文法を基いた逐次選択アプローチによるゲーム記述生成

田中 恒彦 [1] シモセラ エドガー [1]
[1]Waseda University
tsunehiko@fuji.waseda.jp ess@waseda.jp

概要

本研究は，Game Description Language(GDL) を基にしたゲーム記述生成において，文法的正確性を改善する新しいプロセスを提案する．GDL は多様なゲームを統一的に表現できるドメイン固有言語であり，ゲームデザインの自動化に広く利用されている．本研究では，大規模言語モデル (LLM) を活用し，文法に基づいた生成候補を逐次提示するアプローチを開発した．さらに，部分出力と文法的に妥当な候補を含むデータセットを構築し，LLM を教師ありファインチューニングで学習させた．その結果，提案手法は生成されたゲーム記述の文法的に正確さにおいて従来手法を上回る性能を示した。

1 はじめに

Game Description Language(GDL) [1, 2, 3] は，多様なゲームを統一的な表記法で表現するために開発されたドメイン固有言語である．たとえば，Ludii GDL [3] はボードゲームを中心に 1,000 種類以上のゲームをモデル化している．図 1 はゲーム Tic-Tac-Toe の Ludii GDL で書かれたゲーム記述である．GDL で記述されたゲームは機械可読性が高く，専用のゲームエンジンを使ってゲームプレイをシミュレーションしやすいという利点がある．このような特性から，GDL はゲームの自動評価に適しており，自動ゲームデザインの研究 [4, 5, 6] において広く利用されてきた．近年では，大規模言語モデル (LLM) を活用し，自然言語テキストから直接ゲーム記述を生成するタスク (Game Description Generation; GDG) に注目が集まっている [7, 8]．自然言語を生成の起点とすることで，ゲームの専門知識を持たない人でも容易にゲームデザインに取り組めるようになることが期待されている．

GDL のように，特別な構造をもつ出力を生成するためには，LLM にドメイン知識を学習させるアプローチが効果的である．その一つの方法として，

***x*：**
Description :
Tic-Tac-Toe is a game of alignment popular among children. It is known from the nineteenth century in England and the United States, but may be older.
Rules :
Play occurs on a 3x3 grid. One player places an X, the other places an O and players take turns placing their marks in the grid, attempting to get three in a row of their colour.

***y*：**
```
(game "Tic-Tac-Toe"
   (players 2)
   (equipment
      {
         (board (square 3))
         (piece "Disc" P1)
         (piece "Cross" P2)
      }
   )
   (rules
      (play (move Add (to (sites Empty))))
      (end (if (is Line 3) (result Mover Win)))
   )
)
```

図 1 ゲーム「Tic-Tac-Toe」に対する Ludii ゲーム記述の例. *x* は自然言語でゲームを説明する文章であり，*y* は Ludii GDL（Game Description Language）によるゲーム記述である．

LLM の In-Context Learning(ICL) [9] を用い，LLM のプロンプトにタスクの文脈情報を含める手法がある．たとえば，いくつかの研究 [7, 8] は，GDL の表記法に関する説明とゲーム記述生成の例をプロンプトに組み込み，ICL を使ったゲーム記述生成を試みた．もう一つの方法として，ゲームを説明する自然言語テキストとゲーム記述のペアデータを用いて LLM に教師ありファインチューニング (Supervised Fine-Tuning; SFT) を施すアプローチも提案されている [8]．これらの研究成果からは，LLM の高い推論能力を活用すれば，より高精度なゲーム記述を生成できる可能性が示唆されている．しかし，LLM が生成するゲーム記述は依然として文法的に不正確な場合があり [8]，そうした不正確な記述はゲームエンジンで正しく解析・読み込みができず，シミュレーションを実行できない．高品質なゲームを生成するには，まず LLM が文法的に正確なゲーム記述

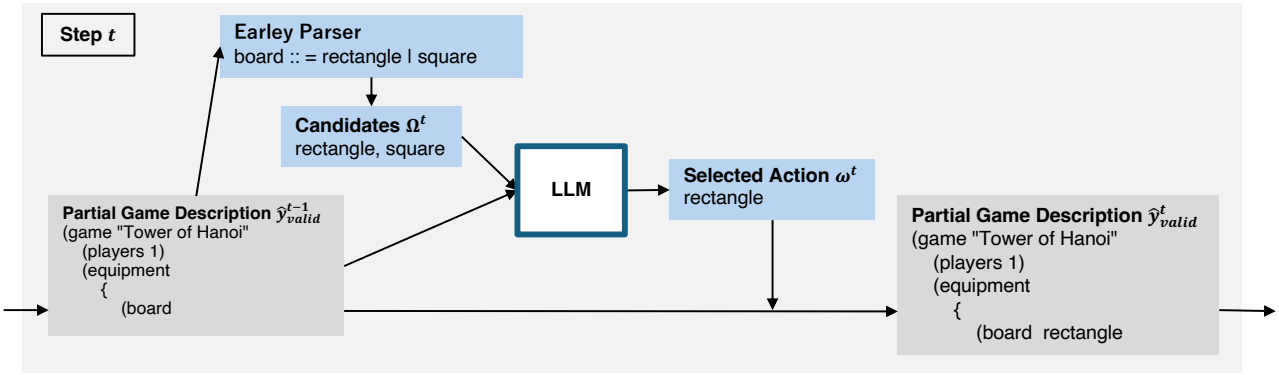

図 2　文法駆動型逐次生成法. t 番目の記号（この場合，rectangle）を生成する様子である．$\hat{y}^{t-1}_{\text{valid}}$ に続く候補記号の集合 Ω^t を Early パーサで獲得し，LLM がその中から最適な記号 ω^t を選ぶ．$\hat{y}^{t}_{\text{valid}} = \hat{y}^{t-1}_{\text{valid}} + \omega^t$ のように更新される．

を生成できるようにすることが不可欠である．

そこで本研究では，ドメイン記述言語の文法を考慮した SFT を導入する．我々は，生成プロセスの各ステップにおいて，それまでに生成された部分出力に続く文法的に妥当な候補を文法から抽出し，その中から LLM に最適な選択肢を選ばせることを考える．その設定下で，部分出力と文法的に許容される続きの候補，および正解を含むデータを用いた SFT によって，GDL のような特別な構造を持つ文法の知識を LLM に学習させることを目指す．具体的には，まずゲーム記述データから GDL 構文の基本単位となる部分ゲーム記述を取り出し，それらに対して文法パーサを使い続きの候補を抽出することでデータセットを構築する．次に，構築したデータセットを用いて LLM に SFT を施す．実験では，文法を組み込まずに自然言語テキストとゲーム記述データのペアを用いて学習した LLM と比較し，提案手法のほうがより多くのケースで GDL に必要とされる機能を満たせることを確認した．

2　準備

2.1　Ludii GDL

GDL は，ゲームを記述するためのドメイン固有言語である．Ludii GDL [3] は，対応しているゲームの数が最も多く，近年注目されている GDL である．Ludii はゲーム関連情報の概念的な単位に分解する Ludemic Apporach [10] に基づいて開発されたゲームシステムであり，1,000 種類以上の伝統的なゲームをモデル化している．この中には，ボードゲーム，カードゲーム，ダイスゲーム，タイルゲームが含まれる．Ludii の文法は拡張バッカス・ナウア形式 (Extended Backus–Naur Form, EBNF) の文脈自由文法 (Context-Free Grammar, CFG) で記述されている．本論文では，Ludii GDL を GDL として使用する．

2.2　ゲーム記述生成

G を GDL 文法とし，$L(G)$ を G によって生成されるゲーム記述の集合とする．ゲームの内容およびルールを記述する自然言語クエリ x を入力し，対応するゲーム記述 $y \in L(G)$ を生成するタスクを「ゲーム記述生成」と呼ぶ．本研究の目標は，生成されたゲーム記述 $\hat{y}$ を真のゲーム記述 y にできるだけ近づけることである．図 1 に Ludii GDL のゲーム記述生成の例を示す．以降，特に指定がない限り，G は Ludii の文法を表すものとする．

3　文法駆動型逐次生成法

3.1　生成プロセス

本セクションでは，文法制約を満たした出力のための文法駆動型逐次生成法 (Iterative Grammar-Guided Generation; IG3) を説明する．図 2 に示すように，我々は，生成プロセスの各ステップにおいて，それまでに生成された部分出力に続く文法的に妥当な候補を文法から抽出し，その中から LLM に最適な選択肢を選ばせることを考える．我々はまず GDL 文法 G に基づく Earley パーサ [11] を構築する．Earley パーサは LLM によって生成されたゲーム記述 $\hat{y}$ の中から文法的に有効な部分列 $\hat{y}_{\text{valid}}$ とその後に続く候補記号の集合 Ω を決定する．$\hat{y}_{\text{valid}}$ は $\hat{y}$ の先頭から取り出せる最も長い文法的に妥当な部分列である．

生成プロセスの各ステップ t では，LLM が候補記号の集合 Ω の中から最適な記号 ω を選択する．これを T ステップ繰り返す．T は生成ステップの反復

回数の上限であり，計算資源や予算によって決定されるものである．ステップ 0 では，開始記号が与えられる．Ludii GDL の場合，開始記号は「(game」である．ステップ t では，ステップ $t-1$ までに生成した文法的に妥当な部分列 $\hat{y}^{t-1}_{valid}$ とステップ t の候補記号の集合 Ω^t が LLM に与えられる．LLM は Ω^t の中から最適な記号 ω^t を生成する．選ばれた記号を使って部分列は $\hat{y}^t_{\text{valid}} = \hat{y}^{t-1}_{\text{valid}} + \omega^t$ のように更新される．これを候補の集合 Ω^t が出現しなくなる，または，反復回数が T に達するまで繰り返す．終了時の $\hat{y}^t_{\text{valid}}$ がモデルの出力 $\hat{y}$ となる．

3.2 データセット構築

我々は 3.1 で説明した推論方法に適した LLM を訓練するためのデータセットを構築する．Ludii ポータルサイトでは，ゲームを説明する自然言語テキスト x とゲーム記述 y のペアが提供されている．我々はゲーム記述 y から部分ゲーム記述 y_{part} とその直後に続く記号 ω を取り出す．y_{part} に Earley パーサを適用し，その続きの候補記号の集合 Omega を得る．そして，y_{part}, Ω を入力，ω を正解とするデータセットを構築する．我々はこのデータセットを用いて LLM に SFT を適用する．なお，本論文ではゲーム記述の長さが 300 トークン以下の 152 のゲーム記述を使用して 8,532 インスタンスのデータセットを構築した．

4 実験

4.1 比較手法

我々は次のベースライン手法と我々の提案手法 IG3 を比較する．

- **GDG-ICL** [8]: SFT を適用していない LLM に，ゲーム記述生成のいくつかのデモンストレーション例を与え，1 度にプログラム全体を生成させる手法．同じゲームカテゴリの異なる 3 つのゲームについて (x, y) のペアをプロンプトに含める．
- **GDG-SFT** [8]: クエリ x とゲーム記述 y のペアデータで SFT を適用した LLM で，1 度にプログラム全体を生成させる手法．訓練データには，長さ 300 トークン以下の 152 のゲームの (x, y) を使う．
- **IG3-SFT(ours)**: セクション 3.2 で提案したデータセットを使って SFT を適用した LLM に，セクション 3.1 の生成プロセスで生成させる提案手法．

4.2 実装

本研究では，LLM として Llama3-8B-Instruct [12] を採用し，SFT を行う際に Low-Rank Adaptation(LoRA) [13] を利用した．具体的には，最大入力シーケンス長を 3072 に設定し，LoRA のパラメータ α と r はともに 16 とした．学習率は 1e-4，warmup rate は 0.03 とし，これらの設定で 3 エポックの学習を実施している．

生成過程においては，出力が文法的に妥当な記号のみからなるよう制約付き生成手法を用い，生成の反復上限 T を 50 に設定した．本研究では問題設定を簡略化するため，[8] に倣い，ゲーム記述 y を生成するのに必要最小限の文法規則のみを含む $G[y]$ を利用している．しかしながら，実際の応用場面では文法 $G[y]$ が生成時に未知である場合が多く，フルサイズの文法 G を用いた生成が必要となる．本稿ではその検証を将来の課題とし，現状では簡便化された条件下での実験結果を報告する．

4.3 評価指標

我々は生成されたゲーム記述を評価するため，次の指標を用いる．

- **Compilability**: Ludii ゲームエンジンで解析およびコンパイル可能なゲームの割合を示す．この指標では，エラーなくコンパイルされるゲームの比率を計測する．ゲームがコンパイルできない場合，次の Functionality では評価されない．このスコアは 0 から 100 まで正規化される．
- **Functionality**: プレイ可能なゲームの割合を示す．ゲームがエラーなくコンパイルされても，例えば駒の動きが未定義の位置に依存する場合など，ゲームがプレイ不能な条件にある場合は「Functionality がない」とみなされる．このスコアも 0 から 100 まで正規化される．
- **ROUGE** [14]: 生成されたゲーム記述と正解のゲーム記述の間の言語的マッチ度を測定する一般的な評価指標．プログラム合成の分野でもよく使用される [15]．この指標は 0 から 100 の範囲で，値が高いほどマッチ度が高いことを示す．この評価では構文的な正確さを考慮せず，テキストの類似性に焦点を当てる．ROUGE-L

表 1　ゲーム記述生成の比較. 最も良い結果を太字にしている.

Method	Compilability↑	Functionality↑	ROUGE↑	NCD↓
GDG-ICL	30.0±1.5	29.3±1.3	**63.3**±0.1	0.72±0.01
GDG-SFT	36.3±1.7	36.0±1.5	62.7±0.3	0.65±0.02
IG3-SFT (ours)	**47.3**±2.4	**38.7**±2.7	59.9±0.4	**0.64**±0.02

の F1 スコアを使用し，すべてのテストデータの平均値を報告する.

- **Normalized Concept Distance (NCD)**: Ludii における予測されたゲームが真のゲームとどれだけ一致しているかを測定する指標．先行研究 [16, 17] に従い，ゲームはセマンティック特徴と自動プレイアウト（ランダムポリシーを使用）から得られた行動データに基づいて，コンセプト値ベクトルとして表現される．これらのベクトル間のコサイン距離を NCD とする．非機能的なゲームはコンセプト距離を計算できないため，距離を 1.0 として設定する．NCD はすべてのテストデータで平均化され，ゲーム記述生成の品質測定として使用される.

4.4　結果

表 1 ゲーム記述生成の結果を示す．IG3-SFT は GDG-ICL および GDG-SFT と比較して，Compilability・Functionality・NCD の指標で優れた性能を示した．これは，提案した生成プロセスとデータセットに基づく学習が，文法的な正確さを効果的に向上させることを示唆している.

一方，ROUGE スコアでは IG3-SFT が最良の GDG-ICL より 3.4 ポイント低い結果となった．これは，文法を考慮しなくても，プロンプトのデモンストレーション例のみで見かけ上は類似したゲーム記述を生成できる可能性を示している．しかし，GDG-ICL の Compilability・Functionality・NCD は GDG-SFT にも及ばず，文法的な正確さを高めるには，SFT などを通して GDL の知識を LLM に与える必要があることが分かる.

さらに，Compilability では IG3-SFT が GDG-SFT を 10.0 ポイント上回っており，他の指標よりも改善幅が大きい．これは，単にテキストとゲーム記述のペアで SFT を適用するだけでなく，GDL 文法の知識を直接的に与えることで，LLM に対してより効果的な学習が可能になることを示している.

図 3 は，IG3-SFT によって生成された「ハノイの塔」のゲーム記述を示している．また，正解を付録の図 4 に示す．必要な要素が不足して破綻が見られるため，さらなる改良が必要であることが分かる．とくに赤枠で囲んだ箇所には不要な繰り返し表現が含まれ，モデルが特定の出力パターンに過度に収束している可能性を示唆している．今後は，こうした偏りを解消し，生成の多様性を確保することが課題となる.

```
(game "Tower of Hanoi"
  (players 3 )
  (equipment
    {
      (board (rectangle 3 1))
      (piece "Counter3" P1)
      (piece "Counter3" P1)
      (piece "Counter3" P1)
    }
  )
  (rules
    (start
      {(place Stack items:
          {"Counter3" "Counter3" "Counter3"} (from)
       )}
    )
    (play
      (move
        (from (sites Occupied by: P1))
        (to (sites Occupied by: P1)
          if:(or (is Empty (to)) (is Empty (to)))
        )
      )
    )
    (end
      {
        (if (is Target {(to) (to) (to)} (count Moves)) (result P1 Win))
        (if (!= (count Moves) (count Moves)) (result P1 Win))
      }
    )
  )
)
```

図 3　IG3-SFT で生成されたハノイの塔のゲーム記述. 赤枠は不要な繰り返しである.

5　おわりに

本研究では，GDL を基にした文法的に正確なゲーム記述生成を実現するための手法を提案した．文法に基づいて候補を逐次選択させるプロセスと，専用のデータセットを用いた SFT を組み合わせた結果，提案手法は従来のアプローチよりも高い文法的正確性を達成した．この成果は，LLM を活用した自動ゲームデザインの可能性を大きく拡げるものであり，今後の応用としてさらなるゲームの多様性や精度向上が期待される.

謝辞

本研究は JSPS 科研費 24KJ2099 の助成を受けたものです.

参考文献

[1] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General game playing: Game description language specification. 2008.

[2] Tom Schaul. A video game description language for model-based or interactive learning. 2013.

[3] É. Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne. Ludii – the ludemic general game system. In **Proc. of ECAI**, 2020.

[4] Cameron Browne. **Evolutionary Game Design**. Springer, 2011.

[5] Ahmed Khalifa, Michael Cerny Green, Diego Perez-Liebana, and Julian Togelius. General video game rule generation. 2017.

[6] Thomas Maurer and Matthew Guzdial. Adversarial random forest classifier for automated game design. 2021.

[7] Chengpeng Hu, Yunlong Zhao, and Jialin Liu. Game generation via large language models, 2024.

[8] Tsunehiko Tanaka and Edgar Simo-Serra. Grammar-based Game Description Generation using Large Language Models. pp. 1–14, 2024.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. 2020.

[10] David Parlett. What' sa ludeme. **Game & Puzzle Design**, Vol. 2, No. 2, pp. 81–84, 2016.

[11] Jay Earley. An efficient context-free parsing algorithm. **Commun. ACM**, Vol. 13, No. 2, p. 94–102, 1970.

[12] Introducing meta llama 3: The most capable openly available llm to date. 2024.

[13] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. 2022.

[14] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In **Proc. of Text Summarization Branches Out**, 2004.

[15] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation, 2024.

[16] Eric Piette, Matthew Stephenson, Dennis J.N.J. Soemers, and Cameron Browne. General board game concepts. 2021.

[17] Matthew Stephenson, Dennis J. N. J. Soemers, Éric Piette, and Cameron Browne. Measuring board game distance. In **Proc. of Computer and Games**, 2022.

```
(game "Tower of Hanoi"
   (players 1)
   (equipment
      {
         (board (rectangle 1 3))
         (piece "Counter3" P1)
         (piece "Counter6" P1)
         (piece "Counter9" P1)
      }
   )
   (rules
      (start { (place Stack items:{ "Counter9" "Counter6" "Counter3"} 0) })
      (play
         (move
            (from (sites Occupied by:Mover))
            (to
               (sites Board)
               if:(and
                  (!= (from) (to))
                  (or
                     (is Empty (to))
                     (<
                        (-
                           (what at:(from) level:(topLevel at:(from)))
                           (who at:(from) level:(topLevel at:(from)))
                        )
                        (-
                           (what at:(to) level:(topLevel at:(to)))
                           (who at:(to) level:(topLevel at:(to)))
                        )
                     )
                  )
               )
            )
         )
      )
      (end
         {
            (if (is Target {3 2 1} 2) (result P1 Win))
            (if (= (count Moves) 7) (result P1 Loss))
         }
      )
   )
)
```

図 4　Ludii GDL で記述されたハノイの塔のゲーム記述（正解データ）.