Wordle in Prolog

Sanyam Agrawal - IIT2021207

***Abstract*— This project aims to implement the popular word puzzle game Wordle using the Prolog programming language. Wordle is a game where players attempt to guess a secret word by proposing words and receiving feedback on the correctness of their guesses. The primary goal is to create an interactive and engaging Prolog program that simulates the Wordle game experience.**

I. PROBLEM STATEMENT

The problem is to implement the classic game Wordle using the Prolog programming language. Wordle is a word puzzle game where the player has to guess a secret word by suggesting words and receiving feedback on the correctness of each suggestion.

Through this project, we aim to not only implement a classic word game in Prolog but also enhance logical thinking and problem-solving skills. The Wordle game in Prolog serves as a practical application of the language's capabilities and offers an opportunity to explore and experiment with logic-based programming paradigms.

II. PROJECT SCOPE

The scope of the project encompasses a mechanism for generating a random secret word, and logic to evaluate the correctness of user guesses. The program will provide clear and concise feedback to the user after each guess, maintaining an intuitive interface for an optimal experience.

A. User Interface for Input and Feedback

1) **Input Interface:** Design an interactive interface to collect user input. This may involve a simple command-line interface where users can input their guesses or a more sophisticated graphical user interface (GUI) if desired.
2) **Feedback Interface:** Implement a mechanism to provide feedback to the user after each guess. Feedback should include information on correct letters, misplaced letters, or any other relevant information guiding the user towards the correct word.

B. Generation of a Secret Word

1) **Word Database:** Develop and utilize a database of words from which the secret word will be randomly selected. This database will be predefined. The database contains several words divided into different categories (E.g. 'india' would be in category 'country')
2) **Random Selection:** A category is chosen randomly from the database and subsequently a word is randomly chosen from the selected category.

C. Logic for Evaluating the Correctness of User Guesses

1) **Word Comparison:** Iterate through each letter of the user's guess and compare it with the corresponding letter in the secret word. Identify letters that are in the correct position and those that are not.
2) **Correct Letters in Correct Position:** Keep track of letters that match both in value and position. These are the letters that the user has correctly guessed and placed in the right order.
3) **Correct Letters in Wrong Position:** Identify letters that match in value but are in the wrong position. These letters are correctly guessed but placed elsewhere in the word.
4) **Generate Feedback:** Two lists are returned, correct-letters (common letters between actual-letters and guess-letters) and correct-positions (letters that appear in both actual-letters and guess-letters in the same positions).
5) **Display Feedback to User:** Present the feedback lists to the user after each guess. Clearly communicate the information about correct letters in the correct and wrong positions to guide the user in refining their subsequent guesses.

D. Constraints for the Game

1) **Attempt Counter:** Implement a mechanism to keep track of the number of attempts the user has made. The number of attempts are initialised to 6.
2) **Game Termination:** Define conditions for terminating the game. This could include reaching the maximum number of attempts or the user correctly guessing the word.

III. LITERATURE REVIEW

A. Wordle Game Word Difficulty Classification

The research paper focuses on Wordle, a word-guessing game designed by Jonah Lupton in 2021, and explores the game from a developer's perspective. While previous research has primarily delved into optimal word-guessing strategies for players, this study aims to aid developers in categorizing potential target words by difficulty.

The analysis reveals a strong correlation between word difficulty and average termination rounds, with words of higher difficulty requiring more rounds on average. The study's findings are supported by a violin plot and statistical analysis, demonstrating the significance of the relationship between difficulty levels and termination rounds.

It introduces a developer-centric approach to Wordle by providing a method to categorize word difficulty, enhancing player engagement and retention. The incorporation of average mutual information, dynamic programming, and Monte Carlo simulations contributes to a comprehensive understanding of the game's dynamics from both a player and developer perspective.

B. Wordle is NP - HARD

This research paper investigates the computational complexity of the Wordle game by formulating it as a decision problem. The primary goal is to determine whether a winning strategy exists for the guesser within a limited number of attempts. The paper establishes the formal decision problem, outlines the concept of feasible words based on previous guesses, and presents an algorithm (WordleGuesser) for playing optimally.

The main results of the research are as follows:

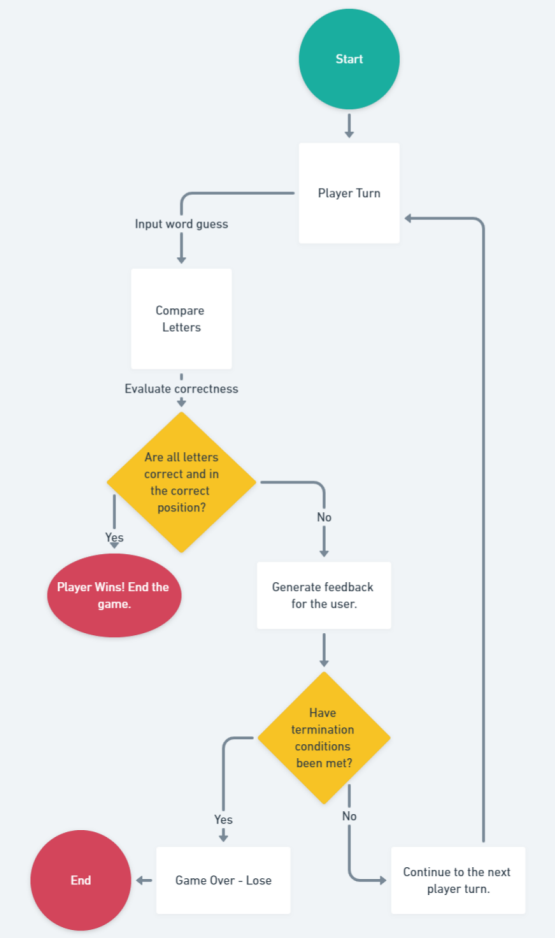
- Theorem 1 states that Wordle is NP-hard and W[2]-hard when parameterized by the number of guesses allowed.
- Theorem 2 asserts that Wordle cannot be solved in polynomial time unless P = NP, even when restricted to instances where the word length (k) is fixed at 5.
- Theorem 3 indicates that Wordle can be solved in polynomial time if the alphabet size is constant.

IV. Requirement Specifications

- **Hardware and Software Requirements**
  - **Hardware:** Standard PC or laptop with sufficient processing power and memory.
  - **Software:** GNU Prolog Console
  - **Why Prolog:** It is a declarative programming language that is particularly well-suited for applications that involve logic, rule-based reasoning, and symbolic computations.
- **Functional Requirements**
  - **User Interface:** Design an interactive interface for user input and feedback.
  - **Word Generation:** Implement a mechanism to generate a random secret word.
  - **Guess Evaluation:** Develop logic to evaluate the correctness of user guesses.
  - **Game Flow:** Control the flow of the game, allowing the user a specified number of attempts.
- **Usability Requirements**
  - **Intuitive Interface:** Ensure that the information exchanged between user and the program is clear and easy to navigate.
  - **Feedback Clarity:** Provide clear and concise feedback to the user after each guess.
  - **Scalability:** Design the program to be scalable, allowing for potential future enhancements or modifications.
  - **Error Handling:** Implement error handling to manage unexpected inputs and edge cases.

V. Activity Diagram

The diagram illustrates the sequential progression of events, beginning with the initiation of the game. The diagram showcases player turns, where users input their guesses, followed by the comparison algorithm that assesses the correctness of each letter in the guess against the secret word. Feedback generation then takes place, indicating correct letters in the correct position and those in the wrong position. This interactive loop between user input and feedback continues until the game reaches its termination conditions, which may include the user correctly guessing the word or reaching the maximum allowed attempts. Branching pathways depict decision points where the game assesses whether the termination conditions are met, guiding the flow accordingly.



VI. Methodology

Prolog programs consist of facts and rules. Facts are statements about relationships between objects, while rules define how new facts can be inferred from existing ones using logical conditions. Predicates are units of logic, consisting of a name and a number of arguments. They can represent relationships, properties, or actions. The predicates we have defined are:

- **is-category(C):** Succeeds if C is one of the available categories.
- **categories(L):** Succeeds if L is a list of all the available categories without duplicates.
- **pick-word(W, C):** Succeeds if there exists a word W in the database which is in category C.
- **correct-positions(ActualLetters, GuessLetters, CorrectPositions):** Succeeds if CorrectPositions contains the letters that appear in both ActualLetters and GuessLetters in the same positions.
- **main:** A predicate that is used to start the game, it first consults the database and then goes into play mode.
- **play:** Starts the game play phase.
- **word(W, C):** Succeeds if there exist a word W in the database in the category C.
- **random-member(M, L):** Succeeds if M is a random member of the list L.
- **intersection(L1, L2, L3):** Succeeds if L3 contains the common items between L1 and L2.
- **list-to-set(L, S):** Succeeds if S is the result of removing duplicates and ordering L in ascending.

These were the predicates explicitly defined. The in-built predicates of GNU prolog used to facilitate the functionality of the above are:

- **write(Message):** Prints a message to the screen.
- **consult(File):** Loads and consults the predicates from the specified file.
- **length(List, Length):** Unifies Length with the number of elements in List.
- **random(Min, Max, RandomNumber):** Generates a random number between Min and Max.
- **nth0(Index, List, Element):** Unifies Element with the element at position Index in List (0-based indexing).
- **read(X):** Reads a term from the user input and unifies it with X.
- **atom-length(Atom, Length):** Unifies Length with the number of characters in Atom.
- **atom-chars(Atom, ListOfChars):** Unifies ListOfChars with the list of characters in Atom.
- **setof(X, Goal, List):** Constructs a list of all solutions to Goal that can be obtained from unifying X.

VII. Result

The complete code can be found at Github Link.

The successful implementation of the Wordle game in Prolog demonstrates how logic programming languages can create engaging experiences. Prolog's logical reasoning and rule-based approach elegantly handle complex game mechanics like word evaluation and feedback. This shows that even within a declarative programming paradigm, intricate games like Wordle can be effectively realized.

Furthermore, this project suggests that Prolog has potential beyond just puzzle games. Its ability to express relationships and constraints intuitively means it could support a variety of entertaining applications. This flexibility opens doors to creating diverse gaming experiences that engage and entertain players in new ways.

In summary, the Wordle project in Prolog showcases how logic programming languages can be adapted to build interactive and captivating games, hinting at a promising future for Prolog in game development and beyond.

Below are the screenshots of the winning and losing cases:

```
| ?- main.
Welcome to Prolog-Wordle!
----------------------
compiling D:/Semester-6/CCPM/wordle-project/db.pl for byte code.
D:/Semester-6/CCPM/wordle-project/db.pl compiled, 4 lines read -
The selected category is country
Game started. You have 6 guesses.

Enter a word composed of 5 letters:
'yummy'.
Correct letters are: [m,y]
Correct letters in correct positions are: [y,m]
Remaining Guesses are 5

Enter a word composed of 5 letters:
'lemon'.
Correct letters are: [e,m,n]
Correct letters in correct positions are: [e,m,n]
Remaining Guesses are 4

Enter a word composed of 5 letters:
'yemen'.
You won!
```

```
Correct letters are: [l,r]
Correct letters in correct positions are: [l]
Remaining Guesses are 3

Enter a word composed of 5 letters:
'sitar'.
Correct letters are: [s,r]
Correct letters in correct positions are: []
Remaining Guesses are 2

Enter a word composed of 5 letters:
'piano'.
Correct letters are: []
Correct letters in correct positions are: []
Remaining Guesses are 1

Enter a word composed of 5 letters:
'flute'.
You lost!
The correct word was lyres!
```

References

[1] Information Theory-based Wordle Game Word Difficulty Classification and Dynamic Planning Optimization Research *Xin Gui *, Chen Su, Keyu Pan School of Civil Engineering, Shandong Jianzhu University, Jinan, China, 250101*

[2] Wordle is N P - H A R D *Daniel Lokshtanov and Bernardo Subercaseaux, University of California Santa Barbara, Carnegie Mellon University, Pittsburgh*

[3] Wordle Demo

[4] GNU Prolog Documentation