<Paragraph 1>
The paper presents an automated grading system (autograder) for evaluating computational thinking skills demonstrated in student code written in Snap!, a block-based visual programming language. Unlike traditional autograders that focus on output correctness, this system aims to assess coding practices like abstraction, decomposition, and algorithm design by analyzing the structure of the student's code represented as an XML tree. This novel approach works directly on the non-textual block-based code, which differs from techniques used for text-based programming languages.
</Paragraph 2>
The core of the autograder involves extracting structural features from the XML tree representation that capture principles of computational thinking like encapsulation of functionality into subblocks and repeated use of code. Various tree-based numerical features were explored, including a custom 1D convolution to measure repetition of code segments. These features were then used to cluster submissions with similar structural properties using algorithms like k-means, Gaussian mixtures, and hierarchical clustering. The quality of the clustering was evaluated by comparing to manual scores provided by instructors for skills like abstraction and algorithm design.
</Paragraph 3>
While the relative inertia analysis showed potential for some clustering approaches to group submissions aligning with the manual rubric scores, the authors acknowledge improvement is needed in the feature engineering process to better capture specifics of computational thinking concepts. The paper outlines directions for continued research including tuning features based on common student pitfalls, investigating generative models to provide customized text feedback, and exploring broader applications of extracting patterns from tree-structured data across domains.