Automatically Verifying and Repairing General Game Descriptions

Yifan He

University of New South Wales, Australia

Abstract

General Game Playing (GGP) is concerned with the development of systems that can play any new game from the mere rules without human intervention. One major challenge in GGP is to endow a player with the ability to prove game-specific knowledge from the mere game rules automatically. Therefore, Automated Theorem Proving has become a popular research area in GGP. Automated theorem provers can assist GGP systems in formally verifying strategic or non-strategic logical properties of general game descriptions. They can also help game designers to check whether game descriptions are correctly encoded and satisfy desired logical properties. My research focuses on three main topics: (1) extending verification methods in GGP to support more expressive logics; (2) identifying a logical framework that can express important game properties, such as strong winnability that can be verified more efficiently through a specialized encoding; and (3) introducing the game description repair problem and developing the first automated method for repairing descriptions that violate formal requirements.

Keywords

General Game Playing, Game Description Language, Answer Set Programming, QBF, Model Checking

1. Introduction

The Game Description Language (GDL) has been developed as a lightweight knowledge representation formalism for describing the rules of arbitrary games [1]. GDL describes game rules in normal logic program syntax similar to Prolog. 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, thereby demonstrating a form of general intelligence. While the original version of GDL, introduced for the first AAAI GGP competition, was restricted to games with complete information, the language was later extended to GDL-II to support the description of games with imperfect information [2].

As a lightweight specification language, GDL merely provides means for *representing* the rules of a game, while a crucial aspect of GGP is the ability to automatically *reason* about a given specification. Important basic properties of games include, for example, termination (i.e., the game described in GDL is guaranteed to terminate), playability (i.e., in all non-terminal states of the game, every player has at least one legal move), and constant-sum (i.e., the sum of scores of all players at terminal states is fixed). Properties about strategies include, for example, strong winnability and the existence of a Nash equilibrium. In early years, successful GGP systems perform random simulations to test the validity of certain properties and rely on their informed guess in case no violation could be detected [3, 4]. However, validating properties through random simulations can be unreliable, and the performance of the GGP system can be affected if incorrect assumptions are made about the game. For example, in a single-player game, if the system wrongly assumes that a subtree does not contain a winning state, it might prune away the actual solution path to the goal.

For this purpose, Automated Theorem Proving was proposed to assist GGP systems in formally analyzing whether a GDL or GDL-II description satisfies certain logical properties [5]. From the perspective of building stronger GGP systems, it is always desirable to have automated theorem provers that can either verify more *general* game-specific properties or are more *efficient* at verifying some

Doctoral Consortium of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR 2025 DC), November 11-17, 2025, Melbourne, Australia

yifan.he1@unsw.edu.au (Y. He)

https://www.linkedin.com/in/yifan-he-unsw/ (Y. He)

properties. Beyond assisting GGP systems, automated theorem provers can also be used by the game designer to reliably check whether game descriptions are correctly encoded and satisfy desired logical properties, although, as for now, if a description violates certain properties, the task of repairing it must be done *manually*. Hence, from the perspective of game designers, it is clearly desirable to have automated methods of repairing ill-defined game descriptions.

The following three research questions can be naturally derived from this background:

R1: Can the Automated Theorem Proving methods in GGP be extended to support more expressive logic and enable the verification of important game properties that are not verifiable by previous theorem provers? (Section 2)

R2: Is there some logical framework that captures some important game properties and can be verified more efficiently through specialized encodings? (Section 3)

R3: Can we design an automated method of repairing game descriptions that violate some formal logic requirements? (Section 4)

Since both GDL and GDL-II are logical languages, we focus on using logic-based methods to address these research questions.

Various works have been carried out in Automated Theorem Proving in GGP. 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 [6, 7]. In addition, the model checker MCK was used to verify Epistemic Computation Tree Logic (CTLK) properties of GDL-II games [8]. However, none of these works handle the strategic dimension. Ruan et al. (2009) provides a framework in which GDL can be understood as a specification language for models of Alternating-time Temporal Logic (ATL) [10]. But their approach is restricted to the original GDL for games with complete state information. Although Ruan and Thielscher (2012) extends the work to allow the verification of Epestemic Alternating-time Temporal Logic (ATEL) properties of GDL-II games, the work stays at the theoretical level, and ATEL cannot express important strategic properties like the existence of a Nash equilibrium.

Although no prior work was carried out in repairing game descriptions that violate formal requirements in GGP, many works have been done regarding repairing formal models in the context of e.g., biological networks [12], service-based processes [13], and Petri nets [14]. One piece of highly related work is repairing unsolvable planning domain descriptions (PDDL) [15]. While PDDL and GDL differ, their work can only repair reachability to the goal (aka. winnability in GDL) instead of arbitrary properties expressed in a certain logical framework.

2. Verification of General Games with Imperfect Information with SLK

2.1. Motivation

To extend the Automated Theorem Proving methods in GGP, we can establish a concrete link between GDL or GDL-II with a more expressive logical language. Notably, no *practical* work in Automated Theorem Proving within GGP considers reasoning about strategic properties in GDL-II games. This raises a natural research question: how can we automatically reason about strategic properties in GDL-II games using existing model checkers?

Epistemic Strategy Logic (SLK) [16] is a rich logical framework for reasoning about multi-agent systems and the strategic behavior of agents with partial observability. In particular, it can express important game properties such as the existence of pure strategy Nash Equilibria, which neither ATEL nor CTLK can express. More importantly, the model checking toolkit MCMAS has an implementation of the SLK model checking algorithm [17]. Our goal, therefore, is to draw a concrete link between GDL-II and SLK models and allow GGP systems to take advantage of this rich formalism for the automatic verification of strategic properties of imperfect information games with MCMAS.

2.2. Approach and Progress

To automatically reason about logical properties specified in the SLK framework, we provide a formal translation from GDL-II games to *interpreted systems*, which are the semantical structures used in both SLK and MCMAS. Our goal is to create a translation $\Pi(G)$ for any GDL-II description G, which represents an interpreted system that is the input to MCMAS. We need to ensure that the model derived from $\Pi(G)$ using MCMAS operational semantics is isomorphic to the interpreted system $\mathcal{I}_G$ derived from G using GDL-II semantics.

In our KR 2024 paper [18], we present the translation and prove its correctness. We also show how to express and verify the strategic properties of players, such as strong winnability and pure-strategy Nash equilibrium in GDL-II games, using MCMAS.

The main limitation of that work lies in scalability. This is due to the complexity of SLK model checking, which is EXPSPACE-complete with respect to the size of the GDL-II description and the formula, and which motivates the development of more efficient, or possibly sound but incomplete, model checkers.

3. Verification of Perfect Information Games with QBF Solvers

3.1. Motivation

Our work on reasoning about GDL-II games with SLK provides a general framework for verifying a broad class of properties in GDL-II games. This significantly extends the range of verifiable properties supported by earlier approaches based on ATL [9]. However, using SLK or ATL model checkers to reason about game-specific properties remains inefficient. This is mainly due to the high complexity of model-checking these logics (e.g., for ATL, it is EXPTIME-complete with respect to the size of the GDL description and formula), and existing implementations of model-checkers for these logics require a full expansion of the state space before property verification can even begin. Due to the potentially large state space of general games, explicitly expanding all game states is certainly undesired.

To address **R2**, we identify a fragment of some popular logical frameworks that can express some interesting game properties, and model-checking for such a logical framework in the context of GGP is within PSPACE with respect to the size of the game description and logical formula. Thielscher and Voigt (2010) defined a temporal logic (GTL) that is similar to the Linear Temporal Logic with only the temporal operator "next" ($\bigcirc$) and allows the formulation of properties that involve finitely many successive game states. Model-checking against such a logic is only co-NP-complete. However, such a logic can only express non-strategic properties. We identify a logical framework that can also express some strategic properties.

3.1.1. Problem Definition

We consider a logic over the ground atoms of GDL descriptions that has the following grammar:

$$\varphi ::= q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \langle\langle C \rangle\rangle \bigcirc \varphi$$

We denote the logic as GCL, which is similar to Coalition Logic [19] or ATL over finite traces [20] with only the temporal operators $\langle\langle C \rangle\rangle \bigcirc$ (i.e., "the coalition C can enforce something in the next state") defined in the context of GDL, which extends the range of verifiable properties supported by GTL. One interesting strategic property that can be expressed in this logic is bounded-depth strong winnability (denoted as $\phi_{win}(p, n)$), which is whether a player p has a strategy to win the game within n steps, regardless of the actions of the remaining players in $R \setminus \{p\}$. This property is useful to GGP systems when conducting endgame searches in perfect information games.

- $\phi_{win}(p, n) = win(terminal \wedge goal(p, 100), n)$ where,
- $win(\gamma, 0) = \gamma$, and $win(\gamma, i) = \gamma \vee \langle\langle C \rangle\rangle \bigcirc win(\gamma, i - 1)$

3.1.2. Approach and Progress

We can show that GCL model-checking is PSPACE-complete with respect to *the size of the GDL description and the formula*. Hence, using a QBF solver to perform the model-checking task is a natural idea. GDL uses stable models for the semantics, while QBF is based on classical models. Converting directly from GDL to QBF is, therefore, challenging as it would require some form of completion technique [21]. Our approach is to extend the Answer Set Programming approach of GTL model-checking [6] by encoding the GCL model-checking task with an intermediate language called Quantified Answer Set Programming (QASP), which is based on stable model semantics and has the same expressive power as QBF. Thanks to the existing work on converting from QASP to QBF [22], we do not need to deal with the completion task from scratch as long as we can encode the model-checking task of GCL in QASP. We can then use QBF solvers [23] to evaluate the resulting QBF, which corresponds to the result of the GCL model-checking task.

Our work on verifying the formula $\phi_{win}(p, n)$ when $|R| = 2$ was published in AAMAS 2024 [24]. In that paper, we compare the efficiency of the QBF approach with forward search on solving several popular two-player GDL games such as Breakthrough, Connect-4, and Dots and Boxes. We show that the QBF approach is comparable with forward search, hence can be used as an alternative approach for GGP systems to evaluate endgame positions. The work on general GCL model-checking was accepted by NMR 2025.

4. General Game Descriptions Repair

4.1. Motivation

Theorem Provers can also help game designers to verify if the game descriptions are encoded correctly. For example, game descriptions used in a GGP competition are required to be well-formed [1], which means: the game must terminate, and is both playable (cf. Section 1) and weakly winnable (i.e., for each player, there is a sequence of joint moves leading to one of its winning states). Game designers can use theorem provers to check if a game description satisfies some desired properties, such as well-formedness, before submitting it to the competition. This is not ideal for non-expert GDL encoders. For this reason, we define the GDL repair problem and propose the first automated method for repairing general game descriptions. Repairing game descriptions is a new research problem. We start by focusing on the repair of GDL descriptions only, and repairing GDL-II is left as future work.

4.2. Problem Definition

We call a GDL rule a legal rule if its head is an instance of the predicate $legal$, and similarly, we call a GDL rule a next rule if its head is an instance of the predicate $next$. We consider a game description G to be broken if it satisfies some undesired properties or dissatisfies some desired properties under some logic L. Informally speaking, a repair to the game description is considered as a set of modifications to the legal and next rules of G or adding a bounded number of new legal or next rules to G. Each modification to G has a cost. To avoid redesigning the entire game, our goal is to calculate the lowest-cost modification to G so that the resulting game description $G' \not\models_L \phi^-$ for all $\phi^- \in \Phi^-$ and $G' \models_L \phi^+$ for all $\phi^+ \in \Phi^+$ where Φ^- (resp. Φ^+) is a set of logical formulas for some logical framework L that G' should dissatisfy (resp. satisfy). We only allow modifications to legal/next rules because changing other rules, such as the definition of base propositions, initial state, and goal, would be akin to defining a new game rather than repairing an existing one.

4.3. Approach and Progress

Game description repair is a broad topic. We start by focusing on solving the lowest-cost repair problem for GDL descriptions that violate logical requirements described in the simple logic GTL (cf. Section 3) (i.e. $L =$ GTL). For example, in the context of GTL, to ensure the repaired game description G' must

terminate within n steps, one can introduce the GTL formula $\phi_{end}(n)$ to Φ^+ which says that for all playing sequences, $terminal$ must hold at some step $0 \leq i \leq n$.

- $\phi_{end}(n) = terminal \vee \bigcirc terminal \vee \ldots \vee \bigcirc^n terminal$

where $\bigcirc^n$ denotes n consecutive $\bigcirc$ connectivities.

To solve the repair problem, we need to identify the complexity class of the problem and encode the problem with some logical language that can express problems in that complexity class. The work was accepted by KR 2025. In the paper, we provide sufficient conditions on when certain repair problems have or do not have solutions. We prove that the complexity class of the lowest-cost repair problem for the logic GTL is $F\Delta_3^P$-complete[1]. Based on the complexity analysis, we apply an Answer Set Programming technique called *guess and check* [26] to solve the lowest-cost repair problem automatically and show its effectiveness on repairing broken GDL descriptions.

5. Future Work

My research mainly focuses on verifying general games and repairing broken game descriptions. By employing logic-based methods to address these problems, we have established connections between General Game Playing research and the research in different branches of logic.

For future work, in terms of verifying general game descriptions, we can try to identify fragments of other logical frameworks that can express interesting game properties and can be verified more efficiently through specialized encodings. In particular, it is worth exploring whether certain fragments of epistemic logics (e.g., SLK), that can express strategic properties, may admit more efficient verification procedures, thereby enabling their practical use in verifying GDL-II games. For repairing broken game descriptions, we can explore how to solve the repair problem in the context of imperfect information games. We plan to extend the definition of the game description repair problem to GDL-II descriptions, which allows the modifications of the $sees$ rules (i.e., modifying what a player can see in the next state). We can then consider how to repair GDL-II descriptions that violate formal requirements specified in some epistemic logics such as GTLK, CTLK, or SLK. Out of these logics, repairing GTLK [7] properties is the most promising one. Since GTLK is an extension of GTL with the knowledge operator and both GTLK and GTL model checking use Answer Set Programming, we might expect that a *guess and check*-based approach for repairing GTLK properties for GDL-II games can be developed by extending our current work on repairing GTL properties.

For the remaining time of my PhD degree, I aim to address the problem of repairing GDL-II descriptions that violate GTLK properties and include the result in my thesis.

Acknowledgments

I want to thank my supervisors, Prof. Michael Thielscher and Dr. Abdallah Saffidine, for their support.

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

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

[2] M. Thielscher, A general game description language for incomplete information games, in: Proc. of AAAI, 2010, pp. 994–999.

[1] $F\Delta_3^P$ is the class of function problems that can be solved in polynomial time with a Σ_2^P oracle [25].

[3] J. Clune, Heuristic evaluation functions for general game playing, in: Proc. of AAAI, 2007, pp. 1134–1139.
[4] M. Świechowski, J. Mańdziuk, Y. S. Ong, Specialization of a uct-based general game playing program to single-player games, IEEE Transactions on Computational Intelligence and AI in Games 8 (2015) 218–228.
[5] S. Haufe, Automated Theorem Proving for General Game Playing, Ph.D. thesis, Dresden University of Technology, 2012. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-89998.
[6] M. Thielscher, S. Voigt, A temporal proof system for general game playing, in: Proc. of AAAI, 2010, pp. 1000–1005.
[7] S. Haufe, M. Thielscher, Automated verification of epistemic properties for general game playing, in: Proc. of KR, 2012, pp. 339–349.
[8] X. Huang, J. Ruan, M. Thielscher, Model checking for reasoning about incomplete information games, in: Proc. of AI, 2013, pp. 246–258.
[9] J. Ruan, W. Van Der Hoek, M. Wooldridge, Verification of games in the game description language, Journal of Logic and Computation 19 (2009) 1127–1156.
[10] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM 49 (2002) 672–713.
[11] J. Ruan, M. Thielscher, Strategic and epistemic reasoning for the game description language GDL-II, in: Proc. of ECAI, 2012, pp. 696–701.
[12] M. Gebser, C. Guziolowski, M. Ivanchev, T. Schaub, A. Siegel, S. Thiele, P. Veber, Repair and prediction (under inconsistency) in large biological networks with answer set programming., in: Proc. of KR, 2010, pp. 497–507.
[13] G. Friedrich, M. G. Fugini, E. Mussi, B. Pernici, G. Tagni, Exception handling for repair in service-based processes, IEEE Transactions on Software Engineering 36 (2010) 198–215.
[14] F. Chiariello, A. Ielo, A. Tarzariol, An ilasp-based approach to repair petri nets, in: Proc. of LPNMR, 2024, pp. 85–97.
[15] A. Gragera, R. Fuentetaja, Á. García-Olaya, F. Fernández, A planning approach to repair domains with incomplete action effects, in: Proc. of ICAPS, 2023, pp. 153–161.
[16] P. Čermák, A. Lomuscio, F. Mogavero, A. Murano, Practical verification of multi-agent systems against slk specifications, Information and Computation 261 (2018) 588–614.
[17] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: an open-source model checker for the verification of multi-agent systems, International Journal on Software Tools for Technology Transfer 19 (2017) 9–30.
[18] Y. He, M. Mittelmann, A. Murano, A. Saffidine, M. Thielscher, Verification of general games with imperfect information using strategy logic, in: Proc. of KR, 2024, pp. 420–430.
[19] M. Pauly, A modal logic for coalitional power in games, Journal of logic and computation 12 (2002) 149–166.
[20] F. Belardinelli, A. Lomuscio, A. Murano, S. Rubin, et al., Alternating-time temporal logic on finite traces., in: Proc. of IJCAI, 2018, pp. 77–83.
[21] V. Asuncion, F. Lin, Y. Zhang, Y. Zhou, Ordered completion for first-order logic programs on finite structures, Artificial Intelligence 177 (2012) 1–24.
[22] J. Fandinno, F. Laferrière, J. Romero, T. Schaub, T. C. Son, Planning with incomplete information in quantified answer set programming, Theory and Practice of Logic Programming 21 (2021) 663–679.
[23] M. N. Rabe, L. Tentrup, Caqe: A certifying qbf solver, in: Proc. of FMCAD, 2015, pp. 136–143.
[24] Y. He, A. Saffidine, M. Thielscher, Solving two-player games with QBF solvers in general game playing, in: Proc. of AAMAS, 2024, pp. 807–815.
[25] M. W. Krentel, Generalizations of opt p to the polynomial hierarchy, Theoretical Computer Science 97 (1992) 183–198.
[26] T. Eiter, A. Polleres, Towards automated integration of guess and check programs in answer set programming: a meta-interpreter and applications, Theory and Practice of Logic Programming 6 (2006) 23–60.