Batman vs Riddler: An Immersive Game Demonstrating The Theory of Finite State Automata

Nawrin Sikder, Syed Bukhari
College of Engineering and Computing Sciences, New York Institute of Technology

CSCI 610: Theoretical Concepts in Computers and Computation

Professor Maherukh Akhtar

Fall Semester December 8, 2023

Contents

Batman vs Riddler: An Immersive Game Demonstrating The Theory of Finite State Automata 3

Implementation 3

Non-Deterministic Finite State Automata (NFAs) 4

Deterministic Finite State Automata (DFAs) 5

Learning Experience Outcome - 7

Learning Experience Outcome - 7

Learning Experience Outcome - 7

References 7

Batman vs Riddler: An Immersive Game Demonstrating The Theory of Finite State Automata

The Java program "Batman vs Riddler" is an engaging and interactive game that combines the thrill of solving riddles with the excitement of the Batman universe. The game employs the use of while loops and switch statements to create a dynamic user experience. Upon starting the game, the player is presented with a series of riddles from the Riddler, one of Batman's iconic villains. The player has three attempts to answer each riddle correctly. The use of while loops ensures that the game continues until the player either successfully answers all the riddles or exhausts their three attempts. The program utilizes switch statements to determine the outcome based upon the user's input. If the player correctly answers a riddle on the first or second try, they progress to the next level, encountering more challenging riddles. However, if the player fails to provide the correct answer within the given three attempts, the game either ends, and the program exits, or falls back onto the previous level and ultimately ends. This simple yet engaging structure not only tests the player's problem-solving skills but also keeps them immersed in the Batman-themed experience. Overall, the "Batman vs Riddler" game demonstrates how basic control flow structures in Java can be harnessed to create an entertaining and interactive gaming experience.

Implementation

Within the "Batman vs Riddler" Java program, while loops play a crucial role in driving the flow of the game. The primary while loop serves as the overarching control structure, ensuring that the game continues until the player either successfully solves all the riddles or exhausts their three attempts. The loop begins by presenting the user with the first riddle, prompting them to input their answer. Inside the while loop, a switch statement evaluates the user's response, determining whether it is correct or incorrect.

If the user answers correctly on the first or second attempt, the while loop facilitates the transition to the next level, introducing a new set of challenging riddles. Importantly, the loop condition is designed to break if the player successfully completes all levels, ensuring a smooth exit from the game. Conversely, if the user provides an incorrect answer within the specified attempts, the loop terminates, and the program exits, concluding the game. This iterative process of presenting riddles, evaluating responses, and progressing through levels encapsulates the essence of the game's design and showcases the effectiveness of while loops in controlling the game's flow. This ensures that the user experiences a progressively challenging set of riddles while adhering to the overarching three-attempt rule. By strategically using while loops, the "Batman vs Riddler" program not only provides an engaging user experience but also demonstrates the power of iterative control structures in creating dynamic and interactive games.

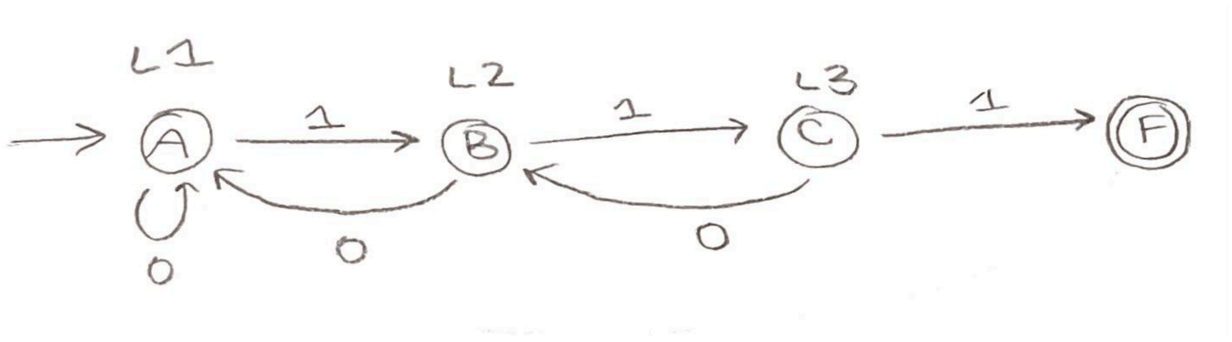
Integrating the "Batman vs Riddler" game into an object-oriented programming (OOP) script can enhance its structure and maintainability. In an object-oriented approach, each aspect of the game, such as levels, riddles, and user interaction, can be encapsulated within objects, contributing to a more modular and extensible design. For example, a "Riddle" class could represent individual riddles, each with its own properties and methods for validation. A "Level" class could manage the progression through different sets of riddles, while a "Game" class could oversee the overall game flow and user interactions.

By adopting an object-oriented paradigm, the game's structure becomes more intuitive and easier to expand upon. The NFA analogy remains relevant, as each object encapsulates a state and transitions between these states can be represented through method calls and interactions between objects. The "while" loops in the original implementation could be replaced or complemented by methods within the relevant classes, enhancing the readability and maintainability of the code. Additionally, the OOP approach allows for the implementation of inheritance and polymorphism, enabling the creation of specialized classes for different types of riddles or levels. This not only facilitates code reuse but also reflects the flexibility inherent in NFAs, where transitions can lead to various states. Overall, incorporating object-oriented principles into the "Batman vs Riddler" game not only aligns with best practices in software development but also provides a clear and structured representation of the game's dynamics, mirroring the non-deterministic transitions found in NFAs.

Non-Deterministic Finite State Automata (NFAs)

While the "Batman vs Riddler" game is an illustrative example of how while loops can control the flow of a program, it is interesting to draw a parallel with Non-Deterministic Finite State Automata (NFA) in theory of computer science. NFAs are theoretical models of computation that involve states, transitions, and an alphabet of symbols. In the context of the game, we can draw an analogy between the game's progression through levels and the transitions between states in an NFA. Each level in the game can be seen as a state in the NFA, and the transitions between these levels are akin to the transitions between states in an NFA. The while loop acts as the mechanism governing these transitions, determining whether the player advances to the next level or exits the game based on their answers. In this way, the game's design aligns with the concept of non-deterministic transitions in NFAs, where the outcome depends on the current state and the input provided by the user. Furthermore, the user's attempts to solve a riddle within a level can be seen as the non-deterministic choices an NFA makes when transitioning between states. The game's branching structure, determined by correct or incorrect answers, reflects the non-deterministic nature of NFAs, where multiple transitions may be possible from a given state.

Figure 1



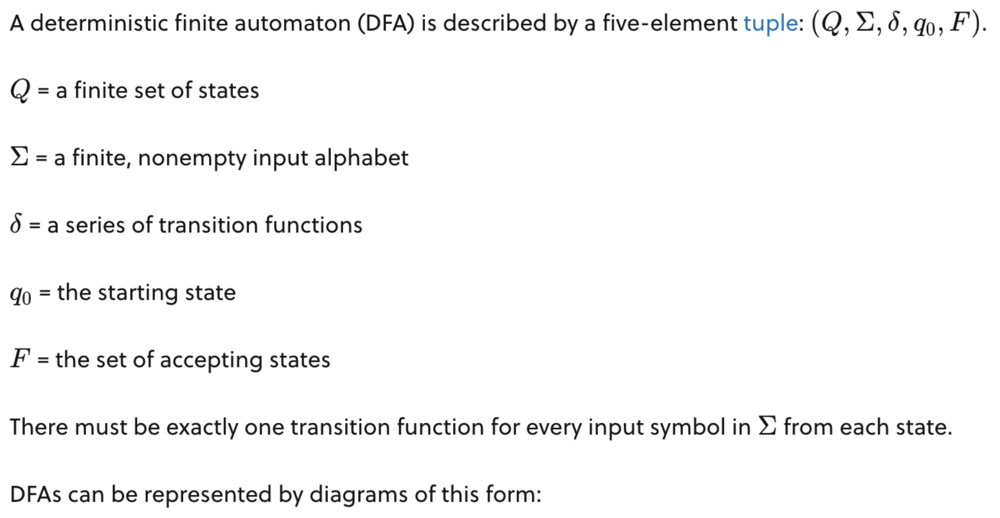
In summary, the "Batman vs Riddler" game provides a practical example of how while loops can emulate the behavior of an NFA by controlling the transitions between different states or levels. While NFAs are theoretical constructs used in formal language theory, the game's structure mirrors the essence of non-deterministic transitions, showcasing the versatile application of programming concepts in both theoretical and practical contexts.

Deterministic Finite State Automata (DFAs)

Not only are important insights drawn from non-deterministic finite state automata but another input that branches from the theory of finite state machines are deterministic finite state automatons (DFAs). DFAs much like NFAs, consist of states, transitions, and an alphabet of symbols that involve a set of five tuples involving a set of all states, a set of input symbols, the initial state, the transition state as well as the final state. In conjunction with an NFA, each level also constitutes as a state, as for each input the user supplies, that determines the way in which the program is able to transition throughout each state. Contrasting to the NFA however, DFAs occur where an input symbol can travel to solely one state in which every input has to be accounted for. In this game, the DFA exemplifies itself, as the user will produce a response to the riddle in question. There are multiple states in which the user can now end up in however, based on their input of an answer, the integration and implementation of switch statements will determine the result of the users progress. From there, while loops continue to ensure that the program is constantly in motion, whether that be progress moving forward, or moving back to previous levels of the game, if and only if that is in the instance of the user being unsuccessful in progressing through the game. The user will thus end up in ultimately one state based on their input of a response. There are multiple states in which the player could end up in when attempting to defeat the Riddler however, based on what the player chooses to respond with, they will wind up in a singular state of the game. The way in which this game implements the concept of DFAs allows the user to be engaged and continues to be engaged in the game to the extent that the desire to fulfill and accomplish each level culminates.

DFAs are able to be derived from the accumulation of NFAs as each input and its outcome or output is considered. In addition, from NFAs to DFAs, the ability to minimize the strewn out DFA can be also produced in which it is able to either eliminate or reduce repeating states in which serves as repetition. Depending on how taxing the player feels as though the riddle is on the given level, their response will determine what occurs thereafter.

Figure 2



A deterministic finite automaton (DFA) is described by a five-element tuple: $(Q, \Sigma, \delta, q_0, F)$.

Q = a finite set of states

Σ = a finite, nonempty input alphabet

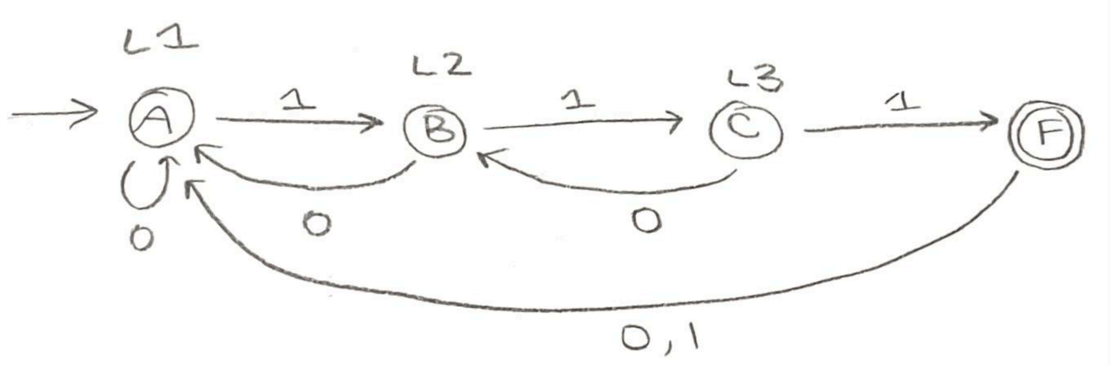
δ = a series of transition functions

q_0 = the starting state

F = the set of accepting states

There must be exactly one transition function for every input symbol in Σ from each state.

DFAs can be represented by diagrams of this form:



Learning Experience Outcome - Nawrin Sikder

In learning about theoretical concepts such as the automata theory with machines such as non-deterministic finite state automata (NFAs) and deterministic finite state automata (DFAs), my understanding of how frequently we encounter machines such as these, has grown exponentially. Grammar, language, NFAs, DFAs, and minimization of DFAs all contribute to attaining a well thought out machine that is not only able to run without error, but be as efficient of a machine as it can be. Learning these concepts not only enhanced my way of thinking, in which it has steered me towards more of an algorithmic approach but it has provided me with more of a structured way in how to solve computational problems. While working on the Batman vs Riddler project and the way in which our code should run, my ability to hone in on the most affective and efficient code was able to be derived. These computational theories have allowed me to problem solve in such a way that being able to take a theoretical concept and apply the same said concept onto a project such as a simple but complex game, is inestimable in value.

Learning Experience Outcome - Syed Bukhari

My learning journey, I have gained proficiency in constructing Non-deterministic Finite Automata (NFA), Deterministic Finite Automata (DFA), and minimizing DFAs. This process involved a deep exploration of formal language theory, allowing me to understand the nuances of computational models and the transition from non-determinism to determinism. The challenge of minimizing DFAs further honed my ability to optimize structures, emphasizing the importance of efficiency in representing language recognition. Beyond theoretical aspects, engaging in programming projects, such as the creation of games like "Batman vs Riddler," has enabled me to seamlessly blend theoretical knowledge with practical application. These projects not only required a profound understanding of automata theory but also demanded creative thinking and problem-solving skills to bring ideas to life. This holistic learning experience has equipped me with a versatile skill set, merging theoretical foundations with the imaginative aspects of software development.

References

1. *Finite State Machines*. Brilliant Math & Science Wiki. (n.d.). https://brilliant.org/wiki/finite-state-machines/#citation-1