<Paragraph 1>
This paper proposes an automated grading system (autograder) for Snap!, a block-based visual programming language aimed at novice programmers. The autograder evaluates student submissions not just on output correctness, but on the computational thinking (CT) principles reflected in the code structure, such as abstraction and use of iteration. This involves extracting features from the XML representation of a student's Snap! program that capture patterns like repeated code blocks and nested functional blocks.
</Paragraph 1>

<Paragraph 2>
The authors explored various tree-based features like number of child nodes, depth, and a novel 1D convolution to detect redundant code. They evaluated clustering algorithms like K-means, DBSCAN, Gaussian Mixture Models, and hierarchical clustering to group similar program structures together. Model performance was measured by the relative inertia of clustering compared to manual scoring of CT criteria by instructors. K-means and Gaussian models showed some promise in separating programs by repetition and abstraction scores.
</Paragraph 2>

<Paragraph 3>
While no single model achieved very low relative inertia compared to the manual rubric, the results suggest potential for an ensemble approach with separate models targeting individual CT criteria. The authors identify better capturing patterns representative of common student mistakes as a key area for improvement in feature engineering. They plan to further tune the model using instructor feedback before deploying it as an autograding tool to provide scalable personalized feedback.
</Paragraph 3>