From Natural Language to Extensive-Form Game Representations: Defining and Automatically Verifying Consistency

Yongzhao Wang*
University of North Florida
Jacksonville, United States
yongzhao.w@unf.edu

Shilong Deng*
University of Liverpool
Liverpool, United Kingdom
shilong.deng@liverpool.ac.uk

Daniel A. Kadnikov
The Alan Turing Institute
London, United Kingdom
dkadnikov@turing.ac.uk

Ed Chalstrey
The Alan Turing Institute
London, United Kingdom
echalstrey@turing.ac.uk

Martin Gairing
University of Liverpool
Liverpool, United Kingdom
gairing@liverpool.ac.uk

Enbo Sun
University of Liverpool
Liverpool, United Kingdom
sgesun@liverpool.ac.uk

Theodore Turocy
The Alan Turing Institute
University of East Anglia
Norwich, United Kingdom
t.turocy@uea.ac.uk

Rahul Savani†
The Alan Turing Institute
University of Liverpool
Liverpool, United Kingdom
rahul.savani@liverpool.ac.uk

ABSTRACT

Game description translation (GDT) converts natural language descriptions of strategic interactions into formal game-theoretic models. Historically, the evaluation of the correctness ("consistency") of such a candidate translation has relied on manual inspection by game theory experts. This manual process is time-consuming and error-prone. Moreover, due to the lack of a formal definition of consistency, evaluation is subjective, which hinders reproducibility. To address these limitations, we formalize consistency, and develop a two-component framework for the automated evaluation of GDT correctness for extensive-form game models. The first component is a set of *translation instances*. Each instance comprises a natural language description, an example of a consistent translation ("reference game") and further constraints. The second component is an *automated checker*. The checker works by first using the reference game and any relevant constraints to check that the set of reduced strategies in the reference game and candidate translation match. Then, if the candidate passes that check, the checker goes on to check any remaining constraints, such as on the relationships between payoffs in the game. Relative to prior practice, the framework shifts the role of experts from repetitive manual evaluation to the one-time creation of translation instances. To support reproducible research on GDT, we provide an extensible instantiation of our framework: a curated dataset of extensive-form game translation instances that were created and verified by a group of game theory experts; and an implementation of automated consistency checkers for this dataset (and future extensions of it). Using the dataset and checkers, we evaluate a range of translators, including a new translator, which demonstrates state-of-the-art performance. The evaluation demonstrates the ability of the framework to effectively assess the performance of translators.

ACM Reference Format:
Yongzhao Wang, Shilong Deng, Daniel A. Kadnikov, Ed Chalstrey, Martin Gairing, Enbo Sun, Theodore Turocy, and Rahul Savani. 2026. From Natural Language to Extensive-Form Game Representations: Defining and Automatically Verifying Consistency. In *Strategic Engineering Workshop on LLMs and Game Theory at the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 15 pages.

1 INTRODUCTION

Game description translation (GDT) is the process of converting descriptions of strategic interactions in natural language into formal game-theoretic representations to enable further strategic analysis. This translation process involves identifying the strategic components of a situation, constructing a game model by specifying players, rules of engagement, and the players' utilities, and finally constructing a formal game model, such as an *extensive-form game* (EFG), which is the standard game-theoretic model for strategic interactions and the focus of this paper. Historically, this has been done by domain experts with game-theoretic knowledge. However, the task is laborious and challenging even for experts, becoming infeasible as the complexity of the strategic situation grows. Thus, GDT research has explored approaches such as large language models (LLMs) to automate the process of generating candidate translations [4, 12, 16]. GDT is a key component of "strategic engineering", a broader nascent research agenda that explores how modern AI tools like LLMs can be used to build automated strategic analysis pipelines.

Automation of GDT, on its own, does not eliminate the bottleneck created by the need for expert time and knowledge. It only shifts the work of the experts from the generation of the EFG to the verification that the EFG is consistent with the natural language description, which is just as laborious and challenging. This is a limiting factor because the ultimate goal of GDT is to develop translators that we can use "in the wild" to translate descriptions of strategic interactions

*Equal contribution.
†Corresponding author.

Strategic Engineering Workshop on LLMs and Game Theory at the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.
© 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). This work is licenced under the Creative Commons Attribution 4.0 International (CC-BY 4.0) licence.

into formal game models without the need for expert supervision of each application. We have a blueprint for how to accomplish this from any number of other machine learning tasks: we need a well-designed dataset of "labeled" examples that allows us both to develop and to evaluate translators, such that we can determine automatically whether a produced translation is consistent with a description.

Most descriptions - especially those likely to arise in the wild - admit many potential EFG representations that could be considered consistent. Experts have been translating descriptions into EFGs since the establishment of game theory as a field, so there exist principles that distinguish consistent and inconsistent translations, implicitly or tacitly. However, there is no existing formal definition of such "consistency" between a game description and an EFG. In this paper, we fill this critical gap, by defining consistency formally. Our main contributions include:

(1) A formal definition of consistency for translation to EFGs, grounded in widely-accepted principles in game theory;
(2) An automated consistency checker and a compatible dataset of translation instances, validated by a team of game theory experts;
(3) A demonstration of our checker and dataset through a benchmarking exercise;
(4) A new translator that shows state-of-the-art performance in the benchmarking exercise.

These four contributions together establish a theoretical and practical foundation for performing and evaluating GDT. Our code and dataset are open source and extensible, and are intended to lay the foundation for reproducible research into GDT.

Our definition of consistency requires certain input data in addition to the natural language description. This data is created by a game theory expert, based on the expert's analysis of the game description. Firstly, the expert creates a ***reference game***, which is a translation that the expert deems consistent with all relevant aspects of the description. An important requirement is that the natural language description is rich enough in detail to allow the expert to ensure the reference game has semantically meaningful *labels* for players, for actions, and for events determined randomly by nature. This is *not* normally a requirement in game theory, but is important for us for two reasons:

- It facilitates efficient checking of properties that determine consistency;
- It enables a user of the translation to understand the correspondence between the EFG and the description from which it was derived.

Secondly, the expert creates any ***additional data*** needed for checking consistency. Strategic equivalence focuses on aspects of the situation which are relevant to a "rational" decision-maker, and therefore, roughly speaking, on information. Consider two variants of the game "rock, paper, scissors". In the most common variant, both players "throw" their choice at the same time. Therefore, neither has information about the other's choice when they make their own. Consider instead a variant in which one player writes down their choice on a piece of paper and places it face down on the table between the players; then the other player "throws" their choice as usual. These games are strategically equivalent because players have the same information in each at the time of their decision. EFGs do not support "simultaneous" moves, so for the standard variant an ordering of the players' decisions in the EFG has to be chosen; because the choices are temporally simultaneous, either ordering of the players would do. Meanwhile, in the second variant, we argue only an EFG in which the first player chooses earlier in the tree would be accepted as consistent with the description. Our methodology uses additional data to differentiate between the two variants.

Game theory captures players' incentives through their payoffs that result from the game. Abstractly payoffs are a numerical representation of a player's preferences over the possible outcomes of the game (their ***utility function***). However, in some applications, payoffs may be specified in whole or part; for example, if players gain or lose amounts of money as a result of a game such as poker, the user might want those net monetary amounts to be recorded as the payoffs to the players. The additional data includes constraints that describe the required relationships among payoffs that must be satisfied by any EFG consistent with the description.

With the reference game and additional data, the first condition of consistency is that the sets of ***reduced strategies***[1] of the reference game and the candidate are identical. This follows established conventions in game theory which focus on the set of reduced strategies as capturing the plans of action available to each player (see e.g. [1, 5]). However, it is not straightforward to check whether the sets of reduced strategies in two arbitrary games match. In designing our dataset, we remove ambiguity about strategic modeling choices in the natural language descriptions, thereby narrowing the set of consistent translations and ensuring that our automated checkers can faithfully align the reduced strategies in the two games using semantic labels. If the sets of reduced strategies match, we deem the candidate translation consistent if it also meets all of the constraints in the additional data.

Paper Outline. In Section 4, we define consistency and discuss automated consistency verification, including semantic-label matching, comparisons of reduced strategies, and verification of constraint satisfaction. We also discuss the general challenges involved in consistency verification. In Section 5, we present our philosophy for dataset design and describe the specification of our dataset. In Section 6, we demonstrate the feasibility and effectiveness of our framework by evaluating existing translation methods alongside a new method introduced in this study. Our experiments show that the proposed evaluation framework successfully identifies both consistent and inconsistent translations produced by all translation methods, and that our new method achieves superior performance across a range of translation tasks.

2 RELATED WORK

Game description translation, the derivation of a game model from a strategic situation described in natural language, has long been a central task in game-theoretic analysis. As games of interest grow in size and complexity, however, performing this translation manually becomes increasingly impractical. Recently, the rapid rise of LLMs has motivated automated methods that generate formal games directly from natural language descriptions. The first work that defines this

[1] In an EFG, a reduced strategy is a pure strategy that removes moves at information sets that can never be reached because of the player's own earlier moves.

automation was proposed by Deng et al. [4]. They translated game descriptions in natural language into extensive-form games (EFGs) using Gambit [15], leveraging LLMs and emphasized that imperfect information poses a key difficulty for reliable translation.

Game description translation also serves as a component in broader objectives. For example, Daskalakis et al. [3] used LLMs to extract and extend EFGs from natural language in order to "chart the shape of stories," following an idea of Kurt Vonnegut[2]. They used LLMs to introduce additional decision nodes, representing alternative choices that players could have made in the story. Once the EFG is constructed, Gambit [15] is used to compute the Nash equilibrium, providing a prediction of the players' behavior if those alternatives had been available.

When translating game descriptions, the choice of target format should follow the translation's purpose. For game-theoretic analysis, EFGs are a natural choice because they are expressive and work well with analysis tools such as Gambit. That said, EFGs are not the only viable choice; other formalisms [7, 10, 14, 16] can be equally appropriate depending on the intended use. For example, in related work by Mensfelt et al. [12], the authors used game description language (GDL) [6, 17, 19] as the target representation and focused primarily on 2×2 bimatrix games. GDL was designed to support the development of AI agents for general game playing. It comes in three versions: GDL I targets deterministic, perfect-information games; GDL II adds constructs for imperfect information and events determined randomly by nature; and GDL III further extends the language to capture epistemic aspects relevant to game-theoretic reasoning. Thielscher [18] showed that, under a perfect-recall assumption, GDL and EFG representations can be converted into each other without loss. When imperfect recall is present, however, EFGs are typically preferred. Mensfelt et. al [13] evaluated consistency between a candidate EFG and a reference EFG by checking whether the payoffs of certain predefined strategies agree with those in the original game. This test provides a necessary—but not sufficient—condition for overall consistency.

As the game size increases, explicit representations in either strategic form or extensive form rapidly become computationally intractable. In those settings, a simulator is often a more appropriate translation target. Wu et al. [22] developed Instruction-Driven Game Engine that allows users to create games using natural language instructions and tested their approach in poker games. Lehrach et al. [11] recently have used in-context learning to construct such simulators directly from natural language descriptions.

3 PRELIMINARIES

An extensive-form game (EFG) consists of the following components:

(1) **Game tree**: The core component of an EFG is a *rooted, directed tree*. Each decision node in the tree is assigned to either one of the players or to the ***Nature*** player, with the latter's nodes being ***chance nodes***. Directed edges represent moves or actions. Each action at a given node has a label unique among the actions at that node. A play of the game begins at the root and progresses through the tree as the player assigned to each decision node selects an action; at a chance node, an action is randomly drawn from a discrete probability distribution specified as part of the data of the game. The play concludes when it reaches a ***terminal*** (leaf) node of the tree.

(2) **Outcomes and payoffs**: Every terminal node corresponds to an outcome, with an associated payoff vector which prescribes the payoff for each player under this outcome.

(3) **Information sets**: An information set is a collection of decision nodes belonging to a single player, representing the situations where the player knows they are at one of these nodes but cannot distinguish exactly which one. Information sets capture a player's uncertainty about previous moves in the game. The set of actions available at every node within an information set must be identical. When every information set contains only a single node, the game has ***perfect information***; otherwise it has ***imperfect information***.

A ***pure strategy*** for a player specifies exactly which action to take at every information set belonging to that player, regardless of whether that node is actually reachable during play. A ***reduced strategy*** further removes actions at information sets that can never be reached because of the player's own earlier actions. The ***reduced strategic form*** of a game tree lists all reduced strategies for each player, and tabulates the expected payoff to each player for each reduced strategy profile. The process to derive the reduced strategic form from an EFG can be found, for example, in Section 4.5 [21].

Extensive-form representations are more expressive than other representations such as strategic-form representations in the sense that they explicitly capture the sequential nature of games, such as the timing of moves. This expressiveness makes them a natural target format for GDT. In this work, we focus on translations into EFGs and use Gambit's EFG format for serialization. In this work, we assume that all games are in extensive form, following Gambit's EFG format for serializing games.[3]

4 AN AUTOMATICALLY-VERIFIABLE DEFINITION OF CONSISTENCY

4.1 Consistency Definition and Translation Instance Format

Our consistency definition is built on a specific ***translation instance*** format, including (i) a game description, (ii) a single correctly translated reference game, and (iii) additional data.

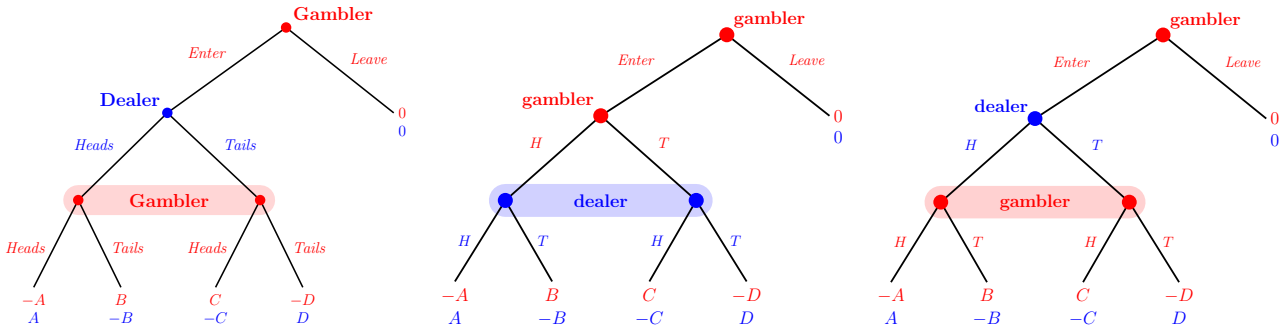
Definition 1 (Consistency)**.** A candidate translation is consistent with a game description if and only if it has the same set of reduced strategies as the reference game and satisfies the constraints in the additional data.

In the following discussion, we use the `gambling` game as a running example to illustrate the components of our consistency definition and explain how our checkers work. This example was chosen intentionally to be small and simple while still containing the features needed to demonstrate many of the key data design issues.

The game description of a gambling game. *A gambler decides whether to enter a casino. If the gambler stays out, the dealer takes no action and both receive 0. If the gambler enters, they play a coin toss game against the dealer: both players simultaneously choose either "Heads" or "Tails". If both choose "Heads", the gambler*

[2] https://www.youtube.com/watch?v=oP3c1h8v2ZQ&ab_channel=DavidComberg

[3] Gambit [15] is a widely-used software suite for game-theoretic analysis.



(a) A reference game.

(b) A candidate translation.

(c) Candidate with reordered players.

Figure 1: The reference game (a) and a candidate translation (b) of the gambling game. The two games differ in the ordering of players in the simultaneous move and in their labels. Then (c) is a variant of the candidate translation with reordered players for the simultaneous move.

receives $-A$ and the dealer receives A; if the dealer chooses "Heads" and the gambler chooses "Tails", the gambler receives B and the dealer receives $-B$; if the dealer chooses "Tails" and the gambler chooses "Heads", the gambler receives C and the dealer receives $-C$; finally if both chooses "Tails", the gambler receives $-D$ and the dealer receives D. The payoffs satisfy $A = D > B = C > 0$.

In our consistency definition, the reference game is a game constructed by a game-theory expert and certified to be consistent with a given description. It characterizes the strategic structure common to the class of games consistent with that description, thereby providing a concrete object of comparison in place of the textual description itself. Figure 1(a) shows an example of the reference game for `gambling`. Given such a reference game, the quality of a candidate translation (e.g., Figure 1(b)) can be evaluated by the extent to which it deviates from the reference game. These deviations may include differences in the sets of players, strategies, and payoffs. Our consistency-verification procedure examines all of these aspects.

4.2 Matching Player Names, Action Labels, and Chance Event Labels

4.2.1 Semantic Label Mapping and Why It Matters. The first step in checking consistency is to match labels for players, actions, and chance events. These labels link the original natural language game description to its consistent translations, and connect different translations to one another (for example, in comparing the reference game and a candidate translation). However, in standard game theory, labels are solely for notational convenience, and otherwise play no role in the formal analysis of the game. This means that, without further restrictions, matching players and actions potentially requires checking all permutations of players and all permutations of labels of actions, which is a combinatorial task with factorial growth.

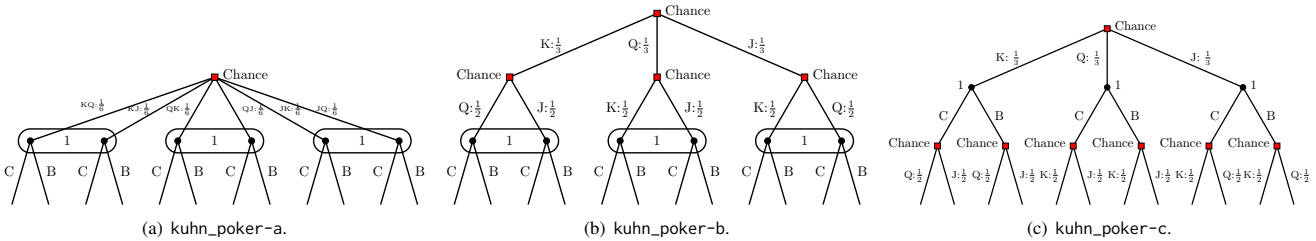
To handle this issue, we introduce ***semantic labels*** to represent player names, action labels, and chance event labels (i.e., action labels but for chance nodes[4]). Semantic labels yield three major advantages. First, they align with common sense of what makes a translation consistent. For example, the description of `gambling` states that the gambler and the dealer can choose either "Heads" or "Tails." Labeling these actions arbitrarily (for example, as "K" and "J" or 1 and 2), or reversing the labels (having the action which is intended to encode "Heads" actually be labeled "Tails") would naturally be considered inconsistent. Second, they make explicit which players and actions in a candidate translation correspond to those in the reference game; in other words, they provide a way to synchronize the two games, which is essential for verifying consistency. Third, by using semantic labels, we can match players and actions (and hence strategies) according to their semantic meanings rather than by exhaustive permutations, which yields a substantial computational benefit in practical applications.

To further illustrate the importance of semantic labels, we provide three examples of issues whose resolution requires semantic labels.

Example 1 – Simultaneous Moves: As already noted, EFGs use information structures to emulate simultaneous moves. Since any ordering of players is valid within a simultaneous move, we need player labels to identify the players so that we can permute them appropriately and efficiently when verifying consistency. This also motivates us to include a ***simultaneous-move identifier*** as the first component of the additional data, which specifies the scope of simultaneous move in the EFG.

Example 2 – Multiple Sensible Action Labels: In many settings, there may be multiple sensible action labels if the natural language description is not specific about this aspect of the problem. For example, Tic-Tac-Toe is a board game between two players, (X) and (O), who take alternate turns on a 3×3 grid, marking empty spaces with (x) and (o), respectively. In our description of `tic-tac-toe endgame` (see Appendix B.22), we specify action labels and their meanings by indexing the grid. Without these labels, that is, without a fixed grid orientation such as taking (0,0) to denote the top-left

[4]Note that chance probabilities are not, strictly speaking, "labels," though they can be matched efficiently while matching action labels. Rather, they impose implicit constraints on payoffs.



(a) kuhn_poker-a. (b) kuhn_poker-b. (c) kuhn_poker-c.

Figure 2: The initial deal for Kuhn poker in three different variants. The full descriptions for kuhn_poker-a, kuhn_poker-b, and kuhn_poker-c are in Appendix B. In kuhn_poker-a, a single chance node deals to both players in one go; in kuhn_poker-b, the deal happens first for player 1 then for player 2; and in kuhn_poker-c, the deal happens for player 1, who then decides what to do, before a card is dealt to player 2.

cell, it would be difficult to align the reference game with a candidate translation for further comparison, since the candidate may use other sensible action labeling, for example, taking (0,0) to denote the bottom-left cell.

Example 3 – Infinitely Many Strategically Equivalent EFGs: In general, as explained by the theory of Thompson transformations [2, 5, 9, 20] and discussed further in Section 4.4, for any EFG there are generally infinitely many *other* EFGs that are strategically equivalent, in the sense that they induce the same reduced strategic form. As a concrete example, in Figure 2, we show the initial chance moves for three strategically-equivalent EFG variants corresponding to the translation instances kuhn_poker-a, kuhn_poker-b, kuhn_poker-c, which differ in how the deal is realized in chance nodes. Only kuhn_poker-b is consistent with the standard game description of Kuhn poker, in which cards are dealt sequentially but before any player makes a choice. So labels are required to distinguish this consistent EFG from other strategically equivalent EFGs. To illustrate the complexity of Kuhn poker, a full game tree is shown in Appendix A.

4.2.2 *Implementation for Matching Labels and Reduced Strategies.*

Matching labels. To match players and actions, we leverage LLMs to identify labels with the same semantic meaning and perform the matching. In particular, given two sets of labels, the LLM maps each label in one set to its counterpart in another. For example, if the player label sets are (*Gambler*, *Dealer*) in the reference game and (*dealer*, *gambler*) in the candidate, then *Gambler* and *Dealer* will be matched with *gambler* and *dealer*, respectively. Similarly, if the two action label sets are (*Heads*, *Tails*) in the reference game and (H, T) in the generated game, then *Heads* and *Tails* will be matched with H and T, respectively. Even though using LLMs could introduce stochasticity in evaluation, we find that LLMs perform extremely well and robustly on this simple language task (with GPT-5-mini in our implementation). This is unsurprising: compared to complex language understanding, semantic label matching is straightforward.

To synchronize the reference game with the candidate game, we traverse both EFGs concurrently using breadth-first search. At each pair of nodes, we use LLMs to determine whether the nodes correspond to the same player and share the same action set including probabilities and associated event names for chance moves. When a simultaneous move occurs, as indicated by a simultaneous-move identifier, we switch the player order, handle the corresponding actions appropriately, and reorder the subgames according to game-theoretic principles.

We exemplify this matching process using gambling. We first apply an LLM to align player labels as discussed. Then we match action labels by traversing the reference (see Figure 1(a)) and candidate (see Figure 1(b)) EFGs in parallel:

(1) First, the LLM successfully matches the player label sets (*gambler*, *dealer*) in the candidate game to (*Gambler*, *Dealer*) in the reference game, respectively.
(2) Then we compare the root nodes. In both games, the root node is assigned to *Gambler*, and the LLM then correctly matches the corresponding actions, namely [*Enter*, *Leave*].
(3) Since the *Leave* branch terminates, the matching proceeds along the *Enter* branch.
(4) Following *Enter*, the acting players differ across the two games: the reference game assigns the move to *Dealer*, whereas the candidate game assigns it to *Gambler*. Since an identifier indicates this is a simultaneous move, the matching process reorders the candidate players to align with the reference ordering, which succeeds and creates an EFG in Figure 1(c).
(5) The LLM then successfully matches the *Dealer* actions [H, T] with [*Heads*, *Tails*].
(6) Conditional on either *Heads* or *Tails* being chosen by the dealer, the LLM subsequently matches the *Gambler* actions [H, T] with [*Heads*, *Tails*] successfully.

In this example, the reference and the candidate match successfully. If a candidate EFG fails any of these matches (e.g., it lacks semantically related labels), we consider it a failure in the consistency test since it indicates a limitation in the ability of the translator.

Comparing the reduced strategy sets. Once the labels have been matched, we use Gambit to directly compute the reduced strategy sets for the reference game and the possibly updated candidate translation, and then check whether the two sets are identical. Note that obtaining reduced strategies is extremely fast with Gambit since it does not require materializing the entire payoff tensor of the reduced strategic form, which in general is exponentially large in the size of the EFG. If the reduced strategy sets match, we proceed to verify whether the payoff constraints in the additional data are satisfied.

4.3 Constraints for Payoffs in Additional Data

A necessary condition for consistency is that the reference game and the candidate game share the same set of reduced strategies. This condition, however, is clearly not sufficient. The next step is therefore to determine whether their payoffs, or the relevant payoff relationships, are consistent. We do so using the payoff constraints constructed for each game. These constraints constitute the final component of additional data in our definition of consistency. We develop three types of constraints tailored to our dataset. Which type is used depends on the game description at hand:

- **Identical payoffs to reference game:** A set of constraints that ensure the payoffs of any consistent translation are identical to the reference game, which is appropriate when the natural language description explicitly gives all payoffs.
- **Total order with ties over player outcomes:** A set of constraints that impose a total order (with ties) on the payoffs of each player across outcomes, but without giving any explicit payoffs.
- **Explicit payoff(s) for a certain outcome:** A constraint that specifies one or more payoffs for a particular outcome.

For example, identical payoff constraints can be used when the description of an instance of `gambling` provides explicit payoff values; for instance, $A = D = 2$ and $B = C = 1$. A total-order constraint can be used when the description of a variant of `gambling` has $A = D > B = C > E = F$ with the payoffs after "Leave" being (E, F). For the description in the running example, we can use both total-order and explicit payoff constraints, with $A = D > B = C > E = F$ and $E = 0$.

If the sets of reduced strategies match and all constraints encoded in the additional data are satisfied, we deem the candidate translation consistent. We note that when creating a translation instance, it is not necessary to enumerate all constraints explicitly. For "identical payoffs to reference game" or "total order with ties over player outcomes," we use their names as keywords to specify the payoffs or payoff relationships that the checker should extract from the reference game and verify in the candidate. However, for "explicit payoff(s) for a certain outcome," we specify each constraint individually. The three types of constraints considered here cover many common use cases in GDT. With the framework established here, it is straightforward to extend the dataset and corresponding checker to consider other constraints such as best-response equivalence.

4.4 General Challenges in Verifying Consistency

A major challenge in automated consistency verification is that a natural language game description may, and typically will, admit a large number of (and possibly infinitely many) consistent translations. The root cause of this challenge is that when it comes to producing an EFG which matches the expectations of a user of game theory. Simply getting the sets of reduced strategies correct does not on its own constrain the translation adequately.

While each EFG has a uniquely-defined reduced strategic form, the same reduced strategic form can be generated by many EFGs. This was noted in the early days of noncooperative game theory by Thompson [20], who defined a set of operations which transform one EFG into another while leaving the reduced strategic form unchanged. These "Thompson transformations" have been further developed by Elmes and Reny [5], Kohlberg and Mertens [9] and Battigalli et al. [2]. One of these transformations, the interchange of simultaneous moves, is straightforward to incorporate via the additional data (i.e., a simultaneous-move identifier). More complications arise with transformations involving the addition or restructuring of actions or information sets. Because these transformations can be composed without affecting the reduced strategic form, it is possible to generate EFGs which are strategically equivalent but bear little or no resemblance to the original description.

Thompson's original analysis considered only games without chance moves, but a similar issue arises when chance moves are considered. The Kuhn poker example in Figure 2 is one illustration. Russian Roulette [8] likewise admits multiple equivalent EFG representations. In Russian Roulette there are two basic consistent forms: one places a chance node at the root, and the other places a chance node after each player move. Suitable combinations of these two forms also yield equivalent representations.

The key intuition as to why the EFG formalism inherently leaves these degrees of freedom in representation is that EFGs, and their corresponding reduced strategic forms, focus on the information a player knows at the time they make a decision. If a player does not have full information at the time they make a decision, it could be because the information will only be determined in the future, or because while the information exists, the player does not (yet) know it. From the perspective of the player as a "rational" decision-maker, these two cases are identical, and so the set of reduced strategies does not distinguish between them.

However, game descriptions in practice are derived from the way the strategic interaction is physically realized in time and space, and not only in terms of information. A user of game theory would typically distinguish among the different physical ways of dealing cards in Kuhn poker, or the different ways of realizing the outcome of turns in Russian Roulette, and would not consider an EFG that is at odds with the physical reality of the game to be consistent even if it is strategically equivalent.

The foregoing has implications for the construction of our dataset. For the purpose of training and testing game description translators, translation instances must be designed carefully to control and calibrate the amount of ambiguity or freedom of choice permitted in translation. For example, in describing a poker game like Kuhn poker, a candidate description might simply describe "dealing" cards to the players. For a typical person who has some familiarity with card games, the physical procedure for dealing cards would be familiar, implicit knowledge. However, if this is not included in the description, either the instance will not be able to place adequate constraints on the generated EFG, or the validator will be testing whether the generator knows about card games rather than whether it correctly extracts information from the description.

5 AN EXPERT-CURATED DATASET OF TRANSLATION INSTANCES

Our dataset consists of 27 translation instances designed to evaluate translators across a range of controlled GDT challenges. Each translation instance contains (i) a game description, (ii) a single correctly translated reference game, and (iii) additional data. The additional data may include simultaneous-move identifiers and the constraints described in Section 4.3. The dataset is intended as an

Games	Perfect Info	#Nodes	#Leaves	Direct	GameInterpreter	MCP	Our Approach
nim(2,2,1)	Yes	90	60	✗	✓[4]	✓[2]	✓
centipede(30 rounds)	Yes	60	61	✓[14]	✓[19]	✗	✓[19]
tic-tac-toe endgame	Yes	17	17	✓[9]	✓[11]	✓[9]	✓
russian roulette-b	Yes	12	12	✓[8]	✓[1]	✓[3]	✓[19]
russian roulette-c	Yes	11	12	✓[14]	✓[13]	✓[17]	✓[19]
bagwell-a	No	7	8	✓[18]	✓[18]	✓[17]	✓
bagwell-b	No	7	8	✓[15]	✓[17]	✓[13]	✓
bagwell-c	No	7	8	✓[13]	✓[14]	✓[15]	✓[19]
gambling-a	No	4	5	✓[18]	✓	✓[18]	✓
gambling-b	No	4	5	✓[19]	✓	✓	✓
gambling-c	No	4	5	✓	✓	✓	✓
market signalling game	No	7	8	✓	✓	✓	✓
monty hall problem	No	13	18	✗	✗	✗	✓[9]
memory one game	No	63	64	✗	✓[5]	✗	✓[19]
kuhn poker(2-player 3 cards)-a	No	25	30	✓[13]	✓[15]	✓[16]	✓
kuhn poker(2-player 3 cards)-b	No	28	30	✓[8]	✓[9]	✓[18]	✓[19]
kuhn poker(2-player 3 cards)-c	No	28	30	✓[14]	✓[15]	✓[16]	✓
kuhn poker(2-player 4 cards)	No	49	60	✓[5]	✓[18]	✓[13]	✓
russian roulette-a	No	22	27	✓[12]	✓[16]	✓[19]	✓
nuclear crisis-a	No	5	6	✓	✓	✓	✓
nuclear crisis-b	No	5	6	✓[18]	✓	✓[18]	✓
nuclear crisis-c	No	5	6	✓[18]	✓	✓	✓
voting game-a	No	7	36	✓[14]	✓[17]	✓[18]	✓
voting game-b	No	7	36	✓[13]	✓[16]	✓[15]	✓
voting game-c	No	7	36	✓[15]	✓[15]	✓[19]	✓
voting game-d	No	7	36	✓[15]	✓[17]	✓[17]	✓
voting game-e	No	7	36	✓[15]	✓[14]	✓[7]	✓

Table 1: A red cross indicates that none of the generated samples passed; a green tick shows that all twenty generated samples passed; a grey tick signifies that between 1 and 20 generated samples passed, with the exact number of successes shown in brackets.

initial but carefully controlled benchmark for developing, training, and testing GDT translators, and as a foundation for developing future translation instances that better approximate the diversity of descriptions encountered "in the wild." The desiderata guiding the selection and design of our translation instances for this first version of the dataset are:

Diversity: We select games that vary in strategic complexity and size, including those with complicated chance events and imperfect recall (no absent-mindedness), to present different challenges to translators. For example, for each variant of Kuhn poker in Figure 2, we create a specific game description that makes the corresponding variant the uniquely consistent EFG. This challenges the translator to correctly interpret the different timing of chance events. We also include games which test whether the translator can correctly capture payoff relationships across strategy profiles (`bagwell-⋆`), and whether it can handle large games (`nim(2,2,1)`).

Description specificity: We write a natural language description that minimizes ambiguity in the relevant strategic modeling choices by specifying explicitly details such as action labels and the order of moves by players and *Chance* while encouraging the modeling choice within each game to be *self-consistent*. This is one reason we emphasize that even having 27 games is highly valuable: substantial care is required to ensure that, up to isomorphism arising from different orderings of actions or chance events, essentially only one EFG form is consistent with the description. In particular, the descriptions are written to rule out the Thompson transformations that would otherwise generate strategically equivalent but inconsistent EFGs. These efforts make the resulting instances amenable to automated

checking and allow the checkers to provide a meaningful measure of translator performance.

Mitigation of memorization risk: We remove the names of strategic situations and customize the descriptions to reduce the chance that LLM-based translators rely on memorized consistent EFGs. For example, a translator that recognizes `kuhn_poker-a` as a Kuhn-Poker-like game, but outputs the EFG corresponding to `kuhn_poker-b`, is marked incorrect, because it has ignored description-specific details about how the deal is realized. In this case, the translator is appropriately penalized for relying on memorization rather than fully understanding and following the description.

Semantic meaningfulness: We ensure that names for players, actions, and chance events are semantically meaningful, clear, and used consistently within each description.

6 BENCHMARKING TRANSLATORS USING OUR DATASET AND CHECKERS

To demonstrate the feasibility and effectiveness of our evaluation framework, we use our dataset and automated consistency checkers to benchmark several GDT translators. Specifically, we test four translators: three existing approaches—a direct LLM-based translator, GameInterpreter [4], and an LLM-based translator that uses the Model Context Protocol (MCP) to communicate with `pygambit` functions and Python math solvers—and one new approach that we introduce, which decomposes EFG construction into tree-form construction and the assignment of information sets and payoffs (details in Appendix C). Although this improvement is conceptually straightforward, we find that it substantially improves the performance of GDT compared with existing baselines.

In Table 1, we present results on 27 games from our dataset. We observed that our method achieves full success on most games and maintains substantially higher accuracy even in the few cases where it does not fully succeed. This advantage is especially clear for large games or those with complex information structures, such as `monty hall problem` and `memory one game`. We attribute this trend to the limited capacity of an LLM to process all relevant information in a single pass, as done in prior work, which is especially prominent for large games with imperfect information. By contrast, the decomposition in our approach creates manageable subproblems for LLMs that are substantially easier for LLMs to solve.

Since the EFGs generated by the translators include both consistent and inconsistent ones, this benchmarking exercise also demonstrates the ability of our checkers to detect both scenarios accurately. The runtime of checking an EFG takes anywhere from a few seconds to several dozen seconds. We provide these consistent and inconsistent EFGs as well as our checkers in the code for reproducibility.

7 LIMITATIONS AND FURTHER WORK

A limitation of our approach is that human expertise and labor are required to construct the ground truth, including reference games and additional data, for each translation instance. However, expert curation is standard and often unavoidable in creating high-quality AI/ML datasets, since evaluation depends on such ground truth and cannot be made fully automatic. ImageNet is a prominent example of such expert-curated data, having helped unlock major advances in computer vision.

In this work, we have developed the methodological and practical foundations for performing and evaluating game description translation to EFGs. Our initial dataset is already a valuable resource for developing, training, and testing translators for the GDT problem. Equally valuable are the data specification and the implementation of automated checkers, both of which are fully open source and extensible by us and the broader community. We believe the carefully selected examples in this initial dataset provide a useful foundation for others to build new translation instances. Going forward, we will work with the community to expand the dataset and develop more advanced automated checkers capable of handling more ambiguous natural-language descriptions.

REFERENCES

[1] Carlos Alós-Ferrer and Klaus Ritzberger. 2016. *The Theory of Extensive Form Games*. Heidelberg: Springer Berlin.

[2] Pierpaolo Battigalli, Paolo Leonetti, and Fabio Maccheroni. 2020. Behavioral equivalence of extensive game structures. *Games and Economic Behavior* 121 (2020), 533–547.

[3] Constantinos Daskalakis, Ian Gemp, Yanchen Jiang, Renato Paes Leme, Christos Papadimitriou, and Georgios Piliouras. 2024. Charting the Shapes of Stories with Game Theory. In *Proceedings of the Advances in Neural Information Processing Systems (Neurips)*.

[4] Shilong Deng, Yongzhao Wang, and Rahul Savani. 2025. From Natural Language to Extensive-Form Game Representations. In *34th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

[5] Susan Elmes and Philip J Reny. 1994. On the strategic equivalence of extensive form games. *Journal of Economic Theory* 62, 1 (1994), 1–23.

[6] Michael Genesereth, Nathaniel Love, and Barney Pell. 2005. General game playing: Overview of the AAAI competition. *AI magazine* 26, 2 (2005), 62–62.

[7] Christopher Hahn, Frederik Schmitt, Julia J Tillman, Niklas Metzger, Julian Siber, and Bernd Finkbeiner. 2022. Formal Specifications from Natural Language. *arXiv preprint arXiv:2206.01962* (2022).

[8] Herman Kahn. 1955. Use of Different Monte Carlo Sampling Techniques. *Symposium on Monte Carlo Methods* (1955), 146–190.

[9] Elon Kohlberg and Jean-Francois Mertens. 1986. On the strategic stability of equilibria. *Econometrica: Journal of the Econometric Society* (1986), 1003–1037.

[10] Jakub Kowalski, Maksymilian Mika, Jakub Sutowicz, and Marek Szykuła. 2019. Regular boardgames. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1699–1706.

[11] Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Jordi Grau-Moya, Marc Lanctot, Atil Iscen, et al. 2025. Code World Models for General Game Playing. *arXiv preprint arXiv:2510.04542* (2025).

[12] Agnieszka Mensfelt, Kostas Stathis, and Vince Trencsenyi. 2024. Autoformalization of Game Descriptions using Large Language Models. *arXiv preprint arXiv:2409.12300* (2024).

[13] Agnieszka Mensfelt, Kostas Stathis, and Vince Trencsenyi. 2024. Autoformalizing and Simulating Game-Theoretic Scenarios using LLM-augmented Agents. *arXiv preprint arXiv:2412.08805* (2024).

[14] Eric Piette, Dennis JNJ Soemers, Matthew Stephenson, Chiara F Sironi, Mark HM Winands, and Cameron Browne. 2019. Ludii–The Ludemic General Game System. *arXiv preprint arXiv:1905.05013* (2019).

[15] Rahul Savani and Theodore L. Turocy. 2026. *Gambit: The package for computation in game theory*. https://www.gambit-project.org Version 16.6.0.

[16] Tsunehiko Tanaka and Edgar Simo-Serra. 2024. Grammar-based game description generation using large language models. *IEEE Transactions on Games* (2024).

[17] Michael Thielscher. 2010. A general game description language for incomplete information games. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24. 994–999.

[18] Michael Thielscher. 2011. The general game playing description language is universal. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Vol. 22. 1107.

[19] Michael Thielscher. 2017. GDL-III: A Description Language for Epistemic General Game Playing.. In *IJCAI*. 1276–1282.

[20] Frederick B Thompson. 1952. *Equivalence of games in extensive form*. Rand Corporation.

[21] Bernhard von Stengel. 2021. *Game Theory Basics*. Cambridge University Press.

[22] Hongqiu Wu, Yan Wang, Xingyuan Liu, Hai Zhao, and Min Zhang. 2024. Instruction-driven game engines on large language models. *arXiv preprint arXiv:2404.00276* (2024).

A KUHN POKER(2-PLAYER 3-CARD)-A FULL GAME TREE

This is a two-player card game between Player One and Player Two, using a deck of three cards: King, Queen, and Jack (King beats Queen beats Jack).

A single move of chance draws an ordered pair of cards uniformly at random and assigns them to Player One and Player Two respectively; the third card is set aside. Each player observes their own card and all actions taken by either player, but neither observes their opponent's card nor the undealt third card.

Player One moves first and chooses either to check or to bet. If Player One bets, Player Two then chooses either to fold or to call. If Player One checks, Player Two chooses either to check or to bet; if Player Two bets, Player One then chooses either to fold or to call.

The game ends in one of three ways. If both players check, the player with the higher-ranked card receives 1 and the other receives -1. If one player bets and the other folds, the betting player receives 1 and the folding player receives -1. If one player bets and the other calls, the player with the higher-ranked card receives 2 and the other receives -2.

B GAME DESCRIPTIONS

B.1 bagwell-a

Consider next the noisy-leader game. In this game, a pure strategy for Player 1 is simply an action, a1 in C, S.

Let the signal received by Player 2 be denoted by phi, and assume for simplicity that phi either takes value C or S. The signal technology works as follows:

Prob(phi = S | S) = 1 - epsilon = Prob(phi = C | C)

In other words, when Player 1 chooses a particular action, the probability that Player 2 will observe a signal specifying that same action is 1 - epsilon. Take epsilon = 0.01. Player 2 observes a signal, either C or S, and then chooses an action from C, S.

Each player's payoff is determined by the realised action profile and does not depend on which signal Player 2 observed.

Player 1's best outcome is when they play C while Player 2 plays S. Their second-best outcome is when both play S, followed by both playing C. Player 1's worst outcome is when they play S while Player 2 plays C.

Player 2's best outcome is when both play C, followed by when Player 1 plays C while Player 2 plays S. The outcome where both play S is third, and Player 2's worst outcome is when they play C while Player 1 plays S.

B.2 bagwell-b

Consider next the noisy-leader game. In this game, a pure strategy for Player 1 is simply an action, a1 in C, S.

Let the signal received by Player 2 be denoted by phi, and assume for simplicity that phi either takes value C or S. The signal technology works as follows:

Prob(phi = S | S) = 1 - epsilon = Prob(phi = C | C)

In other words, when Player 1 chooses a particular action, the probability that Player 2 will observe a signal specifying that same action is 1 - epsilon. Take epsilon = 0.01. Player 2 observes a signal, either C or S, and then chooses an action from C, S.

Each player's payoff is determined by the realised action profile and does not depend on which signal Player 2 observed.

Player 1's best outcome is when they play C while Player 2 plays S. Their second-best outcome is when both play S, followed by both playing C. Player 1's worst outcome is when they play S while Player 2 plays C.

Player 2's best outcome is when both play C, followed by when Player 1 plays C while Player 2 plays S. The outcome where both play S is third, and Player 2's worst outcome is when they play C while Player 1 plays S.

Additionally, Player 1 receives a payoff of 3 in their worst outcome.

B.3 bagwell-c

Consider next the noisy-leader game. In this game, a pure strategy for Player 1 is simply an action, a1 in C, S.

Let the signal received by Player 2 be denoted by phi, and assume for simplicity that phi either takes value C or S. The signal technology works as follows:

Prob(phi = S | S) = 1 - epsilon = Prob(phi = C | C)

In other words, when Player 1 chooses a particular action, the probability that Player 2 will observe a signal specifying that same action is 1 - epsilon. Take epsilon = 0.01. Player 2 observes a signal, either C or S, and then chooses an action from C, S.

Each player's payoff is determined by the realised action profile and does not depend on which signal Player 2 observed.

Player 1's best outcome is when they play C while Player 2 plays S. Their second-best outcome is when both play S, followed by both playing C. Player 1's worst outcome is when they play S while Player 2 plays C.

Player 2's best outcome is when both play C, followed by when Player 1 plays C while Player 2 plays S. The outcome where both play S is third, and Player 2's worst outcome is when they play C while Player 1 plays S.

When Player 1 plays S and Player 2 plays C, Player 1 receives a payoff of 3 and Player 2 receives a payoff of 1.

B.4 centipede(30 rounds)

Consider two players, Alice and Bob, seated across a table from each other. Alice moves first.

At the start of the game, two piles of coins sit on Alice's side of the table: one pile contains 4 coins and the other contains 1 coin.

On each turn, the active player, the player on whose side the piles currently sit, chooses between two actions. They may "take", keeping the larger pile and giving the smaller pile to the other player, in which case the game ends immediately; or they may "push", sliding both piles across the table to the other player, who then becomes the active player. Each push doubles both piles. Each player observes all previous moves.

For instance, if Alice takes on her very first move she gets 4 coins and Bob gets 1. If instead Alice pushes and Bob then takes, Bob gets 8 and Alice gets 2; one push later, if Alice takes, she gets 16 and Bob gets 4.

The game lasts 60 moves at most: Alice makes the odd-numbered moves (1, 3,..., 59) and Bob makes the even-numbered moves (2, 4,..., 60). If no one ever takes, Alice obtains the larger pile and Bob gets the smaller one.

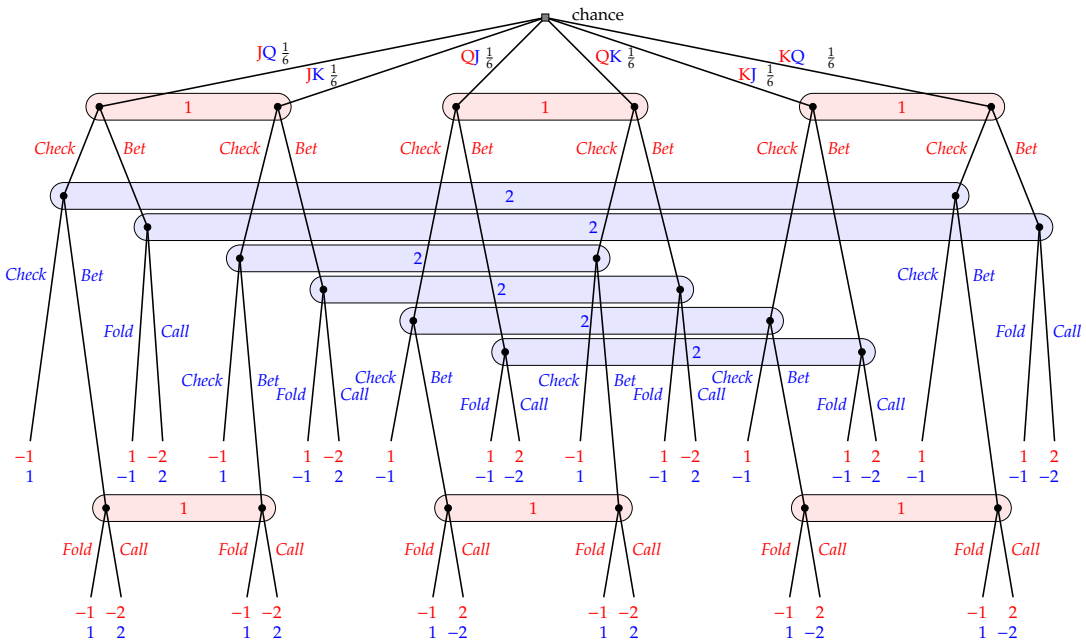


Figure 3: Full Kuhn poker.

The game lasts 60 moves at most: Alice makes the odd-numbered moves (1, 3, 5, . . . , 59) and Bob makes the even-numbered moves (2, 4, . . . , 60). If no one ever takes, Alice obtains the larger pile and Bob gets the smaller one.

B.5 gambling-a

A gambler decides whether to enter a casino.

If the gambler stays out, the dealer takes no action and both receive 0. If the gambler enters, they play a coin toss game against the dealer: both players simultaneously choose either Heads or Tails.

If the choices match, the gambler receives 1 and the dealer receives -1; otherwise, the dealer receives 1 and the gambler receives -1.

B.6 gambling-b

A gambler decides whether to enter a casino.

If the gambler stays out, the dealer takes no action and both receive 0. If the gambler enters, the gambler first chooses either Heads or Tails, and the dealer then chooses either Heads or Tails without observing the gambler's choice.

If the choices match, the gambler receives 1 and the dealer receives -1; otherwise, the dealer receives 1 and the gambler receives -1.

B.7 gambling-c

A gambler decides whether to enter a casino.

If the gambler stays out, the dealer takes no action and both receive 0. If the gambler enters, the dealer first chooses either Heads or Tails, and the gambler then chooses either Heads or Tails without observing the dealer's choice.

If the choices match, the gambler receives 1 and the dealer receives -1; otherwise, the dealer receives 1 and the gambler receives -1.

B.8 kuhn poker(2-player 3-card)-a

This is a two-player card game between Player One and Player Two, using a deck of three cards: King, Queen, and Jack (King beats Queen beats Jack).

A single move of chance draws an ordered pair of cards uniformly at random and assigns them to Player One and Player Two respectively; the third card is set aside. Each player observes their own card and all actions taken by either player, but neither observes their opponent's card nor the undealt third card.

Player One moves first and chooses either to check or to bet. If Player One bets, Player Two then chooses either to fold or to call. If Player One checks, Player Two chooses either to check or to bet; if Player Two bets, Player One then chooses either to fold or to call.

The game ends in one of three ways. If both players check, the player with the higher-ranked card receives 1 and the other receives -1. If one player bets and the other folds, the betting player receives 1 and the folding player receives -1. If one player bets and the other calls, the player with the higher-ranked card receives 2 and the other receives -2.

B.9 kuhn poker(2-player 3-card)-b

This is a two-player card game between Player One and Player Two, using a deck of three cards: King, Queen, and Jack (King beats Queen beats Jack).

Before any player acts, chance deals the cards in two stages: In the first stage, Player One is dealt a card uniformly at random from the three cards; In the second stage, Player Two is dealt a card uniformly at random from the remaining two. Each player observes their own card and all actions taken by either player, but neither observes their opponent's card nor the undealt third card.

Player One moves first and chooses either to check or to bet. If Player One bets, Player Two then chooses either to fold or to call. If Player One checks, Player Two chooses either to check or to bet; if Player Two bets, Player One then chooses either to fold or to call.

The game ends in one of three ways. If both players check, the player with the higher-ranked card receives 1 and the other receives -1. If one player bets and the other folds, the betting player receives 1 and the folding player receives -1. If one player bets and the other calls, the player with the higher-ranked card receives 2 and the other receives -2.

B.10 kuhn poker(2-player 3-card)-c

This is a two-player card game between Player One and Player Two, using a deck of three cards: King, Queen, and Jack (King beats Queen beats Jack).

Chance deals the cards in two stages. In the first stage, Player One is dealt a card uniformly at random from the three cards. In the second stage, after Player One's first decision, Player Two is dealt a card uniformly at random from the remaining two. Each player observes their own card and all actions taken by either player, but neither observes their opponent's card nor the undealt third card.

Player One moves first and chooses either to check or to bet. If Player One bets, Player Two then chooses either to fold or to call. If Player One checks, Player Two chooses either to check or to bet; if Player Two bets, Player One then chooses either to fold or to call.

The game ends in one of three ways. If both players check, the player with the higher-ranked card receives 1 and the other receives -1. If one player bets and the other folds, the betting player receives 1 and the folding player receives -1. If one player bets and the other calls, the player with the higher-ranked card receives 2 and the other receives -2.

B.11 kuhn poker(2-player 4-card)

This is a two-player card game between Player One and Player Two, using a deck of four cards: Ace, King, Queen, and Jack (Ace beats King beats Queen beats Jack).

A single move of chance draws an ordered pair of cards uniformly at random and assigns them to Player One and Player Two respectively; the other two cards are set aside. Each player observes their own card and all actions taken by either player, but neither observes their opponent's card nor the two undealt cards.

Player One moves first and chooses either to check or to bet. If Player One bets, Player Two then chooses either to fold or to call. If Player One checks, Player Two chooses either to check or to bet; if Player Two bets, Player One then chooses either to fold or to call.

The game ends in one of three ways. If both players check, the player with the higher-ranked card receives 1 and the other receives -1. If one player bets and the other folds, the betting player receives 1 and the folding player receives -1. If one player bets and the other calls, the player with the higher-ranked card receives 2 and the other receives -2.

B.12 market signalling game

A new manufacturer plans to enter the market. Chance determines its type: strong with probability 2/3 and weak with probability 1/3. The new manufacturer observes its own type and then chooses a signal, either strong or weak, to send to the current manufacturer. The current manufacturer observes the signal but not the new manufacturer's actual type, and chooses either to fight or to accommodate. The payoffs to the new and current manufacturers respectively are as follows.

If the new manufacturer is strong and signals strong: fight yields (1, 0), accommodate yields (3, 1).

If the new manufacturer is strong and signals weak: fight yields (0, 0), accommodate yields (2, 1).

If the new manufacturer is weak and signals strong: fight yields (0, 2), accommodate yields (2, 1).

If the new manufacturer is weak and signals weak: fight yields (1, 2), accommodate yields (3, 1).

B.13 memory one game

Consider a game played between two players, A and B, over six stages. Player A acts at the odd-numbered stages and Player B at the even-numbered stages; at every stage, the acting player selects an action from the set X, Y. The six stages are grouped into three rounds: round 1 comprises stages 1 and 2, round 2 stages 3 and 4, and round 3 stages 5 and 6.

Neither player observes the other's actions. The players also have imperfect recall; specifically, when acting, a player's information consists only of the current stage number and their own most recent action (if any).

At the end of stage 6, payoffs are determined round by round. In each round, if both players selected the same action that round, each receives 1 point for that round; if their actions differed, each receives 0. A player's final payoff is the sum of points across the three rounds.

B.14 monty hall problem

This is a single-player decision problem. There are three closed doors, numbered 1, 2, and 3: behind one is a car, and behind each of the other two is a goat.

Nature first chooses uniformly at random which door hides the car. Without observing nature's choice, the player then chooses one of the three doors.

After the player's first choice, the lowest-numbered unchosen door that hides a goat is opened to reveal the goat. The player observes which door was opened. So now there is one chosen door, one opened door, and a final remaining unopened door. Then, remembering their own first choice, the player chooses between the originally chosen door and the remaining unopened door.

If the car is behind the player's final choice, the player receives 1; otherwise, the player receives 0.

B.15 nim(2,2,1)

Consider a game played between two players, Alice and Bob, that begins with three piles containing 2, 2, and 1 stones, respectively. Alice moves first, and the players alternate turns. Each player observes all previous moves.

On their turn, the acting player chooses a non-empty pile and removes either one stone, or two stones if the pile contains at least two.

The game ends when all piles are empty. The player who removes the final stone wins, receiving a payoff of 1; the other player receives -1.

B.16 nuclear crisis-a

Consider a multi-stage game played between Country A and Country B.

Country A moves first, choosing whether to issue a threat or to ignore the provocation. If Country A ignores, the game ends. If Country A issues a threat, Country B observes this and chooses whether to back down or to issue a counter-threat. If Country B backs down, the game ends. If Country B counter-threatens, the game enters a crisis stage in which both countries simultaneously choose to retreat or to detonate. After the countries make their choices in the crisis stage, the game ends.

The terminal outcomes of the game are:

- "ignore": Country A ignores the provocation;
- "B backs down": Country A threatens and Country B backs down;
- "A wins crisis": in the crisis, Country A detonates and Country B retreats;
- "B wins crisis": in the crisis, Country B detonates and Country A retreats;
- "mutual retreat": in the crisis, both countries retreat;
- "mutual detonation": in the crisis, both countries detonate.

Country A's preferences, from most to least preferred, are: "A wins crisis", "B backs down", "ignore", "mutual retreat", "B wins crisis", "mutual detonation".

Country B's preferences, from most to least preferred, are: "B wins crisis", "ignore", then "B backs down" and "mutual retreat" tied, then "A wins crisis", then "mutual detonation".

B.17 nuclear crisis-b

Consider a multi-stage game played between Country A and Country B.

Country A moves first, choosing whether to issue a threat or to ignore the provocation. If Country A ignores, the game ends. If Country A issues a threat, Country B observes this and chooses whether to back down or to issue a counter-threat. If Country B backs down, the game ends. If Country B counter-threatens, the game enters a crisis stage in which both countries simultaneously choose to retreat or to detonate. After the countries make their choices in the crisis stage, the game ends.

The terminal outcomes of the game are:

- "ignore": Country A ignores the provocation;
- "B backs down": Country A threatens and Country B backs down;
- "A wins crisis": in the crisis, Country A detonates and Country B retreats;
- "B wins crisis": in the crisis, Country B detonates and Country A retreats;
- "mutual retreat": in the crisis, both countries retreat;
- "mutual detonation": in the crisis, both countries detonate.

Country A's preferences, from most to least preferred, are: "A wins crisis", "B backs down", "ignore", "mutual retreat", "B wins crisis", "mutual detonation".

Country B's preferences, from most to least preferred, are: "B wins crisis", "ignore", then "B backs down" and "mutual retreat" tied, then "A wins crisis", then "mutual detonation".

Both countries receive a payoff of -100 at "mutual detonation".

B.18 nuclear crisis-c

Consider a multi-stage game played between Country A and Country B.

Country A moves first, choosing whether to issue a threat or to ignore the provocation. If Country A ignores, the game ends. If Country A issues a threat, Country B observes this and chooses whether to back down or to issue a counter-threat. If Country B backs down, the game ends. If Country B counter-threatens, the game enters a crisis stage in which both countries simultaneously choose to retreat or to detonate. After the countries make their choices in the crisis stage, the game ends.

The terminal outcomes of the game are:

- "ignore": Country A ignores the provocation;
- "B backs down": Country A threatens and Country B backs down;
- "A wins crisis": in the crisis, Country A detonates and Country B retreats;
- "B wins crisis": in the crisis, Country B detonates and Country A retreats;
- "mutual retreat": in the crisis, both countries retreat;
- "mutual detonation": in the crisis, both countries detonate.

Country A's preferences, from most to least preferred, are: "A wins crisis", "B backs down", "ignore", "mutual retreat", "B wins crisis", "mutual detonation".

Country B's preferences, from most to least preferred, are: "B wins crisis", "ignore", then "B backs down" and "mutual retreat" tied, then "A wins crisis", then "mutual detonation".

Both countries receive a payoff of -100 at "mutual detonation", and Country A receives a payoff of 10 at "A wins crisis".

B.19 russian roulette-a

Two players alternate turns with a six-chamber revolver. Before any player acts, chance first determines which one of the six chambers contains the bullet, each with probability 1/6. This chamber remains fixed for the entire game, but neither player observes which chamber was selected.

On each turn, a player chooses either to quit, and live on for sure, or to pull the trigger, and potentially kill himself. If he quits, he withdraws and the other player wins. If he pulls the trigger and the current chamber is the loaded one, he shoots himself and the other player wins; otherwise, play passes to the other player.

If shooting himself, a player gets -1. If quitting, a player gets 0. If winning, a player gets 1.

B.20 russian roulette-b

Two players alternate turns with a six-chamber revolver, with Player 1 acting first. On each turn, a player chooses either to quit, and live on for sure, or to pull the trigger, and potentially kill himself. Each trigger pull and its outcome are common knowledge before the next move. The chambers are numbered 1 through 6 according to the order in which they would be fired if players keep pulling the trigger.

If the player quits, he withdraws and the other player wins. If he pulls the trigger, chance determines whether the gun fires; the firing probability equals one divided by the number of chambers that have not yet been pulled. If he pulls the trigger and the current chamber is the loaded one, he shoots himself and the other player wins; otherwise, play passes to the other player.

If shooting himself, a player gets -1. If quitting, a player gets 0. If winning, a player gets 1.

B.21 russian roulette-c

Two players alternate turns with a six-chamber revolver, with Player 1 acting first. On each turn, a player chooses either to quit, and live on for sure, or to pull the trigger, and potentially kill himself. Each trigger pull and its outcome are common knowledge before the next move.

If the player quits, he withdraws and the other player wins. If he pulls the trigger and more than one chamber remains unpulled, chance determines the outcome: the gun fires with probability one over the number of remaining chambers, and if the gun does not fire play passes to the other player. If only one chamber remains, the trigger pull fires for certain. When the gun fires, the acting player shoots himself and the other player wins.

If shooting himself, a player gets -1. If quitting, a player gets 0. If winning, a player gets 1.

B.22 tic-tac-toe endgame

Consider a game played between two players, X and O, who alternate turns on a 3x3 grid, marking empty spaces with "x" and "o" respectively.

The game ends immediately when a player places three of their marks in a horizontal, vertical, or diagonal row; that player wins the game. The winner receives a payoff of 1, and the loser receives a payoff of -1. If the grid is completely filled without a winner, the game is a draw and both players receive a payoff of 0.

Assume the game begins from the following board state, where "e" represents an empty space and the outer numbers represent the coordinate axes:

```
    0   1   2
0 | e | x | e |
1 | x | e | o |
2 | x | e | o |
```

Player O is the next to move.

B.23 voting game-a

Player 1 and Player 2 vote on three candidates, A, B, and C. Each player's ballot is a strict ordering of the three candidates, and the two players submit their ballots simultaneously.

Ballots are scored by Borda count:

- a candidate receives two points each time it is ranked first on a ballot,

- one point each time it is ranked second, and

- zero points each time it is ranked third.

The candidate with the highest total score wins. If two or more candidates are tied for the highest score, the tie is broken according to the priority order A, B, C. Among the tied candidates, the one appearing earliest in this order wins.

Each player's payoff depends only on which candidate wins, not on which ballots were submitted to produce that win. Player 1 strictly prefers A winning to B winning, and B winning to C winning. Player 2 strictly prefers C winning to B winning, and B winning to A winning.

B.24 voting game-b

There are two players and three candidates, A, B, and C. Each player's ballot is a strict ordering of the three candidates, and the two players submit their ballots simultaneously.

Ballots are scored by Borda count:

- a candidate receives two points each time it is ranked first on a ballot,

- one point each time it is ranked second, and

- zero points each time it is ranked third.

The candidate with the highest total score wins. If two or more candidates are tied for the highest score, the tie is broken according to the priority order C, B, A. Among the tied candidates, the one appearing earliest in this order wins.

Each player's payoff depends only on which candidate wins, not on which ballots were submitted to produce that win. Player 1 strictly prefers A winning to B winning, and B winning to C winning. Player 2 strictly prefers C winning to B winning, and B winning to A winning.

B.25 voting game-c

There are two players and three candidates, A, B, and C. Each player's ballot is a strict ordering of the three candidates, and the two players submit their ballots simultaneously.

Ballots are scored by Borda count:

- a candidate receives two points each time it is ranked first on a ballot,

- one point each time it is ranked second, and

- zero points each time it is ranked third.

The candidate with the highest total score wins. If two or more candidates are tied for the highest score, the tie is broken according to the priority order A, B, C. Among the tied candidates, the one appearing earliest in this order wins.

Each player's payoff depends only on which candidate wins, not on which ballots were submitted to produce that win. Player 1 strictly prefers A winning to B winning, and B winning to C winning. Player 2 strictly prefers C winning to B winning, and B winning to A winning.

Player 1's payoff from C winning is 0.

B.26 voting game-d

There are two players and three candidates, A, B, and C. Each player's ballot is a strict ordering of the three candidates, and the two players submit their ballots simultaneously.

Ballots are scored by Borda count:

- a candidate receives two points each time it is ranked first on a ballot, - one point each time it is ranked second, and

- zero points each time it is ranked third.

The candidate with the highest total score wins. If two or more candidates are tied for the highest score, the tie is broken according

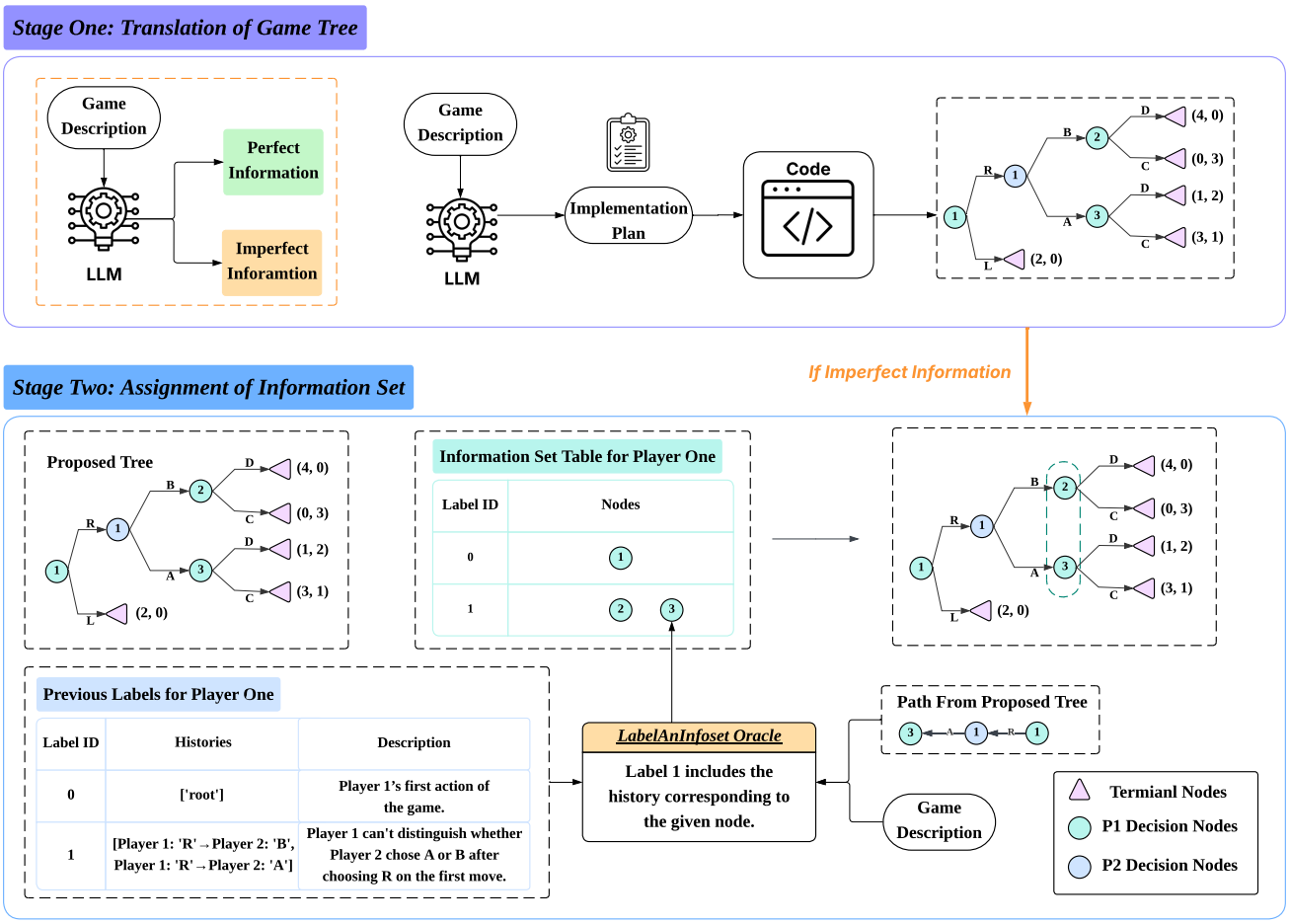


Figure 4: The illustration of our improvement for game description translation.

to the priority order A, B, C. Among the tied candidates, the one appearing earliest in this order wins.

Each player's payoff depends only on which candidate wins, not on which ballots were submitted to produce that win. Player 1 strictly prefers A winning to C winning, and C winning to B winning. Player 2 strictly prefers B winning to C winning, and C winning to A winning.

B.27 voting game-e

Player 1 and Player 2 vote on three candidates, A, B, and C. Each player's ballot is a strict ordering of the three candidates. Player 2 votes first; Player 1 votes afterward without knowing Player 2's choice.

Ballots are scored by Borda count: - a candidate receives two points each time it is ranked first on a ballot, - one point each time it is ranked second, and - zero points each time it is ranked third.

The candidate with the highest total score wins. If two or more candidates are tied for the highest score, the tie is broken according to the priority order A, B, C. Among the tied candidates, the one appearing earliest in this order wins.

Each player's payoff depends only on which candidate wins, not on which ballots were submitted to produce that win. Player 1 strictly prefers A winning to B winning, and B winning to C winning. Player 2 strictly prefers C winning to B winning, and B winning to A winning.

C AN IMPROVED APPROACH FOR GAME DESCRIPTION TRANSLATION

We present a straightforward improvement for translating natural language game descriptions into EFGs. We decompose the translation process into two stages: (i) the construction of the perfect-information game tree (with payoffs); and (ii) partitioning the decision nodes into information sets if the game has imperfect information. The workflow is illustrated in Figure 4.

In the first stage, an LLM is tasked with generating Gambit code that constructs a perfect-information game tree with payoffs. We encourage the LLM to build the tree efficiently using loops or recursion, which leads to more concise code for large games with recursive structures, such as `centipede` or `nim`. This approach also motivates

constructing the game tree node(s) by node(s), thereby decomposing the task into manageable subproblems that are substantially easier for LLMs to solve.

In the second stage, we use LLMs to iteratively assign each node to an information set. For some games, nodes within the same information set can be represented by a short natural-language description. A new node can then be classified either into an existing information set whose description has the same semantic meaning, or into a newly created information set with a new descriptive label. For games in which such descriptive labels are not available, we instead use the set of action histories as the label for an information set, corresponding to the formal representation of information sets in game theory. We then determine whether the history leading to the current node appears in any existing label.