GAME-BASED LEARNING WITH AUTOMATA THEORY

Meghana K, M Chaithra,

Anu S M, N Nagamani

Abstract:

This research explores a comprehensive and interdisciplinary approach to game-based learning, leveraging the principles of Automata Theory. There are distinct papers collectively contribute to the development of an integrated educational framework. The initial paper introduces a simulator and game tailored for engineering students, emphasizing active learning in automata theory. Building upon this foundation, the second paper expands the scope, showcasing applications of Computational and Automata Theory across various domains, with a specific emphasis on arcade game design. The third paper seamlessly integrates computation, automata tools, and game theory in the development of a complex roller coaster game. This interdisciplinary approach highlights the practical application of theoretical concepts, extending implications to computer networks, law, and economics. The fourth paper introduces a finite state automata simulator and a robot-based game, incorporating Java and Lego NXT Robot set and many more. This exemplifies a hands-on, active learning pedagogy tailored for theory of computation courses. Collectively, these papers contribute to a game-based learning concept that spans multiple dimensions of Automata Theory. The provided simulator and games offer visual, interactive platforms for building, modifying, and simulating finite state machines, thereby enhancing the learning experience for engineering students. The incorporation of robots and diverse applications in arcade game design and roller coaster simulations further enriches the educational landscape. This integrated framework not only reinforces theoretical concepts but also promotes collaborative learning, active engagement, and motivation among students. The multifaceted applications showcased underscore the versatility of Automata Theory in solving real-world problems, establishing it as a pivotal component in the broader landscape of game-based learning within computer science education.

Keywords—***: Game-Based Learning, Automata Theory, Active Learning, Finite State Machines, Computational Theory, Game Design, Educational Framework.***

I. INTRODUCTION

This research delves into a holistic exploration of automata theory in game design, encompassing distinct papers that collectively contribute to the development of an integrated educational framework. The primary objective is to present a robust game design methodology using automata theory, specifically focusing on the development of an infinite runner game featuring dynamic objects. This involves showcasing the ease of implementation through automaton tools, emphasizing the advantages, such as fewer bugs and faster implementation. Building upon this objective, the second paper in the series focuses on addressing challenges faced by novice learners in understanding fundamental automata concepts. The aim is to enhance their motivation and active participation in the learning process by developing a visual and interactive simulator and a robot-based game for learning automata theory topics, particularly tailored for engineering students. The third paper extends the exploration into the realm of arcade game design, emphasizing computational and automata theory, and game theory's application. The objective is to create an engaging and educational arcade game that can serve as a valuable learning tool. The background emphasizes the increasing trend of using educational games, highlighting their role in accommodating different learning styles. Lastly, the fourth paper aims to leverage automata theory in multidisciplinary computing and scientific research, specifically within the context of game design. The primary objective is to demonstrate the application of non-deterministic finite state automata in designing a roller coaster game. The background provides a comprehensive overview of game theory's relevance in explaining various phenomena and its applications in computer science, telecommunication networks, law, and economics.

Collectively, these papers contribute to a game-based learning concept that spans multiple dimensions of automata theory. The provided simulators, games, and methodologies offer visual, interactive platforms for building, modifying, and simulating finite state machines, enhancing the learning experience for engineering students. The incorporation of robots, diverse applications in arcade game design, and roller coaster simulations further enrich the educational landscape. This integrated framework not only reinforces theoretical concepts but also promotes collaborative learning, active engagement, and motivation among students, establishing automata theory as a pivotal component in the broader landscape of game-based learning within computer science education.

.

II. THEORY/CALCULATIONS/ METHODOLOGY

A. An Infinite Runner Game Design

The Game Hungry bird is endless runner until player hits any hurdle, enemy or fuel runs out. This game is made using infinite loops with some condition on which it breaks the loop and exit to Game End state.

Table 1: Transition Table Mealy Machine

Present State	Next State							Output						
Running State	**Input**	CC	FBC	S	HH			**Input**	CC	FBC	S	HH		
	State	Q1	Q1	Q2	Q4			**Output**	COIN++	FUEL++	FUEL--	GO		
Jump State	**Input**	CC	FBC	SS	BD	HH	FBE	**Input**	CC	FBC	SS	BD	HH	FBE
	State	Q2	Q2	Q3	Q1	Q4	Q4	**Output**	COIN++	FUEL++	FUEL-	FU	GO	GO
Fly State	**Input**	CC	FBC	BD	HH	FBE		**Input**	CC	FBC	BD	HH	FBE	
	State	Q3	Q3	Q1	Q4	Q4		**Output**	COIN++	FUEL++	FU	GO	GO	
Game Over State	HALT STATE													

These games have Seven States.

Game Description:

1) Running State: Game will be in running state and all the game objects will performing according to job assigned.

2) Game Over State: Game will goes to game over state when player hits the enemy, hurdle or fuel barrel ends.

3) Jump State: Health bar will decrease when player is in jump state on "S" input.

4) Fly State: Player will be in fly state when "SS" input comes and it will decrease fuel.

• There will be two background images creating an impression of live/animated moving background while player will be still at its position.

• Four different types of hurdles will come from horizontal path and the player has to avoid them.

• Player can jump by input "S" and on S pressed player will be fly.

• Player will move on input "UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW.

• Fuel will be subtracted from Fuel barrel on jump and fly.

• Fuel will be added on catching fuel bonus barrel..

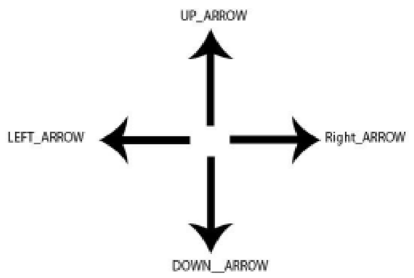


Figure 1- Player Motion Controls

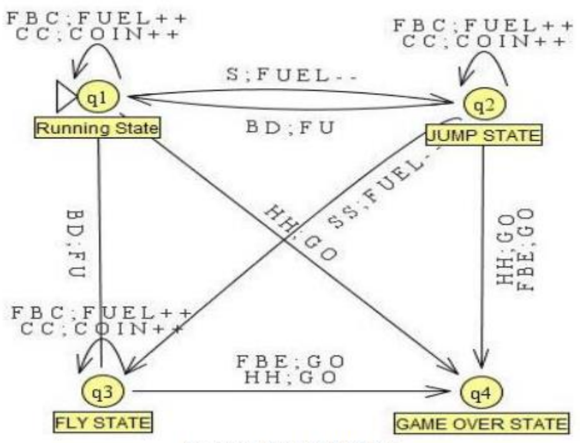


Fig. 2. Main Design Mealy Machine

B. A Game-based Learning System Using Lego NXT Robot

The methodology employed in the research involves the development and evaluation of a Finite State Machine (FSM) simulator and a robot-based game for active learning in theory of computation related courses. The initial phase of the methodology involved identifying the challenges faced by novice learners in understanding basic automata concepts and the lack of motivation to actively participate in traditional lecture-driven teaching methods. Subsequently, the research team designed and implemented a visual and interactive simulator and a robot-based game using Java language, ensuring portability, machine independence, and web-based functionality.

The development process integrated the active construction learning model, emphasizing the role of teachers as facilitators and the importance of collaborative learning environments. The simulator and game were designed to align with these principles, providing a clear workflow of learning activities and ease of use for new users. The methodology also included conducting experiments to evaluate the effectiveness of the integrated environment tools. The preliminary results of these experiments demonstrated improvements in learners' performance and motivation, validating the efficacy of the developed tools.

Furthermore, the methodology involved providing a comprehensive overview of finite state machines and their applications in software and hardware, emphasizing their relevance to various courses including theory of computations, discrete mathematics, programming languages, and compiler design. Overall, the methodology focused on addressing the limitations of existing automata learning tools and providing a model for interactive online collaborative learning tools, specifically tailored to the needs of engineering students.

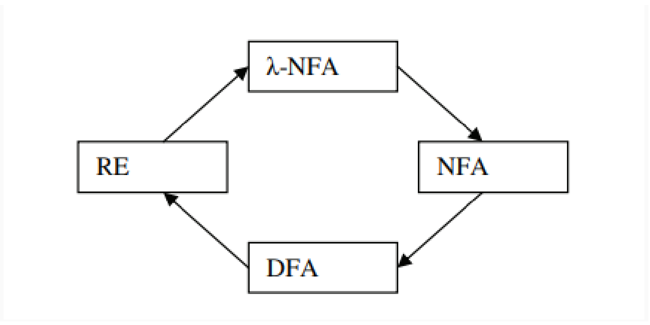


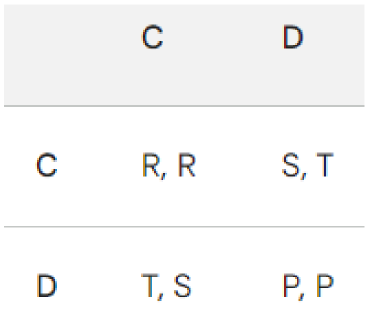
Fig 3:Transformation between Finite state machine and Regular Expression

C. Solution of Goore game using modules of stochastic learning automata

The Goore game, also known as the Gur game, is a simple yet fascinating game with profound implications in distributed coordination and learning automata.The theory of the paper includes

1. Game Structure:

- Players: Finite and identical (symmetric) in terms of actions and rewards.
- Actions: Each player has two choices: cooperate (C) or defect (D).
- Payoffs: Cooperating with a cooperator yields the highest reward (R), while defecting against a cooperator yields the highest individual gain (T), but less than mutual cooperation. Defecting against a defector earns nothing (P).



	C	D
C	R, R	S, T
D	T, S	P, P

Fig 4: Payoff Matrix

2. Equilibrium Analysis:

- Nash Equilibrium: The Goore game has two Nash equilibria: (C,C) and (D,D).
- (C,C) is Pareto Optimal: Both players maximize their joint reward in (C,C).
- Defecting Dilemma: Temptation to defect for individual gain exists, leading to the (D,D) equilibrium, even though it's worse for both players combined.

3. Distributed Coordination:

- The challenge lies in achieving (C,C) without explicit communication or centralized control.
- This is where learning automata come in.

Methodology:

1. Learning Automata:

- Finite-state machines that learn and adapt their behavior based on received rewards.
- Each player is represented by an automaton with internal states and transition rules.
- Actions (C/D) and state transitions depend on past actions and rewards received.

2. LR_1 Algorithm:

- A popular and simple learning automata algorithm for the Goore game.
- Each automaton maintains two internal states: "cooperate" and "defect".
- After each action, the state with higher reward probability is reinforced.

Over time, automata converge to playing C most of the time, leading to (C,C) coordination.

3. Convergence and Properties:

- LR_1 is guaranteed to converge to a Nash equilibrium under certain conditions.
- It is robust to noise and errors in reward signals.
- Can be extended to multi-agent systems and more complex games.

D. LARPA: A learning automata-based resource provisioning approach for massively multiplayer online games in cloud environments

The paper proposes a learning-based resource provisioning approach for MMOG services that is based on the combination of the autonomic computing paradigm and learning automata (LA). The proposed approach is aimed at dealing with fluctuating demands due to variability in the arrival rate of players of the MMOG services. The authors claim that the proposed approach has a remarkable performance in terms of response time, cost, and allocated virtual machines (VMs).

1. Autonomic Computing: LARPA is said to combine utonomic computing principles with learning automata. Autonomic computing emphasizes self-management and adaptation of systems, which aligns with the use of learning automata in LARPA. Learning automata can adjust their behavior based on feedback from the environment, enabling dynamic resource allocation in response to fluctuating MMOG demands.

2. Learning Automata: The paper likely employs specific learning automata models, like the Q-learning model, to learn optimal resource allocation strategies. These models involve states, actions, transitions, and reward functions. The automata choose actions based on their current state and update their internal states based on received rewards (e.g., improved response time or reduced cost) and external observations (e.g., player arrival rate).

3. Feedback Loops: Learning automata rely on feedback loops to improve resource allocation. LARPA likely uses metrics like response time, resource utilization, and cost as feedback signals to guide the automata's learning process.

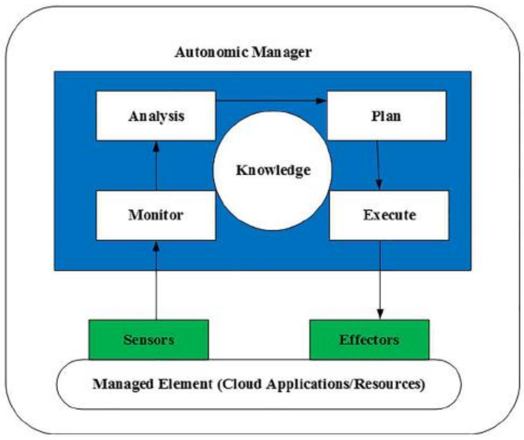


Fig 5. The MAPE-K Feedback loop

E. Hangman–Hangaroo Game Design Using Automata Theory

This section describes a modern single-player version of the classic Hangman game. In the traditional version, there are two players, one of whom thinks the word and the other guesses it. However, in this modern version, there is only one player and the goal is to guess the correct word without making more than six mistakes. The game involves choosing a category, showing a blank space for a word based on the chosen category, and having the player guess the letters from A to Z. The player has six chances to guess correctly, and each mistake results in a character being drawn. part of the figure of the executioner. If the executioner and the character are done, the player loses; otherwise, they win if all the letters are correctly guessed within the given possibilities.

It also mentions a similar game called Hangaroo which follows a similar concept but only allows the player to make three mistakes. A link with automatism theory is highlighted, suggesting that games such as Hangman are easy to play and develop by adding knowledge from different perspectives, especially in the context of language and words. Overall, the section provides an overview of the game's rules, mechanics, and its educational and entertainment aspects.

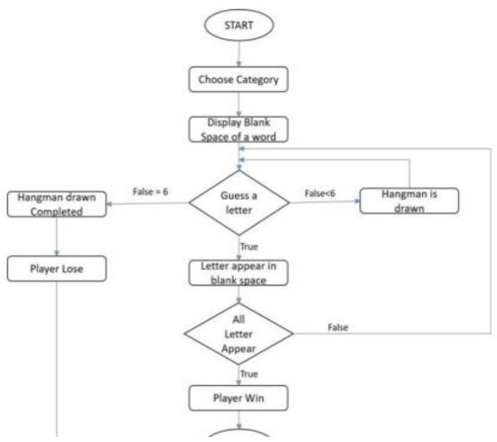


Fig 6: Flowchart of Hangman Game

F. Automata Theory : A Gamification Approach

This section introduces game concepts and mechanics, highlighting their potential to increase user engagement. Gamification has been applied in various contexts such as training, risk management, usability problem solving and marketing. The definition of gamification is presented as a package of services that enhances a core service using a rule-based system with feedback and interaction mechanisms.

The section distinguishes between playfulness and playful design and emphasizes that playfulness includes structural elements, goals and rules. It deals with the application of game design in non-game contexts such as commercial applications and collaborative programs.

Differences between play and serious games:

This article describes several related concepts, including game-inspired design, play, serious games, simulations, and games. All of them use important elements of games to support learning and improve user engagement. Game-inspired design uses ideas common to games without new game elements, while gamification applies game elements in different contexts to influence user behaviour. Serious games are goal-based models like training, while simulations replicate real-world scenarios. Games encompass all these concepts and are primarily meant for entertainment.

Gaming:

Gaming is described as simply awarding points and tokens; it implies understanding and influencing the behaviour of people to encourage certain actions. The section highlights the basics of play in human psychology, behavioural science, computer science, motivation, skill levels, and triggers. The key to behaviour change lies in triggers, skills and motivation, the latter two of which are trade-offs. Effective play fosters engagement and learning with elements such as storytelling, challenge, immediate feedback, curiosity, problem solving, achievement, autonomy and mastery**.**

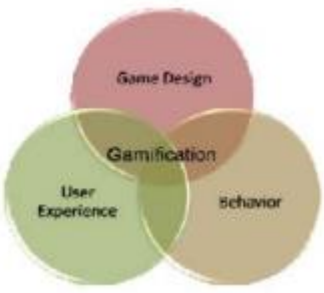


Fig 7: Gamification Overview

Game Mechanics and Dynamics:

This article discusses the definition of game mechanics and dynamics, including statistics, scores, points, leaderboards, achievements, badges, missions, quests, progress, and challenges. This emphasizes that multiple mechanics and dynamics are often used together.

G. An Educational Game Introduction Animal Ecosystem using Finite Automata

Finite State Automata (FSA) to design NPC enemy behavior in a game. FSA is chosen for its simplicity and ease of implementation. The state diagrams, illustrated for easy and normal levels, define NPC actions. In the easy level, players avoid thorny cactus plants, while in the normal level, a moving hunter creates obstacles. The design integrates difficulty progression and conditions to ensure smooth gameplay flow.

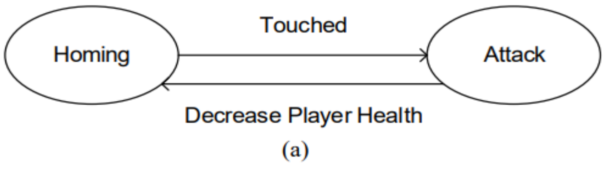
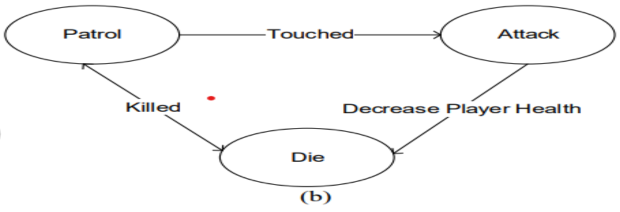


Figure (a) shows, easy level category finite state automata diagram. Figure (b) shows, normal level category finite state automata diagram.



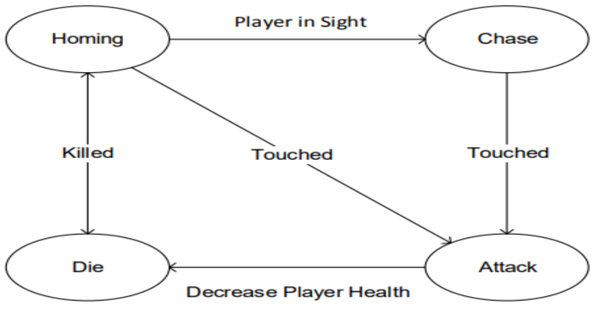
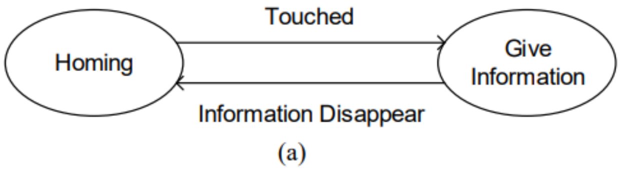
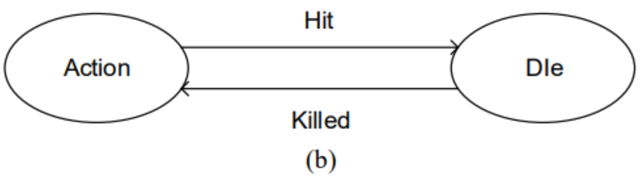


Fig: Finite State Automata Diagram for Hard Level Categories.





Figure(a) shows, animal category finite state automata diagram. Figure 3 (b) shows, finite state automata diagram of category the adventurer main character. If the main character will step on or destroy the enemy / NPC if the enemy has disappeared then continue the game.

H. Procedural Generation of Roguelike Video Game Levels.

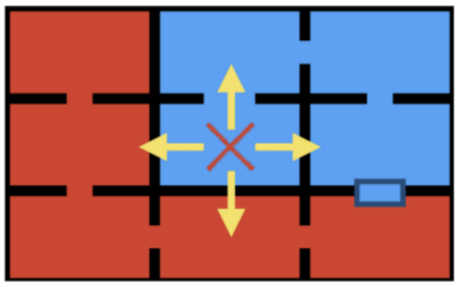
Procedural content generation (PCG) for creating replay-able and enjoyable games, focusing on the roguelike genre known for random levels and permanent player death. It reviews various PCG methods, identifying an opportunity for a hybrid approach to minimize individual technique weaknesses.
prototype level generator for a roguelike dungeon crawler combines mission and level space generators using context-free grammars and cellular automata-inspired rules.

Markov Chains and Random Markov fields: Fast, aesthetically varied, outputs resemble statistically similar distributions of components.

Cellular Automata: Cellular automata (CA) are spatial, discrete time models represented on a uniform grid, which can be used to model different aspects of game environments.

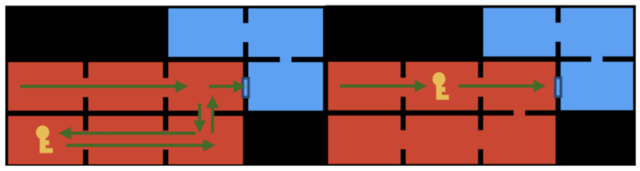
Grammars are linguistic based models consisting of a set of terminal and non-terminal symbols, which are traditionally used in code parsers to validate syntactical correctness.

Machine Learning: Machine learning (ML) techniques are similar to Markov Chains, in that ML is heavily reliant on statistical properties found within datasets of existing manually designed game levels.



(a)

Fig (a): The first CA rule. Subsections shown in red/blue, X denotes active room, and arrow search directions. This scenario shows no valid options for room placement.



(b)

Fig (b): The interestingness of a generated dungeon. Left section shows a more interesting path, whereas the right section shows a more direct path.

I. RESULT AND DISCUSSIONS

A. An Infinite Runner Game Design and Results.

The experimental results in the paper demonstrate the successful implementation of game design using automaton tools, particularly Mealy machines.

The results include a transition state table that illustrates the game scenario, its states, possible inputs, and corresponding outputs. This provides clear evidence of the practical application of automata theory in game design. The observed results affirm the feasibility and effectiveness of utilizing automaton tools in the development of the infinite runner game with dynamic objects. The seamless integration of theoretical concepts from automata theory into a practical and functional game design is evident. The results validate the theoretical framework presented in the paper, showcasing the successful translation of theoretical principles into a real-world application. This substantiates the practical relevance and applicability of automata theory in the context of game design, particularly in the development of dynamic and interactive gaming experiences.

B. A Game-based Learning System Using Lego NXT Robot

The project involved the development of a Finite State Machine Simulator and a Robot-Based Game to facilitate active learning in theory of computation related courses. Implemented in Java, the simulator features a graphical interface for building and simulating finite state machines and integrates with a Lego Mindstorm NXT robot to simulate automaton transitions. The project aimed to address the difficulties novice learners face in understanding basic concepts and to enhance their motivation to actively participate in the learning process. The simulator provides a clear workflow of learning activities and is designed to be easy-to-use and easy-to-learn for new users. It has shown to improve learners' performance and motivation, making it suitable for theory of computation, automata theory, discrete mathematics, computational models, programming languages, and compiler design courses. The integration of robots into the learning process is a novel approach, attracting less motivated students. The robot can display inputted strings, start motion according to the inputted string and automaton, and display whether the automaton accepts or rejects the input. Overall, the project has demonstrated the effectiveness of integrating interactive and online collaborative tools into the learning environment, providing a model for a wide range of topics at the undergraduate level.

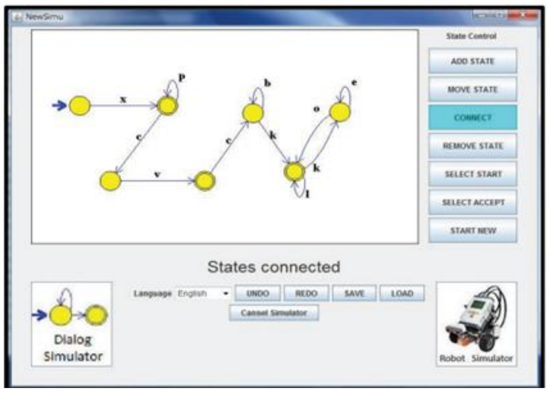


Fig 8: The simulator's Interface

C. Solution of Goore game using modules of stochastic learning automata

1. Equilibrium Analysis: As mentioned previously, the Goore game has two Nash equilibria: (C, C) and (D, D). The (C, C) equilibrium, where all players cooperate, is Pareto optimal as it maximizes the joint reward for all players. However, the game faces the "defecting dilemma," where it's individually more tempting to defect and exploit others, leading to the less beneficial (D, D) equilibrium.

2. Learning Automata Performance: Research shows that learning automata algorithms like LR_1, Q-learning, and SARSA can successfully achieve cooperation in the Goore game. Studies have demonstrated how these algorithms converge towards the (C, C) equilibrium with high probability, particularly under conditions where cooperation offers sufficient advantages over defection.

3. Applications: The Goore game serves as a model for various real-world scenarios involving decentralized coordination and resource management. Insights gained from the game can be applied in fields like:

- Resource allocation: Coordinating agents in distributed systems to share resources efficiently.
- Sensor networks: Optimizing data collection and communication among decentralized sensors.
- Distributed optimization: Solving optimization problems involving multiple agents without a central controller

D. LARPA: A learning automata-based resource provisioning approach for massively multiplayer online games in cloud environments

1. Reduced Response Time: If the automata learn to allocate adequate resources promptly as player numbers increase, response times for players could improve.

2. Lower Cost: Efficient resource allocation based on demand might lead to cost savings by avoiding unnecessary VM provisioning.

3. Optimal VM Allocation: Learning automata, by continuously adapting to demand fluctuations, might achieve optimal VM allocation, using only the necessary resources to maintain performance.

E. Hangman–Hangaroo Game Design Using Automata Theory

Formal definition: The automaton designed for Hangaroo is a finite automaton with five times denoted by M = (Q, ∑, δ, q0, F). The official definition includes:
states (Q): {Start Game, Choose Class, Guess Star, Design Hangaroon, Win, Lose, Game End}.

alphabet (∑): {sg, cg, ig, hi, hc, wc}.

Transition function (δ): Represents transitions between states based on input alphabetic symbols. Certain transitions determine the flow of a Hangaroo game from one state to another.

Start state (q0): {Start game} indicating the starting point of the game.
end states (F): {End}, indicates the state where the game ends.

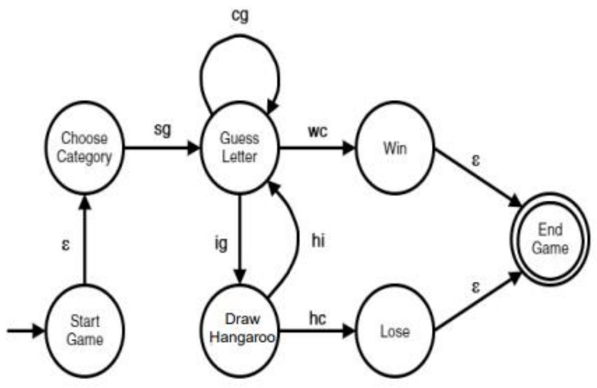


Fig 9: State Transition Diagram

F. Automata Theory : A Gamification Approach

The study emphasizes that effective gamification is consistent with the goals of the system and its users. A case study of the concept of a gamified social network is presented, emphasizing three key factors: intrinsic motivation, game mechanics and immersive dynamics. In this context, user engagement is considered effective if the gamified system provides feedback and ensures that users receive information after a certain level of engagement. The importance of feedback is emphasized, as bad feedback can weaken user engagement and cause the game platform to fail. The effectiveness of gamification depends on how well it promotes the specific purpose of the system and fulfills the goals of the users. Overall, research suggests that incorporating automation theory into gamification frameworks can improve their effectiveness in achieving both system and user goals.

G. An Educational Game Introduction Animal Ecosystem using Finite Automata.

An educational game introducing animal ecosystems, involves various menu displays and gameplay elements. The start menu includes the game's name, a play button, sound controls, tones, and game pointers. The home menu displays buttons for hints, sound toggling, music toggling, exit, and ecosystem options.

The level menu showcases difficulty options: easy (dangerous plants), moderate (patrolling hunter), and hard (chasing hunter). Different ecosystem levels, such as desert, polar and forest, provide diverse gameplay environments.

Each level introduces animals with informational popups. Questions are presented through a popup layout, when picking up boxes, requiring players to select multiple answers.

H. Procedural Generation of Roguelike Video Game Levels.

A hybridized approach for PCG of game levels, tested in a roguelike dungeon-style level generator. The proposed method, based on generative grammars and cellular automata-inspired behavior, aimed to create a simple and effective dungeon generator. We introduced a new heuristic-based evaluation method and tested various variations of our basic generation rules. Random Neighbor Search, faced challenges in generating completable levels. Only 53.2% of the time, it successfully placed an end room and created a playable level. However, two other methods, with a wider pool of available actions, consistently generated playable levels without encountering generative failures. The Path Difference heuristic, designed to assess the layout's linearity, favored the Persistent Subsection Search approach, which achieved a higher score.

The results indicate that our hybrid approach effectively overcomes generative weaknesses, producing interesting procedurally generated content. The aesthetic results align with similar experiments, providing varied yet structured dungeon layouts. Our method proves sufficient as a constructive PCG approach, preventing unplayable levels during generation. Moreover, the heuristic allows for potential use as a search-based approach, although the speed considerations should be taken into account. Overall, hybridized generative techniques appear promising for further investigation and refinement, offering a balance between simplicity, effectiveness, and variety in generating game content.

II. CONCLUSION AND FUTURE SCOPE

A. *Conclusion*

In conclusion, the integration of automata theory in education and game design has shown promising results. The Finite State Machine Simulator and Robot-Based Game have been well-received by students, indicating interest and convenience in learning automata theory. Despite positive feedback, areas for improvement in the simulator have been identified, with plans to enhance graphical editing and eliminate command prompt usage. The benefits of using automaton and computational theory tools in game design are evident, showcasing fewer bugs, simplified design processes, and effective handling of invalid inputs. The future scope involves refining the simulator, integrating tools comprehensively, and exploring advanced artificial intelligence in game design. There's potential for broader applications in solving real-life complex problems, extending the behavioral model, and applying automata theory to diverse scenarios.

B. *Future Scope*

The future direction of this research includes refining the Finite State Machine Simulator, streamlining user interfaces, and integrating the simulator and robot game into a comprehensive set of automata tools. Further enhancements aim to improve usability for both novice and advanced learners. Additionally, the exploration of advanced artificial intelligence using complexity theory in game design is proposed. Extending the application of automata theory to solve real-life complex problems and refining behavioral models for broader adaptability are avenues for future research. The potential integration of automata theory with software engineering practices and the optimization of gaming strategies opens new dimensions for multidisciplinary computing and scientific research.

III. APPENDIX

Comprehensive Methods and Development Information

I. Finite State Machine (FSM) Simulator and Robot-Based Game Development

This section provides a comprehensive explanation of the methods used to develop the Finite State Machine (FSM) simulator and the related robot-based game. The chosen programming language, Java, is based on its suitability for portability, machine independence, and inclusion of web-based features. Architecture diagrams, code snippets, and design considerations are included to give readers an overview of the technical aspects of the development process.

II. Identifying Challenges and Learning Gaps

The initial phase of the methodology identified the challenges that novice learners face in understanding basic automata concepts. This section details methods such as surveys, interviews or focus groups to gather insights into specific student difficulties. A comprehensive overview of identified challenges and learning gaps is provided, which lays the foundation for further development of the simulator and game.

III. Integration of the Active Constructive Learning Model

This section describes the seamless integration of the Active Constructive Learning Model into the simulator and game design. It introduces the principles of active learning, collaborative environments and the role of teachers as facilitators of learning. Descriptions of aligning simulators and games with these principles are provided, emphasizing the importance of user-friendly interfaces and engaging learning activities.

IV. Overview of Finite State Machines and Applications

To contextualize the importance of the tools, this section provides a comprehensive overview of Finite State Machines (FSMs) and their various applications in software and hardware. Methods for introducing the significance of FSMs to a variety of courses including theory of computation, discrete mathematics, programming languages, and compiler design are discussed

IV. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our Professor, Ms. Sini Anna Alex for her invaluable guidance throughout the course of this project and assistance on writing the Technical Report on Game Based learning with automata theory.

V. REFERENCES

1. Norman Qureshi, “Computing Game Design with Automata Theory”, Engineering University of Engineering & Technology, Lahore 2012.
2. Zohaib Abbas, “A Roller Coaster Game Design using Automata Theory”, Engineering, University of Engineering & Technology, Lahore, 2012.
3. Visch, V. T., et al. "Persuasive Game Design: A model and its definitions." CHI 2013: Workshop Designing Gamification: Creating Gameful and Playful Experiences, Paris, France, 27 April-2 May 2013. ACM, 2013.
4. Nelson, Mark J., and Michael Mateas. "Towards automated game design." AI* IA 2007: Artificial Intelligence and Human-Oriented Computing. Springer Berlin Heidelberg, 2007. 626-637.
5. Claypool, Kaval, and Mark Claypool. "Teaching software engineering through game design." ACM SIGCSE Bulletin 37.3 (2005): 123-127.
6. Andrew Rolling and E. Adams. On Game Design. New Riders Publishing, 2003.
7. H. Bergstrom, "Applications, Minimization, and Visualization of Finite State Machines." Master Thesis. Stockholm University, 1998. Related website at: http://www.dsv.su.se/~henrikbe/petc/.
8. J. Bovet, "Visual Automata Simulator, a tool for simulating automata and Turing machines." University of San Francisco. Available at: http://www.cs.usfca.edu/~jbovet/vas.html, 2004.
9. N. Christin, "DFApplet, a deterministic finite automata simulator." Available at: http://www.sims.berkeley.edu/~christin/dfa/, 1998. 1952 Mohamed Hamada et al. / Procedia Computer Science 4 (2011) 1944–1952.
10. S. Hadjerrouit, "Toward a constructivist approach to e-learning in software engineering." Proc. E-Learn-World Conf. E-Learning Corporate, Government, Healthcare, Higher Education, Phoenix, AZ, pp. 507-514, 2003.
11. M. Hamada, "An Integrated Virtual Environment for Active and Collaborative e-Learning in Theory of Computation." IEEE Transactions on Learning Technologies, Vol. 1, No. 2, pp. 1-14, 2008.
12. E. Head, "ASSIST: A Simple Simulator for State Transitions." Master Thesis. State University of New York at Binghamton. 1998. Related website at: http://www.cs.binghamton.edu/~software/.
13. LEGO.com MINDSTORMS: "What is in the box?"[Online] Available: http://mindstorms.lego.com/en-us/history/default.aspx.
14. LEJOS Java for LEGO Mindstorms [Online] Available: http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/Intro.htm.
15. LEJOS Java for LEGO Mindstorms [Online] Available: http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/GettingStartedWindows.htm.

16. LEJOS java for LEGO Mindstorms [Online] Available: http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/CompileAndRun.htm.
17. M. Mohri, F. Pereria, and M. Riley, "AT&T FSM Library." Software tools. 2003. Available at: http://www.research.att.com/sw/tools/fsm/.
18. S. Rodger, "Visual and Interactive tools." Website of Automata Theory tools at Duke University, http://www.cs.duke.edu/~rodger/tools/. 2006.
19. "Transforming undergraduate education in science, mathematics, engineering, and technology." In “Committee on Undergraduate Science Education”, Center for Science, Mathematics, and Engineering Education. National Research Council ed. Washington, DC: National Academy Press, 1999.
20. G. Wilson, Ed., "Constructivist Learning Environments: Case Studies in Instructional Design." Englewood Cliffs, NJ: Educational Technology, 1998.