Abstract
Student experience in introductory computer science classes can be enhanced by applying static analysis techniques to automatically grade assignments. Since resources for teaching large introductory programming courses are limited, it is infeasible to have teaching staff individually examine each student’s answer for small in-lecture exercises. However, qualitative data regarding student code independent from execution is still valuable (and in some cases required) to assess progress. When static analysis utilities are made available to instructors and integrated with an automatic assignment testing platform, instructors are able to judge student performance and provide feedback at a scale that would otherwise be infeasible. There are clear advantages to applying static analysis techniques in comparison to less sophisticated methods (e.g., regular expression-based tools). For one, students are unable to subvert grading by placing certain keywords within comments or string literals. Since analysis can also be applied to easily grade students on patterns that would be nontrivial to detect using a more naive method, for example in enforcing that all member variables of a C++ class must be private, or verifying that a function takes the appropriate number and type of arguments.

A Motivating Use-Case
- Computer Science I at RPI is very large. Fall 2016 numbers: 650 students, ∼50 undergraduate TAs.
- There are clear advantages to applying static analysis techniques in comparison to less sophisticated methods (e.g., regular expression-based tools).
- Student code using goto
- Student rewrite using while

Design Requirements
- Grading must be completely automatic.
- Instructor setup must be quick and simple.
- Tests should be robust to common student errors.
- Students should be unable to subvert grading requirements.

Examples of Instructor Usage
- These methods were introduced in the Fall 2016 introductory Computer Science course at RPI.
- Students received immediate feedback about specific assignments.
- Students could correct any issues and make additional submissions.
- Instructors were able to provide feedback at a scale that would otherwise be infeasible.

Results from Fall 2016
- Custom grader to determine loop depth.
- This simple code re-writing exercise is often used early in the semester to help students learn the somewhat less intuitive while loop in Python.

Details of Implementation
- Lexical analysis tool and parser available for C, C++, Python, and Java.
- Instructors use these tools through a language-agnostic Python interface.
- Scripts using this interface are provided for instructor convenience when dealing with particularly common use cases.
- Different tools on the backend are used for each language, but the instructor-visible interface is consistent, allowing course-wide sharing of problems.

Performance
- At peak load (Thursday at midnight, the typical homework deadline for multiple courses), wait times for students averaged 15 seconds, despite thousands of submissions.

Instructor Feedback
- Overall, feedback from instructors teaching the introductory course has been very positive.
- Teaching assistant workload for the mundane portions of homework grading (verifying student-level assignment) has been significantly reduced.
- Teaching assistants are free to give more detailed and personalized feedback to each student regarding overall technique and style rather than being forced to focus on programmatic minutia.

Ongoing Work
- Maintenance and bug fixing, security analysis.
- Increased language support.
- Expanded library of grading techniques for ease of use.
- Eleven more instructors are aware of the tool.

Design Requirements
- Grading must be completely automatic.
- Instructor setup must be quick and simple.
- Tests should be robust to common student errors.
- Students should be unable to subvert grading requirements.

Examples of Instructor Usage
- These methods were introduced in the Fall 2016 introductory Computer Science course at RPI.
- Students received immediate feedback about specific assignments.
- Students could correct any issues and make additional submissions.
- Instructors were able to provide feedback at a scale that would otherwise be infeasible.

Results from Fall 2016
- Custom grader to determine loop depth.
- This simple code re-writing exercise is often used early in the semester to help students learn the somewhat less intuitive while loop in Python.

Details of Implementation
- Lexical analysis tool and parser available for C, C++, Python, and Java.
- Instructors use these tools through a language-agnostic Python interface.
- Scripts using this interface are provided for instructor convenience when dealing with particularly common use cases.
- Different tools on the backend are used for each language, but the instructor-visible interface is consistent, allowing course-wide sharing of problems.

Performance
- At peak load (Thursday at midnight, the typical homework deadline for multiple courses), wait times for students averaged 15 seconds, despite thousands of submissions.

Instructor Feedback
- Overall, feedback from instructors teaching the introductory course has been very positive.
- Teaching assistant workload for the mundane portions of homework grading (verifying student-level assignment) has been significantly reduced.
- Teaching assistants are free to give more detailed and personalized feedback to each student regarding overall technique and style rather than being forced to focus on programmatic minutia.

Ongoing Work
- Maintenance and bug fixing, security analysis.
- Increased language support.
- Expanded library of grading techniques for ease of use.
- Eleven more instructors are aware of the tool.

Design Requirements
- Grading must be completely automatic.
- Instructor setup must be quick and simple.
- Tests should be robust to common student errors.
- Students should be unable to subvert grading requirements.

Examples of Instructor Usage
- These methods were introduced in the Fall 2016 introductory Computer Science course at RPI.
- Students received immediate feedback about specific assignments.
- Students could correct any issues and make additional submissions.
- Instructors were able to provide feedback at a scale that would otherwise be infeasible.

Results from Fall 2016
- Custom grader to determine loop depth.
- This simple code re-writing exercise is often used early in the semester to help students learn the somewhat less intuitive while loop in Python.

Details of Implementation
- Lexical analysis tool and parser available for C, C++, Python, and Java.
- Instructors use these tools through a language-agnostic Python interface.
- Scripts using this interface are provided for instructor convenience when dealing with particularly common use cases.
- Different tools on the backend are used for each language, but the instructor-visible interface is consistent, allowing course-wide sharing of problems.

Performance
- At peak load (Thursday at midnight, the typical homework deadline for multiple courses), wait times for students averaged 15 seconds, despite thousands of submissions.

Instructor Feedback
- Overall, feedback from instructors teaching the introductory course has been very positive.
- Teaching assistant workload for the mundane portions of homework grading (verifying student-level assignment) has been significantly reduced.
- Teaching assistants are free to give more detailed and personalized feedback to each student regarding overall technique and style rather than being forced to focus on programmatic minutia.

Ongoing Work
- Maintenance and bug fixing, security analysis.
- Increased language support.
- Expanded library of grading techniques for ease of use.
- Eleven more instructors are aware of the tool.