Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Verification of General Games with Imperfect Information using Strategy Logic

Yifan He[1] , **Munyque Mittelmann**[2] , **Aniello Murano**[2] , **Abdallah Saffidine**[1] , **Michael Thielscher**[1]

[1]University of New South Wales, Australia
[2]University of Naples Federico II, Italy

{yifan.he1,mit}@unsw.edu.au, {munyque.mittelmann, aniello.murano}@unina.it, abdallahs@gmail.com

Abstract

The Game Description Language with Imperfect Information (GDL-II) is a lightweight formalism for representing the rules of arbitrary games, including those where players have private information. Its purpose is to build general game-playing systems, that is, automated players that can understand the rules of games and learn how to play them without human intervention. Epistemic Strategy Logic (SLK), on the other hand, is a rich logical framework for reasoning about multi-agent systems and the strategic behavior of agents with partial observability. To enable a general game-playing system to take advantage of this rich formalism for the automatic verification of properties of games, we present a formal translation from GDL-II games to SLK models. We prove the correctness of this translation and show how crucial properties of general games, including playability and the existence of Nash equilibria, can be expressed as formulas in SLK. Finally, we demonstrate the application of an existing model-checking system to verify the properties of GDL-II games.

1 Introduction

The Game Description Language (GDL) has been developed as a lightweight knowledge representation formalism for describing the rules of arbitrary games. It is used as the input language for general game-playing (GGP) systems, which can learn to play any new game from the mere rules and without human intervention, thus exhibiting a form of general intelligence. While the original GDL defined for the first AAAI GGP competition (Love, Genesereth, and Hinrichs 2006; Genesereth and Björnsson 2013) was restricted to games in which players have complete information about the game state, the language has later been extended to GDL-II to be able to describe any game with imperfect information (Thielscher 2011; Schiffel and Thielscher 2014).

As a lightweight specification language, GDL merely provides means for *representing* the rules of a game while a crucial aspect of general game playing is the ability to automatically *reason* about a given specification. GGP systems require this ability as a basis for search (Kuhlmann, Dresner, and Stone 2006; Clune 2007), to use machine learning for game playing (Goldwaser and Thielscher 2020; Gunawan 2023), and for solving games (Thielscher 2009; He, Saffidine, and Thielscher 2024). Automated reasoning can also be used for analysing general games (Ruan, Van Der Hoek, and Wooldridge 2009). Important basic properties of games include, for example, universal termination, i.e., ensuring that a game described in GDL is guaranteed to terminate; playability, i.e., ensuring that in all reachable states, every player knows their legal moves; and properties about strategies such as the existence of Nash equilibrium.

Strategy Logic (SL) (Mogavero et al. 2014) provides a rich logical framework for reasoning about multi-agent systems and the strategic behavior of agents. Its extension, Epistemic Strategy Logic (SLK) (Berthon et al. 2021), enables reasoning about games with partial observability. Taking advantage of this formalism to automatically verify properties of general games requires a formal translation from GDL-II games to *interpreted systems*, which are the semantical structures used in both SLK and the model checking toolkit MCMAS (Lomuscio, Qu, and Raimondi 2017; Čermák et al. 2018). In this paper, we propose such a translation and prove its correctness. We also show how to express and verify properties regarding the strategic behavior of agents in GDL-II games using MCMAS. Our main contribution is thus a concrete link between the general Game Description Language and Strategy Logic that bridges the research in GGP with the recent developments in formal methods for strategic reasoning. Specifically, for the first time, we show that GDL-II descriptions can be translated into interpreted systems, which can be used as models for logic for strategic reasoning under imperfect information and model-checked using MCMAS.

Related work. For the verification of temporal invariance properties in GDL and of epistemic properties in GDL-II, Answer Set Programming has been used in the past (Haufe, Schiffel, and Thielscher 2012; Haufe and Thielscher 2012). In addition, the model checker MCK (Gammie and Van Der Meyden 2004) was considered to verify GDL-II properties (Huang, Ruan, and Thielscher 2013). However, none of these works handle the strategic dimension, which also holds for the translation of epistemic GDL-III into Dynamic Epistemic Logic (DEL) that focuses on automated epistemic reasoning and planning (Engesser et al. 2021). Ruan, Van Der Hoek, and Wooldridge (2009) provide a framework in which GDL can be understood as a specification language for models of Alternating-time Temporal Logic (ATL). But their approach is restricted to the original GDL for games with complete state information.

Outline. We start by presenting basic definitions in Section 2. Then, we present our translation from GDL-II games to interpreted systems in Section 3 and prove its correctness in Section 4. In Section 5 we demonstrate its use to verify properties expressed in SLK. We conclude in Section 6.

2 Preliminaries

We recall basic definitions of the Game Description Language with Imperfect Information and Epistemic Strategy Logic. We consider a set of atomic propositions AP, a set of agents R, a set of actions Ac_r for each $r \in R$, and a set of strategy variables S. We let $Ag = R \cup \texttt{Env}$, where $\texttt{Env}$ is a distinguishing element also called the *environment*.

2.1 GDL-II

The Game Description Language (GDL) is a general language for expressing game rules using a syntax similar to that of logic programs. The extension GDL-II enables the description of incomplete information games and contains the following predefined keywords:

$role(R)$	R is a player
$init(F)$	F holds in the initial position
$true(F)$	F holds in the current position
$legal(R, M)$	R can do action M in the current position
$does(R, M)$	player R does action M
$next(F)$	F holds in the next position
$terminal$	the current position is terminal
$goal(R, N)$	R gets N points in the current position
$sees(R, P)$	R perceives P in the next position
$random$	the random player

Plain GDL considers only the first eight keywords (Love, Genesereth, and Hinrichs 2006). The final two were added in GDL-II to support the description of games of incomplete information. The keyword $sees(R, P)$ is used to specify the conditions under which a player R gets information P, while $random$ denotes a special player that is assumed to always perform random actions (Schiffel and Thielscher 2014). GDL-II is a universal language that can express all finite extensive-form games with imperfect information (Thielscher 2011).

As an example, consider the simple 2-player game in extensive form depicted in Fig. 2. Fig. 1 gives a GDL-II description of this game. The players take alternating moves. When it is their turn, they can both choose between actions l and r , otherwise they can only play $noop$. After each round, each player perceives the most recent action that she played.

Valid game descriptions must satisfy certain syntactic restrictions; for details, we refer to Love, Genesereth, and Hinrichs (2006). A GDL-II description is *grounded* if it is variable-free. The *atom dependency graph* of a grounded GDL-II description G can be obtained by creating a node for each atom in G and a directed edge between any two atoms b to h such that there exists a rule in G with h appears in the head and b in the body (Genesereth and Thielscher 2014). A game description is *acyclic* if its atom dependency graph contains no directed cycle. For simplicity, in this paper, we only consider GDL-II descriptions that are both **grounded** and **acyclic**. A method of obtaining a grounded version of any valid GDL-II description has been described by Huang, Ruan, and Thielscher (2013), and while not compulsory, all game descriptions used in past GGP competitions are acyclic.

Let At be the set of ground atoms in G. We define At_{other} as the set of all atoms in At that are **not** of form $does(r, a)$, $init(f)$, $true(f)$, $next(f)$, or $sees(r, a)$. By $Base$ we mean the set of all *fluents* f such that $init(f) \in At$, $true(f) \in At$, or $next(f) \in At$. Given a set of fluents $Q \subseteq Base$, we introduce the notation $Q^{true} \overset{\text{def}}{=} \{true(f). \mid f \in Q\}$. Finally, for a *joint action* $A = \langle A(r_1), \ldots, A(r_n)\rangle$, i.e. an action $A(r_i)$ for each player r_i, let

$$A^{does} \overset{\text{def}}{=} \{does(r_1, A(r_1)).,\ \ldots,\ does(r_n, A(r_n)).\}$$

Definition 1 (GDL-II semantics (Ruan and Thielscher 2011))**.** *Let G be a GDL-II specification over a set of ground terms σ. The semantics of G is the state transition system $\langle R, Q_1, T, l, u, I, g\rangle$ defined as follows:*

- $R = \{r \in \sigma \mid G \models role(r)\}$ *(players);*
- $Q_1 = \{f \in Base \mid G \models init(f)\}$ *(initial position);*
- $T = \{Q \subseteq Base | G \cup Q^{true} \models terminal\}$ *(terminal positions);*
- $l = \{(r, a, Q) | G \cup Q^{true} \models legal(r, a)\}$*, where* $r \in R$*,* $a \in \sigma$*, and* $Q \subseteq Base$ *(legality relation);*
- $u(A, Q) = \{f \in Base \mid G \cup Q^{true} \cup A^{does} \models next(f)\}$*, for all* $A : R \to \sigma$ *and all* $Q \subseteq Base$ *(update function);*
- $I = \{(r, A, Q, p) \mid G \cup A^{does} \cup Q^{true} \models sees(r, p)\}$*, for all* $r \in R \backslash \{random\}$*,* $A \in \sigma^{|R|-1}$*,* $Q \subseteq Base$*, and* $p \in \sigma$ *(information relation, determining players' percepts);*
- $g = \{(r, v, Q) \mid G \cup Q^{true} \models goal(r, v)\}$*, where* $r \in R \setminus \{random\}$*,* $v \in \mathbb{N}_0$*, and* $Q \subseteq Base$ *(goal relation).*

Different runs of a game can be described by *developments*, which are sequences of states and legal joint moves by each player up to a certain round:

$$\langle Q_1, A_1, Q_2, \ldots, Q_{k-1}, A_{k-1}, Q_k\rangle$$

Informally, under perfect recall, a role $r \in R \backslash \{random\}$ cannot distinguish two developments if, and only if, these are of the same length and the player takes the same moves and gets the same percepts in every round (Ruan and Thielscher 2011). Formally, if $d = \langle Q_1, A_1, Q_2, \ldots, Q_n\rangle$ and $d' = \langle Q_1, A'_1, Q'_2, \ldots, Q_m\rangle$ are two developments, then player r cannot distinguish d and d' (written as $d \sim_r d'$) iff

- $n = m$.
- $\{p|(r, A_i, Q_i, p) \in I\} = \{p|(r, A'_i, Q'_i, p) \in I\}$, for all $1 \leq i \leq m$,
- $A_i(r) = A'_i(r)$, for all $1 \leq i < m$.

Based on the semantics, the epistemic game model for a GDL-II description is defined as follows.

Definition 2 (GDL-II Epistemic Game Model (Ruan and Thielscher 2011))**.** *Consider a GDL-II description G with semantics $\langle R, Q_1, T, l, u, I, g\rangle$. The epistemic game model of G is a structure $\langle W, R, (\sim_i)_{i \in R \backslash \{random\}}, V\rangle$, where*

```
role(x).   role(o).  init(control(x)).  init(s1).
legal(P,l) <= true(control(P)).   legal(x,noop) <= true(control(o)).
legal(P,r) <= true(control(P)).   legal(o,noop) <= true(control(x)).
sees(Player,Move) <= does(Player,Move).
next(control(x)) <= true(control(o)).   next(s2) <= true(s1), does(x,l).
next(control(o)) <= true(control(x)).   ... next(s11) <= true(s9), does(o,r).
goal(x,7) <= true(s3).  ...  goal(x,6) <= true(s11).   terminal <= true(s3).
goal(o,9) <= true(s3).  ...  goal(o,6) <= true(s11).   ... terminal <= true(s11).
```

Figure 1: A simple GDL-II game description (given in infix notation).

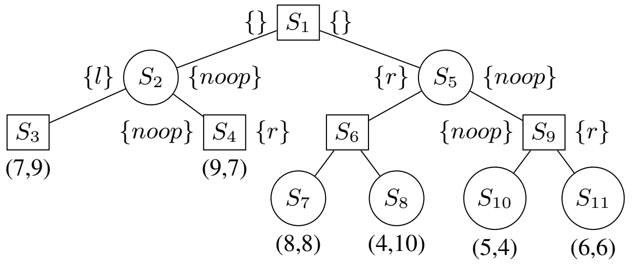

Figure 2: Game tree of the simple GDL-II game in Fig. 1 with the goal value for players x and o given at each terminal state and the *percepts* of the two players (here: their own preceding move) shown to the left and right of some states.

- *W is the set of developments of G;*
- *$\sim_i \in W \times W$ is the indistinguishability relation for player $i \in R \backslash \{random\}$;*
- *$V : W \to 2^{\sigma}$ is an interpretation function, associating each development d with the set of ground terms in d that are true in the last state of d.*

2.2 Epistemic Strategy Logic

Epistemic Strategy Logic (SLK) is an extension of SL (Mogavero et al. 2014) to handle imperfect information.

Definition 3. *The syntax of* SLK *is defined as follows:*

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\langle x \rangle\rangle\varphi \mid (r, x)\varphi \mid \mathbf{K}_r\varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

where $p \in \mathrm{AP}$, $x \in \mathrm{S}$, and $r \in R$.

The formula $\langle\langle x \rangle\rangle\varphi$ means that there exists a strategy x such that φ holds; $(r, x)\varphi$ means that, when strategy x is assigned to r, φ holds; $\mathbf{K}_r\varphi$ means that agent r knows that φ holds; $\mathbf{X}$ and $\mathbf{U}$ are the usual temporal operators "next" and "until".

We consider the SLK semantics by Čermák et al. (2018), which is based on interpreted systems.

Definition 4. *An* interpreted system *(IS) is a tuple $\mathcal{I} = \big\langle (St_r, Ac_r, \mathrm{P}_r, tr_r)_{r\in Ag}, \mathrm{I}, \mathrm{h}\big\rangle$ where*

- *St_r is a finite, non-empty set of local agent states. For each agent $r \in R$, we assume that $St_r = St_r^{\mathsf{p}} \times St_{Env}^{\mathsf{vis}_r}$, where St_r^{p} is the set of internal states of agent r and $St_{Env}^{\mathsf{vis}_r}$ is an image of the set of environment states visible to agent r via her visibility function $\mathsf{vis}_r : St_{Env} \to St_{Env}^{\mathsf{vis}_r}$.*
  We denote by $St = (\prod_{r\in R} St_r^{\mathsf{p}}) \times St_{Env}$ the set of global states. Given a global state $\mathrm{s} \in St$, we denote by $s_{Env}(\mathrm{s})$ and s_r^{p} the environment state and the internal state of agent r in s, respectively. We also let $s_r(\mathrm{s}) = \big(s_r^{\mathsf{p}}, \mathsf{vis}_r(s_{Env}(\mathrm{s}))\big)$ denote the local state of agent r in s;
- *Ac_r is a finite non-empty set of actions available to agent r. The set of decisions for all agents is $Dc = \prod_{r\in Ag} Ac_r$. Given $\boldsymbol{c} \in Dc$, we write c_r for r's action in $\boldsymbol{c}$;*
- *$\mathrm{P}_r : St_r \to 2^{Ac_r} \setminus \{\emptyset\}$ is the protocol of agent r, i.e., a function specifying the available actions for r at a given state. The global protocol $\mathrm{P} : St \to 2^{Dc}$ is defined as $\mathrm{P}(\mathrm{s}) = \{\boldsymbol{c} \in Dc \mid \forall r \in Ag.c_r \in \mathrm{P}_r(s_r(\mathrm{s}))\}$ for all $\mathrm{s} \in St$;*
- *$tr_r : St_r \times Dc \to St_r^{\mathsf{p}}$ is an evolution function, mapping every local state of an agent and decision to a new internal state of the same agent. The global evolution is a function $tr : St \times Dc \to St$, defined as follows: $tr(\mathrm{s}, c) = \mathrm{s}'$ iff, for all agents $r \in Ag$, it holds that $tr(s_r(\mathrm{s}), c) = s_r(\mathrm{s}')$;*
- *$\mathrm{I} \subseteq St$ is a finite non-empty set of initial global states;*
- *$\mathrm{h} : \mathrm{AP} \to 2^{St}$ is a* valuation function, *mapping each proposition to the set of global states in which it is true.*

A (memoryless) *strategy* is a function $\sigma : St \to c$ mapping each global state to an action. Given an agent r, we say that σ is r-coherent if, for each $\mathrm{s} \in St$, $\sigma(\mathrm{s}) \in \mathrm{P}_r(s_r(\mathrm{s}))$.

A strategy σ is uniform w.r.t. the agent r if, for every pair of global states $\mathrm{s}_1, \mathrm{s}_2 \in St$ with $s_r(\mathrm{s}_1) = s_r(\mathrm{s}_2)$ (i.e., s_1 and s_2 are indistinguishable by r) it holds that $\sigma(s_1) = \sigma(s_2)$. Given a group of agents $A \subseteq R$, by Str_A we denote the set of all r-coherent uniform strategies, for each $r \in A$. A strategy profile $\boldsymbol{\sigma} = (\sigma_r)_{r\in R}$ is a tuple of strategies, where σ_r is r-coherent, for $r \in Ag$.

An *assignment* $\mathcal{A} : Ag \cup \mathrm{S} \to Str$ is a function from players and strategy variables to strategies such that $\mathcal{A}(r)$ is r-coherent for each agent r. For an assignment $\mathcal{A}$, an agent r and a strategy σ for r, $\mathcal{A}[a \mapsto \sigma]$ is the assignment that maps a to σ and is otherwise equal to $\mathcal{A}$, and $\mathcal{A}[x \mapsto \sigma]$ is defined similarly, where x is a strategy variable.

Given a strategy variable x and a SLK formula φ, we let shr_x^{φ} be the set of agents associated to x in φ, i.e., $\mathrm{shr}_x^{\varphi} \overset{\text{def}}{=} \{r \in R : (r, x)\psi \text{ is a subformula of } \varphi\}$.

Definition 5. *The satisfaction relation $\models$ for a state $\mathrm{s} \in St$, an IS $\mathcal{I}$, an assignment $\mathcal{A}$, and* SLK *formula φ is defined inductively. We only give the inductive cases for strategy quantification and epistemic operators, and refer to (Čermák et al. 2018) for the other standard cases.*

- $\mathcal{I}, \mathcal{A}, \mathrm{s} \models \langle\langle x \rangle\rangle\, \varphi$ *iff* $\exists\sigma \in \mathrm{Str}_{\mathrm{shr}_x^{\varphi}}$ *s.t.* $\mathcal{I}, \mathcal{A}[x \mapsto \sigma], \mathrm{s} \models \varphi$
- $\mathcal{I}, \mathcal{A}, \mathrm{s} \models (r, x)\varphi$ *iff* $\mathcal{I}, \mathcal{A}[r \mapsto \mathcal{A}(x)], \mathrm{s} \models \varphi$
- $\mathcal{I}, \mathcal{A}, \mathrm{s} \models \mathbf{K}_r\varphi$ *iff* $\mathcal{I}, \mathcal{A}, \mathrm{s}' \models \varphi$ *for all* s' *s.t* $s_r(\mathrm{s}') = s_r(\mathrm{s})$

We define and use the standard abbreviations as follows: $\bot \overset{\text{def}}{=} \neg\top$, $\varphi \wedge \varphi' \overset{\text{def}}{=} \neg(\neg\varphi \vee \neg\varphi')$, $\varphi \rightarrow \varphi' \overset{\text{def}}{=} \neg\varphi \vee \varphi'$, $\mathbf{F}\psi \overset{\text{def}}{=} \top\mathbf{U}\psi$, $\mathbf{G}\psi \overset{\text{def}}{=} \neg\mathbf{F}\neg\psi$ and $[[x]]\,\varphi \overset{\text{def}}{=} \neg\langle\langle x\rangle\rangle\,\neg\varphi$.

The *model-checking problem* for SLK consists in deciding, given a formula φ, an IS $\mathcal{I}$, and a state s in $\mathcal{I}$, whether $\mathcal{I}, \mathcal{A}, s \models \varphi$ for some assignment $\mathcal{A}$. This problem is undecidable for memoryfull strategies (Berthon et al. 2021; Dima and Tiplea 2011) and PSPACE-complete (Čermák et al. 2018; Maubert et al. 2021) for memoryless strategies.

MCMAS (Lomuscio, Qu, and Raimondi 2017) is a model-checking toolkit for the verification of multiagent systems (MAS) described utilizing Interpreted Systems Programming Language (ISPL) programs.[1] A basic ISPL program is described by: (i) *local states*, which are internal states of the agents, declared using *local variables*, and not observable by the other agents. The only exception is on some local variables of the environment, which can be specified to be visible to other agents using the keyword **Lobsvars**; (ii) a *local protocol*, which represents how the agents can interact with each other and the environment; (iii) a *initial state*, which represents the initial assignment of local variables; and (iv) a *local evolution function*, which describes how local states change value over time. We consider the MCMAS extension MCMAS-SLK, which handles SLK with memoryless strategies (Čermák et al. 2018).

3 Translation

Given a GDL-II description G, we generate an IS $\mathcal{I}$ described in ISPL. $\mathcal{I}$ should be equivalent to G in such a way that every ground atom in G is mapped to some variables in $\mathcal{I}$, and each state of G is mapped to some global state of $\mathcal{I}$. As a result, if a property holds in the game model of G under GDL-II semantics, it should also hold in $\mathcal{I}$ under SLK semantics. We first introduce some auxiliary notations.

- An action history h is a (possibly empty) finite sequence $A_1; A_2; \ldots; A_m$ where A_i is a joint action, for each $1 \leq i \leq m$, and $m \in \mathbb{N}_0$.
- For an n-player GDL-II game, $Q^{\emptyset}_{h}$ denotes the unique game state reached after the players performed the history of *legal* joint actions h, and Q^{A}_{h} denotes that the legal joint action A was played after the history h.
- For any state Q^{A}_{h} in G (A can be $\emptyset$), we use $Q^{A}_{h} = \{f_1, \ldots, f_k\}$ to denote the set of ground terms that hold at that state, and $(Q^{A}_{h})^{true} \overset{\text{def}}{=} \{true(f_1).,\ldots, true(f_k).\}$
- For any state $Q^{\emptyset}_{h}$ in G, $(Q^{\emptyset}_{h})^{sees}$ represents the set of percepts of all players in the *current* state. $(Q^{\emptyset}_{h})^{sees}(r)$ represents the percepts of player r at the state $Q^{\emptyset}_{h}$. More formally, $(Q^{\emptyset}_{\emptyset})^{sees}(r) = \{\}$, and $(Q^{\emptyset}_{h;A})^{sees}(r) = \{sees(r, f) \mid G \cup (Q^{A}_{h})^{true} \cup A^{does} \models sees(r, f)\}$.
- We use $h[-i]$ to represent the i-th last element of a list, and $h[: -i]$ to represent the sublist of h from the first element up to the i-th last element of h (the list obtained by removing the final i-1 elements). We denote by $h[-i](r)$ the action of player r in the i-th last element of h.

When the history is empty, the (initial) game state is $Q^{\emptyset}_{\emptyset}$, and based on the last item, we have $h = h[: -1]$. When a joint action A is played at state $Q^{\emptyset}_{h} \in G$, we write Q^{A}_{h} for an **intermediate** state *before* updating the ground terms in $Q^{\emptyset}_{h}$, and then the game evolves to $Q^{\emptyset}_{h;A}$ where these terms have been updated. Here $h; A$ means appending A to the end of h. Note that only $Q^{\emptyset}_{h}$ correspond to "real" game states in a game G; the intermediate states are introduced for the purpose of the IS.

Since the SLK model checker considers imperfect recall semantics while the standard semantics for GDL-II assumes that each player has perfect recall, we introduce the concept of *recall depth* in GDL-II.

Definition 6 (Recall depth)**.** *Suppose G is an n-player GDL-II description. A player r has recall depth D iff r cannot distinguish any two states $Q^{\emptyset}_{h_1}$ and $Q^{\emptyset}_{h_2}$ in the game that satisfy both the following properties.*

- $\forall 1 \leq i \leq D$, $h_1[-i](r) = h_2[-i](r)$
- $\forall 1 \leq i \leq D+1$, $(Q^{\emptyset}_{h_1[:-i]})^{sees}(r) = (Q^{\emptyset}_{h_2[:-i]})^{sees}(r)$

If $i \geq length(h)$, $h[-i](r) = \emptyset$, and $(Q^{\emptyset}_{h_1[:-i]})^{sees}(r) = \emptyset$. We say a game is playable at recall depth D iff all players have a recall depth D, and for any two indistinguishable states $Q^{\emptyset}_{h_1}$ and $Q^{\emptyset}_{h_2}$, for any player r, if $G \cup (Q^{\emptyset}_{h_1})^{true} \models legal(r, a)$ then $G \cup (Q^{\emptyset}_{h_2})^{true} \models legal(r, a)$.

The above definition can be understood as follows. If a player has a recall depth of D, then at each state, they only memorize their actions in the previous D steps, and their percepts in the last $D + 1$ steps. If the game is playable at recall depth D, each player can still derive their legal actions at all game states under this recall depth. We call players *memoryless* if they have a recall depth of 0. As an example, the game described in Fig. 1 has been designed so as to be playable at any recall depth.

We can now describe the conversion of a GDL-II description G to an IS $\mathcal{I}$ in the MCMAS framework (Lomuscio, Qu, and Raimondi 2017). In the following, we assume the game G to be playable at recall depth 0 and that all players are memoryless; we discuss the generalization of our translation in Section 4.2.

3.1 Local variables

We define all variables in $\mathcal{I}$ as local variables associated to the environment `Env` while letting the players observe some local variables of `Env` with the keyword **Lobsvars**. Each atom in G is mapped to several local variables of `Env` in the translated IS $\mathcal{I}$, which are defined as follows.

Definition 7 (Local variables of the Environment)**.** *Suppose G is a grounded GDL-II description with atoms At, then the set of local variables Var of the environment `Env` of $\mathcal{I}$ is the union of the following sets of variables.*

- $Var_{next} = \{next(f) \mid f \in Base\}$.
- $Var_{true} = \{true(f) \mid f \in Base\}$.

[1]ISPL syntax is available at https://sail.doc.ic.ac.uk/software/mcmas/manual.pdf

- $Var_{sees} = \{sees(r,a) \mid sees(r,a) \in At\}$.
- $Var_{seen} = \{seen(r,a,0) \mid sees(r,a) \in At\}$.
- $Var_{does} = \{does(r,a) \mid does(r,a) \in At\}$.
- $Var_{other} = At_{other}$.
- *act: a boolean variable.*
- *cnt: an integer between 0 and $\delta + 1$.*

Here, δ is the length of the longest chain of the atom dependency graph of G. For the game described in Fig. 1, $\delta = 1$. The reason for introducing cnt, act, and δ will be discussed in Section 3.3. The above definition creates an implicit association between the atoms in G and variables in $\mathcal{I}$. The association can be explicitly described as follows.

Definition 8 (Association between atoms in G and variables in $\mathcal{I}$)**.** *Suppose G is a grounded GDL-II description with atoms At, and $\mathcal{I}$ is the IS with local variables Var. For each $p \in Var \backslash \{act, cnt\}$, the corresponding atom in G is denoted as $g(p)$; we write $g(p) = q$ to denote that p is associated with the atom q in G:*

- *If $p \in Var_{true} \cup Var_{other} \cup Var_{does}$, then $g(p) = p$.*
- *If $p \in Var_{next}$ and $p \in At$, then $g(p) = p$.*
- *If $p \in Var_{next}$ and $p \notin At$, then $g(p)$ is undefined.*
- *If p is of form $sees(r, a)$, then $g(p) = sees(r, a)$.*
- *If p is of form $seen(r, a, 0)$, then $g(p) = sees_{curr}(r, a)$.*

where $sees_{curr}$ is the observation token (i.e., percept) of an agent in the current state. Conversely, we define $i(q)$ as a mapping from GDL-II atoms q to its corresponding variable $p \in Var$ such that $i(q) = p$ iff $g(p) = q$.

Note that $sees_{curr}$ do not correspond to any "real" atoms in G at a given state, they are used to describe the precepts of the players (i.e., $(Q^{\emptyset}_{h})^{sees}$). Some $next(f) \in Var$ do not have an associated $next(f) \in At$ (e.g., $next(s1)$ in the game introduced in Section 2). These "redundant" atoms are added to Var to reduce some casework in Section 3.3.

We now introduce the completion rule. This was originally introduced in GDL by Ruan, Van Der Hoek, and Wooldridge (2009), and we adapt it to GDL-II.

Definition 9 (Completion rule)**.** *Suppose G is a grounded GDL-II description, and $\mathcal{I}$ is the IS that models G, with the local variables Var. For every $p \in Var_{next} \cup Var_{other} \cup Var_{sees}$, the completion rule of p is:*

$$cp(p) \equiv \begin{cases} \bot, & \text{if } g(p) \text{ is undefined} \\ \bigvee_{r \in G, hd(r) = g(p)} \bigwedge bd(r), & \text{otherwise.} \end{cases}$$

Here, $hd(r)$ means the atom in the head of a rule $r \in G$ with body $bd(r) = \{l_1, \ldots, l_n\}$ and $\bigwedge bd(r) = i(l_1) \wedge \ldots \wedge i(l_n)$. For an empty body rule, $bd(r) = \top$.

E.g. for the GDL-II description in Fig. 1, the completion $cp(terminal)$ of $terminal$ is given by: $true(s3) \vee true(s4) \vee true(s7) \vee true(s8) \vee true(s10) \vee true(s11)$.

Since we deal with imperfect-information games, we need to define the local variables of `Env` that are observable by each player in the game.

Definition 10 (Local observations of players)**.** *Suppose G is a grounded GDL-II description, and $\mathcal{I}$ is the IS that models G, then the set of local observation variables Obs_r of any player $r \neq random$ in $\mathcal{I}$ is the union of the following sets.*

1. *$\{cnt, act\}$*
2. *$Legal_r = \{a \mid a \in Var_{other} \wedge \exists ac.\ a = legal(r, ac)\}$*
3. *$Seen_r = \{a \mid a \in Var_{seen} \wedge \exists f.\ a = seen(r, f, 0)\}$*

If r is the $random$ player, then $Obs_r = Var$.

3.2 Initial State and Protocol

The initial state and protocol of the IS are defined as follows.

Definition 11 (Initial state)**.** *Suppose G is a grounded GDL-II description with ground atoms At and initial state $Q^{\emptyset}_{\emptyset}$. For the initial state of our IS $\mathcal{I}$, the variables are initialized as follows.*

- *$cnt = 0$ and $act = \top$.*
- *If $p \in Var_{seen} \cup Var_{does}$, then $p = \bot$.*
- *If $p \in Var_{next}$ and $g(p)$ is undefined, then $p = \bot$.*
- *If $p \in Var_{next} \cup Var_{true} \cup Var_{other} \cup Var_{sees}$ and $g(p)$ is defined, then $p = \top$ if and only if $G \cup (Q^{\emptyset}_{\emptyset})^{true} \models g(p)$.*

Definition 12 (Legal actions)**.** *Suppose G is a grounded GDL-II description, and $\mathcal{I}$ is the IS that models G. Suppose "none" is a dummy action not in the move domain of any player in G, then for any player $r \in R$ and any state $S \in I$, it is legal for player r to play the action a from their move domain if $legal(r, a) = \top \wedge act = \top \wedge cnt = 0$ at state S. If no action in the move domain is legal at a state S, it is legal for player r to play "none." "none" is the only legal action for `Env` in all states of $\mathcal{I}$.*

3.3 Evolution function

The global evolution function of $\mathcal{I}$ describes how the values of the local variables change after a joint action of all the players. For each variable $p \in Var$, we use p to denote its value in the current state and p' its value in the next state. The main challenge of defining the evolution function of $\mathcal{I}$ is that the evolution of atoms in GDL-II is based on the standard stable model semantics while in MCMAS the values of variables in the next state can only depend on the values of the variables in the current state.

Example 1. *Let P be a logic program with just two rules:*

- $a \text{ :- } true(f)$. $\quad terminal \text{ :- } a$.

Initially, $a = true(f) = terminal = \bot$. By forcing $true(f) = \top$, the only stable model of the program is $a = true(f) = terminal = \top$ in the next state. It is tempting to model the evolution function of an IS $\mathcal{I}$ describing P with the completion rule as in the existing GDL to ATL translation (Ruan, Van Der Hoek, and Wooldridge 2009).

- $a' = true(f)$. $\quad terminal' = a$.

The model checker Mocha (Alur et al. 1998) used in the GDL to ATL translation can execute the first function before the second function such that after one evolution step, $a = true(f) = terminal = \top$. However, in every *evolution step of MCMAS, all evolution functions are executed* ***simultaneously****. Hence, after one evolution step, $terminal = \bot$ and $true(f) = a = \top$, which does not match the stable model.*

The issue here is that in the stable model, the value of $terminal$ in the next state depends on the value of a in the next state while in MCMAS, the value of variables in the next state should only depend on the value of variables in the current state. We must address the simultaneous update issue to enable verification of GDL games with MCMAS. To do so, first observe that P in the example above is acyclic and that the length of the longest chain of the dependency graph of P is 2. If we apply the completion rule to all $v \in Var$ simultaneously two or more times, the value of all variables v will converge to true. We therefore introduce a counter-based method to model the evolution function of $\mathcal{I}$. The structure of this IS $\mathcal{I}$ is illustrated in Fig. 3. We use $S_h^{c,a}$ to represent a state in $\mathcal{I}$ with $cnt = c$, $act = a$, and the list of joint actions performed before this state being h. $S_h^{c,a}$ corresponds to a "real" game state of G if and only if $c = 0$ and $a = \top$. For each state not of the form $S_h^{0,\top}$, the only legal joint action in the given state is $\langle none, \ldots, none\rangle$. Note that only legal joint actions performed by the players at states with $c = 0$, and $a = \top$ are appended to the list h.

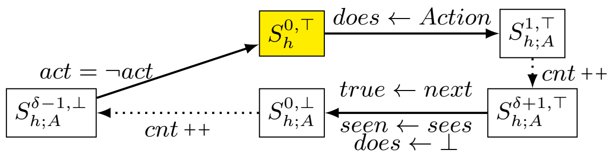

Figure 3: Evolution of local variables in the IS

The intended behavior of the constructed $\mathcal{I}$ is as follows. For each state $S_h^{0,\top}$ (initial state is $S_\emptyset^{0,\top}$, where $\emptyset$ denotes empty history), if $terminal \neq \top$, a legal joint action $A = \langle A(r_1), \ldots, A(r_n)\rangle$ is picked and we transit to $S_{h;A}^{1,\top}$ with all the $does(r_i, A(r_i))'$ set to true. If $terminal = \top$, we transit to $S_h^{0,\top}$ no matter what A we pick. Note that we leave out the action of `Env` in A since it is "$none$" in all states of $\mathcal{I}$. During the evolution from $S_{h;A}^{\delta+1,\top}$ to $S_{h;A}^{0,\bot}$ we set all $does(r, a)' = \bot$, $seen(r, f, 0)' = sees(r, f)$, and $true(f)' = next(f)$. To ensure that the variables in $\mathcal{I}$ converge, at least δ many intermediate evolutions are introduced between $S_{h;A}^{1,\top}$ and $S_{h;A}^{\delta+1,\top}$, and $S_{h;A}^{0,\bot}$ and $S_{h;A}^{0,\top}$.

Formally, we define the global evolution function of $\mathcal{I}$ as the union of the following 6 types of evolution functions.

E1. For each $true(f) \in Var_{true}$,

$$true(f)' = \begin{cases} next(f), & cnt = \delta + 1 \wedge act = \top \\ true(f), & otherwise \end{cases}$$

E2. For each $does(r, a) \in Var_{does}$,

$$does(r,a)' = \begin{cases} \top, & cnt = 0 \wedge act \wedge \neg terminal \wedge R.Action = a \\ \bot, & cnt = \delta + 1 \wedge act = \top \\ does(r,a), & otherwise \end{cases}$$

E3. For each $seen(r, a, 0) \in Var_{seen}$,

$$seen(r, f, 0)' = \begin{cases} sees(r, f), & cnt = \delta + 1 \wedge act = \top \\ seen(r, f, 0), & otherwise \end{cases}$$

E4. For the variable cnt,

$$cnt' = \begin{cases} 0, & terminal = \top \wedge cnt = 0 \\ 0, & act = \top \wedge cnt = \delta + 1 \\ 0, & act = \bot \wedge cnt = \delta - 1 \\ cnt + 1, & otherwise \end{cases}$$

E5. For the variable act,

$$act' = \begin{cases} \neg act, & cnt = \delta + 1 \wedge act = \top \\ \neg act, & cnt = \delta - 1 \wedge act = \bot \\ act, & otherwise \end{cases}$$

E6. For each variable $p \in Var_{next} \cup Var_{sees} \cup Var_{other}$,

$$p' = cp(p)$$

4 Correctness and Generalization

4.1 Correctness of the Translation

We prove the correctness of the mapping from a GDL-II description G to an interpreted system $\mathcal{I}$, again under the assumption that G is playable at a recall depth of 0 and that all players are memoryless. We use $q(Q_h^A)$ to denote the truth value of atom q in the state Q_h^A of G while $p(S_h^{c,a})$ denotes the value of variable p in the state $S_h^{c,a}$ of $\mathcal{I}$. We say atom $q(Q_h^A) = \top$ in G if and only if $G \cup (Q_h^\emptyset)^{true} \cup A^{does} \models q$ or $q \in (Q_h^\emptyset)^{sees}$. Note that if $A = \emptyset$ then $\emptyset^{does} = \{\}$. We can now define the concept of *matching* states between G and $\mathcal{I}$.

Definition 13. *Suppose G is a GDL-II description of a game and $\mathcal{I}$ is the interpreted system with variables Var that models G. A state $S_{h_1}^{c,a} \in \mathcal{I}$ matches a state $Q_{h_2}^A \in G$, denoted as $S_{h_1}^{c,a} \simeq Q_{h_2}^A$, if and only if the following conditions hold.*

1. $h_1 = h_2$.

2. For all $p \in Var \setminus Var_{seen}$, $p(S_{h_1}^{c,a}) = g(p)(Q_{h_2}^A)$.

3. If $A = \emptyset$, for all $p \in Var_{seen}$, $p(S_{h_1}^{c,a}) = g(p)(Q_{h_2}^A)$.

Here, if $g(p)$ is undefined, $g(p)(Q_h^A) = \bot$ for all Q_h^A.

The correctness of the translation under the memoryless assumption can be proved using induction, as captured by a series of theorems. Theorem 1 will show that the initial state of the interpreted system $S_h^{0,\top}$ matches the initial state of the game $Q_h^\emptyset$. Theorem 2 will show that if a state $S_h^{0,\top} \in \mathcal{I}$ matches a state $Q_h^\emptyset \in G$, then $S_h^{0,\top}$ and $Q_h^\emptyset$ have the same set of legal joint actions. Theorem 3 will show that if we apply the same joint action at $S_h^{0,\top} \in \mathcal{I}$ and $Q_h^\emptyset \in G$ with $S_h^{0,\top} \simeq Q_h^\emptyset$ such that $Q_h^\emptyset$ is not a terminal state, then the "real" successor state of $S_h^{0,\top}$ matches the successor state of $Q_h^\emptyset$. The total number of intermediate states between $S_h^{0,\top}$ and the "real" successor state is $2 \cdot \delta + 1$, which is independent of the legal joint action played at state $S_h^{0,\top}$. Theorem 4 will show that if $S \in \mathcal{I}$ corresponds to a "real" terminal state of the game, any legal joint action performed at this state will not change the values of any local variable at that state.

Lemma 1. *Suppose G is a grounded acyclic GDL-II description, and $\mathcal{I}$ is an interpreted system with local variables Var that models G. Let $V_{fact} \subseteq Var$ be the set of variables v with $g(v)$ appearing as facts in G; $V_t \subseteq Var_{true}$ be the* ***only*** *set of variables in Var_{true} assigned to true; and $V_d \subseteq Var_{does}$ be the* ***only*** *set of variables in Var_{does} assigned to true. By setting all variables in $V_{fact} \cup V_t \cup V_d$ to be true, for an arbitrary initialization of the remaining variables in $Var_{other} \cup Var_{sees} \cup Var_{next}$, if we apply the completion rule to all $p \in Var_{other} \cup Var_{sees} \cup Var_{next}$* ***simultaneously*** *$\delta$ or more times, then $p = \top$ iff $G \cup P_t \cup P_d \models g(p)$. Here, $P_t = \{true(f).|true(f) \in V_t\}$ and $P_d = \{does(r,a).|does(r,a) \in V_d\}$.*

Proof (Sketch). Since G is a valid grounded acyclic game description, only atoms of the form $true(f)$ or $does(r,a)$ or which appear as facts in G can have an in-degree of 0 in the atom dependency graph of G. Hence, we have initialized all variables $p \in Var$ such that $g(p)$ might have an in-degree of 0 in the dependency graph of G. We have initialized all p in such a way that $p = \top$ if and only if $G \cup P_t \cup P_d \models g(p)$. We define $dist(s,t)$ to be the length of the longest path from atom s to t in the atom dependency graph of G, and $d(t) = \max_{s \in At} dist(s,t)$. By induction on the value of $d(t)$, we can prove that if we apply the completion rule i times to all $p \in Var_{other} \cup Var_{sees} \cup Var_{next}$, the values of all variables with $d(g(p)) \leq i$ will converge. Since the longest chain of the dependency graph of G is δ, if we apply the completion rule to all $p \in Var_{other} \cup Var_{sees} \cup Var_{next}$ *simultaneously* δ or more times, the values of the variables will converge to their values in the stable model of the program of $G \cup P_t \cup P_d$ as a consequence of Clark's completion (Clark 1977). □

Theorem 1. *Suppose G is a GDL-II description and $\mathcal{I}$ is the interpreted system that models G, then $S^{0,\top}_{\emptyset} \simeq Q^{\emptyset}_{\emptyset}$.*

Proof. By the GDL-II semantics, $(Q^{\emptyset}_{\emptyset})^{sees} = \{\}$. The claim then follows directly from our Definitions 11 and 13. □

Theorem 2. *Suppose G is a valid GDL-II description and $\mathcal{I}$ is an interpreted system with local variables Var that models G. Suppose further that h is a valid history of joint actions of the players and $S^{0,\top}_h \simeq Q^{\emptyset}_h$ such that $G \cup (Q^{\emptyset}_h)^{true} \not\models terminal$. Then, A is a legal joint action at $Q^{\emptyset}_h$ iff A is a legal joint action at $S^{0,\top}_h$.*

Proof. Since h is a valid history of joint actions of the players and G is a valid GDL-II game, each player has at least one legal action in every non-terminal state. The claim then holds by construction (Definitions 10, 12, and 13). □

Theorem 3. *Suppose G is a valid GDL-II description and $\mathcal{I}$ is an interpreted system with local variables Var that models G. Suppose h is a valid history of joint actions of the players and $S^{0,\top}_h \simeq Q^{\emptyset}_h$ such that $G \cup (Q^{\emptyset}_h)^{true} \not\models terminal$. For any legal joint action A at $Q^{\emptyset}_h$, we have*

1. *$S^{\delta+1,\top}_{h;A} \simeq Q^A_h$.*
2. *$S^{0,\top}_{h;A} \simeq Q^{\emptyset}_{h;A}$.*
3. *The total number of intermediate states between $S^{0,\top}_h$ and $S^{0,\top}_{h;A}$ is $2 \cdot \delta + 1$.*

Proof (Sketch). Since $S^{0,\top}_h \simeq Q^{\emptyset}_h$, we conclude A is also a legal action at $S^{0,\top}_h \in \mathcal{I}$ according to Theorem 2. Therefore, $S^{1,\top}_{h;A} \in \mathcal{I}$, and for any $p \in Var_{does}$, $p(S^{1,\top}_{h;A}) = g(p)(Q^A_h)$ according to evolution function E2. According to evolution function E1, for any variable $p \in Var_{true}$, $p(S^{1,\top}_{h;A}) = p(S^{0,\top}_{h;A})$. According to the evolution functions E1–E2 and E4–E5, all $S^{i,\top}_h$ $(1 < i \leq \delta + 1)$ exist, and for any $p \in Var_{true} \cup Var_{does}$, $p(S^{i,\top}_{h;A}) = p(S^{i+1,\top}_{h;A})$ for all $1 < i \leq \delta$. Since the completion rule (evolution function E6) is applied to all $p \in Var_{sees} \cup Var_{next} \cup Var_{other}$ simultaneously δ times, we conclude that $p(S^{\delta+1,\top}_{h;A}) = g(p)(Q^A_h)$ according to Lemma 1. Using Definition 13, $S^{\delta+1,\top}_{h;A} \simeq Q^A_h$.

Since $S^{\delta+1,\top}_{h;A} \simeq Q^A_h$, for any $p \in Var_{next}$, we have $p(S^{\delta+1,\top}_{h;A}) = g(p)(Q^A_h)$ if $g(p)$ is defined. If $g(p)$ is undefined, $p(S^{\delta+1,\top}_{h;A}) = \perp$. Due to the semantics of GDL-II, if $G \cup (Q^A_h)^{true} \cup A^{does} \models next(f)$ then $G \cup (Q^{\emptyset}_{h;A})^{true} \models true(f)$. Similarly, if $G \cup (Q^A_h)^{true} \cup A^{does} \models sees(r,f)$ then $sees_{curr}(r,f) \in (Q^{\emptyset}_{h;A})^{sees}$. Considering the evolution functions E2–E3, at state $S^{0,\perp}_{h;A}$, for all $p \in Var_{does}$ we have $p(S^{0,\perp}_{h;A}) = \perp$; for all $seen(r,f,0) \in Var_{seen}$ we have $seen(r,f,0)(S^{0,\perp}_{h;A}) = \top$ iff $sees(r,f)(S^{\delta+1,\top}_{h;A}) = \top$; and for all $true(f) \in Var_{true}$ we have $true(f)(S^{0,\perp}_{h;A}) = \top$ iff $next(S^{\delta+1,\top}_{h;A}) = \top$. We conclude that $S^{0,\top}_{h;A} \simeq Q^{\emptyset}_{h;A}$ using Definition 13 and Lemma 1.

Considering evolution function E4, we can easily verify the number of intermediate states between $S^{0,\top}_h$ and $S^{0,\top}_{h;A}$ to be $2 \cdot \delta + 1$ for an arbitrary legal joint action A at $S^{0,\top}_h$. □

Theorem 4. *Suppose G is a valid GDL-II description and $\mathcal{I}$ is an interpreted system with local variables Var that models G. Suppose h is a valid history of joint actions of the players and $terminal(S^{0,\top}_h) = \top$. For any legal joint action A at $S^{0,\top}_h$, let S' to be the successor state of $S^{0,\top}_h$ after A is applied to the system. Then, for any $p \in Var$ we have that $p(S^{0,\top}_h) = p(S')$.*

Proof. Note that at state $S^{0,\top}_h$, $cnt = 0$, and $act = \top$. Since $terminal(S^{0,\top}_h) = \top$, according to the evolution functions E2 and E4, cnt and all $does(r,a) \in Var_{does}$ will preserve their value in S'. The values of $p \in Var_{true} \cup Var_{seen} \cup \{act\}$ will preserve due to evolution function E1, E3, and E5. Finally, the values of $p \in Var_{next} \cup Var_{sees} \cup Var_{other}$ will preserve in S' due to the consequence of Lemma 1. □

The direct consequence of the above theorems is that for any legal joint action history h of the game G, $S^{0,\top}_h \simeq Q^{\emptyset}_h$. Thus, for any **non-random** player, two states $Q^{\emptyset}_{h1}$ and $Q^{\emptyset}_{h2}$

are indistinguishable in G if and only if $S_{h1}^{0,\top}$ and $S_{h2}^{0,\top}$ are indistinguishable in $\mathcal{I}$ under the memoryless assumption (cf. Definition 6 and 10). Furthermore, for all the states in $\mathcal{I}$ that are not of the form $S_h^{0,\top}$, all players can only play the dummy action "*none*" (cf. Definition 12). Hence, $\mathcal{I}$ is equivalent to the epistemic game model of G (cf. Definition 2) in the required sense (despite having $2 \cdot \delta + 1$ intermediate dummy states between $S_h^{0,\top}$ and $S_{h;A}^{0,\top}$ to ensure that the update of the local variables in $\mathcal{I}$ converges).

The behavior of the random player We end the correctness proof with a remark on the behavior of the random player. In GDL-II, one can view *random* as a special role with perfect information about the game state and who always chooses a random legal action with uniform probability at each step of the game. Our translation treats the percepts of the *random* player differently from all the other players, by letting *random* observe **all** local variables of Env (cf. Definition 7). Recall that in SLK, all players in the IS can only play uniform pure strategies. In our translation, the *random* player is equivalent to a player playing uniform pure strategies under perfect information. This is the expected behavior of *random* since modeling mixed strategies is beyond the expressiveness of SLK.

4.2 Relaxing the memoryless assumption

One weakness of the SLK model-checking is the memoryless assumption (Čermák et al. 2018). In GDL-II, many game-specific properties (even legality) are provable only if the agent has a perfect recall. The translation provided in Section 3 assumes that each player has a recall depth of 0. We can easily generalize our translation to allow the players to have a bounded recall of depth D in $\mathcal{I}$. To achieve this, we extend each $seen(r, f, 0) \in Var_{seen}$ to $D + 1$ variables $seen(r, f, 0), \dots, seen(r, f, D)$. We also augment the local variables of Env with a set of variables $Var_{done} = \{done(r, a, 1..D) \mid does(r, a) \in Var_{done}\}$ that records the action each player plays in previous rounds, with $done(r, a, i)$ meaning that player r performed a i steps ago. The evolution of $done$ can be defined as follows. For each $done(r, a, 1) \in Var_{done}$:

$$done(r,a,1)' = \begin{cases} does(r,a) & \text{if } cnt = \delta+1 \wedge act = \top, \\ done(r,a,1) & \text{otherwise.} \end{cases}$$

And for each $2 \leq i \leq D$, the evolution of $done(r, a, i)$ is

$$done(r,a,i)' = \begin{cases} done(r,a,i-1) & \text{if } cnt = \delta+1 \wedge act, \\ done(r,a,i) & \text{otherwise.} \end{cases}$$

The evolution of $seen(r, f, i)$ can be defined similarly. All variables of the form $done(r, a, i)$ or $seen(r, f, i)$ should be inserted into the set of local observation variables of player r. Our translation can now be applied to any arbitrary acyclic GDL-II description with finite grounding, assuming that each player has a bounded recall depth and follows a uniform pure strategy. Since every practical game in GGP competitions terminates within finitely many steps, by setting D as the longest valid playing sequence of a game, we can thus model any GDL-II game that is playable under perfect recall.

5 Experimental Results

The overarching motivation for translating GDL-II games into interpreted systems is that SLK is a powerful logic language that can express many game properties. Once these properties are represented in SLK, it is now possible to use an SLK model checker to automatically verify any of these properties for any given GDL-II game.

In this section, we demonstrate how to express some important properties of general n-player GDL-II games in SLK (Section 5.1) and use the model-checker MCMAS-SLK to verify these properties on two small games (Sections 5.2 and 5.3). The complete GDL-II descriptions of these games and the paper's source code are available online.[2] All experiments were run on a Latitude 5430 laptop.

For a GDL-II description G and an interpreted system $\mathcal{I}$, we let t represent the "real" terminal state of G in $\mathcal{I}$. Here, we use the syntax of *evaluation* in ISPL to define t. The proposition t is evaluated to be true in all global states such that $terminal = \top$ along with $act = \top$ and $cnt = 0$. We use α to represent the strategy quantifier "for all uniform strategies of all the n players". Note that, in MCMAS we also need to bind the environment (Env) to a strategy.

$$t \texttt{ if } terminal = \top \; \&\& \; cnt = 0 \; \&\& \; act = \top$$

$$\alpha = [[x_0]]\,[[x_1]] \dots [[x_n]]\,(\texttt{Env}, x_0)(r_1, x_1) \dots (r_n, x_n)$$

We introduce the notation $\mathcal{I}_G^d$ to represent the interpreted system that models the GDL-II game G with each player in G having a recall depth of d. We write $\mathcal{I} \models \psi$ when $\mathcal{I}, \mathcal{A}, s_0 \models \psi$ for any assignment $\mathcal{A}$, that is, the interpreted system entails ψ for any assignment at the initial state.

5.1 Expressing strategic properties

Simple strategic properties Let us see how to model some universally quantified strategic properties of GDL-II games in SLK. Firstly, any player should know all the true fluents for any GDL-II game G at the initial state, which is captured by the following formula:

$$\psi_{init} = \alpha \bigwedge_{r \in R} \bigwedge_{init(f) \in At} \mathbf{K}_r \; true(f)$$

Similarly, a general property of GDL-II games is that at the initial state, each player knows the legal actions of all the players. This property can be modeled as follows.

$$L(r', m) \equiv G \cup (Q_\emptyset^\emptyset)^{true} \models legal(r', m)$$
$$\psi_l = \alpha \bigwedge_{r \in R} \bigwedge_{L(r',m)=\top} \mathbf{K}_r legal(r', m)$$

In addition, for an ideal GDL-II description, all players should know the termination of a game. This constraint can be modeled by the formula below, which can be read as: For all uniform strategies of all players, it is always the case that when the game terminates, then every player knows this.

$$\psi_{terminal} = \alpha \, \mathbf{G} \bigwedge_{r \in R} t \rightarrow \mathbf{K}_r terminal$$

[2]https://github.com/hharryyf/gdl-ii2slk

A similar property for an ideal GDL-II description is that when the game terminates, every player should know the goal value it gets. We can model this as follows.

$$\psi_g = \alpha\, \mathbf{G} \bigwedge_{goal(r,g)\in At} ((t \wedge goal(r,g)) \rightarrow \mathbf{K}_r goal(r,g))$$

Complex strategic properties Note that most of the game-specific properties we listed above can be expressed not only by SLK but also by CTLK (Huang, Ruan, and Thielscher 2013). For example, $\psi_{terminal}$ can be stated in CTLK as

$$\psi'_{terminal} = \mathbf{AG}\ (t \rightarrow \bigwedge_{r\in R} \mathbf{K}_r terminal)$$

The main reason for using SLK model checkers to verify GDL-II properties is that SLK can express *solution concepts* in games. We discuss two such properties: (1) Nash equilibrium and (2) strong winnability. Note that Nash equilibria cannot be checked by either the GDL-II to CTLK translation (Huang, Ruan, and Thielscher 2013) or the GDL to ATL translation (Ruan, Van Der Hoek, and Wooldridge 2009).

The existence of a uniform pure strategy Nash equilibrium of a GDL-II game G can be expressed as follows.

$$\psi_{ne} = \ \gamma \bigwedge_{r\in R} \bigvee_{goal(r,g)\in At} \psi_{br}(r,g)$$

Where,

$$\gamma = \langle\langle x_0\rangle\rangle\, \langle\langle x_1\rangle\rangle \ldots \langle\langle x_n\rangle\rangle\, (\mathtt{Env}, x_0)(r_1, x_1)\ldots(r_n, x_n)$$

$$\psi_{dev}(r,g) = [[x'_r]]\,(r, x'_r)\, \mathbf{F} \bigvee_{\substack{goal(r,g')\in At,\\ g'\leq g}} t \wedge goal(r,g')$$

$$\psi_{br}(r,g) = (\mathbf{F}(t \wedge goal(g,r))) \wedge \psi_{dev}(r,g)$$

Here, ψ_{ne} is true if and only if there exists a strategy profile such that all players achieve some goal value when the game terminates. And all players are playing the best response to their opponent's strategy (i.e., $\psi_{br}(r,g)$). In other words, when any player deviates from her current strategy, and the opponent's strategy is unchanged, her reward at the terminal state cannot exceed her current reward (i.e., $\psi_{dev}(r,g)$). This is exactly the definition of Nash equilibrium under the assumption that the players follow a uniform pure strategy.

Another solution concept is strong winnability, which gives a lower-bound estimate of a player's utility no matter what the other players do. This property was originally verified in the context of GDL (Ruan, Van Der Hoek, and Wooldridge 2009), and we adapt it to GDL-II. We denote $\psi_{sw}(r_i, U)$ as the proposition that player r_i has a uniform pure strategy to achieve a utility of at least U for all uniform pure strategies of all the other players. The strong winnability property can be expressed as follows.

$$\beta = \langle\langle x_i\rangle\rangle\, [[x_0]]\, [[x_{-i}]]\, (r_i, x_i)(\mathtt{Env}, x_0)(r_{-i}, x_{-i})$$

$$\psi_{sw}(r_i, U) = \beta\, \mathbf{F}\ (t \wedge \bigvee_{\substack{goal(r_i,p)\in At,\\ p\geq U}} goal(r_i,p))$$

Here, we use (r_{-i}, x_{-i}) to abbreviate bounding a strategy x_k to all players except player i.

5.2 Model Checking a Simple Game

We use MCMAS-SLK to verify the above strategic properties for the GDL-II game described in Fig. 1. We assume that both x and o have a recall depth of 3 (i.e., perfect recall). Since the game is small, MCMAS-SLK can verify all the above strategic properties within 1 second. The result is displayed below. We record the result of a strategic property ψ as $\top$ if and only if $\mathcal{I}^3_G \models \psi$.

Strategic property	MCMAS-SLK output
ψ_{init}	$\top$
ψ_l	$\top$
$\psi_{terminal}$	$\bot$
ψ_g	$\bot$
ψ_{ne}	$\top$
$\psi_{sw}(x,7)$	$\top$
$\psi_{sw}(x,8)$	$\bot$

We can see that the game in Fig. 1 satisfies $\mathcal{I}^3_G \models \psi_{init}$ and $\mathcal{I}^3_G \models \psi_l$. Yet $\mathcal{I}^3_G \not\models \psi_{terminal}$ since player o is not able to distinguish terminal state S_4 (resp. S_3) from non-terminal state S_9 (resp. S_6). For similar reasons we have $\mathcal{I}^3_G \not\models \psi_g$. For the complex strategic properties, we obtain $I^3_G \models \psi_{ne}$, and the strategy output by the SLK model checker is player x moves r at state S_1 and l at state S_6 while player o moves l at state S_5. Both players achieve a goal value of 8 at the terminal state. It is easy to check that this strategy profile is indeed a Nash equilibrium. For the strong winnability property, MCMAS verifies that $I^3_G \models \psi_{sw}(x,7)$ but also that $I^3_G \not\models \psi_{sw}(x,8)$. This means that although in the Nash equilibrium of the game player x can achieve a utility of 8, this is not guaranteed if player o doesn't know to cooperate at state S_5 and moves l. However, player x *can* always achieve a utility of at least 7 by moving l at state S_1 regardless of the action of player o at S_2.

5.3 Number Guessing

We conclude this section by reporting on our experimental study with a more complicated GDL-II game called Number Guessing (Schofield and Thielscher 2015). In this game, player o (who can be considered to play the role of Nature) selects a hidden integer n between 1 and N ($N \geq 2$). At each round, player x can guess a number g between 1 and N and is informed about her previous guess g and whether $g < n$, $g > n$, or $g = n$. The game terminates after player x guesses the correct number or after $N+1$ rounds. If the player guesses the correct number at round i, her utility is $\left\lfloor \frac{100\cdot(N-i)}{N-1} \right\rfloor$. The utility of player o is always 0.

For this game, the pure strategy Nash equilibrium corresponds to the strategy profile in which player x directly guesses the number picked by player o. The utility of player x under this strategy profile is 100. Note, however, that such an equilibrium does not make sense because Nature does not follow any strategy here. Hence, we only focus on the strong winnability property in this game. For simplicity, let us assume that player x has a recall depth of 0, which means at each state of the game, she only observes the number she guessed in the previous round and whether that number is larger or smaller than the target. We use $\psi(N,i)$ to represent the property that the player can always guess the number between 1 and N correctly within i rounds:

$$\beta = \langle\langle x_1 \rangle\rangle\, [[x_0]]\, [[x_2]]\, (x, x_1)(\texttt{Env}, x_0)(o, x_2)$$

$$\psi(N,i) = \ \beta\, \mathbf{F}\, t \wedge \bigvee_{goal(x,p) \in At \wedge p \geq \lfloor \frac{(N-i)\cdot 100}{N-1} \rfloor} goal(x,p)$$

We check $\psi(N,i)$ for different combinations of N and i, and report the runtime of MCMAS in the table below.

i	1	2	3	4
$\psi_{sw}(5,i)$	0.81	0.81	**0.82**	**0.85**
$\psi_{sw}(6,i)$	4.68	4.83	**4.85**	**5.06**
$\psi_{sw}(7,i)$	11.66	12.26	**14.42**	**40.73**
$\psi_{sw}(8,i)$	13.67	14.11	144.95	*killed*

We record the runtime in **bold** if $\mathcal{I}^0_G \models \psi(N,i)$. For $5 \leq N \leq 7$, the model-checker can prove that the player has a strategy to guess the number correctly within 3 rounds no matter what number was chosen. One optimal strategy for the number guessing game is obviously binary search, with which the player can guess the number correctly within $\lfloor log_2 N \rfloor + 1$ rounds. Our result matches this optimal bound. However, we were not able to verify the strong winnability property for $N \geq 8$ and $i \geq 4$ because the model-checker ran out of memory. This is due to the theoretical difficulty of SLK model-checking and with the only publicly available system MCMAS-SLK a prototypical implementation that just works for small GDL-II games, which motivates future work on more efficient model checking algorithms.

6 Conclusion

We studied how to verify properties related to the strategic behavior of players in GDL-II games with imperfect information. To do so, we provided a translation from GDL-II games to interpreted systems, which are the semantical structures considered for Epistemic Strategy Logic (SLK) and in the model checking toolkit MCMAS. We proved correctness of the translation and showed how important properties of general games can be expressed in SLK.

While we focused on SLK, the proposed translation can be used with other specification languages supported by MCMAS both for perfect and imperfect information, such as Epistemic ATL (Lomuscio, Qu, and Raimondi 2017) and the one goal fragment of Strategy Logic (Cermák, Lomuscio, and Murano 2015). For future work we intend to explore the verification of GDL games under those formalisms. Also, we intend to extend the proposed translation to handle public actions (Belardinelli et al. 2020), hierarchical information (Berthon et al. 2021), and natural strategies (Jamroga, Malvone, and Murano 2019; Belardinelli et al. 2022), settings in which the model checking problem for Strategy Logic is known to be decidable even for memoryfull strategies (and imperfect information).

Acknowledgements

This work has been partially funded by the EU H2020 Marie Skłodowska-Curie program (grant No. 101105549), MUR PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013-FAIR).

References

Alur, R.; Henzinger, T. A.; Mang, F. Y.; Qadeer, S.; Rajamani, S. K.; and Tasiran, S. 1998. Mocha: Modularity in model checking. In *Computer Aided Verification: 10th International Conference*, 521–525. Springer.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2020. Verification of multi-agent systems with public actions against strategy logic. *Artificial Intelligence (AIJ)* 285:103302.

Belardinelli, F.; Jamroga, W.; Malvone, V.; Mittelmann, M.; Murano, A.; and Perrussel, L. 2022. Reasoning about human-friendly strategies in repeated keyword auctions. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022*, 62–71. IFAAMAS.

Berthon, R.; Maubert, B.; Murano, A.; Rubin, S.; and Vardi, M. Y. 2021. Strategy logic with imperfect information. *ACM Trans. Comput. Log.* 22(1):5:1–5:51.

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.

Cermák, P.; Lomuscio, A.; and Murano, A. 2015. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *Proceedings of AAAI*, 2038–2044.

Clark, K. L. 1977. Negation as failure. In *Logic and Data Bases*. Springer. 293–322.

Clune, J. 2007. Heuristic evaluation functions for general game playing. In *Proceedings of AAAI*, volume 7, 1134–1139.

Dima, C., and Tiplea, F. L. 2011. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *arXiv preprint arXiv:1102.4225*.

Engesser, T.; Mattmüller, R.; Nebel, B.; and Thielscher, M. 2021. Game description language and dynamic epistemic logic compared. *Artificial Intelligence (AIJ)* 292:103433.

Gammie, P., and Van Der Meyden, R. 2004. MCK: model checking the logic of knowledge. In *Proceedings of the 16th International Conference on Computer Aided Verification*, 479–483. Boston: Springer.

Genesereth, M., and Björnsson, Y. 2013. The international general game playing competition. *AI Magazine* 34(2):107–107.

Genesereth, M., and Thielscher, M. 2014. General game playing. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8(2):1–229.

Goldwaser, A., and Thielscher, M. 2020. Deep reinforcement learning for general game playing. In *Proceedings of AAAI*, 1701–1708.

Gunawan, A. 2023. *Investigating Novel Representations and Deep Reinforcement Learning for General Game Playing*. Ph.D. Dissertation, Auckland University of Technology.

Haufe, S., and Thielscher, M. 2012. Automated verification of epistemic properties for general game playing. In *Proceedings of KR*, 339–349.

Haufe, S.; Schiffel, S.; and Thielscher, M. 2012. Automated verification of state sequence invariants in general game playing. *Artificial Intelligence (AIJ)* 187:1–30.

He, Y.; Saffidine, A.; and Thielscher, M. 2024. Solving two-player games with QBF solvers in general game playing. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 807–815.

Huang, X.; Ruan, J.; and Thielscher, M. 2013. Model checking for reasoning about incomplete information games. In Cranefield, S., and Nayak, A., eds., *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, volume 8272 of *LNCS*, 246–258. Dunedin: Springer.

Jamroga, W.; Malvone, V.; and Murano, A. 2019. Natural strategic ability. *Artificial Intelligence (AIJ)* 277:103170.

Kuhlmann, G.; Dresner, K.; and Stone, P. 2006. Automatic heuristic construction in a complete general game player. In *Proceedings of AAAI*, 1457–62.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. *Int. J. Softw. Tools Technol. Transf.* 19(1):9–30.

Love, N.; Genesereth, M.; and Hinrichs, T. 2006. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford University, Stanford, CA.

Maubert, B.; Mittelmann, M.; Murano, A.; and Perrussel, L. 2021. Strategic reasoning in automated mechanism design. In *Proceedings of KR*, volume 18, 487–496.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. Y. 2014. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.* 15(4):34:1–34:47.

Ruan, J., and Thielscher, M. 2011. The epistemic logic behind the game description language. In *Proceedings of AAAI*, 840–845.

Ruan, J.; Van Der Hoek, W.; and Wooldridge, M. 2009. Verification of games in the game description language. *Journal of Logic and Computation* 19(6):1127–1156.

Schiffel, S., and Thielscher, M. 2014. Representing and reasoning about the rules of general games with imperfect information. *Journal of Artificial Intelligence Research* 49:171–206.

Schofield, M., and Thielscher, M. 2015. Lifting model sampling for general game playing to incomplete-information models. In *Proceedings of AAAI*, 3585–3591.

Thielscher, M. 2009. Answer set programming for single-player games in general game playing. In Hill, P., and Warren, D., eds., *Proceedings of the International Conference on Logic Programming (ICLP)*, volume 5649 of *LNCS*, 327–341. Pasadena: Springer.

Thielscher, M. 2011. The general game playing description language is universal. In *Proceedings of IJCAI*, 1107–1112.