mathcal{C}_1$ and $\mathcal{C}_2$. $\square$

Model Checking Memoryless SL_{IEDS}

In this section, we prove that model checking memoryless SL_{IEDS} is EXPTIME-complete. This complexity is higher than that of model-checking memoryless SL, which is PSPACE-complete (Čermák et al. 2018).

We first define the model checking problem for SL_{IEDS}.

Definition 20. Given a CGS $\mathcal{G}$, a state w on $\mathcal{G}$, an SL_{IEDS} sentence φ, the full memoryless or memoryful strategy space Σ_f, the problem is to check if $\mathcal{G}, w, \Sigma_f \models \varphi$.

Upper Bound

Our model checking algorithm MC for memoryless SL_{IEDS} is derived directly from the semantics. Algorithm MC takes a CGS $\mathcal{G} = \langle W, L, P, \tau, w^0 \rangle$, a state $w \in W$, a strategy space Σ, a formula φ and two assignments χ and α as parameters, and returns whether $\mathcal{G}, w, \Sigma \models_{\chi,\alpha} \varphi$ by checking all the subformulas of φ recursively. All the cases except the case of $\mathbf{U}$ are straightforward. To process $\varphi_1 \mathbf{U} \varphi_2$, we need to check if φ_2 turns true before φ_1 turns false on the computation $out(w, \alpha)$. As $out(w, \alpha)$ forms a loop in memoryless semantics, we only need to check at most $|W|$ states on $out(w, \alpha)$. Since the definition of the strategy elimination operator is involved, we give the subprocess RS for processing it in Algorithm 1. The iterated elimination process RS^∞ is done by calling RS on Σ repeatedly until reaching a fixed point.

Theorem 5. *Model checking memoryless SL_{IEDS} is in EXPTIME, and can be done in time exponential to the model size m and the formula size l.*

Proof Sketch. We prove that the algorithm takes $\mathcal{O}(2^{ml})$ time by induction on l. Processing $\varphi_1 \mathbf{U} \varphi_2$ takes $\mathcal{O}(m(2^{m|\varphi_1|} + 2^{m|\varphi_2|})) = \mathcal{O}(2^{ml})$ time. Processing $\exists s$ calls MC for every strategy. There are $\mathcal{O}(2^m)$ strategies, thus it takes $\mathcal{O}(2^m 2^{m(l-1)}) = \mathcal{O}(2^{ml})$ time. As to $[\pi]\varphi_1$ and $[\pi]^\infty\varphi_1$, we have: since there are $\mathcal{O}(2^m)$ agent strategy

Algorithm 1: Reducing strategy spaces

```
$RS(\mathcal{G}, w, \Sigma, \chi, \pi)$:
1: for each $i \in Ag$ do
2:     for each $\sigma_i \in \Sigma(i)$ do
3:         for each $\alpha$ of $-i$ that is restricted to $\Sigma$ do
4:             if $MC(\mathcal{G}, w, \Sigma, [\![\pi]\!](i), \chi, \alpha[i \mapsto \sigma_i])$ then
5:                 add $\alpha$ to $M_{\pi,w,\chi,\Sigma}(\sigma_i)$
6:     for each $\sigma_i, \sigma_i' \in \Sigma(i)$ do
7:         if $M_{\pi,w,\chi,\Sigma}(\sigma_i) \supset M_{\pi,w,\chi,\Sigma}(\sigma_i')$ then
8:             $\Sigma(i) \leftarrow \Sigma(i) - \{\sigma_i'\}$
9: return $\Sigma$
```

assignments, calling RS takes $\mathcal{O}(2^m 2^{m|\pi|})$ time; since RS^∞ performs RS at most $\mathcal{O}(2^m)$ times, calling RS^∞ takes $\mathcal{O}(2^m 2^m 2^{m|\pi|})$ time; thus processing both $[\pi]\varphi_1$ and $[\pi]^\infty\varphi_1$ takes $\mathcal{O}(2^{m|[\pi]|} + 2^{m|\varphi_1|}) = \mathcal{O}(2^{ml})$ time. □

Lower Bound

To prove the EXPTIME-hardness, we reduce a EXPTIME-hard problem to model checking memoryless SL$_{\texttt{IEDS}}$. Pauly (2016) proved the 2-Simultaneous IEDS problem is P-hard. By a result of Papadimitriou and Yannakakis (1986), we obtain that 2-Simultaneous IEDS on succinct representation of payoff matrices is EXPTIME-hard. Then, we reduce this problem to model checking memoryless SL$_{\texttt{IEDS}}$.

We first need to introduce succinct representation sof matrices proposed by Galperin and Wigderson (1983):

Definition 21 (Succinct representation)**.** Let A be a Boolean matrix with $N = |I| \leq 2^n$ rows and $|J| \leq 2^n$ columns, where I and J are the sets of row numbers and column numbers. Let $\overline{x}$ be the binary representation of a number x. A circuit C_A with size polynomial in n (polylograthmic in N) is a succinct representation of A if the following properties hold: C_A is a combinational circuit with two inputs of n bits each, and for $i \in I, j \in J$, $C_A(\bar{i}, \bar{j}) = 1$ iff $A_{ij} = 1$.

We now introduce a generalization of the conclusion by Papadimitriou and Yannakakis (1986) about succinct representations. We begin with introducing the P-bounded halting problem and the notion of projections (Skyum and Valiant 1985). Given a Turing machine M, an input x and a number T, the bounded halting problem is the decision problem that decides whether M accepts x in T steps. This problem is P-complete by definition if T is polynomial to the size of M and x. A mapping π from a language $L_1 \subseteq \{0,1\}^*$ to another L_2 is a projection iff the following conditions hold: (1) For any $x \in L_1$ (denoted as $x_1 \dots x_m$) with length m, its image $y = \pi(x)$ (denoted as $y_1 \dots y_l$) has length $l = m^c$, where c is a constant; (2) There exists a PTIME algorithm that computes the mapping $\delta : \{1, \dots, l\} \to \{0, 1, x_1, \dots, x_m, \neg x_1, \dots, \neg x_m\}$ s.t. $y_1 \dots y_l = \delta(1) \dots \delta(l)$. That is, any output bit can be computed from some input bit. It follows that if L_1 can be projected to L_2, and L_2 can be projected to L_3, then L_1 can be projected to L_3.

Theorem (Papadimitriou and Yannakakis 1986)**.** *Given a decision problem Π on Boolean matrices, if there is a projection from the P-bounded halting problem to Π, then the problem on succinct representations of matrices is EXPTIME-hard.*

This conclusion relates a P-hard problem on Boolean matrices to an EXPTIME-hard one on succinct representations. To apply it, we need the 2-Simultaneous IEDS given and proven to be P-hard by Pauly (2016). Given a finite two-player game with Boolean payoff, represented by two payoff matrices, the problem 2-Simutaneous IEDS is to decide whether a given strategy of player 1, represented by the row index, remains after IEDS. We consider this problem on succinct representations, where we use a circuit to represent payoff matrices of both players. Without loss of generality, we assume that a single input bit decides whether the circuit is calculating payoff of player 1 (if the bit is 0) or 2 (if the bit is 1).

Proposition 6. *2-Simultaneous IEDS on succinct representation of payoff matrices is EXPTIME-hard.*

Proof Sketch. It is proven by Pauly (2016) that 2-Simultaneous is P-complete. Pauly's proof is a projection from a variation of the monotone circuit value problem. By a similar proof as the one shown in (Greenlaw, Hoover, and Ruzzo 1995), we can show that the monotone circuit value problem variation can be projected from the P-bounded halting problem. □

By a reduction from this problem, we have:

Theorem 7. *Model checking memoryless SL$_{\texttt{IEDS}}$ is EXPTIME-hard.*

Proof. Given a circuit C representing payoff matrices of both players and a strategy of player 1 represented with a binary number i, we construct in polynomial time a CGS $\mathcal{G}_c$ with 3 agents to simulate the game and a formula φ_c, s.t. φ_c holds on $\mathcal{G}_c$ in memoryless semantics iff the answer of 2-Simultaneous on C and i is true. The idea of constructing $\mathcal{G}_c$ and φ_c is as follows. We divide the gates of C into three types: input gates of player 1, input gates of player 2, and the other gates. Thus we introduce 3 agents corresponding to the 3 types of gates. We number the u gates of C, including the input gates, topologically. $\mathcal{G}_c$ has $3u$ states. Figure 5 shows the general structure of $\mathcal{G}_c$. Each s_j state has a controlling agent. For example, if s_j is corresponding to an input gate of player 1, then the controlling agent of s_j is agent 1. Agent 1 or 2 acts on the states controlled by her as follows: if the input bit of the corresponding gate is 1 (resp. 0), then agent

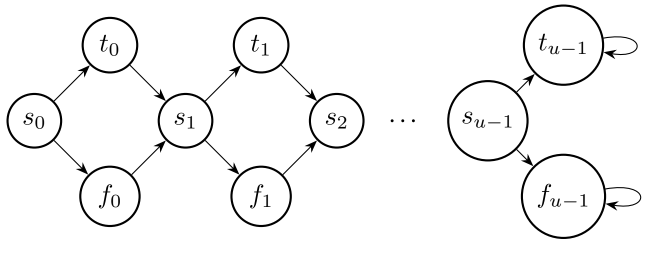

Figure 5: The CGS $\mathcal{G}_c$. The initial state $w^0 = s_0$.

1 on s_j chooses an action which makes the transition to t_j (resp. f_j). Agent 3 also acts under the states controlled by her. The formula φ_c will encode the constraint that the acting of agent 3 must follow the circuit. For example, if gate j is an AND gate of the outputs of gate j_1 and j_2, then agent 3 chooses action leading to t_j iff the controlling agents of s_{j_1} and s_{j_2} choose actions leading to t_{j_1} and t_{j_2}. Formula φ_c states that agents 1 and 2 do IEDS according to the correct payoff calculated by agent 3, and after IEDS, there is a strategy of agent 1 left that performs the same as the strategy i of player 1. We now give the detailed construction of the CGS $\mathcal{G}_c$ and the formula φ_c.

The CGS $\mathcal{G}_c$ is constructed as follows:

- $AP = \{eval_t, eval_a, win, in_1, in_2, \dots, in_u\}$;
- $Ac = \{\mathsf{yes}, \mathsf{no}, \mathsf{idle}\}$, and $Ag = \{1, 2, 3\}$;
- $W = \{s_j, t_j, f_j \mid 0 \leq j < u\}$.
- Every t_j is labeled with $eval_t$. The state f_j corresponding to the input gate controlled by agent 3 is labeled with $eval_a$. The state t_{u-1} is labeled with win. We label the states s_j controlled by 1 with the atoms in_k, the first one is labeled in_1, the second one in_2, respectively.
- For any agent i, if she controls state s_j, then $P_i(s_j) = \{\mathsf{yes}, \mathsf{no}\}$. Any other states w have $P_i(w) = \{\mathsf{idle}\}$.
- τ is defined as in Figure 5. In state s_j controlled by i, the next state will be t_j (resp. f_j) if i does yes (resp. no).

Intuitively, $eval_t$ means that the gate is determined to have value 1, $eval_a$ means that the circuit is evaluating the payoff of player 1, win means that the circuit outputs 1. Atoms in_k labels the states corresponding to input gates of player 1. We assume there are $u' < u$ such states, and let their numbers be $p_1, p_2, \dots, p_{u'}$. Therefore, the set of strategies on $\mathcal{G}_c$ for agent 1 and 2 would correspond to the set of strategies of player 1 and 2 in the matrix form of the finite two-player game. We denote the strategy that corresponds to the given strategy i as σ_1.

To construct φ_c, we first give the LTL formula ψ_c that represents the correctness of the evaluation, namely, every gate's value should be in line with their inputs. For an AND gate numbered j with input gates $j_1, j_2, \dots, j_k$, the formula ψ_j is $\mathbf{X}^{2j+1} eval_t \leftrightarrow \bigwedge_{1 \leq m \leq k} (\mathbf{X}^{2j_m+1} eval_t)$, which means that the value of this gate is true iff the values of all inputs to the gate are decided by the agents to be true. The formula stating the correctness for an OR gate or a NOT gate can be defined in a similar way. Our formula ψ_c that represents the correctness of the evaluation is written as the conjunction of all the ψ_j, while j ranges over all of the non-input gates.

Now it is possible to write the agents' goals in the IEDS to let it correspond to the elimination in the original two-player game on matrices. Agent 1's goal φ_1 is written as $\psi_c \wedge \mathbf{F} eval_a \to \mathbf{F} win$, which means if the evaluation is correct, and it is evaluating agent 1's payoff, then the evaluation should be true, which means agent 1 got payoff 1. Similarly, agent 2's goal φ_2 is written as $\psi_c \wedge \mathbf{G} \neg eval_a \to \mathbf{F} win$. The goal expression π is written as $(1 : \varphi_1), (2 : \varphi_2), (3 : \top)$.

The formula φ_c is given as follows. Let $f(\mathsf{yes}) = eval_t$, $f(\mathsf{no}) = \neg eval_t$. As the strategies of agent 1 have their behaviors only differ in the u' states controlled by 1, and in any play these state will always be visited, we can write it as

$$[\pi]^{\infty} \exists x. \forall y. \forall z. (1, x)(2, y)(3, z) \bigwedge_{1 \leq k \leq u'} \mathbf{F}(in_k \wedge \mathbf{X} f(\sigma_1(s_{p_k}))).$$

This formula means that after the IEDS, there exists a strategy σ of 1 that in all states s_{p_k}, which are labeled as in_k and controlled by 1, will act in a way that the result corresponds to σ_1's actions, *i.e.*, for $k = 1, \dots, u'$, if $\sigma_1(s_{p_k}) = \mathsf{yes}$, then on the computation generated with σ, in_k will be followed by $eval_t$, and if $\sigma_1(s_{p_k}) = \mathsf{no}$, then in_k will be followed by $eval_f$. σ is effectively σ_1. Therefore, the original problem of 2-Simultaneous on succinct representations is reduced to model checking φ_c on $\mathcal{G}_c$ in the memoryless case.

The formula φ_c and the CGS $\mathcal{G}_c$ both have sizes polynomial to u, the number of gates in C. Since u is polynomial in n, both structures can be constructed in time polynomial in n, thus the reduction is polynomial. ☐

Combining Theorem 5 and Theorem 7, we have

Corollary 8. *Model checking memoryless SL$_{\mathtt{IEDS}}$ is EXPTIME-complete.*

Liu et al. (2020) proved that model checking memoryless JAADL is in EXPTIME, but the lower bound was left open. Note that our technique for proving the complexity lower bound of model checking memoryless SL$_{\mathtt{IEDS}}$ can be used in JAADL as well, with the modification that the decision problem used should be 2-Simultaneous in the fully cooperative setting. Such problem can also be proved as P-complete by a projection from the monotone circuit value problem variation. Therefore, we can conclude that model checking memoryless JAADL is EXPTIME-complete as well.

Mogavero et al. (2014) defined a number of SL fragments such as SL[NG], SL[BG], and SL[1G], where SL[1G] is the most restrictive and its model-checking problem is 2EXPTIME-complete while model-checking SL is non-elementary. We can define the corresponding SL$_{\mathtt{IEDS}}$ fragments by extending SL fragments with EDS and IEDS operators. However, note that the SL$_{\mathtt{IEDS}}$ formula φ_c in our proof of Theorem 7 is in SL$_{\mathtt{IEDS}}$[1G]. Thus, model-checking memoryless SL$_{\mathtt{IEDS}}$[1G] is already EXPTIME-complete.

Conclusion

In this work, we propose SL$_{\mathtt{IEDS}}$ – an extension of strategy logic SL with EDS and IEDS operators. Different from JAADL (Liu et al. 2020) and similar to rational verification (Wooldridge et al. 2016), when defining dominance of strategies, different agents can have different goals. With this extension, we can reason about rational strategic abilities in multi-agent systems based on the concepts of Nash Equilibrium, IEDS, and bounded rationality. We can also use SL$_{\mathtt{IEDS}}$ to reason about joint abilities. We prove that the EDS operator can be expressed in SL, but SL$_{\mathtt{IEDS}}$ is strictly more expressive than SL. Finally, we prove that model checking memoryless SL$_{\mathtt{IEDS}}$ is EXPTIME-complete. Our future work will focus on exploring the model checking problem in the memoryful case.

Acknowledgments

We thank the anonymous reviewers for helpful comments. We acknowledge support from the Natural Science Foundation of China under Grant No. 62076261.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM (JACM)*, 49(5): 672–713.

Aminof, B.; De Giacomo, G.; Rubin, S.; et al. 2021. Best-effort synthesis: Doing your best is not harder than giving up. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*.

Berwanger, D. 2007. Admissibility in infinite games. In *24th Annual Symposium on Theoretical Aspects of Computer Science (STACS-07)*.

Bulling, N.; Jamroga, W.; and Dix, J. 2008. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1): 51–114.

Čermák, P.; Lomuscio, A.; Mogavero, F.; and Murano, A. 2018. Practical verification of multi-agent systems against SLK specifications. *Information and Computation*, 261: 588–614.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Information and Computation*, 208(6): 677–693.

Galperin, H.; and Wigderson, A. 1983. Succinct representations of graphs. *Information and Control*, 56(3): 183–198.

Gigerenzer, G.; Todd, P. M.; and the ABC Research Group. 2000. *Simple heuristics that make us smart*. Oxford University Press.

Greenlaw, R.; Hoover, H. J.; and Ruzzo, W. L. 1995. *Limits to parallel computation: P-completeness theory*. Oxford University Press.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2014. Reasoning about equilibria in game-like concurrent systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference (KR-14)*.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. J. 2017. Reasoning about equilibria in game-like concurrent systems. *Annals of Pure and Applied Logic*, 168(2): 373–403.

Gutierrez, J.; Murano, A.; Perelli, G.; Rubin, S.; Steeples, T.; and Wooldridge, M. J. 2021. Equilibria for games with combined qualitative and quantitative objectives. *Acta Informatica*, 58(6): 585–610.

Gutierrez, J.; Najib, M.; Perelli, G.; and Wooldridge, M. J. 2023. On the complexity of rational verification. *Annals of Mathematics and Artificial Intelligence*, 91(4): 409–430.

Huang, X.; and Ruan, J. 2017. ATL Strategic Reasoning Meets Correlated Equilibrium. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*.

Hyland, D.; Gutierrez, J.; Krishna, S.; and Wooldridge, M. J. 2024. Rational Verification with Quantitative Probabilistic Goals. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-24)*.

Kupferman, O.; Perelli, G.; and Vardi, M. Y. 2016. Synthesis with rational environments. *Annals of Mathematics and Artificial Intelligence*, 78(1): 3–20.

Li, Y.; Lorini, E.; and Mittelmann, M. 2025. Rational Capability in Concurrent Games. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-25)*.

Libkin, L. 2004. *Elements of Finite Model Theory*. Springer.

Liu, Z.; Xiong, L.; Liu, Y.; Lespérance, Y.; Xu, R.; and Shi, H. 2020. A Modal Logic for Joint Abilities under Strategy Commitments. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*.

Lorini, E. 2016. A minimal logic for interactive epistemology. *Synthese*, 193(3): 725–755.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4): 1–47.

Osborne, M. J.; and Rubinstein, A. 1994. *A course in game theory*. MIT press.

Papadimitriou, C. H.; and Yannakakis, M. 1986. A note on succinct representations of graphs. *Information and control*, 71(3): 181–185.

Pauly, A. 2016. The computational complexity of iterated elimination of dominated strategies. *Theory of Computing Systems*, 59: 52–75.

Russell, S. J.; and Wefald, E. 1991. *Do the right thing - studies in limited rationality*. MIT Press.

Simon, H. A. 1955. A behavioral model of rational choice. *The quarterly journal of economics*, 99–118.

Skyum, S.; and Valiant, L. G. 1985. A complexity theory based on Boolean algebra. *Journal of the ACM (JACM)*, 32(2): 484–502.

Wooldridge, M.; Gutierrez, J.; Harrenstein, P.; Marchioni, E.; Perelli, G.; and Toumi, A. 2016. Rational verification: From model checking to equilibrium checking. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.