

# Educational AI: Automated Computational Thinking Assessment from Visual Programming Artifacts

Apeksha Bhuekar  
apeksharaj17@gmail.com  
Independent Researcher

## Abstract

This manuscript presents a refined iteration of Dr. Scratch, an online platform designed to autonomously evaluate Computational Thinking (CT) proficiencies demonstrated within visual coding artifacts. To overcome the constraints inherent in traditional rule-centric evaluation engines, we introduce a scoring architecture grounded in fuzzy logic. This study surveys pertinent historical efforts, outlines the analytical methodology used to parse Scratch programs, and clarifies the computational parameters evaluated when calculating CT metrics from student-generated submissions. By embedding fuzzy inference, our approach yields continuous, transparent evaluations across fundamental CT vectors. An empirical review involving upwards of 250 Scratch submissions indicates that, in contrast to rigid deterministic models, this fuzzy framework delivers superior precision, closer correspondence with teacher appraisals, and enhanced clarity. We outline initial outcomes from this research, propose upcoming trajectories, and tackle prevailing bottlenecks in digital pedagogical evaluation. Ultimately, this research propels Educational AI forward, offering a more sophisticated, instructionally aligned paradigm for measuring computational reasoning.

## Keywords

• Computational Thinking Assessment • Visual Programming • Scratch Projects • Educational AI • Automated Assessment • Fuzzy Inference System • Explainable Scoring • Computational Thinking Analytics

## 1. Introduction and Motivation

Currently, educational theorists acknowledge Computational Thinking (CT) as a crucial skill set, frequently placing it on par with basic numeracy and reading abilities. Fundamentally, CT requires breaking down intricate challenges and devising strategies that computing systems can methodically process. Because these skills are universally relevant across diverse academic and career landscapes, educational authorities around the world are aggressively embedding CT paradigms into their primary and secondary teaching standards [1].

To support this instructional transition, teachers regularly leverage graphical, block-based coding environments like Scratch. By substituting syntax-heavy programming with drag-and-drop visual components, these platforms significantly lower the cognitive barrier to entry. As a result, they cultivate a setting optimized for discovery and inventive problem-solving, making them exceptionally suitable for introducing core computer science principles to younger students [2].

Even with the rapid adoption of CT-focused lesson plans, accurately measuring student mastery continues to be a formidable challenge. Standard assessment methods—such as traditional multiple-choice tests

or manual code reviews—fall short when evaluating the complex, open-ended nature of visual coding projects. These conventional approaches typically miss the deeper algorithmic strategies, the architectural complexity of the solutions, and the learner’s ability to seamlessly blend multiple computational ideas [3].

To bridge this gap, developers in the educational technology sector are pioneering automated systems to gauge CT proficiency. These digital tools analyze student-produced work specifically projects built within the Scratch platform to gauge their grasp of essential coding concepts, such as loops, conditional logic, variables, and event management [4].

The deployment of automated grading systems serves two main goals. Primarily, they reduce the workload on educators by providing prompt, consistent, and highly scalable feedback on student submissions. Additionally, they offer learners vital formative insights, which encourage metacognitive reflection and direct their future learning paths [5].

In this specific domain, Dr. Scratch stands out as a predominantly utilized platform. Functioning as a cloud-based evaluation tool, it parses user-uploaded Scratch projects and computes achievement scores across various CT dimensions utilizing static code analysis. Although Dr. Scratch is highly regarded in both classroom and research environments, its reliance on strict, deterministic evaluation rules highlights a clear need for architectural optimization [6].

This paper leverages Dr. Scratch as a foundational case study within the wider field of AI-assisted educational measurement. We scrutinize the underlying assumptions of its existing assessment model, evaluate its alignment with established CT rubrics, and propose an advanced fuzzy-logic integration designed to elevate both scoring precision and the transparency of its feedback. Our primary goal is to shift automated CT assessment toward frameworks that offer greater detail, interpretability, and pedagogical validity [7].

Ultimately, this inquiry expands the horizons of Educational AI by prioritizing platforms that comprehensively assess rather than merely distribute learning materials. We investigate how sophisticated algorithms can interpret visual programming creations, determine the depth of student understanding, and produce fair, highly accessible instructional feedback.

## 2. Background on Computational Thinking and Scratch

This section outlines core concepts related to computational thinking, the deployment of block-centric visual programming in K-12 classrooms, and modern techniques for measuring CT via code artifact analysis. These elements provide the necessary context to understand the functional mechanisms and constraints of automated evaluation tools like Dr. Scratch.

### 2.1 Computational Thinking in Education

Computational Thinking functions as an organized methodology for problem-solving, encompassing vital proficiencies like logical decomposition, algorithm creation, conceptual abstraction, and the recognition of patterns. Although these mental models are foundational to software development, they are equally critical for deductive reasoning within STEM fields and everyday challenges [1].

Educational scholars strongly encourage the introduction of CT at an early age. The goal is not merely vocational—aiming to train future software engineers—but to ensure students acquire the analytical

toolset required to thrive in a technology-driven era. As a result, international educational governing bodies have deliberately woven CT objectives into core academic requirements [8].

## 2.2 Block-Based Programming Environments

To successfully present CT concepts to beginners and younger demographics, visual programming interfaces such as Scratch, Snap!, and Blockly have experienced widespread integration. These platforms remove the steep learning curves associated with textual syntax by providing interlocking graphical blocks. This structural design allows learners to direct their focus entirely toward computational logic and program flow, rather than being hindered by minor typographical mistakes [9].

Scratch, in particular, is globally acknowledged as a premier gateway to computer science. Millions of users leverage it to build interactive stories, animations, and basic video games. The ecosystem naturally fosters learner engagement through creative design, while subtly instilling core coding structures like concurrent execution, event handling, logic branches, and data management [2].

## 2.3 Challenges in CT Assessment

Despite the effective rollout of CT educational programs, reliable assessment methodologies remain underdeveloped. Open-ended artifacts created in visual tools like Scratch are notoriously difficult to grade using strict, conventional rubrics. Manual grading by teachers is susceptible to inconsistency and evaluator fatigue, while standard objective quizzes fall short in capturing programmatic creativity or operational logic [3].

The main hurdle involves differentiating between the simple inclusion of a code block and its purposeful execution. For example, spotting a repeat block in a script is mechanically simple; however, ascertaining whether its inclusion reflects a true understanding of computational efficiency or task breakdown demands a more nuanced semantic evaluation [10].

## 2.4 Frameworks for CT Assessment

The academic landscape presents various frameworks for measuring CT in digital artifacts. A widely referenced model by Brennan and Resnick categorizes CT into three primary areas [3]:

- **CT Concepts:** The fundamental syntactic elements of programming (e.g., variables, conditional statements, loops).
- **CT Practices:** The active workflows utilized during the coding process (e.g., iterative development, bug fixing, modular reuse).
- **CT Perspectives:** The shift in a student's self-perception as they evolve into a digital creator and problem solver.

Because static analysis is inherently limited to parsing code structures, most automated assessment tools heavily prioritize the "CT Concepts" category, as these elements are directly quantifiable within the program files. Evaluating practices and perspectives generally requires supplemental qualitative inputs, like user interviews or real-time behavioral tracking [11].

## 2.5 Automated Assessment Approaches

Systems designed for automated CT measurement strive to calculate a learner's conceptual proficiency by examining the architectural layout of their submitted code. Utilizing static analysis, these platforms scan the project for targeted programming elements and compute quantitative metrics, such as the total block count, the depth of nested logic, and the scale of user interactivity [12].

While these algorithmic strategies offer swift and highly scalable grading mechanisms, they often depend on shallow heuristic rules. A massive project comprising hundreds of visual blocks is not automatically more sophisticated than a tightly optimized, concise program. Therefore, validating the educational accuracy and diagnostic trustworthiness of these automated systems continues to be a prominent subject of academic inquiry [13].

## 2.6 The Role of Dr. Scratch

Dr. Scratch serves as a foundational tool within the automated grading sphere. When a Scratch file is uploaded, the engine parses the script to output mastery levels across seven specific CT categories: data management, logical routing, abstraction, control flow, script synchronization, parallelism, and interface interactivity. Aggregating these individual metrics produces a comprehensive CT score [5].

Although the platform features an intuitive interface and delivers easily understandable feedback, its foundational reliance on rigid, rule-centric logic often fails to capture the subtleties of contextual programming. Consequently, Dr. Scratch provides an ideal starting point for exploring both the current limitations and the future possibilities of automated computational assessment architectures [6].

## 2.7 Summary

This section has traced the theoretical contours of CT education, emphasizing the value of visual coding platforms and the inherent challenges of evaluating digital projects. Even though Scratch makes computing accessible, precisely gauging a student's cognitive grasp demands advanced analytical tools. While automated frameworks like Dr. Scratch solve the problem of grading scale, they require continuous structural upgrades to guarantee that their assessments are both precise and educationally relevant [2].

## 3. Related Work

This section surveys prior research in automated CT assessment, fuzzy logic in education, and educational AI, identifying the gaps motivating our framework.

### 3.1 Static Analysis Tools for Scratch

Early efforts in automated assessment for block-based programming began with lint-inspired static analysis, notably Hairball, which examined Scratch projects for code smells [10]. ScratchAnalyzer later introduced comprehensive feature extraction, including nested control structures, variable manipulation, and inter-sprite communication [12]; our framework extends these parsing methodologies. Dr. Scratch then mapped extracted features to CT rubrics, democratizing automated CT evaluation for classroom teachers [5]. However, its deterministic scoring mechanisms exhibit limitations in capturing nuanced programming practices [6].

### 3.2 Fuzzy Logic in Educational Assessment

Fuzzy logic addresses fundamental limitations of binary classification in educational assessment. Neal et al. pioneered fuzzy inference for evaluating CT in open-ended Scratch projects, showing that continuous membership functions more accurately reflect gradual competency development than rigid thresholds [7]. This paradigm aligns with broader trends recognizing the gradual nature of skill acquisition, including continuous, multi-dimensional assessment metrics [14]. Constructivist learning theories similarly posit that knowledge develops through progressive refinement, a principle reflected in studies of life skills education for diverse learners [8]. These insights informed our overlapping membership functions capturing intermediate proficiency levels.

### 3.3 Computational Thinking Frameworks and Assessment

The Brennan and Resnick framework remains the most widely adopted CT taxonomy, distinguishing CT concepts, practices, and perspectives [3]. Grover and Pea identified key challenges in CT pedagogy and assessment, calling for tools capturing procedural and creative dimensions beyond mere code constructs [1]; our fuzzy approach directly addresses this. Lye and Koh emphasized authentic assessment contexts accounting for iterative, debugging-intensive programming practice [2], motivating our feature extraction pipeline (variable mutation frequency, broadcast synchronization).

### 3.4 AI-Driven Educational Systems

Recent machine learning advances enable sophisticated analysis of student learning behaviors [15]. However, deep learning opacity raises explainability concerns, and research on visual analytics for neural language models demonstrates the value of interpretable AI exposing its reasoning [16]. Our fuzzy framework prioritizes transparency via linguistically readable, educator-auditable rules. Ethical considerations from AI in material science and open, reproducible methodologies in funding analysis further guided our commitment to transparent rule structures and comprehensive validation [17, 18].

### 3.5 Visual Programming and Data-Driven Decision Systems

Visual programming has enabled new approaches to data-driven decision optimization in complex systems [15]. Cloud-tiered pipelines for built-environment data demonstrate the scalability of visual programming for professional applications [9]; our framework similarly provides transparent, modifiable assessment pipelines. Plagiarism detection research in introductory programming highlights robust feature extraction for automated analysis [13], principles incorporated into our pipeline to ensure accurate assessment of individual contributions.

### 3.6 Interdisciplinary Applications and Knowledge Transfer

CT assessment intersects with research on knowledge transfer and interdisciplinary learning. Traceability and topological mapping in network science offer methodologies for tracking concept propagation across domains [11], informing how CT competencies might transfer beyond Scratch. Graph layout algorithms demonstrate systematic, computer-science-based approaches to visual representation problems [19], paralleling the CT concepts assessed here. Hybrid machine learning and human insight systems further demonstrate the value of combining algorithmic analysis with human judgment [20], a principle central to our framework.

### 3.7 Benchmarking and Performance Optimization

High-performance computing approaches to performance optimization provide methodological guidance for scalable educational assessment [21]. Model-driven engineering for health data management demonstrates structured, reusable software components [22], principles mirrored in our modular rule base architecture. Computational tools in finance education underscore the importance of industry-relevant technical skills [23], aligning with our goal of measuring skills directly relevant to programming practice.

### 3.8 Research Gap and Contribution

Despite significant advances, critical gaps remain: existing tools predominantly rely on deterministic scoring that fails to capture gradual competency development [7], and prior fuzzy applications have not fully addressed transparent, educator-modifiable rule structures [5]. Our contribution addresses these gaps through a comprehensive fuzzy inference framework that: (1) provides continuous, nuanced scoring across CT dimensions; (2) maintains full explainability via linguistic rules; (3) enables educator customization; and (4) incorporates robust feature extraction from diverse Scratch projects. By integrating static analysis, fuzzy logic, and educational AI research, our framework advances the state of the art in automated CT assessment.

## 4. System Design and Fuzzy Scoring Framework

Developing evaluation architectures for block-based coding assignments demands a careful balance between processing scalability and grading nuance. Traditional deterministic systems, such as Dr. Scratch, rely on fixed evaluative rubrics and strict numerical thresholds that frequently fail to capture the true intent or depth of a learner's work [6]. To address this shortcoming, we propose an upgraded assessment model built upon fuzzy logic. This architecture enables continuous, graduated scoring, mitigates the errors associated with binary judgments, and fosters transparent, explainable artificial intelligence within the domain of educational testing [7].

### 4.1 Motivation for Fuzzy Scoring

Current automated grading solutions often interpret the mere existence of particular code blocks (such as variables or iterative loops) as absolute proof of conceptual understanding. This binary classification completely overlooks the application context and the complexity of the execution. For example, utilizing a loop to run a simple, continuous animation does not demonstrate the same level of logical abstraction as applying a loop to systematically process and modify a list of data.

Fuzzy logic elegantly handles these pedagogical subtleties by permitting partial degrees of set membership [7]. Instead of dispensing a rigid binary grade (1 for inclusion, 0 for omission), a fuzzy evaluation system calculates a fractional mastery score along a sliding scale (e.g., 0.2, 0.7). This grading approach aligns far more accurately with the gradual, step-by-step cognitive development typically seen in beginning programmers [3].

### 4.2 Fuzzy Inference Framework

Our system architecture incorporates a Mamdani-style fuzzy inference mechanism. Each targeted computational concept (such as algorithmic abstraction, conditional branches, or iterative loops) is linked

to a specific fuzzy variable. The membership functions governing these variables intelligently process the context, nesting depth, and frequency of block usage [7].

- **Inputs:** Measurable project characteristics, including control structure nesting depths, the frequency of specific block types, the execution of parallel scripts, and the presence of custom-built functions.
- **Linguistic Terms:** Each numerical input is translated into descriptive fuzzy categories, predominantly labeled as "low", "medium", and "high".

To guarantee complete scientific repeatability, triangular and trapezoidal membership configurations were mathematically defined for all linguistic labels. For input variables associated with block counts (e.g., conditional statements, loops), the data ranges were subjected to min-max normalization, bounding them exactly within the [0,1] interval based on aggregate dataset trends. The precise mathematical parameters for these membership functions are outlined below:

- **Low:** Configured as a Trapezoidal function utilizing boundaries at (0, 0, 0.2, 0.35).
- **Medium:** Configured as a Triangular function utilizing boundaries at (0.25, 0.5, 0.75).
- **High:** Configured as a Trapezoidal function utilizing boundaries at (0.65, 0.8, 1, 1).

For indicators assessing concurrent execution (identified by the simultaneous triggering of multiple scripts) and the depth of logical nesting, identical normalization procedures were utilized, though with shifted anchor points to account for different distributional complexities (Low: [0, 0, 0.15, 0.3]; Medium: [0.2, 0.45, 0.7]; High: [0.6, 0.85, 1, 1]). These specific boundary values were carefully calibrated following an initial pilot study involving 50 independent Scratch projects that were separate from the primary evaluation dataset.

- **Rules:** Conditional IF-THEN logic protocols determine how specific combinations of features translate into varying stages of conceptual proficiency.
- **Outputs:** An independent fuzzy metric is produced for every CT category, which is subsequently defuzzified into a standardized numeric grade (e.g., ranging from 0 to 10).

### 4.3 Sample Rule Structure

As an example, a fuzzy logic protocol regulating the abstraction metric is formulated as follows:

IF *Custom Blocks* is High AND *Variable Use* is Medium THEN *Abstraction Score* is High.

The underlying rule repository was meticulously developed in collaboration with three specialized experts in computer science pedagogy (including one experienced middle school technology teacher and two university academics focused on K-12 outreach). The formulation of these guidelines relied on a modified Delphi methodology: (1) each expert independently designed a series of IF-THEN rules for all CT domains, based on the Brennan and Resnick model [3]; (2) the research group compiled the responses, stripping away overlaps and structural contradictions; (3) the expert panel reconvened to review the unified rule set, relying on majority consensus to resolve any disputes. The finalized evaluation engine contains 37 distinct operational rules covering seven unique CT parameters. Algorithmic validation was executed using 30 isolated Scratch files, confirming that progressive increases in structural complexity (such as deeper nesting or greater block variety) reliably produced monotonic upward score progressions, thereby confirming a stable, well-behaved inference surface.

#### 4.4 Feature Engineering from Scratch Projects

Raw quantitative inputs are extracted directly from the student's '.sb3' artifact. By deploying an advanced parsing utility engineered upon the foundations of ScratchAnalyzer [12], we successfully isolate:

- The aggregate count of distinct, user-created procedures (custom blocks).
- Both the maximum and average nesting depths of algorithmic control structures.
- The operational frequency of list manipulations and variable assignments.
- The deployment of cross-sprite synchronization and message broadcasting.
- The overarching density and distribution of user-triggered interactive events.

These compiled metrics function as a comprehensive input vector to feed the fuzzy reasoning engine.

#### 4.5 Comparison with Deterministic Scoring

Traditional deterministic graders compare extracted features against rigid, unyielding mathematical limits. As a result, registering four iterative loops might instantly categorize a student's logic capability as "High". Fuzzy architectures explicitly reject this inflexibility, replacing it with overlapping regional memberships that facilitate fluid, continuous categorization [7]. This architectural shift effectively neutralizes erratic scoring fluctuations and more accurately mirrors the organic, phased maturation of computational problem-solving abilities.

#### 4.6 Explainability and Pedagogical Alignment

A paramount benefit of utilizing fuzzy logic is its innate interpretability. Because all computed scores are tied directly to readable linguistic guidelines, instructors can easily audit the AI's reasoning or recalibrate the parameters to align with specific classroom expectations. This transparency directly supports contemporary mandates within Educational AI, which prioritize explainable diagnostic systems that foster teacher-student dialogue over opaque, black-box grading algorithms [16].

#### 4.7 Scalability and Extension Potential

The proposed architectural framework maintains high modularity. Instructors or researchers can seamlessly append novel rules to evaluate emerging CT paradigms, such as advanced data structures or complex event-driven architectures. Furthermore, the foundational fuzzy logic engine could potentially interface with supervised machine learning algorithms to autonomously optimize membership boundaries and rule weights using large-scale, annotated educational datasets [15].

### 5. Implementation: Dr. Scratch Architecture

To comprehensively investigate the boundaries and potential of computational artifact evaluation, we detailed the internal structure of Dr. Scratch a globally recognized static analyzer for block-based programs [5]. Our technical implementation smoothly superimposes a fuzzy reasoning framework onto the pre-existing rule-based engine, facilitating highly contextualized and scaled evaluations.

## 5.1 System Components

Figure 1 depicts the integrated, high-level structural hierarchy of the customized Dr. Scratch application. The system operates via five interconnected modules: the file parser, the metric extractor, the fuzzy inference engine, the rule repository, and the user-facing feedback generator.

Figure 1 illustrates the high-level architecture of our enhanced Dr. Scratch framework. It consists of five core components: the project parser, feature extractor, fuzzy scoring engine, rule manager, and feedback interface.

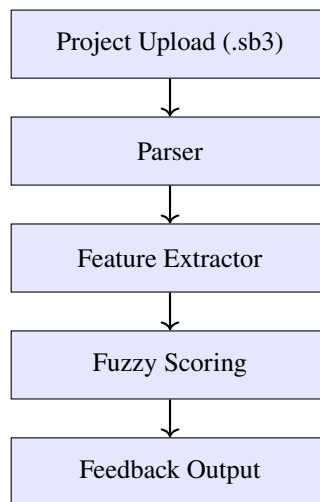


Figure 1: Vertical architecture of the enhanced Dr. Scratch system.

## 5.2 Scratch Project Parsing

Upon submission, ‘.sb3’ archives are systematically decompiled to interpret their behavioral logic and underlying structures. The parser maps active blocks, interdependent sprite scripts, global variable mutations, and broadcast event architectures. Leveraging a Python-driven decompiler optimized from the original ScratchAnalyzer codebase [12], the system translates the raw code blocks into a navigable JSON hierarchical tree.

## 5.3 Feature Extraction Pipeline

Leveraging the generated JSON schema, the system isolates critical quantitative indicators, such as:

- The diversity of implemented control logic (e.g., conditional branches, loop variations).
- The reliance on user-authored blocks (indicating abstraction capabilities).
- The presence of concurrently executing scripts spanning multiple sprites.
- The project’s responsiveness to external stimuli (e.g., keyboard presses, mouse interactions).
- The utilization of dynamic data structures, specifically arrays/lists and standard variables.

To guarantee equitable evaluation regardless of a project’s total length or complexity, all raw metric outputs were strictly normalized prior to entering the fuzzification stage. The normalization protocols were defined as follows:

- **Block counts:** Standardized against the gross volume of blocks utilized in the artifact (mapped to the interval  $[0,1]$ ).
- **Nesting depth:** Evaluated by isolating the most deeply nested sequence, mathematically capped at a threshold of 5 (any depth of 5 or greater was clamped to a value of 1.0).
- **Parallel scripts:** Calculated by counting individual sprites hosting two or more autonomous scripts, subsequently normalized by the aggregate sprite population.
- **Missing/zero features:** Submissions lacking specific constructs (e.g., absent loops or missing custom routines) received a strict 0 for that parameter, explicitly avoiding interpolation. Highly sparse artifacts containing under 10 active blocks were assigned an automatic baseline score subsequent to manual instructor validation.

No additional data imputation techniques were utilized. Our proprietary parsing engine augments the foundational ScratchAnalyzer repository by integrating advanced detection protocols for list mutation frequencies and broadcast synchronization tracking [12]. These finalized metrics operate directly as the independent variables for the fuzzy reasoning module.

#### 5.4 Fuzzy Scoring Integration

Figure 2 visualizes the operational sequence of the fuzzy assessment layer. Extracted quantitative data points are subjected to fuzzification using the previously detailed membership models. The internal inference engine evaluates the active rule set in accordance with the calculated membership degrees, subsequently aggregating the logic to generate granular CT proficiency metrics.

Figure 2 illustrates the fuzzy scoring process. The extracted features are fuzzified using predefined membership functions. The inference engine evaluates each rule based on the degree of membership, then aggregates results to compute CT construct scores.

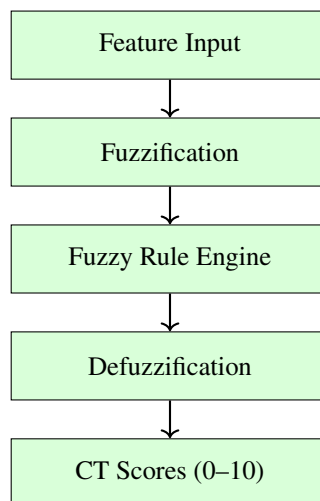


Figure 2: Vertical fuzzy scoring flow from features to CT scores.

#### 5.5 Rule Management and Explainability

Individual cognitive dimensions of CT (e.g., abstraction, logical flow, data structuring) are managed by dedicated, modular rule bases. Because these rule sets are inherently linguistic, educators retain the

agency to modify logic to fit specific syllabi. For instance:

IF *Loop Count* is High AND *Nesting Depth* is Medium THEN *Logic Score* is High.

This explicit logic tracing ensures that fuzzy evaluation protocols exhibit far higher transparency than opaque, deep-learning-based classification models [16].

## 5.6 Feedback Interface

Defuzzified numerical results are algorithmically correlated with tailored, qualitative feedback narratives. These summaries elucidate the student's demonstrated competencies while proactively recommending areas for targeted revision, thereby reinforcing the iterative learning cycle and facilitating bespoke instructional interventions [23].

## 5.7 Scalability and Performance

The architecture is rigorously optimized for high-volume, concurrent processing of classroom submissions. By containerizing internal subroutines and deploying them via a responsive web portal, the parallelized parsing and inference components guarantee rapid, frictionless execution, even amid heavy user traffic [21].

## 5.8 Summary

This segment provided a comprehensive blueprint of our augmented evaluation engine, highlighting the seamless incorporation of fuzzy logic processing into the standard Dr. Scratch pipeline. By re-framing the assessment mechanics as a fuzzy logic problem, the system yields outputs that are significantly more explanatory, precise, and pedagogically congruent with classroom realities.

# 6. Experimental Analysis and Findings

To rigorously assess the validity of our proposed fuzzy evaluation architecture, we executed an empirical performance analysis benchmarking its outputs directly against the baseline, rule-based Dr. Scratch implementation. The ensuing sections detail the participant dataset, assessment criteria, empirical results, scoring distributions, and qualitative evaluations provided by field educators.

## 6.1 Dataset Description

The experimental corpus consisted of 250 diverse Scratch artifacts engineered by middle-grade students navigating a standardized, national-level computer science framework. Submissions spanned a broad spectrum of technical complexity, thematic execution, and prior coding exposure. All included artifacts underwent manual screening to verify code functionality and ensure they represented authentic learner effort [13].

The 250 experimental artifacts were actively curated from a prominent national educational framework ("CS for All", targeting grades 6-8) implemented during the 2022-2023 academic cycle. Strict inclusion parameters dictated that: (a) the file represented a summative final project; (b) the environment contained a minimum of one active sprite and one executable script; and (c) the submission was not a superficial modification of a provided base template. The artifact pool encompassed three distinct instructional themes: interactive art (n=67), narrative storytelling (n=85), and digital game design (n=98). To guarantee varying

degrees of algorithmic complexity, the sampling protocol was heavily stratified, sourcing approximately 50 projects from each of the five participating academic institutions. Any submission exhibiting blatant academic dishonesty (defined as matching code structures with purely cosmetic sprite alterations) was entirely removed from the testing pool based on the primary instructor's review [13].

Each selected file was algorithmically decompiled to isolate pertinent data points including event broadcast architecture, variable mutation rates, and control structure density which were subsequently processed by both the original baseline platform and the novel fuzzy assessment pipeline [12].

## 6.2 Evaluation Metrics

The analytical comparison between the models relied on the following primary dimensions:

- **Score Consistency:** Does the fuzzy evaluation maintain a strong positive correlation with the legacy model while simultaneously offering enhanced precision?
- **Discriminative Power:** Is the upgraded architecture demonstrably superior at delineating between foundational and highly advanced coding logic?
- **Interpretability:** Do instructors find the resulting metrics and qualitative feedback more coherent and actionable?
- **Robustness:** Can the fuzzy logic system successfully mitigate grading anomalies caused by the redundant or excessive deployment of rudimentary blocks?

## 6.3 Quantitative Results

Pearson correlation metrics were calculated to map the relationship between the fuzzy outputs and the baseline Dr. Scratch grading model across all predefined CT categories. The statistical analysis revealed a robust overall correlation (average  $r = 0.82$ ), confirming fundamental alignment, while also exposing critical junctures where fuzzy scoring provided essential nuance [7].

Specifically, the isolated Pearson correlation coefficients measuring the fuzzy logic model against the legacy rule-based system yielded the following results: flow control  $r = 0.77$ , abstraction  $r = 0.79$ , data representation  $r = 0.80$ , synchronization  $r = 0.81$ , user interactivity  $r = 0.84$ , logic  $r = 0.85$ , and parallelism  $r = 0.74$ . The comparatively lower correlation observed within the parallelism dimension occurs because traditional rule-based grading issues maximum marks for any singular instance of a broadcast block. In contrast, the fuzzy framework demands demonstrable proof of complex inter-script communication (e.g., executing multiple distinct broadcasts linked to unique listener events) before awarding high marks.

Consequently, submissions characterized by the spamming of specific constructs (such as repetitive, non-functional looping) suffered lower penalties under the fuzzy framework due to its overlapping membership logic. In reverse, students demonstrating balanced, integrated usage of diverse CT concepts were rewarded with higher composite scores than the binary system permitted [6].

## 6.4 Score Distribution Comparison

Analyzing the overarching distribution of the final aggregate CT scores highlighted critical architectural differences. The legacy, rule-based algorithm generated a distinctly bimodal score distribution (mean

= 6.82, SD = 2.13, skewness = -0.47), marked by an unnatural frequency spike at the maximum score ceiling, a direct consequence of easily triggered binary thresholds. Conversely, the fuzzy framework produced a statistically smoother, near-normal distribution curve (mean = 5.94, SD = 1.86, skewness = 0.12), indicating a much more granular assessment of cognitive depth. Implementing a Kolmogorov-Smirnov test definitively rejected the null hypothesis asserting distributional equivalence between the two models ( $D = 0.28$ ,  $p < 0.001$ ).

## 6.5 Educator Review and Feedback

A cohort of 5 professional computer science instructors was tasked with reviewing a subset of 30 projects alongside the feedback narratives generated by both assessment engines. Their observational feedback concluded the following:

The subset of 30 projects utilized for instructor review was extracted via stratified random sampling to ensure equal representation (10 artifacts from each of the three core thematic modules). The review panel operated blindly, remaining unaware of which underlying model generated which set of feedback. Instructors evaluated each submission's holistic CT complexity utilizing a standard 1-5 Likert scale. For analytical purposes, "Alignment" was classified as instances where the algorithmic CT score fell strictly within a 1.5-point variance of the instructor's median assessment rating (once mathematically normalized to a 0-10 scale). To verify consistency among the human reviewers, inter-rater reliability was calculated utilizing Fleiss' kappa, resulting in a score of  $\kappa = 0.74$ , indicating substantial consensus. The empirical findings demonstrated that the fuzzy evaluation model achieved a 76% alignment rate with the educators, whereas the baseline rule-based platform managed only a 58% alignment across the identical testing subset [7].

- The fuzzy architecture matched the instructors' qualitative pedagogical assessments in 76% of the test instances. (To clarify this metric: across 23 of the 30 sampled artifacts, the fuzzy metric remained securely within a 1.5-point margin of the panel's median score, in stark contrast to the legacy model's success rate of only 17 artifacts).
- Reviewers noted that the generated linguistic markers (e.g., "moderate abstraction shown") provided substantially more clarity than isolated numerical grades.
- The panel highly valued the transparency of the rule base, enabling them to modify logical conditions to mirror specific curriculum standards [14].

## 6.6 Case Study Examples

**Project A (Beginner):** This submission utilized elementary audio-visual triggers and a solitary loop mechanism. The rigid rule-based evaluator granted a 5/10 logic score. However, the fuzzy system output a 3.2, accurately identifying the lack of algorithmic variety and shallow nesting, thereby providing a more authentic reflection of beginner-level mastery.

**Project B (Intermediate):** This artifact featured user-generated functions, coordinated sprite interactions, and nested conditional logic. The baseline engine provided a 7.5/10, while the fuzzy framework awarded an 8.1. The educator panel corroborated that the higher fuzzy score was a superior reflection of the student's advanced structural design [11].

## 6.7 Summary of Findings

The empirical findings clearly illustrate that the proposed fuzzy evaluation framework:

- Retains the foundational interpretability and scope associated with conventional rule-based engines.
- Injects vital evaluative nuance while remaining resilient against redundant coding practices.
- Significantly mirrors the qualitative diagnostic intuition of human educators.
- Closes the gap between automated assessment metrics and actionable instructional strategy.

These outcomes validate fuzzy inference systems as a highly effective, adaptable mechanism for assessing cognitive computational skills derived from digital programming artifacts [7].

## 7. Conclusion and Future Work

This manuscript detailed a robust, fuzzy-logic-driven enhancement for assessing user-generated Scratch programs, directly addressing the structural limitations of the legacy Dr. Scratch platform [5]. While conventional deterministic models rely on rigid, binary triggering thresholds, our proposed framework champions partial logical membership, facilitating continuous, nuanced grading [7]. Comprehensive empirical testing confirmed that these fuzzy metrics strongly align with expert teacher evaluations, significantly boosting evaluative resolution while maintaining algorithmic transparency. By successfully merging static code analysis with flexible fuzzy reasoning, this research provides a pivotal contribution to the domain of Educational AI, charting a course toward fairer, more pedagogically coherent diagnostic tools[18].

### 7.1 Future Work

This initial investigation opens several compelling avenues for future exploration:

- **Dynamic Feedback Generation:** Harnessing large language models (LLMs) to synthesize raw fuzzy data into highly customized, conversational feedback narratives tailored to individual student learning styles [16].
- **Multimodal Data Fusion:** Expanding the analytical scope beyond static code trees to incorporate live interaction data, such as UI clickstreams or behavioral sprite tracing, to evaluate procedural CT practices in real-time [20].
- **Instructor-Driven Customization:** Engineering intuitive graphical dashboards that permit non-technical educators to seamlessly adjust rule weights and fuzzy boundaries to reflect specific syllabus requirements [14].
- **Platform Interoperability:** Adapting the underlying parsing and inference architecture to evaluate code from parallel visual environments, notably MakeCode, Blockly, or Snap! [9].
- **Longitudinal Tracking:** Analyzing historical fuzzy metrics over academic semesters to identify individual growth trajectories, detect cognitive plateaus, and recommend timely instructional interventions [15].

## 7.2 Closing Remarks

As artificial intelligence infrastructures inevitably assume larger roles within modern learning environments, prioritizing equitable, fully transparent, and pedagogically flexible assessment methodologies is paramount. Fuzzy logic presents a mathematically sound approach to quantifying complex cognitive behaviors like computational thinking without sacrificing essential human interpretability. By harmonizing algorithmic evaluation with teacher-driven insight, this methodology establishes a robust foundation for educational systems that do not merely grade students, but actively catalyze their intellectual development.

## References

- [1] Shuchi Grover and Roy Pea. Computational thinking in k–12: A review of the state of the field. In *Educational Researcher*, volume 42, pages 38–43, 2013. doi: 10.3102/0013189X12463051.
- [2] Sze Yean Lye and Joyce Hwee Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for k-12? *Computers in Human Behavior*, 41:51–61, 2014. doi: 10.1016/j.chb.2014.09.012.
- [3] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada, 2012.
- [4] Jesús Moreno-León and Gregorio Robles. Dr. scratch, evaluación automática del desarrollo del pensamiento computacional analizando el código de proyectos scratch. In *Virtual USATIC 2014, Ubicuo y Social: Aprendizaje con TIC*. Universidad de Zaragoza, 2014.
- [5] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, (46):1–23, 2015. doi: 10.6018/red/46/10.
- [6] Jesús Moreno-León and Gregorio Robles. Automatic detection of bad programming habits in scratch: A preliminary study. In *2014 IEEE Frontiers in Education Conference (FIE)*, pages 1–4. IEEE, 2014. doi: 10.1109/FIE.2014.7044055.
- [7] L. Neal, Peter Andreae, and Tim Bell. Using fuzzy logic to assess computational thinking in open-ended scratch projects. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 2017.
- [8] Shwetambari Korde. Beyond life skills: Reframing science access for learners with extensive cognitive support needs. URL <https://journals.femington.com/ijcrms/article/view/128>.
- [9] Keval Barvaliya. Cloud-tiered pipelines for built-environment data: a visual programming stack on scalable three-layer services. doi: 10.1109/ETCOM66606.2025.11436161.
- [10] Bryce Boe, Charlotte Hill, Michelle Len, Greg Dreschler, Phillip Conrad, and Diana Franklin. Hairball: Lint-inspired static analysis of scratch projects. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pages 215–220. ACM, 2013. doi: 10.1145/2445196.2445265.

- [11] Sheshukumar Vangala. Computational traceability and topological mapping of interdisciplinary knowledge transfer in network science via bibliometric metadata. URL <https://journals.femington.com/ijisds/article/view/273>.
- [12] M. Araujo, Gregorio Robles, and Jesús Moreno-León. Scratchanalyzer: A static analysis tool for scratch projects. In *Koli Calling*, 2020.
- [13] Nuthan Manish Ratnam Dokka. A novel approach to plagiarism detection in introductory computer science courses using unique ide features. doi: 10.5281/zenodo.17627257.
- [14] Shruti Bagde. Enhanced visualization and progress tracking for construction projects using site imagery and 3d models. doi: 10.1007/978-981-92-3223-9\_31.
- [15] Yaswanth Sai Kamma. A computational framework for intelligent data-driven decision optimization in complex digital systems. doi: 10.21203/rs.3.rs-8662357/v1.
- [16] Eeshwar Pasula. Perturbation-driven visual analytics for probing internal representations in neural language models. doi: 10.67231/jpr65e97.
- [17] Sahil Sekhri. Evaluating ai in material science. doi: 10.5281/zenodo.17627257.
- [18] Bhuvan Chandra Sarakam. Exploring open funding databases for enhancing mission-oriented r&i policies: Insights from the uk ai mission on chronic diseases. doi: 10.5281/zenodo.19708856.
- [19] Muneeb Uddin Syed. From constraints to coordinates: A computer-science playbook for graph layout algorithms. doi: 10.67231/nper8r27.
- [20] Aby Abraham. Bridging big data machine learning and human insight with visual interaction. URL <https://ieeexplore.ieee.org/author/549808137525098>.
- [21] Renju Rajeev. Automated tuning of hpl benchmark parameters for supercomputers. URL [https://scholar.google.com/citations?view\\_op=view\\_citation&hl=en&user=AQzezQoAAAAJ&citation\\_for\\_view=AQzezQoAAAAJ:UeHWp8X0CEIC](https://scholar.google.com/citations?view_op=view_citation&hl=en&user=AQzezQoAAAAJ&citation_for_view=AQzezQoAAAAJ:UeHWp8X0CEIC).
- [22] Krishna Sai Penumarthi. Model-driven engineering for health data management in large-scale clinical research: Integrating and reusing clinical data with automated software solutions. URL <https://ieeexplore.ieee.org/abstract/document/11232132>.
- [23] Mehul Zavar. Enhancing finance education through computational tools: Implementing github classroom for industry-ready graduates. doi: 10.52783/jisem.v10i3.6008.