Evaluating the Impact of a Graph Grammar-Based Activity for Introducing Automata Theory in K-12*

Júlia Veiga da Silva, Federal University of Pelotas,
jvsilva@inf.ufpel.edu.br, https://orcid.org/0000-0002-6839-3117

Simone André da Costa Cavalheiro, Federal University of Pelotas,
simone.costa@inf.ufpel.edu.br, https://orcid.org/0000-0002-7442-7379

Luciana Foss, Federal University of Pelotas,
lfoss@inf.ufpel.edu.br, https://orcid.org/0000-0002-0473-4111

Abstract. This paper presents the application of an educational activity for K-12 students, aligned with the National Common Curricular Base, which explores Automata Theory through Graph Grammar. Formal Methods, important in complex systems, promote skills such as abstraction, analysis, and critical thinking, which are essential for 21st-century students. Despite the growing interest in Computing Education, formal specifications, part of Formal Methods, remain underexplored. Thus, this activity focused on Deterministic Finite Automata aims to bridge this gap. A pilot experiment with a group of students was conducted, and the results indicated that the students not only understood the concepts related to automata but also successfully manipulated the grammars with the tool used.

Keywords: computing education, k-12, automata theory, graph grammar.

Avaliação do Impacto de uma Atividade Baseada em Gramática de Grafos para Introdução à Teoria dos Autômatos na Educação Básica

Resumo. *Este trabalho apresenta a aplicação de uma atividade educacional para estudantes da Educação Básica, alinhada à Base Nacional Comum Curricular, que explora a Teoria dos Autômatos por meio de Gramática de Grafos. Os Métodos Formais, importantes em sistemas complexos, promovem habilidades como abstração, análise e pensamento crítico, essenciais para os estudantes do século XXI. Apesar do crescente interesse pela Educação em Computação, a especificação formal, parte dos Métodos Formais, ainda é pouco explorada. Assim, a atividade focada em Autômatos Finitos Determinísticos busca minimizar essa lacuna. Um experimento piloto com um grupo de alunos foi realizado e os resultados indicaram que os estudantes compreenderam tanto os conceitos relacionados aos autômatos, quanto souberam manipular as gramáticas na ferramenta utilizada.*

Palavras-chave: *educação em computação, educação básica, teoria dos autômatos, gramática de grafos.*

1. Introduction

As computer systems become more complex, the need for precise specifications to describe their expected behavior becomes increasingly critical. Formal specifications, part of Formal Methods, provide a rigorous, mathematical approach to defining system requirements and behaviors, avoiding the ambiguities and misinterpretations often arising from informal specifications (RIBEIRO, 2000). This precision is essential not only for ensuring implementation correctness but also for solving problems effectively, as poorly defined solutions can lead to

*This work was supported by CAPES – Brazil – Financing Code 001, the SACCI Network, SMED/Pelotas, PREC, and PRPPG / UFPel.

critical errors in complex systems. To create a formal specification, a Formal Specification Language (FSL) with well-defined syntax and semantics is necessary to ensure precision in the specification.

In a scenario where systems are increasingly critical and sensitive to failures, the use of FSL to ensure the correctness and reliability of complex systems has expanded, extending into the field of education. Computing Education, once primarily focused on technical programming skills, is now broadening to include more fundamental and theoretical areas, such as FSL, which fosters skills in abstraction, formal reasoning, and problem-solving. This shift is evident in the expansion of computing curricula to K-12[1] education, where the development of these skills is becoming increasingly valued (NOBLE *et al.*, 2022). Consequently, FSL plays a crucial role in Computer Science education, going beyond coding to promote a deeper understanding of the principles underlying computational systems (BROY *et al.*, 2024).

This growing attention to FSL in the educational context reflects one aspect of the transformation that Computing Education is undergoing. This transformation has broadened its focus in higher education, extending beyond technical programming to include more foundational and theoretical areas. The field is evolving into a more general-purpose discipline, even within K-12 education, with an emphasis on problem-solving. This shift is encapsulated in the concept of Computational Thinking (CT), which posits that computing extends beyond programming and encompasses a set of problem-solving skills, such as abstraction, decomposition, automation, and evaluation, benefiting everyone – not just computer scientists (WING, 2006).

Despite the expansion of research on CT in Brazil in recent years (FRANÇA e TEDESCO, 2019), the approval of the norms on computing in K-12 – in addition to the National Common Curricular Base (BNCC) (BRAZIL, 2022) –, which designates CT as one of the three main core areas for developing specific skills in computing, presents several challenges for educators. Issues such as professional training, the availability of appropriate teaching materials, and the implementation of effective methodologies pose significant hurdles. To facilitate this change, it is essential to create activities that promote the development of the learning objectives outlined by the BNCC for each level of K-12 education.

Although Computing Education has expanded with the rise of CT reached K-12, its efforts have predominantly focused on areas such as visual programming, primarily using Scratch (KITE; PARK e WIEBE, 2021), games (KRATH; SCHÜRMANN e von KORFLESCH, 2021), and robotics (YANG; LIU e CHEN, 2020). On the other hand, Theoretical Computing, where FSL is inserted, receives less attention in education. To bridge this gap, an educational activity based on FSL concepts was proposed (SILVA *et al.*, 2024). The activity aims to develop the EF09CO03 skill outlined in the BNCC, which involves using automata to describe behaviors abstractly and automate them through an event-based programming language. It focused on the Deterministic Finite Automata (DFA) concept and was developed using GrameStation (SILVA JUNIOR; CAVALHEIRO e FOSS, 2021). GrameStation is a game engine based on Graph Grammar (GG), the language employed for activity specification. Thus, this work aims to present the results of the application and evaluation of this activity.

The rest of this work is organized as follows: Section 2 presents the theoretical background; Section 3 describes the methodology, including a synthesis of the Systematic Literature Review (SLR) conducted, which is detailed in (SILVA; CAVALHEIRO e FOSS, 2024), as well as the methods used for the development and evaluation of the activity; Section 4 gives a brief presentation of the activity; Section 5 outlines the application and evaluation of the activity; Section 6 presents the results and discussion; and Section 7 concludes the work and suggests directions for future research.

[1]In this work, “K-12” refers to primary and secondary education, ranging from kindergarten through high school.

2. Theoretical Background

This section outlines the theoretical context of our study. It addresses the definitions of Automata Theory – focusing on DFA – and GG, and introduces the tool employed to create and apply the activity.

2.1. Automata Theory

Automata are FSLs that can be used for abstract modeling of systems that follow a specific set of instructions to perform particular tasks. It can be envisioned as a machine comprising a finite set of states capable of reading input symbols, transitioning between states according to the symbols encountered, and, in certain cases, producing an output. Automata are classified into various types, including Finite Automata and Pushdown Automata. This work, however, concentrates on DFA, a specific subcategory of Finite Automata. Formally, a DFA is defined as follows (MOGENSEN, 2024):

A DFA is represented as a tuple $M = (Q, \Sigma, \delta, q0, F)$, where: Q denotes a finite set of states; Σ signifies a finite set of input symbols, referred to as the input alphabet; $\delta : Q \times \Sigma \to Q$ represents the transition function that maps a state and an input symbol to a new state, though it is not necessarily defined for every possible combination of state and input symbol; $q0 \in Q$ is the initial state; and $F \subseteq Q$ indicates the set of accepting (or final) states.

Thus, the automaton processes symbols from an input alphabet and transitions between states based on these symbols and its transition function, δ. This function maps a current state and an input symbol to a new state. The transition occurs sequentially for each symbol in the input, beginning with the initial state, $q0$. After processing the entire input string, if the DFA is in an accepting state (belonging to the set F), the input string is accepted by the DFA, signifying that it is part of the language recognized by the DFA. Conversely, if the DFA is in a non-accepting state or fails to complete the reading of the input string, the input string is not recognized by the DFA.

2.2. Graph Grammar

A GG models a system by representing its states as graphs composed of vertices and edges, while defining events that can change its current state as a set of graph transformation rules (EHRIG *et al.*, 1997). It is essential for a GG to specify how its state graph, known as the *initial graph*, is initialized. Additionally, a GG differentiates and restricts its elements through a *type graph*, which defines the various components of the system. It also outlines a *set of rules* that describe the potential state changes within the system.

As an example, Figure 1 illustrates the type graph (on the left) and the initial graph (on the right) of the Pac-Man game represented as a GG. The type graph specifies the elements that compose the game, while the initial graph depicts a Pac-Man, a ghost, and fruits arranged on a 3x4 grid, along with a counter (represented by a pink triangle) related to Pac-Man to track the number of fruits consumed. Specifically, the Pac-Man game includes four rules (Figure 2): *PacMove*, *GhostMove*, *PacEat*, and *GhostKill*, each defined by a pair of graphs connected through a graph homomorphism (HAHN e TARDIF, 1997).

The graphs representing each rule consist of a Left-Hand Side (LHS), which sets a condition for the rule to be applied, and a Right-Hand Side (RHS), which specifies the consequence of applying the rule. In *PacMove* (Figure 2), for instance, the LHS defines the condition where Pac-Man is located in a place connected to another, while the RHS defines the consequence by removing Pac-Man from the initial location and placing it in another. This structure also requires mappings between elements of the graphs (morphisms), indicating for each element in one graph its corresponding element, if any, in the other graph. When an element

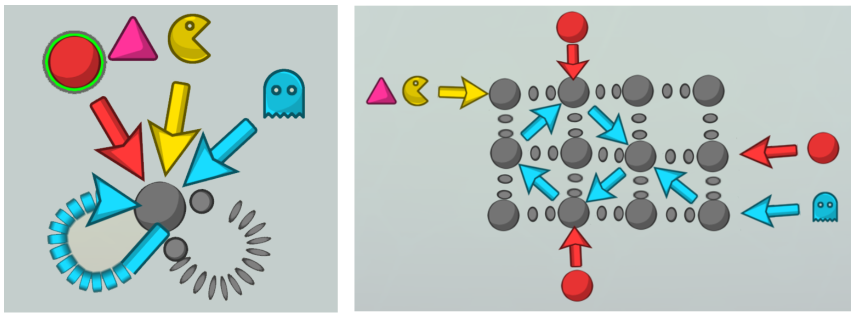

Figure 1: Type graph (left) and initial graph (right) of the Pac-Man game as Graph Grammar

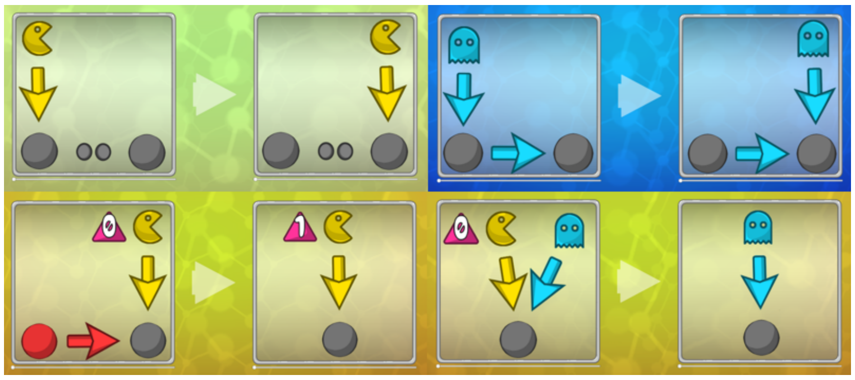

Figure 2: *PacMove* (top, left), *GhostMove* (top, right), *PacEat* (bottom, left), and *GhostKill* (bottom, right)

is mapped successfully, it means the rule preserves it, as with Pac-Man in *PacMove* (where Pac-Man's position is preserved while its connection to the place is removed and recreated). If an element in the LHS remains unmapped, the rule deletes it, as in Pac-Man's deletion in *GhostKill*. Conversely, if an element is in the RHS but unmapped, the rule creates it. To apply a rule, thereby changing the system's current state, it is necessary to find a match by mapping the LHS elements to their corresponding elements in a state graph.

2.3. GrameStation

GrameStation (SILVA JUNIOR; CAVALHEIRO e FOSS, 2021) is a tool based on GG designed for creating and executing games that follow this formal modeling language. Since games are represented as GGs, the tool also fosters the development of skills associated with CT. These skills are developed (SILVA JUNIOR, 2020) by both the game creator (who specifies the GG) and the player (who simulates the GG). GrameStation is structured into three modules: Grame Explorer, Grame Builder, and Grame Player, which enable users to locate, create, and execute games, respectively.

Within the tool, the type graph serves as a declaration area, the initial graph represents the game's starting layout, and the rules outline the player's possible actions. To play a game (using the Grame Player module), users can select, map, and apply the specified rules throughout gameplay. While the game runs in Grame Player, the specified rules appear at the top of the screen. The LHS and RHS graphs are displayed, allowing the player to establish a match by selecting elements in the initial graph corresponding to the highlighted elements in the LHS. If an invalid mapping occurs, such as mapping a Robot to a Ship, an error symbol appears on the screen, and the current match is canceled.

3. Methods

An SLR was conducted in (SILVA; CAVALHEIRO e FOSS, 2024) to investigate how Automata Theory is addressed in Computing Education. This review identified best practices and existing gaps that informed the development of the proposed activity (SILVA *et al.*, 2024). The findings from the SLR emphasized the advantages of using interactive tools and gamification, as interactive simulations and game-like elements enhanced student engagement and comprehension, making abstract concepts more accessible through active learning. Gamified activities were also noted to promote motivation, collaboration, and competition, potentially improving learning outcomes. However, assessment practices in this area often lacked formalization, relying on custom methods, which highlighted the need for standardized approaches to evaluate student learning consistently. In Brazil, despite recent BNCC updates incorporating Computing Education in K-12, research and educational initiatives related to Automata Theory remain limited, underscoring the need for resources and teacher training to fully integrate these methods.

The creation of the activity was based on the ENgAGED method. ENgAGED (EducatioNAl GamEs Development) is a process designed for developing educational games for computer teaching, integrating principles of both instructional design and game design (BATTISTELLA, 2016). This process is structured into five main phases based on instructional design, with phase 3 — focused on the Development of the Educational Game — divided into five additional subphases that specifically address game design. Considering that creating a game modeled as a GG involves well-defined tasks, which is considerably different from creating games in other models, a series of adaptations to the ENgAGED process is proposed to facilitate the development of educational games using the GradeStation platform (SILVA *et al.*, 2021). The proposed process modifies the original ENgAGED methodology to suit the GrameStation environment, incorporating new phases and simplifying steps that the platform automates.

The activity was evaluated using MEEGA+KIDS (VON WANGENHEIM; PETRI e BORGATTO, 2020). This questionnaire aims to assess the quality of educational games regarding usability and player experience from students' perspective within the context of Computing Education in K-12. The model consists of three categories: demographic information, usability, and player experience. The response format for items in the usability and player experience categories is based on a 5-point Likert scale (DEVELLIS, 2003), with options ranging from -2 ("strongly disagree") to 2 ("strongly agree"). In this model, usability is defined as the degree to which a product (educational game) can be effectively and efficiently used by specific users (students) to achieve particular goals in a specific context of use (K-12) (VON WANGENHEIM; PETRI e BORGATTO, 2020). In contrast, player experience is a quality factor encompassing the depth of a student's involvement in the game task (VON WANGENHEIM; PETRI e BORGATTO, 2020).

The questionnaire underwent several adaptations to address the specific requirements of this study. In particular, the "social interaction" dimension was excluded since GrameStation does not currently support online play. Additionally, items that referenced the subject matter covered by the activity (e.g., "I learned the subject content with this game.") were omitted because the activity is not directly related to any of the subjects in the students' original classroom curriculum, which the questionnaire item addresses. Finally, three questions were added to gather data regarding the students' perceptions of automatons: "What did you find easiest about the game?"; "What did you find most difficult about the game?"; and "Which of the concepts presented did you find most challenging?".

4. Activity

The methodological steps involved the proposal's conception, and the detailed proposed tasks are presented in (SILVA *et al.*, 2024). The activity places students as a robot explorer on a space mission, with each task designed to gradually introduce concepts such as initial and final states, undefined transitions, and word acceptance. Table 1 provides a brief description of each task.

Table 1: Design of tasks at different levels

Approach	**Description**
#1	Students analyze a complete tape to predict the final state of the automaton before simulating its execution. This task introduces the concepts of states and transitions.
#2	Students specify an initial state and observe the final state after each simulation, varying the initial state to explore different outcomes. This task highlights the importance of the initial state and transitions.
#3	Students have to verify whether the automaton reaches a final state when processing a sequence of symbols, addressing the concept of acceptance or rejection of a word.
#4	Students analyze whether the robot completes the reading of a tape, considering the possibility of undefined transitions. This exercise explores the concept of undefined transitions.
#5	Students create a sequence of commands to move the automaton from an initial state to a final state, addressing initial and final states, transitions, and acceptance or rejection of a word.
#6	Students must identify the instructions necessary to complete the task, eventually recognizing only a subset of the language. This introduces the concept of recognized language.
#7	Students create an automaton that accepts a specific, regular language, emphasizing automata specification.

5. Activity Application

To test the feasibility of the activity, a pilot experiment[2] was conducted with nine students. The experiment was conducted in person, with all participants being K-12 students (7th grade, 9th grade, and 1st year of high school) from four schools (Mario Quintana, São José, SESI e IFSul) in the city of Pelotas. A range of ages was selected to determine if age differences would affect the results. Prior to the activity, all guardians of the participants read and signed the Free and Informed Consent Form (TCLE), indicating their voluntary consent for their children to participate in the research and their awareness of how the data would be used.

Before the activity started, students were introduced to the basics of Automata Theory necessary for completing the task, such as states (initial and final), transitions, tape (and symbols), words, and the acceptance or rejection of a word. These concepts were presented[3] appropriately for the students' age and grade level, using ludic examples and simulations. Additionally, the procedure for performing the activity in GrameStation was explained without delving into the details of GG. Specifically, the concepts of rules and rule application (related to GG) were presented simply as "the way to play" in GrameStation. The explanation was delivered to the

[2]Project approved by the ethics committee. CAAE: 73891417.0.0000.5317.
[3]The materials used are available at: https://bit.ly/4fc4VYU.

students as a group, while the activity itself was carried out individually (student and tutor). The tutor was responsible for outlining the tasks that the student needed to perform, initiating the first tape of the first task together with the student, but refraining from interfering with the execution of the remaining tasks.

The activity involved an automaton with five states (Figure 3), requiring students to complete two tasks: identify the final state and determine whether a word would be accepted or rejected by the automaton. In the first task, the student received two different and complete tapes and had to navigate the robot (the main character) through the automaton (by applying rules to the grammar) until reaching the final state. To accomplish this, they needed to apply transition rules. In the second task, the student received three additional tapes and had to ascertain which state the robot would reach after processing the sequence of symbols, identifying the final state. For this task, in addition to the transition rules, they also had to apply a rule that distinguished the final state from the others. Finally, the student analyzed the state where the robot stopped and indicated whether the word was accepted or rejected.

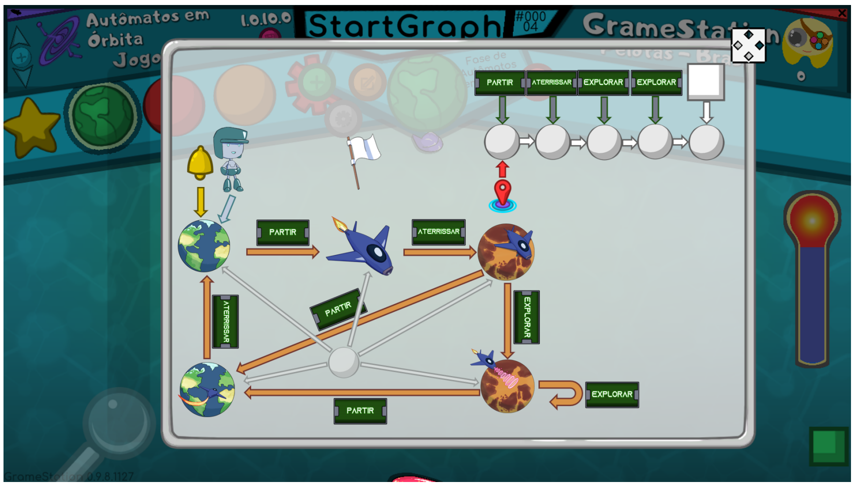

Figure 3: Initial graph of the task in GrameStation

6. Results and Discussion

All nine participants completed the activity, and all responses were included in the analysis. Figure 4 presents the demographic information of the participants: 33% (3) were female, and 67% (6) were male. In terms of age, 22% (2) were 12 years old, 11% (1) was 14 years old, 45% (4) were 15 years old, and 22% (2) were 16 years old. Figure 4 illustrates the percentage of students from different institutions and their respective school grades. Additionally, it provides information on the participants' familiarity with both digital and non-digital games.

Figures 5 and 6 display the results concerning usability and player experience based on the participants' responses. Each item's rating is represented by the median, ranging from -2 ("strongly disagree") to 2 ("strongly agree"). The evaluation of the activity revealed a mostly positive reception regarding usability. Most participants found the design appealing and noted that the combination of text, colors, and fonts was well coordinated. The ease of learning and gameplay received favorable ratings, with 6 out of 9 respondents finding the game easy to learn and play. While text readability was generally well-rated, there were some discrepancies. The clarity of the game rules garnered divided opinions, with 4 participants considering the rules clear and 5 expressing mixed feelings. Conversely, all participants strongly agreed that the colors used were easy to understand.

The medians indicated in the final column suggest that for most questions, the median

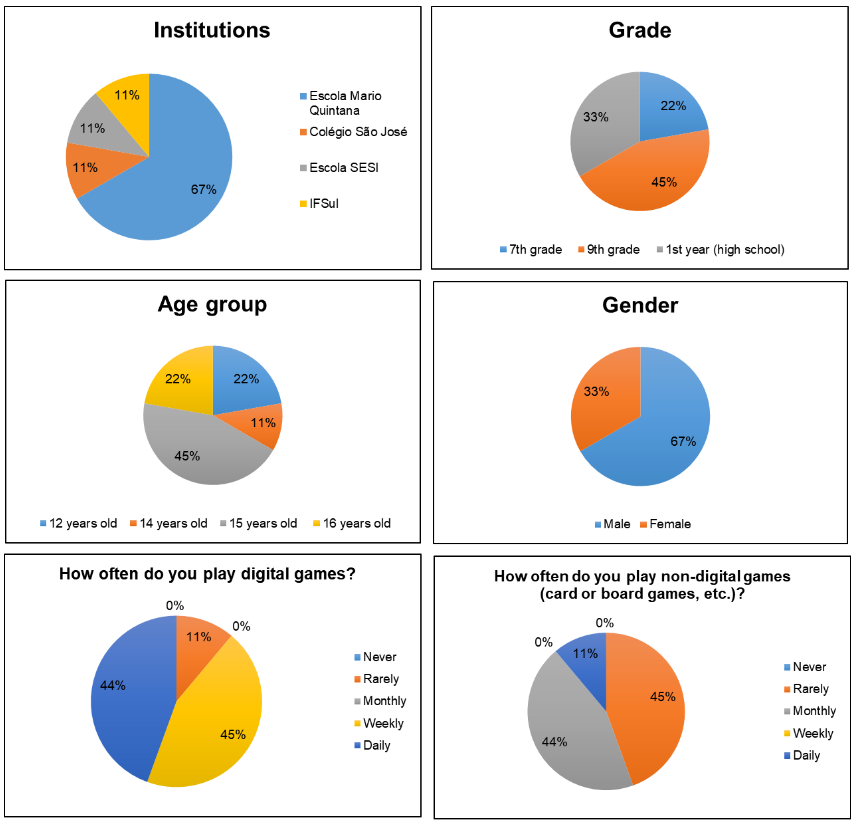

Figure 4: Demographic information of participants

response was "agree", except for the question concerning the game rules, where the median was "neutral". Overall, the game was well-received in terms of design, ease of learning, and gameplay. However, the game rules may require further clarification, as this was the only area with a "neutral" median and a higher number of negative responses.

Participants' experiences with the game were predominantly positive, though some variations were noted. Most participants (6) believed that the content's organization contributed to their confidence in learning. They felt that the game offered new challenges at an appropriate pace and did not become monotonous. Perceptions of the challenge varied; more than half either disagreed or felt neutral, while 3 participants found the game challenging. Satisfaction upon completing the tasks was high, with 7 participants expressing satisfaction and 5 believing that their progress reflected their effort.

Analyzing the data by age group (7th grade, 9th grade, and 1st year of high school)[4], we found that, despite variations in the perceived challenge and pace of the game across different grade levels, some aspects of the student experience remained consistent. Across all age groups, students demonstrated high confidence in the learning provided by the game and reported enjoyment and fun while playing, feeling that the game was not monotonous. However, the perceived level of engagement, including the tendency to lose track of time, showed significant differences, with younger students being more engaged and finding the game more challenging. Furthermore, in our experiment, the preference for traditional learning methods increased with age, with older students expressing a inclination toward these methods, indicating a possible need to adjust the activity to better meet their expectations.

Regarding the essay questions[5], the analysis focused on summarizing the key points from the students' perceptions of the activity. The responses revealed varied perceptions of

[4]Data divided by groups is available at: https://bit.ly/4fc4VYU.
[5]Available at: https://bit.ly/4fc4VYU.

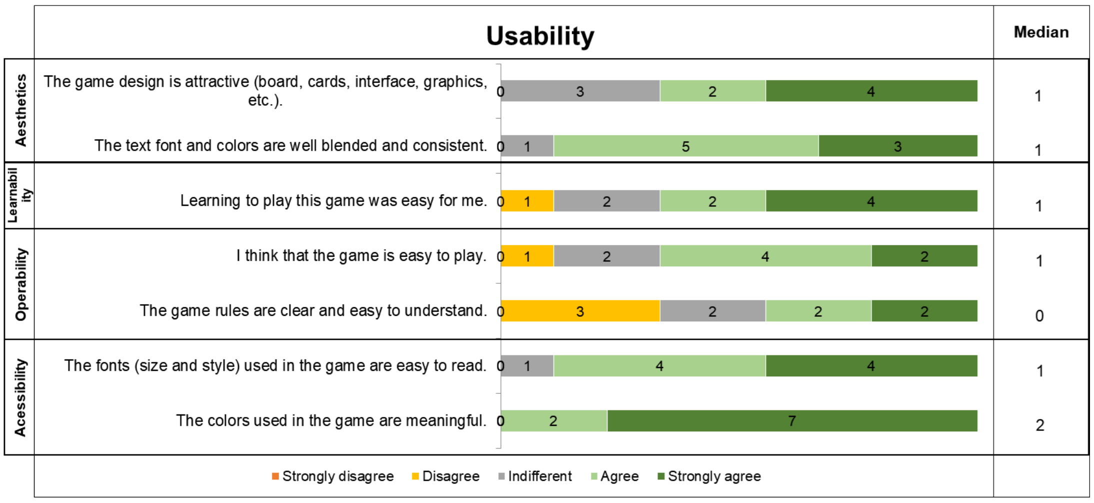

Figure 5: Overall usability evaluation

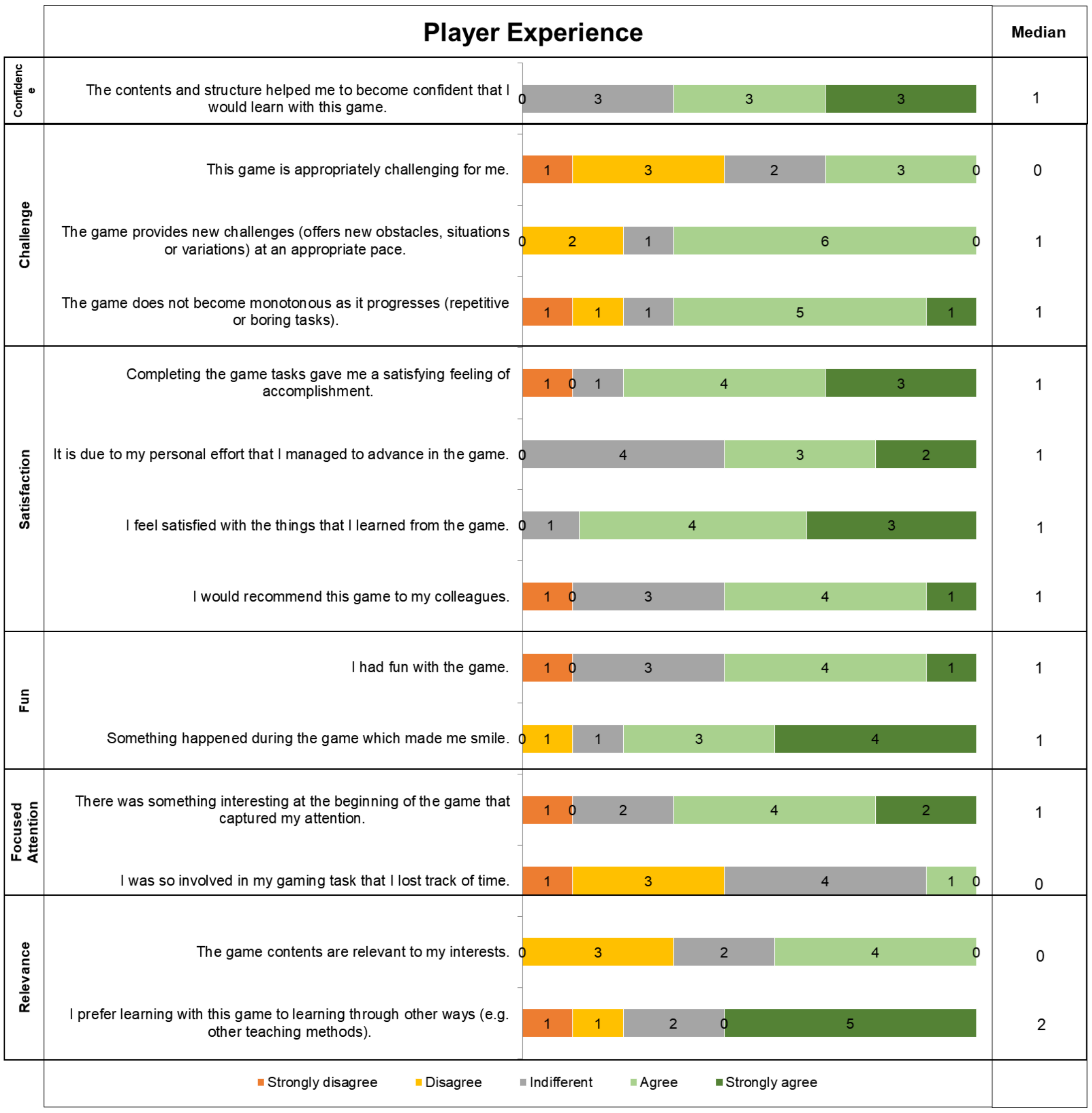

Figure 6: Overall player experience evaluation

the game, highlighting both positive aspects and challenges faced. The 7th graders noted that they learned how the tape works and understood the concept of states, finding the game easy to grasp. However, they struggled with symbol selection and the location of the rules, and they reported technical issues, such as bugs, that hindered their experience. The 9th graders appreciated how the game simplified the concepts of accepting and rejecting words. Still, they found the mechanics difficult to understand and encountered technical issues, such as difficulty clicking on icons. Despite these challenges, they valued the game's aesthetics and the simplified content presentation, recommending bug fixes and game mechanics improvements. Lastly, the 1st graders demonstrated a more detailed understanding of the technical concepts but struggled with interaction and clarity of the rules. They praised the design and the differentiated teaching approach but suggested improved rule clarity and bug fixes. These bugs have been reported to the GrameStation development team.

7. Conclusion

This paper presented the application and evaluation of a proposed activity that introduced Automata Theory in K-12 education, specifically using the formal language GG. The activity is based on the EF09CO03 skill from the BNCC and was developed in GrameStation, a game engine based on GG. In addition to imparting computational concepts, the proposed approaches also promote CT skills. Alongside the relations established by (SILVA JUNIOR, 2020) using GG, it is important to note that the manipulation and specification of automata further enhance these skills.

The use of automata to represent the robot's behavior during missions exercises *abstraction*, particularly through the states that represent the robot's current condition and the transitions that signify actions capable of modifying that condition. By specifying automata, students define algorithms using an event-based language to solve problems. When defining the tape for the robot to execute a specific mission, they outline the sequence of actions (*algorithm*) necessary to complete its task. Additionally, to identify the language recognized by an automaton, students must discern *patterns* in various input tapes and *generalize* the formation rules for all accepted words. *Debugging* skills are also developed as students are challenged to define an input tape that guides the robot in completing a mission, enabling them to simulate the tape processing and verify whether the mission was accomplished.

The feasibility of the activity was demonstrated through a pilot experiment conducted with a group of K-12 students from the city of Pelotas. In this experiment, an evaluation questionnaire was applied (VON WANGENHEIM; PETRI e BORGATTO, 2020). The results indicated that the students understood the concepts presented during the activity and could manipulate the grammars in the tool used. Thus, the activity exemplifies using FSL in K-12 and presents an alternative approach to developing CT. It is important to emphasize that the pilot experiment was not aimed at validating the activity but at identifying aspects that could be improved. Consequently, it involved a small participant sample and a diverse age range among the students. Some improvements could enhance the effectiveness of future applications, such as incorporating group work to promote peer learning and discussion, allowing students to share different problem-solving strategies. Additionally, refining the assessment questionnaire based on initial findings to generate more accurate perceptions of student understanding and engagement. The complexity of the grammars posed some challenges for the execution of the activity. In this case, an incremental approach to developing the activity could be a possible solution.

In future work, we intend to expand the activity to include additional phases, gradually increasing its complexity and exploring other types of automata, such as Non-Deterministic Automata and Pushdown Automata. At the end of the implementation, a new experiment will be conducted with the activity's target audience (9th graders).

References

BATTISTELLA, P. E. ENgAGED: Um Processo de Desenvolvimento de Jogos para Ensino em Computação. Florianópolis: UFSC, 2016. Tese de Doutorado.

BRAZIL. Normas sobre Computação na Educação Básica. 2022. Available at: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=182481-texto-referencia-normas-sobre-computacao-na-educacao-basica&category_slug=abril-2021-pdf&Itemid=30192>. Accessed on: December 10, 2024.

BROY, M. *et al.* Does every computer scientist need to know formal methods? Formal Aspects of Computing, Association for Computing Machinery, 2024. ISSN 0934-5043. Available at: <https://doi.org/10.1145/3670795>. Accessed on: December 10, 2024.

DEVELLIS, R. F. Scale development: Theory and applications. London: Sage, 2003. 176 p.

EHRIG, H. *et al.* Algebraic Approaches to Graph Transformation. Part II: Single Pushout Approach and Comparison with Double Pushout Approach. In: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations. Netherlands: World Scientific Publishing Co., Inc., 1997.

FRANÇA, R.; TEDESCO, P. Pensamento Computacional: Panorama dos Grupos de Pesquisa no Brasil. In: XXX BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION. Proceedings [...]. Brasília: SBC, 2019. v. 30, n. 1, p. 409-418.

HAHN, G.; TARDIF, C. Graph homomorphisms: structure and symmetry. In: Graph symmetry: algebraic methods and applications. Netherlands: Springer, 1997. p. 107–166.

KITE, V.; PARK, S.; WIEBE, E. The Code-centric Nature of Computational Thinking Education: A Review of Trends and Issues in Computational Thinking Education Research. Sage Open, v. 11, n. 2, p. 1–17, 2021.

KRATH, J.; SCHÜRMANN, L.; von KORFLESCH, H. F. Revealing the Theoretical Basis of Gamification: A Systematic Review and Analysis of Theory in Research on Gamification, Serious Games and Game-based Learning. Computers in Human Behavior, v. 125, p. 1–33, 2021.

MOGENSEN, T. Æ. Introduction to Compiler Design. Berlin: Springer Nature, 2024.

NOBLE, J.; STREADER, D.; GARIANO, I. O.; SAMARAKOON, M. More Programming than Programming: Teaching Formal Methods in a Software Engineering Programme. In: NASA FORMAL METHODS SYMPOSIUM. Proceedings[...]. Pasadena: Springer, 2022. p. 431–450.

RIBEIRO, L. Métodos Formais de Especificação: Gramática de Grafos. Escola de Informática da SBC-Sul, v. 8, p. 1–33, 2000.

SILVA, J.; CAVALHEIRO, S.; FOSS, L. Automata theory in computing education: A systematic review. In: XXXV SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. Proceedings [...]. Rio de Janeiro: SBC, 2024. p. 301–313.

SILVA, J.; SILVA JUNIOR, B.; CAVALHEIRO, S.; FOSS, L. Exploring automata theory with an educational activity using graph grammar for k-12 education. In: XXXV SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. Proceedings [...]. Rio de Janeiro: SBC, 2024. p. 329–342.

SILVA, J. V.; SILVA JUNIOR, B. A.; FOSS, L.; CAVALHEIRO, S. Adaptação do processo engaged para o desenvolvimento de conteúdos curriculares em uma plataforma de jogos baseada em gramática de grafos. In: XXXII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. Proceedings [...]. Belém: SBC, 2021. p. 316–327.

SILVA JUNIOR, B. A.; CAVALHEIRO, S. A. D. C.; FOSS, L. GrameStation: Specifying Games with Graphs. In: XXXII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO. Proceedings [...]. Belém (online): SBC, 2021. p. 499–511.

SILVA JUNIOR, B. A. D. GGasCT: bringing formal methods to the computational thinking. Pelotas: UFPEL, 2020. Dissertação de Mestrado.

VON WANGENHEIM, C. G.; PETRI, G.; BORGATTO, A. F. MEEGA+ KIDS: a model for the evaluation of games for computing education in secondary school. Revista Novas Tecnologias na Educação, v. 18, n. 1, 2020.

WING, J. M. Computational Thinking. Communications of the ACM, v. 49, n. 3, p. 33–35, 2006.

YANG, K.; LIU, X.; CHEN, G. The Influence of Robots on Students' Computational Thinking: A Literature Review. International Journal of Information and Education Technology, v. 10, n. 8, p. 627–631, 2020.