This paper presents an automated grader (autograder) for evaluating students' programming assignments in an introductory computer science course that uses Snap!, a block-based visual programming language. The autograder assesses computational thinking concepts reflected in the structure of a student's code submission, rather than just the accuracy or runtime of the output.

The key novelty is extracting features from the XML representation of a student's Snap! program to capture patterns of abstraction, encapsulation, and iterative thinking. These features include measuring code repetition using a sliding 1D filter, counting nested blocks, and analyzing the tree structure. Various clustering algorithms were explored to group submissions with similar computational thinking patterns.

The authors conducted experiments on actual student assignments and instructor feedback to validate their approach. They found that certain clustering models like Gaussian Mixture Models and Hierarchical Clustering could effectively group submissions based on the engineered features, aligning well with the manual scoring rubric for assessing computational thinking skills. The paper discusses promising results as a proof-of-concept, while acknowledging the need for further tuning of feature extraction and exploring other clustering techniques to scale the autograder to larger and more diverse classrooms.