Maastricht University

Department of Data Science and Knowledge Engineering

Master's Program in Artificial Intelligence
and Data Science for Decision Making

Master Research Project Report

Group 6

January 20, 2021

Ludii - English Translation of Formal Rule Sets

Authors:
Adam Lackner
Le Duc Huy Ngo
Prakash Gupta
Sven Kerstjens

Supervisor:
Cameron Browne
Matthew Stephenson

Coordinators:
Dr. Kurt Driessens
Dr. Gijs Schoenmakers

Abstract

Ludii is a general gaming system where games can be designed, evaluated, and played. There are many games implemented in the Ludii system, such as board games (i.e. Tic-Tac-Toe, Chess, etc.), mathematical games, card games, and many more. To understand and appreciate these games, we need descriptions of their rules. In the Ludii system, games are represented in a formal set of rules, consisting of Ludemes. In the future, it will be possible to generate games within the Ludii framework; however, to be able to play those games, English descriptions will be necessary, as Ludemic formal rule sets of those games are incomprehensible for most of the people. To solve this problem, we propose a translation mechanism that converts formal rule sets of games within the Ludii framework to a plain English description. During this paper, we consider a substitution and a contextual approach, after which we consider some directions of future research.

Keywords: *Ludii language, Ludemes, English translation*

Contents

1 Introduction

Since time immemorial, games have been part of human society. Many researchers have acknowledged the vital role of games and their influence on human culture [1]. Games are not only used for fun, entertainment, or for achieving rewards, such as solving puzzle games to get a prize, but they also have educational purposes. With the aim to deepen the understanding of traditional games, using state-of-the-art AI technologies, discovering their historical evolution and their role in the development of human society and the dissemination of mathematical ideas, the Ludii Portal - a general gaming platform - was created [2].

This platform stores hundreds of traditional games from many different countries around the world. To play one of these games, players have to follow the rules of the game. This includes the definition of how to set it up and the way to win it. The set of rules of each game is necessary for the players to understand the game. In the use of that, players can identify legal moves and anticipate replies from their opponents.

The rules of the games within the Ludii platform are primarily defined within a context-free grammar, allowing all implemented games to be defined in a structured way. However, these so-called Ludii game description language (LGDL) are incomprehensible for users who are unfamiliar with it. Luckily, most games in the Ludii framework are games that are known by humans. Therefore, English descriptions of the games can easily be found online and are often already properly explained within the Luddi framework. However, it is still unclear how to describe the rule sets of games that are not known to humans in plain English. In the near future, it will be possible to (randomly) generate games within Ludii; all games that this feature will produce will not have been seen before by humans. Therefore, to be able to play, evaluate or appreciate these games, we need to find a way to describe their rule sets in plain English.

This project's primary goal is to develop a function for the Ludii platform that automatically translates formal rule sets described in Ludemic form into explicit English descriptions. It aims to describe games that humans have not yet seen before, using games currently existing in the Ludii system. Having a feature that translates generated games into plain English, allows users to understand, play, and evaluate those games, without having to understand their generated Ludemic description. Without it, even those with LGDL domain knowledge would have to spend a considerable amount of time trying to make sense of the Ludemic description before being able to play such a game.

In this report, we clarify first the definitions and meanings used in the paper in Section 2. To achieve the main goals of this research, some research questions in Section 3 are posed. Following this, Section 4 explains approaches applied in the project to translate Ludemes. The research experiments and results are shown in Section 5 and 6 with several examples from easy to more complicated games. This section also explains the pros and cons of our current approach. Based on the final results, in Section 7, we offer some suggestions which may be helpful in expanding the proposed translation mechanism. Section 8 is the last section which concludes this research project.

2 Background

In this section, we will consider the main concepts that we will be dealing with throughout the project.

2.1 Ludii

During our project, we will be working within the Ludii portal [3]. Ludii is a general gaming system that can be used to play, evaluate and design games [4]. The system is also used to test Artificially Intelligent search algorithms. One of the system's strengths is, that it allows a wide variety of games to be modelled and played within the same architecture.

Within the project, we have had to get familiar with the Ludii code base. As all of us have varying backgrounds, it has been more difficult for some than for others to gain a thorough understanding of the codebase.

2.1.1 The Ludii Language

Ludii is a complete generalized game system implemented in Java using a class grammar approach in which the language of the game description is created automatically from constructor functions in the source code of Ludii's class hierarchy [5]. Game descriptions represented in grammar are automatically initialized to the corresponding compilation library code, providing a guaranteed 1:1 mapping between the source code and the grammar[6]. The games described using this approach can be effortlessly modified by tailoring a game depending on what you want it to be. The Ludeme library is Ludii's core, which includes several classes, each implementing a specific Ludeme.

Within the Ludii Language, there exist several hundreds of so-called Ludemes, which are its building blocks. Ludemes describe the game and all its aspects in the form of Ludemic phrases. One such Ludemic phrases can be:

(Move Add (To (Sites Empty)))

which could possibly translate to: *"Add a piece of your colour to an empty cell."*. Games within the Ludii system are built out of a set of these kinds of formal rules or Ludemic phrases, which consist of Ludemes.

2.1.2 Ludemes

According to David Parlett [7]: "A ludeme is an element of play, comparable to, but distinct from, a game component or instrument of play." A major advantage of the Ludemic approach is that it encapsulates and gives them concrete names, key game concepts. This provides for the automatic description of sets of game rules, game comparisons, and possibly the automated explanation of learnt strategies incomprehensible terms[6].

2.1.3 Translation

We have to consider what we consider to be a satisfying translation. When some Ludemic phrase is translated to written English, we should make sure that the English phrase yields the same information as the Ludemic phrase as well as possible. The generated English phrase but also the connection between phrases should flow as natural as possible. While formal correctness has our priority, the user should be able to understand the generated English translations without trouble, and ideally, without noticing they are generated as opposed to written.

2.2 Ludii Game Description Language

To understand what Ludii Game Description Language (LGDL) is, first off, we need to know about Game Description Language (GDL).

Game Description Language is a logic programming language [8]. The basis for GDL is a concept of games in terms of entities, actions, propositions, and players: Entities include relevant objects to the state of a game such as game pieces, ranks, files, board squares, etc.; Actions are performed by the player to show the movement of game object such as character; Propositions are conditions of true or false in each game state; Players are the active entities in games. However, with any game description, it is important to precisely define from scratch the main structural aspects of games, such as board or card decks and arithmetic operators. In addition, the equipment and rules have a strong connection with each other, which means that modifying any part of the game entails a substantial rewrite of the whole code. Moreover, the potential applications outside of game AI are restricted to GDL as well.

Based on the main idea of GDL, Ludii Game Description Language was built to overwhelm these problems. According to the research [6], Ludii is not merely more efficient but also performs better than GDL. This approach also opens up potentials for future AI-related works.

3 Research Goals

Our final goal is to implement a translation mechanism within the Ludii General Gaming System, which will take any formal game ruleset and translate it into a plain English description. However, due to the shier size of the Ludii framework, it will not be possible to build a translation mechanism that will cover all possible rule sets. Nevertheless, we aim to implement a mechanism that is able to deal with a substantial amount of cases. The generated translations will be tested on correctness and readability.

During this process, we aim to implement a translation mechanism, using a naive and a sophisticated approach. Moreover, we explore future ways to improve the algorithm with techniques drawn from the Natural Language Processing (NLP) domain. Within the scope of our research, we aim to answer the following research questions:

- How can we convert formal rule sets of Ludemic forms into plain English descriptions?
- To what extent is it possible to translate Ludemes to English in isolation as opposed to translating them within a context?

4 Methods

In this section, we will discuss our approach to building the translation mechanism. Firstly, formal rule sets and their structure will be explained in more detail. Secondly, we will cover a substitution approach, after which we will consider a contextual approach to translating these rule sets.

4.1 Ludemic Structures

Before we can understand how we approach translating games to English, we have to address the structure in which games occur in the Ludii framework. The Ludii framework allows games to be defined in a context-free grammar [5]. Consequently, as mentioned before, each game consists of a combination of so-called *Ludemes* which are the building blocks of the Ludii framework. Generally, the Ludii game structure is given by a tuple of three main parts: Players, Equipment, and Rules. The Players part describes the number of players playing in each turn. There two subsections of the equipment, the first subsection denotes a list of containers which describes the main board of game and the next one presents a list of component presenting which piece is played by each player. The Rules set defines the operations of the game containing three subsets: Start, Play, and End. The Start is an optional section describing the pre-actions of the game. The Play section is one of the important part where define all legal actions (moves) in the game. The condition to end the game is described in the End section. However, these descriptions can become arbitrarily complex. In figure 1 we can see an easy example of a Ludemic description of a game.

4.1.1 From Ludemes To English

More concretely, each Ludeme is represented in the code as a Java class, which defines the working of that Ludeme in different contexts, taking into account its parameters. Before our project, a placeholder "*toEnglish()*" function was already implemented in each class/Ludeme, of which the working was still to be defined.

```
(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)))
    )
)
```

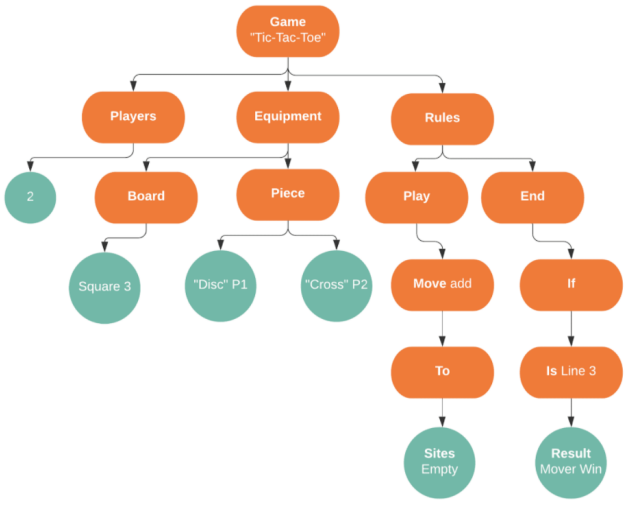


Figure 1: On the left we can see the game Tic-Tac-Toe in Ludemic form, constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result*. On the right we can see the same information visualized as a tree.

4.2 Our Approach

Most of the core classes already implement the *Ludeme* interface which provides the *toEnglish()* function definition. Also, most of them extend the *BaseLudeme* class which also implements the *Ludeme* interface and provides a base implementation for *toEnglish()*. The translation mechanism makes use of it by calling *toEnglish()* in a recursive manner on the tree of Ludemes, starting by the Game-Ludeme (see figure 2).

Because the Game-Ludeme is the main entry point in every game, the major combining logic is implemented here. Every sub-Ludeme in the tree assumes that they will never be used as the root node for translation. Because of that, the results of the toEnglish() function of every not-Game-Ludeme will be structured in such a way, that it can easily be plugged in the other sentence structure. So ,it will not always be possible to generate proper English translations of the Ludemic sub-trees.

Also, the Game-Ludeme is not the only one who has a special role in the translation mechanism. The three Ludemes shown in figure 2 that are the direct child's of the Game-Ludeme also fulfill an additional behavior. So the Player- Equipment- and Rules-Ludeme add some hard coded sentences to the overall translation to improve the concatenation and makes it more natural.

Overall, we started by implementing the translation mechanism for Ludemes occurring in simple games such as "Tic-Tac-Toe" until the translation was sufficient. Afterward, we extended the mechanism along the way for Ludemes occurring in increasingly complex games. In many cases, this included updating the translation mechanism for some Ludemes several times when noticing the translation of that Ludeme fitted in some contexts but turned out to be incorrect in others.

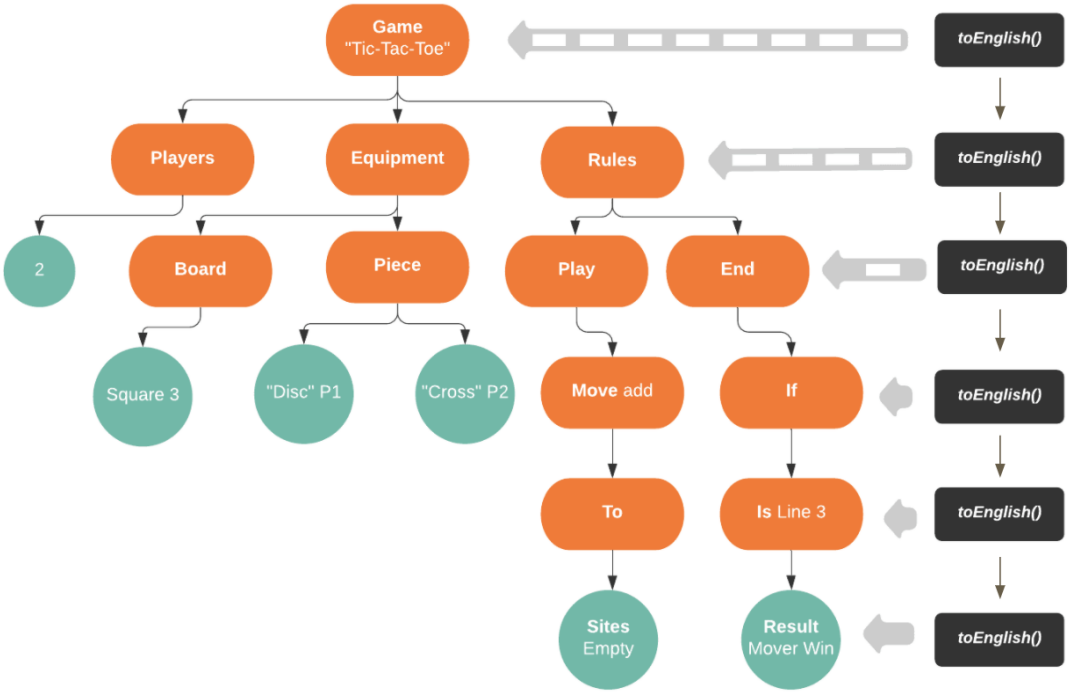


Figure 2: Recursive toEnglish() translation mechanism for the game "Tic-Tac-Toe".

4.2.1 Substitution Approach

Our first intuition was to make a simple approach. With this approach, we tried to figure out a single hardcoded word or set of words that can be used as a substitution for a specific Ludeme. So that we get, after substituting every Ludeme in the description by their so built counter part and a concatenation of the result, a fully translation of the game. This approach may work for some easy Ludemes or Ludeme combinations; however, it is not expected to be sufficiently flexible to deal with the great variety of Ludeme combinations it may encounter.

4.2.2 Contextual Approach

To overcome the problems occurring when translating Ludemes always in the same way, we have to consider translating Ludemes or sets of Ludemes depending on their context. Ludemes can occur in many different places, with different arguments, surrounded by many different other Ludemes. As the translation of a Ludeme may depend on that, the translation function of that Ludeme has to take this into account. So we check for some Ludemes, whether there are specific direct child Ludemes or specific arguments, where we need to change the translation.

Also, the structure of the tree can vary in some cases. So, it is possible that the pieces can contain additional rule sets for determining their movement for example. This could also be handled in the Rules-Ludeme branch. Because of that we have to check, whether Rules-Ludemes exists in the Equipment-Ludeme branch and handle them separately to structure the final outcome.

In section 6 we will go into more detail, regarding specific complexities we encountered.

5 Experiments

This section will discuss our experimental setup. Five experiments have been executed to test the state of the toEnglish() translation mechanism. Firstly, "Tic-Tac-Toe" was tested; secondly, "Flowers" was tested; thirdly, "Sudoku" was tested; fourthly, "Amazons" was tested; and lastly, "Chess" was tested. These games are chosen to be increasing in complexity, each having some specific challenges. Each of the experimental games has some unique challenges and problems, which have helped enhance the algorithm. Moreover, these games already have an existing non-generated English description within Ludii. These descriptions are stated in the following subsections and will act as ground truths for our generated translations.

5.1 Tic-Tac-Toe

The current English description of "Tic-Tac-Toe" in Ludii:

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.

We focused on building all parts of the game. At the time of implementation, we faced the problem with grammatical correctness. Plurals of nouns such as "Disc" and "Cross" need to be generated correctly when necessary, as well as suiting punctuation.

5.2 Flowers

The current English description of "Flowers" in Ludii:

Players take turns placing a piece of their colour, and win by making an orthogonally connected line of 4 of their colour (diagonal steps do not count).

We faced the challenge of dealing with the gaming board's dimension, which has unusual dimensions. For example, "Tic-Tac-Toe" has a "3x3" board, but for the game "Flowers," we have to deal with a "T33336 Hexagon Dual" board.

5.3 Sudoku

The current English description of "Sudoku" in Ludii:

Played on a 9x9 grid divided into 3x3 groups of 9 or 'nonets.' Numbers from 1 to 9 are placed in the squares such that each row, column, or nonet has each number appearing only once.

We faced the challenge of dealing with describing regions that are not clearly visually defined in the board itself. Besides, the end conditions are different from typical two or more player games.

5.4 Amazons

The current English description of "Amazons" in Ludii:

Played on a 10x10 board. Each player has four Amazons (chess queens), with other pieces used as arrows. Two things happen on a turn: an amazon moves like a Chess queen, but cannot cross or enter a space occupied by another amazon or arrow. Then, it shoots an arrow to any space on the board that is along the path of a queen's move from that place. The last player able to make a move wins.

We encountered mention of rules of the game defined within pieces. This was different from what we had seen before as rules are typically defined within the "Rules" branch of the Ludemic description.

5.5 Chess

The current English description of "Chess" in Ludii:

Played on an 8x8 board with pieces with specialized moves: Pawns (8): can move one space forward; Rooks (2): can move any number of spaces orthogonally; Bishops (2): can move any number of spaces diagonally; Knight (2): moves in any direction, one space orthogonally with one space forward diagonally; Queens (1): can move any number of spaces orthogonally or diagonally; Kings (1): can move one space orthogonally or diagonally. Players capture pieces by moving onto a space occupied by an opponent's piece. Player wins when they capture the other player's king.

Chess has many challenges. There are many different pieces, each having specific rules bound to them. Also, there are many rules which are defined specifically for Chess, such as *En Passant* or *Castling*. Moreover, rules for pieces can be complicated, as the possible ways a piece can move depends on the state of the game.

6 Results

In this section, we will discuss the generated translations. During the project, we have implemented around **120** Ludemes, which is **7.5%** of all currently existing Ludemes. The mechanism provides a valid translation for (at least) **50** games, which occupy roughly **10%** of all games in Ludii. However, this number is only count the games which are **100%** translated, while there are more than **100** games with at least 80% complete translation.

We will now go into the results for several games specifically.

6.1 Tic-Tac-Toe

"Tic-Tac-Toe" is the most straightforward game to translate in Plain English (See figure 3). It has two players and a simple rule and winning conditions. The problem we faced to translate in English is singular or plural of the words such as disc or discs, player or players, and cross or crosses. For the grammatical correction, we implemented the logic to select the plural or singular words in the sentences. As plurals of nouns in English can be constructed in more than one way depending on the word. Therefore, we had to make sure that correct plurals are generated based on the necessary English grammar rules [9].

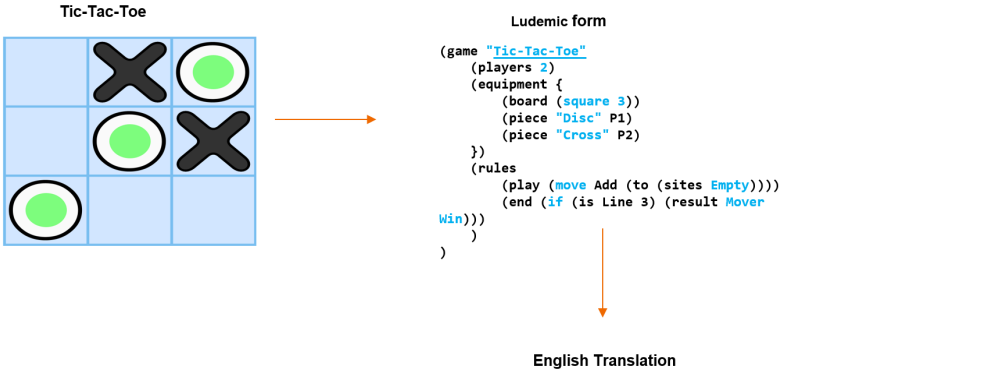


Figure 3: *Top-left*: A visual representation of the game "Tic-Tac-Toe". *Top-right*: "Tic-Tac-Toe" in Ludemic form, constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result* *Bottom-right*: Generated English translation of "Tic-Tac-Toe".

When comparing the generated translation with the ground truth, we can see that the information is almost equal. Yet, they differ in sentence structure. Specifically, the sentence structure of the ground truth description is more complex, while the generated description has more concise sentences.

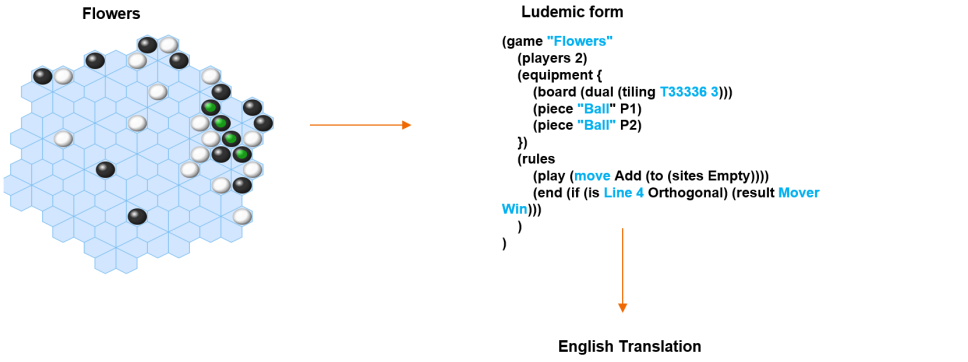


Figure 4: *Top-left*: A visual representation of the game "Flowers". *Top-right*: "Flowers" in Ludemic form, constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result* *Bottom-right*: Generated English translation of "Flowers".

6.2 Flowers

In the game "Flowers", we encountered a new type of board dimensions (see figure 4). The shape of the flowers board is *T33336 Hexagon Dual board* which has different conventions than usual. Using a case-based approach, we implemented language utilities, which are capable of translating many types of unusual boards to an English description.

When comparing the generated translation with the ground truth, we find that the ground truth is much shorter; it is even only one sentence. Also, the ground truth does not give any information about players or the board, while the generated description does.

6.3 Sudoku

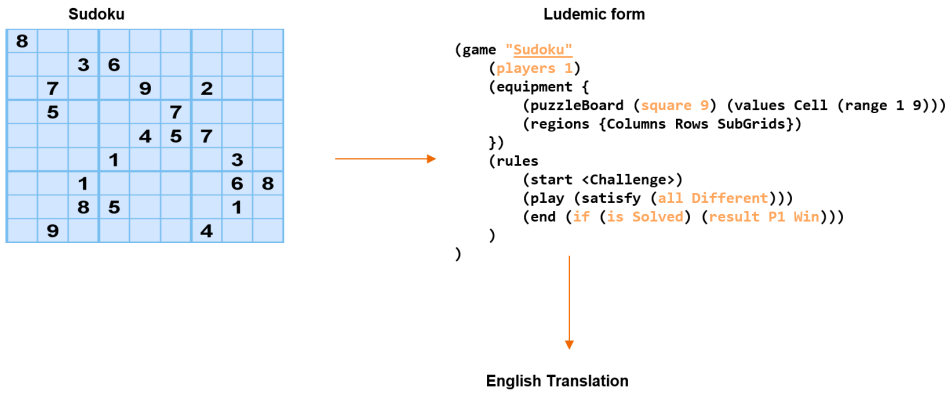
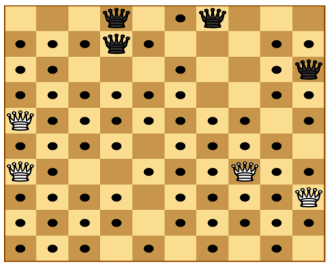


Figure 5: *Top-left*: A visual representation of the game "Sudoku". *Top-right*: "Sudoku" in Ludemic form, constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result* *Bottom-right*: Generated English translation of "Sudoku".

In the game "Sudoku" we encountered the notion of having some relevant regions which are not immediately visible from the playing board. For "Sudoku" one would need all digits within columns, rows, and sub-grids to be different. Where sub-grids are visible in the playing board by the bold lines, rows and columns are not. Therefore we need to mention these regions in the translation. In a previous state of the mechanism, we introduced code that produced a translation of the form: "The played regions are Columns, Rows, and Sub-grids." However, later we had to remove it for the sake of generality in cases where the *regions* Ludeme was used differently. Up until now, we have not come to a solution to make the translation for *regions* give no conflicting outputs in different games. It seems that in some cases, the relevant region information is present within the *region* Ludeme, wherein other games, it is given in the *regiontype* Ludeme.

When comparing the generated translation with the ground truth, we find that the information about regions is missing in the generation, while it is present in the ground truth. Besides, the ground truth has information on the actual numbers which can be used in the game, while the generated description refers to "items".

6.4 Amazons



Ludemic form

```
(game "Amazons"
   (players 2)
   (equipment {
      (board (square 10))
      (piece "Queen" Each (move Slide (then
(moveAgain))))
      (piece "Dot" Neutral)
      }
   )
   (rules
      (start {
         (place "Queen1" {"A4" "D1" "G1" "J4"})
         (place "Queen2" {"A7" "D10" "G10" "J7"})
      })

      (play
         (if (is Even (count Moves))
            (forEach Piece)
            (move Shoot (piece "Dot0"))
         )
      )

      (end (if (no Moves Next)  (result Mover
Win) ) )
   )
)
```

English Translation

The game "Amazons" is played by two players on a 10x10 square board.
No player plays Dots. Player one plays Queens. Player two plays Queens.
Rules for Pieces:
    Queens slide in the Adjacent direction with the condition: empty cell, then move again.
The game is initialized as follows:
    Place a Queen for player one on sites: A4, D1, G1 and J4.
    Place a Queen for player two on sites: A7, D10, G10 and J7.
Rules of play:
    if the number of moves is even, during your turn, move one of your pieces, else shoot the piece Dot0.
End conditions:
    If the next player cannot move, the moving player wins.

Figure 6: *Top-left*: A visual representation of the game "Amazons". *Top-right*: "Amazons" in Ludemic form, constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result* *Bottom-right*: Generated English translation of "Amazons".

In the game "Amazons" we encountered rules that are defined within the pieces (see figure 6). For example, in a game involving chess queens, one needs to know the moves that a chess queen can make. We chose to devote a section of the English translation to rules for pieces, which checks if there are any rules defined within the pieces, after which it will describe them if necessary. The rules that are defined can often by fairly complicated, which can be seen in the game Chess.

Compared to the ground truth we can see that the generated description is quite different. The ground truth seems to appeal more to ones common sense, as it describes turns, consisting of two moves, where the toEnglish() descriptions talks about the number of moves being even or not. Again, more complex sentences occur in the ground truth than in the toEnglish translation.

6.5 Chess

The game "Chess" is the most difficult game of which we have tried to implement the English translation because of its many complexities. It has many rules that depend on many specific scenarios in the game. Moreover, every piece has unique moves, which are not always the same (e.g.: pawns can move forward one square,

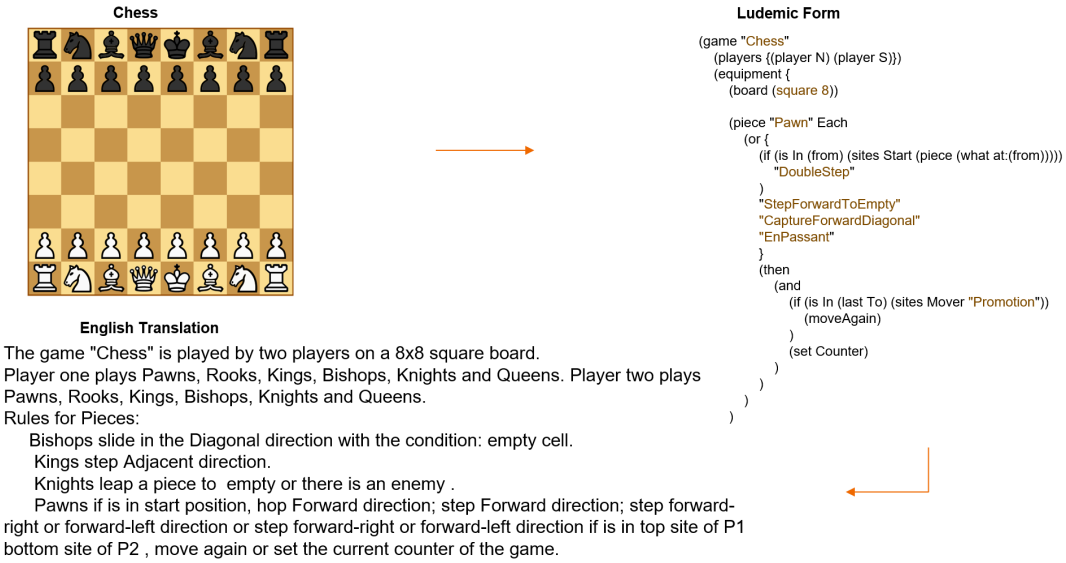


Figure 7: *Top-left*: A visual representation of the game "Chess". *Top-right*: part of the Ludemic form of "Chess", constructed from combinations of Ludemes such as: *equipment*, *board*, *piece*, *rules*, *play* and *result* *Bottom-right*: Part of the generated English translation of "Chess".

but can move forward two squares in its first move, but when it wants to take a piece, it can only do so in a forward diagonal direction. On top of that, it can also perform an "*en passant*"). Many rules of Chess such as "*en passant*" and "*castling*" are defined specifically for chess, making it difficult to translate them in a generic way.

In the generated translation (see Appendix A), we can see that many parts of the rules of pieces are translated, but not all of them are correct (see figure 7). Also, it has shown to be complicated to produce proper English sentences from Ludemic phrases, which consist of many Ludemes, such as *(or (if (is In (from) (sites Start (piece (what at: (from)))))*, as each part of the phrase is highly dependent of its context. Particularly, it has shown to be difficult to generically generate words in English in the correct order, as the Ludemes from which they are generated are often processed in a different order.

When comparing the generated translation to the ground truth we find that the generated translation is much lengthier. It is also much more chaotic, as some rules are described in a rather cumbersome fashion, for example in the piece rules for the Pawn. This is both due to short-comings of the mechanism, as well as the complex notation in the Ludemic descriptions.

6.6 Evaluation

The validation of the results we have found, has been a continuous process of evaluating the translations ourselves, being complemented by getting feedback from others, including native English speakers. Upon finding mistakes in translations or untranslated rules, we have responded by fixing the mistakes by updating the

mechanism, or by implementing Ludemes that were not yet translated. As seen, the translations are not always completely correct. This is both due to the constrained scope of this project, and the limitations that will be mentioned in section 7.

7 Discussion

In this section, we will evaluate the results from section 6 and answer our research questions. To answer the question "How can we convert formal rule sets of Ludemic forms into plain English descriptions?", we have considered our approach as described in section 4. As we have seen, our approach yields some interesting results, producing adequate English descriptions for several games. While the generated descriptions look different from the provided game descriptions, they are often equally understandable.

To answer the question "To what extent is it possible to translate Ludemes to English in isolation as opposed to translating them within a context?" we have to consider that processing and translating Ludemes without taking into account the context they occur in is often not sufficient to translate whole games. However, it provided a proper basis for dealing with Ludemes that do not depend much on their context. such as: *board*, *players* and *game*. While not requiring a complicated solution, translating these Ludemes is still very useful as they occur in most games.

When Ludeme structures become more complicated, the quality of our results shows to highly depend on the "intuitiveness" of the Ludemic description. More specifically, when Ludemes occur in roughly the same order as they would occur in natural language, their translation is more likely to be correct. For example, the phrase "...(piece "Bishop" Each (move Slide Diagonal (to ...)))" gets translated to "... Bishops slide in the diagonal direction with the condition: empty cell ...". This translation shows to be appropriate despite consisting of several Ludemes.

However, due to the tree structure of the mechanism, Ludemes can be processed in an order that does not always match the order in which their translations would occur. Moreover, in several cases, Ludemic descriptions describe rules from a different perspective than their natural language counterpart while meaning the same thing. For example, in the game "Amazons," the Ludemic rule "...(if Even (count Moves))..." occurs. The toEnglish() function translates this to "...if the number of moves is even...", which seems roughly correct. However, in the context of the game Amazons, this rule exists to clarify that after moving a Queen, that Queen can shoot an arrow, meaning that something – the shooting of an arrow – happens only when the number of moves is even. While the Ludemic rule "...(if Even (count Moves))..." formally makes sense, the phrase "...if the number of moves is even..." is very unlikely to occur in any in non-generated natural language descriptions of a game, while likely being among the best possible translations for the phrase in question.

7.1 Outcomes

The final outcome of the project is the code for the toEnglish() function. This code is reworking a part of the current Ludii code base. Not every Ludeme will have a

new implementation for the toEnglish() function. Every Ludeme which has no new function will fall back to the base implementation, which just adds a placeholder to the output. Because of that, not every game will be fully translatable at the moment. However, the current translation leads to the right direction, also if it is not complete. So the outcome of this project is usable in the official Ludii code base and is useful for the general case, but should be stated as an early alpha-version. The complete code can be found on: https://github.com/cambolbro/Ludii .

7.2 Future Research

By extending the current algorithm, several improvements can be made; firstly, by dealing with the current mistakes in the translations which are still generated for some Ludemes or Ludeme combinations (See Section 6 for examples); secondly, by implementing the translation mechanism for a greater variety of Ludemes. Finally, we have not yet been able to test our mechanism on actual generated games as the game generation function of the Ludii gaming system is still under constructing. In the future, it will be interesting to test the translation mechanism on generated games. While we expect to get similar results on generated games as on existing games as they are fundamentally the same, this should be tested experimentally when possible.

Furthermore, it can be useful to consider different approaches. Within the domain of extracting natural language from some kind of machine language, researchers have tried to summarize Java source code by using methods from natural language processing.[10] They aimed to build a model that provided short generated descriptions for Java methods using Neural Machine Translation (NMT). While this technique is usually used to translate one natural language to another, it seems that it can also be applied for code summarizing. An encoder-decoder system within these methods usually translates sentences one word at a time. In their research, they use a similar framework to summarize source code, yielding good results.

In the future, it might be possible to consider a similar approach as LeClair et al. (2019) [10] to translate formal Ludemic rule sets into plain English. One major challenge will be to deal with the relatively small data set that could be created from the Ludii framework. As a comparison, within Ludii, several hundreds of games and Ludemes exist, while Leclair et al. used 51 million Java methods to train their model. Therefore, it will likely be necessary to take a more general approach, not restricting to the Ludii framework.

8 Conclusion

In our project, we have implemented a translation mechanism that translates formal rule sets of games into plain English descriptions. The approach we have taken has shown to yield good results for a variety of Ludemes occurring in a variety of games. We have found that in most cases translating Ludemes in isolation will yield unsatisfactory translations. Therefore, resorting to contextually based translations was necessary. While this approach yields good results, more work is necessary to

obtain a fully functioning translation mechanism covering the entire Ludii framework. Moreover, to overcome some of the problems we have faced, it might be useful to explore NLP based approaches as described in section 7.2. Finally, in the future, it would be interesting to test the translation mechanism on generated games when they become available.

References

[1] Zuzana Václavíková. Game: Experience as an educational tool. https://www.intechopen.com/books/game-design-and-intelligent-interaction/game-experience-as-an-educational-tool.

[2] Digital ludeme project. http://ludeme.eu/index.html.

[3] Matthew Stephenson, Eric Piette, Dennis Soemers, and Cameron Browne. An overview of the ludii general game system. In *IEEE Conference on Games*, pages 864–865, August 2019.

[4] Cameron B Browne. Ludii portal. https://ludii.games/index.php.

[5] Cameron B Browne. A class grammar for general games, 2016.

[6] Éric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H. M. Winands, and Cameron Browne. Ludii - the ludemic general game system. *CoRR*, abs/1905.05013, 2019.

[7] D. Parlett. *What's a Ludeme?* 2007.

[8] Game definition language. http://games.stanford.edu/games/gdl.html.

[9] Plurals of english nouns. https://lexico.com/grammar/plurals-of-nouns.

[10] Alexander LeClair, Siyuan Jiang, and Collin McMillan. A neural model for generating natural language summaries of program subroutines. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 795–806. IEEE, 2019.

List of Figures

A Appendix

The game “Chess” is played by two players on a 8x8 square board.

Player one plays Pawns, Rooks, Kings, Bishops, Knights and Queens. Player two plays Pawns, Rooks, Kings, Bishops, Knights and Queens.

Rules for Pieces:

Bishops slide in the Diagonal direction with the condition: empty cell.

Kings step Adjacent direction.

Knights leap a piece to empty or there is an enemy .

Pawns if is in start position, hop Forward direction; step Forward direction; step forward-right or forward-left direction or step forward-right or forward-left direction if is in top site of P1 bottom site of P2 , move again or set the current counter of the game.

Queens slide in the Adjacent direction with the condition: empty cell.

Rooks slide in the Orthogonal direction with the condition: empty cell, then if reaches to the location of the last move equals to default piece value.

The game is initialized as follows:

Place one Pawn for player one at each cell of the second row.

Place one Pawn for player two at each cell of the seventh row.

Place a Rook for player one on sites: A1 and H1.

Place a Knight for player one on sites: B1 and G1.

Place a Bishop for player one on sites: C1 and F1.

Place a Queen for player one on site D1.

Place a King for player one on site E1.

Place a Rook for player two on sites: A8 and H8.

Place a Knight for player two on sites: B8 and G8.

Place a Bishop for player two on sites: C8 and F8.

Place a Queen for player two on site D8.

Place a King for player two on site E8.

Rules of play:

if in the same turn, a piece of the moving player reaches to the location of the last move, this piece can promote into Queen, Knight, Bishop or Rook, else during your turn, move one of your pieces or if King of current moving player equals to King of the moving player, King of current moving player equals to default piece value and not King of the moving player is threatened, if RookLeft of current moving player equals to default piece value and can move slide from RookLeft of current moving player to 3 cell in the east direction with the condition: empty, slide from King of current moving player to 2 cell in the west direction with the condition: empty and not King of the moving player is threatened, then or slide from RookLeft of current moving player to 3 cell in the east direction with the condition: true or if RookRight of current moving player equals to default piece value and can move slide from RookRight of current moving player to 2 cell in the west direction with the condition: empty, slide from King of current moving player to 2 cell in the east direction with the condition: empty and not King of the moving player is threatened, then or slide from RookRight of current moving player to 2 cell in the west direction with the condition: true.

End conditions:

If King of the next player is threatened and not can move during your turn, move one of your pieces, the moving player wins. If the moving player cannot move or Counter equals to 100, it's a draw.