Here is a concise 3-paragraph summary of the paper:

<Paragraph 1> The paper proposes an automated grader (autograder) for the Snap! block-based programming language that evaluates students' coding assignments not just on output accuracy, but on their demonstrated computational thinking skills as reflected in the structure of their code submissions. Instead of simply checking if the code produces the correct output, the autograder analyzes patterns of abstraction, repetition, and encapsulation in the XML tree representation of the student's Snap! program. </Paragraph 2>
The key features extracted from the code include repetition (measured by sliding a filter over the code to detect repeated sections), average number of unique child nodes per parent node, and other tree structure metrics. The authors explore various clustering algorithms like k-means and Gaussian mixtures to group submissions with similar computational thinking traits. They validate the clustering against manual scoring of abstraction, algorithms, data representation, and documentation provided by human instructors. </Paragraph 3> While promising, the current approach has some limitations such as inability to provide specific feedback beyond scores, lack of personalized alternative solutions, and failure to appreciate the artistic elements of multimedia programming assignments. The authors discuss potential avenues like using the autograder for data-driven refinement of features, developing generative models to suggest code improvements, and gathering more training data from Snap! creators themselves.