Project Description

This research seeks to devise and evaluate mechanisms that will enable students to get useful and reliable assessments of the proofs they write in mathematics and computer science classes from their peers—both students currently enrolled in the class and those who have taken the course recently and are now serving as graders. It also seeks to maximize the educational benefit to the students performing the assessments, by teaching them how to critically review and evaluate proofs. Within the scope of this proposal, the work will focus on developing and evaluating a system to support the assessment of proof-based assignments in large classes, with enrollments of 200 or more students. It will also begin the design and extension of a system that can support web-scale courses with 10,000 or more participants.

Our long-term goal is to create educational technology for helping students learn to write good proofs in large-scale learning enviroments. The resulting technology will also provide a platform for experimenting with different teaching and assessment strategies that will enable quantititative evaluation with an unprecedented degree of scale and granularity.

1 Motivation

Writing clear and concise proofs is a fundamental skill in mathematics. Writing a good proof requires the ability to rigorously decompose an argument into a sequence of steps that lead from a set of premises to a desired conclusion. It also requires the writer to communicate to the reader enough insight and information to guide them through the reasoning process at an appropriate level of detail.

Being able to write proofs is also a fundamental requirement in the education of computer scientists. Not only does it provide a mechanism where they can formally verify that a program will function correctly [6], it also enables them to analyze the time and space complexity of an algorithm or program. Even though few programmers prove theorems about their programs on a routine basis, knowing how to do so sharpens their ability to systematically reason about code as they create it. It also helps them reason through a specific execution when they try to debug it, determining both how they arrived at this point in the program and what changes must be made to prevent an error from occuring.

Learning how to write good proofs can be very challenging for many students. In many ways, it is like learning to write a compelling essay or to paint a beautiful picture—it requires internalizing a core set of rules and technical skills, while also learning how to communicate concepts to others. Just as with creative writing or art, the ideal learning environment for proof writing is one where students get personal attention by expert mentors.

Unfortunately, most undergraduate students get their early training in mathematics and computer science in large classes, where it is impractical for the instructor to give each student extensive individual guidance. If students are to learn to write proofs starting at an early point in their mathematical and computer science education, there must be a system for providing them accurate and timely assessment of their work even in the context of high enrollment classes. Such assessments are essential to help them improve their abilities (formative) and to give them proper credit for their

efforts (summative.)

As a case in point, we recently revised the undergraduate computer science curriculum at Carnegie Mellon University so that all majors and many nonmajors take two programming courses—one on imperative programming and one on functional programming—that have students writing and proving preconditions, postconditions, and invariants [2]. Students also take courses in discrete mathematics and introductory theoretical computer science that involve proofs and formal reasoning, all within their first two years. These courses each have total annual enrollments of 200–500 students. We handle our assessment needs with large teams of undergraduate graders, who are among our best students, but they still lack extensive experience in the subject material and in critically evaluating proofs. Ensuring reliable, consistent, and uniform grading in such environments has proved challenging.

Going beyond the need to provide assessments for classes with enrollments in the hundreds, consider the recent advent of *web-scale* courses, such as Massively Open Online Courses (MOOCs), offered to 10,000 or more students for free over the Internet [3, 8]. Although it is easy enough to provide simple assessments, such as multiple-choice questions, to these students, there is no feasible way to have students attempt to write proofs and then get reliable feedback on their efforts. This shortcoming seriously reduces the degree to which the participants can gain a deep understanding of the theoretical aspects of computer science.

In computer science terms, we require a system for providing assessments of proofs that is *scalable*, meaning that it can meet the quantitative and qualitative demands for assessments using the limited resources available. We consider scaling at both the large-class and web-scale levels, but with resources limited to: 1) an instructor, 2) a small team of assistants, who may range in talent from almost novices to near experts, and 3) the students themselves, who are just learning the material and who have almost no experience in critically evaluating proofs.

We approach this problem as one of *human computation*, using computer technology to devise systems in which useful work can be done by harnessing the power of large numbers of nonexperts [10, 12]. This requires the task to be decomposed into smaller units of work, each of which can be performed by the available talent pool. It also requires mechanisms to generate these units, supply them to the workers, and then aggregate the results, such that the outcome is valid, even when some of the workers either inadvertently or maliciously misuse the system.

2 Existing Work on Peer Assessment

As background, we consider the work done by others, as well as the prototype system we have developed.

2.1 Published work

The published literature on teaching students how to write good proofs and how to assess their work is remarkably sparse. Most published work focuses on creating a tightly knit group of students working under the close supervision of an instructor [4, 7], or a small-enrollment, seminar-style course [11]. These environments do allow students to discuss and evaluate each others work, but

they cannot be scaled to larger classes. Indeed, it seems that most math programs avoid proof-based assignments with their large-enrollment, introductory courses.

The common approach for grading any assignment for a large class with multiple graders is for the instructor to generate a *scoring rubric* giving a detailed description of what components the solution must contain, how each of these components should be evaluated, and how points should be assigned [9].

Perhaps the most relevant prior work was done by Zerr at the University of North Dakota [14]. He instituted formal peer evaluation, but in a way that actually increased the required effort by the instructor, rather than reducing it. Even so, it produced some useful insights into the challenges of peer assessment. They found the students were good at identifying correct proofs and then supplying useful advice on how to improve the presentation. For incorrect proofs, however, many of the reviewers failed to even detect that they were incorrect, and even fewer were able to pinpoint the exact error.

Other computer scientists have created generic, online peer review systems [5, 13], but these did not address the assessment of proofs explicitly, and indeed provided little or no framework for guiding the assessment process.

2.2 Our Prototype System

At Carnegie Mellon University, one of our undergraduate graders, Adam Blank, created a system that serves as the prototype for the proposed research. Indeed, he is now a PhD student and this proposal seeks funding to support him. The system is designed to support large classes, with the assessment performed by the students in the class and by undergraduate graders. It has been used in classes with enrollments ranging from 60 to 380 students, in both computer science (introduction to theoretical computer science) and mathematics (discrete math.)

Elements of this system demonstrate how a scalable and reliable peer assessment system can work. First, the task of assessing a proof is defined as one of identifying a set of *attributes* that are applicable to a particular proof. These are positive and negative facts that hold for the proof. Example attributes include:

- The proof is by induction on the size of the array.
- The proof uses linearity of expectation.
- The base case of the induction is correct.
- The proof contains an arithmetic error.

The list of possible attributes for a problem is generated by the instructor and by experienced graders, based on grading a sample of around 15 proofs. Of the 148 problems that have been graded with attribute-based grading, nine had 100 or more applicable attributes, while the average was 40.6.

The actual grade for a proof is computed using a formula, devised by the instructor, based on which attributes apply. This mechanism nearly eliminates any subjective evaluation by the less

experienced graders or the student reviewers and ensures uniformity of the grading standards across the team of graders. Each proof is reviewed by 3–5 students and by one grader. The task for the student reviewers is to identify which attributes they believe apply to a proof, while for the grader it is to filter through the proposed attributes to see which ones really do apply. Having multiple reviews increases the chance that the set of possible attributes will be complete. To avoid collusion, students are required to identify their collaborators, forming edges in an undirected graph with students as nodes. The set of students is partitioned according to the connected components of the graph, and reviewers are assigned randomly, but such that they never review assignments generated by students in their own components.

Reviewers are graded on the quality of their assessments, comparing their results with those generated by the grader. This is done with a simple 0–2 point scale (0 = useless, 1 = somewhat helpful, and 2 = accurate.) These reviews comprise around 10% of the course grade, and so students are highly motivated to do a good job, although more by coercion than by any perception of benefit on their part.

We can see how this prototype system implements a form of human computation, enabling nonexperts to perform useful and reliable work. The normally subjective task of assessing a proof is put into a structured framework, such that it can be decomposed into a number of simpler tasks that can be performed by nonexperts. Workers are assigned different types of tasks according to their capabilities and trustworthiness. Randomization, redundancy, and selectivity are used in the task assignment to improve quality and to ensure integrity of the process.

Although the conditions for and scale of the deployment of our prototype system were insufficient to perform a systematic evaluation, we have gained some insights from anecdotal evidence and from a survey conducted in one of the classes. First, the system worked better than existing rubric-based approaches. The graders found that the student reviewers generally identified all of the applicable attributes for a proof, although their selections had to be pruned down and occasionally augmented. The graders found having this input from the student reviewers useful, even though they still had to carefully review the proofs themselves. Second, the instructors found the grades assigned to the proofs to generally be fair assessments of their quality. The idea of generating a score automatically from the set of applicable attributes was seen as greatly improving the uniformity of the grading, while also yielding appropriate scores. Of course, we cannot claim these results without a more rigorous evaluation, but they lend hope that the overall approach is viable.

A survey conducted of students in one class indicated that they had mixed opinions about being required to participate in the reviewing process. The major complaint was that they were being asked to do extra work in a course that was already perceived as requiring too much work. Even so, when asked whether they found performing the reviews to be helpful to their understanding of the material, 42% said yes, 28% said no, and 30% were neutral. So, there is some justification for claiming that having students perform peer reviews has an educational value, but more must be done to maximize the real and perceived educational benefit of doing reviews.

2.3 Summary of Prior Work

We see from this survey of prior work that our system stands out as unique in providing a structured environment for students to review each other's proofs in a structured way. This enables quality assessment to be performed using the students themselves and a large staff of undergraduate graders. Nonetheless, our system is only a prototype and requires much more work to solidify its design and to evaluate its effectiveness.

3 Coping with Scale

Most large classes, including those using our prototype system, use a two-level hierarchy for assessing student work. A single instructor guides the work of a staff of graders, who then oversee the efforts of a much larger groups of students. One simple model for such a system is to assume that a single person can monitor the work of around k others. Then a two-level hierarchy can have one instructor running a class with k staff members and k^2 students. A typical value range for k might be 20–40, making it possible to have classes with enrollments ranging from 400 to 1,600. Although this model is obviously simplistic, we can see some validation with very large courses. For example, Prof. James Maas of Cornell University was renowned for teaching an introductory psychology course with 1,600 students, staffed by 22 teaching assistants.[1]. Harvard's introductory computer science course CS 50 has an enrollment of around 600 students, and their web page lists over 50 staff members. In the former case, the graduate teaching assistants working for Prof. Maas could probably handle a heavier load than could the undergraduates staffing the Harvard course. (We can also see more than a simple two-level structure in the Harvard class, with some staff members serving as teaching assistants and others as graders, but our simplified model suffices for discussion purposes.

A two-level hierarchy suffices for almost any course at a single institution, but it cannot grow to web scale, in that it would require our factor k to be 100 or more to staff a course with over 10,000 students. One could institute a deeper hierarchy, requiring $\log_k n$ levels of staffing for a course with n students. This would require a total staff of around $n/(k-1)$ people. For example, for $k = 30$, a three-level hierarchy could handle 27,000 students, but it would require over 900 staff members. This would not be economically feasible for a MOOC, where the intention is to offer the course at little or no cost. Instead, we must find ways to fill out the lower ranks of the hierarchy using volunteer labor. This can be done by identifying the most capable students and providing them with incentives to help the others. In addition, it may be possible to recruit volunteers who already know the material, perhaps by already having taken the course. The challenge with this approach is then to devise a suitable incentive system and to provide the necessary framework for the volunteers, and the students themselves, to handle most of the load in performing assessment.

4 Proposed Work

Our long-term goal is to create educational technology that enables students to learn how to write and evaluate proofs. We propose building on the ideas embodied in our prototype to support

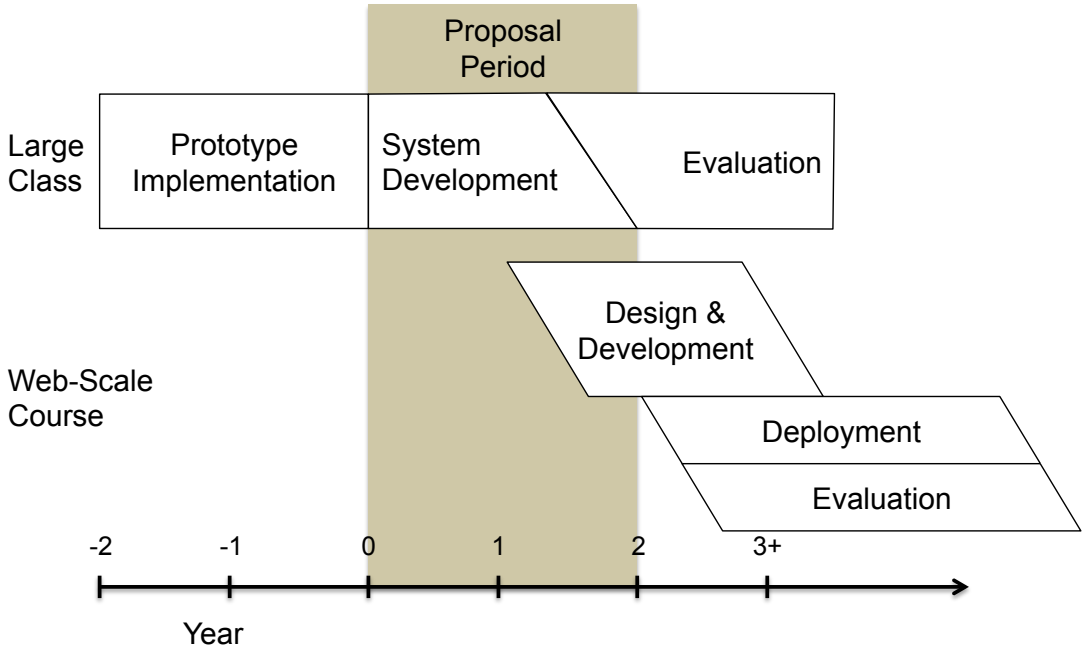

Figure 1: Proposed Activity Schedule

both large classes and web-scale courses. We will design our systems with the twin objectives of maximizing the quality of the assessments produced, as well as value students gain by performing the peer assessments.

Figure 1 illustrates how work over the two-year period funded by this proposal fits into the overall plan. As is indicated, we have already invested two years of work in building and deploying our prototype system. During the proposal period, we would concentrate mainly on supporting large classes using an improved system, and also doing some work to evaluate its effectiveness.

We would also begin work on a system to support web-scale courses, but we anticipate that the majority of its design, implementation, deployment and evaluation would take place after we have gained more experience in large-class environments.

In the following we identify specific areas requiring further attention and ideas for how they could be improved. We also describe possible evaluation approaches.

Just as with our prototype, we plan to deploy and evaluate the technology as it is being developed within computer science and mathematics courses at Carnegie Mellon University. This will give us access to actual students and undergraduate graders in the context of actual courses, all at no cost to the research project.

4.1 Improving Assessment Quality

Our first goal will be to create an assessment system that provides reliable and accurate assessments in the context of a large, undergraduate class. It assumes that we have a staff of undergraduate

graders, ranging from those who have just taken the class to those who have several semesters of grading experience and who have gained mathematical maturity by taking more advanced courses.

Overall, the strategy of breaking the assessment of a proof down into a smaller set of attributes seems effective, both to make the task manageable by both student reviewers and graders, and to also enable a uniform system for assigning scores. In our current system, however, the set of attributes is simply tabulated as a flat list. We have found that this lack of assessment structure fails to provide enough guidance to the reviewers and the graders regarding how the different aspects of a proof should fit together.

Pathway verification

We would like to create a more structured representation of proofs we call *pathway verification,* where the required components of a proof are explicitly identified. Some example structures include:

- A proof by induction must contain an induction hypothesis, proofs of the base cases, and proofs of the induction steps.
- The proof of an if-and-only-if property must have proofs of the "if" part and the "only if" part.
- A case analysis must include proofs of each possible case.

In addition, more complex proofs should be broken down into a set of lemmas with a valid dependency structure.

For problems assigned in the introductory and intermediate-level courses we are targeting, there are only be a limited number of possible pathway structures students can use. These will be determined by the instructional staff when devising the assignment and by grading a sample of 15–25 proofs. In doing their reviews, the students will first identify the relevant pathway structure, and then they will select attributes related to how successfully each component of the pathway is covered in the proof. The graders will then refine the student reviews to devise the precise set of attributes.

We believe that the greater structure provided by pathways will provide helpful guidance to the students and the graders when doing their assessments. It will also provide a useful tool for helping students better understand how a proof should be structured. One could imagine, for example, assigning exercises where the desired pathway for a proof is given, and the students must then structure their proofs accordingly.

Assigning graders and reviewers

Our current method of assigning reviewers to proofs attempts to avoid conflicts of interest, but it otherwise is completely random. We believe we could achieve better results by making use of the heterogeneity of the students, both as proof generators and reviewers, and the grading staff. For example, a common strategy in grading papers is to first look at the ones from the top students

to determine what the best quality results are likely to be. We can gain some idea of the likely difficulty of assessing a proof from the prior history for this student, as well as by some statistical analysis of the text. Assuming Zerr's experience that correct proofs are much easier to grade than incorrect ones, we can bias the assignment of reviewers and graders so that the more challenging ones are assigned to the most qualified assessors.

Evaluation

There are multiple methods for measuring the effectiveness of an assessment technique. The most straightforward is to compare the results to those generated by qualified assessors. This can readily be done by having a graduate student independently grade a set of assignments, and then doing a comparison with the assessments generated by our system. In addition, we can evaluate the uniformity of the assessment system by having multiple, independent assessments of a single set of proofs, and similarly measure the effectiveness of different assessment systems.

4.2 Improving Learning Experience

We can force students to provide peer reviews of each others proofs by making this an aspect of their course grades, but we would rather motivate by having it be a useful learning exercise. We want to assign each student a set of proofs that represents a diversity of possible solutions and that will be appropriate for their skill level. In addition, we want their assessment efforts to give them a better understanding of how a correct proof should be structured, and the likely ways in which a proof can be invalid.

We believe the learning value of peer assessment can be increased by a more careful assignment of proofs to student reviewers. Based on the prior history of the student writing the proof and by statistical language analysis, we should be able to estimate properties, such as 1) whether the proof likely to be correct, 2) how clear the presentation will be, and 3) what pathway structure does it follow. From this, we can ensure that students are assigned different types of proofs. Moreover, we can exploit the heterogeneity of the students and graders, assigning the most challenging proofs to the people with the highest abilities.

In addition, we believe that peer reviewers will better internalize the appropriate proof structures by performing their reviews in the context of pathway structures. This will get a detailed view of alternate ways to prove something and to pinpoint places where a proof is either invalid or incomplete.

Evaluating the students' perception of the learning benefit of performing peer reviews can readily be done via surveys. Evaluating the actual learning benefit is far more difficult, especially with the many factors that cannot be controlled, and the small sample size provided by a single class. We believe that this aspect of the assessment system will best be evaluated when we move to web-scale courses. Such an environment provides more opportunities to do A/B testing, where the students are partitioned into different groups, with different variations of the assessment system applied to each group. We can then monitor the performance of the different student groups in their later assignments.

4.3 Scaling to Web-Scale Courses

Within the two-year period covered by this proposal, we would only begin our efforts to support web-scale courses. Nonetheless, it seems worthwhile to describe some of our thinking about what would be required to handle very large courses with limited resources, as well as some discussion on what a rich environment this would create for rigorously experimenting with different learning and assessment methods.

As we have discussed, a true web-scale course cannot rely on a simple two-level hierarchy. In fact, if the course is to be offered at little or no cost, it cannot even involve a system where assistants are being paid to directly oversee the efforts of the students. Instead, we must find a method to recruit volunteer labor, including current and former students from the course. In addition to providing incentives to motivate these volunteers, we must have a very structured and reliable framework that will ensure that the assessments are of high quality.

On the other hand, given that current and planned MOOCs do not provide any form of authenticated credential, the purpose of assessment in these courses is more to help the student (formative) rather than to measure performance (summative.) There is therefore less need for uniformity and reliability of the assessment and for preventing collusion among the students and between students and assessors.

We believe that our system for supporting assessments of proofs in large classes can be scaled to handle web-scale classes. Typically, the web-scale course would be based on one that had already been taught using our system, and so a sufficient set of possible attributes and pathway structures would already have been generated.

Evaluation: Online course provides an ideal platform for evaluation. Can do A/B testing. Have huge amount of digital data. Scaling to much larger environments

- Limit effort by instructor and staff to setting up structure and testing by grading small set of assignments.
- Typically, would prototype with class of 200+ students and use this to generate set of possible attributes and pathways.
- Students must be able to reliably assess correctness.
- Must provide motivation to students to participate and do a good job.
  - Reward verifiers with more verifications of their own work.
  - Identify most skilled verifiers and reward with special status or job opportunities.
- Exploit range of verifier capabilities. Most capable would identify cases where new attributes or pathways required.

Evaluation of effectiveness

- Analyze accuracy and consistency of student-generated assessments, compared to ones generated by experts and to each other.

- Measure effectiveness of verification as a learning experience. Compare student performance to that in other offerings of course. In web-scale environment, could do more A/B testing where different students are given different degrees of verification support.
- Use as data set for understanding how students learn to do proofs.
- Use as structured methodology for teaching how to write good proofs.
  - Explicit use of pathway structure.
  - Distill into Proof Concept Inventory.

4.4 Additional Deployments

- Incorporate into 15-122 for evaluating program invariants.
- Deploy at some other institution
  - Need large classes to ensure redundancy and anonymity.
  - Need to test on students with range of abilities.

4.5 Beyond grading proofs

- Use to assess and teach good programming style.

5 Results from Prior NSF Support

References

[1] K. W. Arenson. Lining up to get a lecture; a class with 1,600 students and one popular teacher. *New York Times*, November 17 2000.

[2] R. E. Bryant, K. Sutner, and M. J. Stehlik. Introductory computer science education at Carnegie Mellon University: A deans' perspective. Technical Report CMU-CS-10-140, Carnegie Mellon University School of Computer Science, 2010.

[3] S. Carson and J. P. Schmidt. The massive open online professor. *Academic Matters*, May 2012.

[4] D. W. Cohen. A modified Moore method for teaching undergraduate mathematics. *American Mathematical Monthly*, 89(7):473–474, 487–490, 1982.

[5] E. F. Gehringer, D. D. Chinn, M. A. Pérez-Quiñones, and M. A. Ardis. Using peer review in teaching computing. *SIGCSE Bulletin*, 37(1):321–322, February 2005.

[6] C. A. R. Hoare. An axiomatic basis for computer programming. *Comm. ACM*, 12(10):576–580, October 1969.

[7] F. B. Jones. The Moore method. *American Mathematical Monthly*, 84(4):273–278, 1977.

[8] T. Lewin. Instruction for masses knocks down campus walls. *New York Times*, March 4 2012.

[9] B. M. Moskal and J. A. Leydens. Scoring rubric development: Validity and reliability. *Practical Assessment, Research and Evaluation*, 7(10), 2000.

[10] A. J. Quinn and B. B. Bederson. Human computation: A survey and taxonomy of a growing field. *Conference on Human Factors in Computing Systems (CHI '11)*, pages 1403–1412, 2011.

[11] R. B. Reisel. How to construct and analyze proofs—a seminar course. *American Mathematical Monthly*, 89(7):490–492, 1982.

[12] L. von Ahn. Human computation. Technical Report CMU-CS-05-193, Carnegie Mellon University School of Computer Science, 2005.

[13] W. J. Wolfe. Online student peer reviews. In *Proceedings of the 5th Conference on Information Technology Education*, pages 33–37. ACM, 2004.

[14] J. M. Zerr and R. J. Zerr. Learning from their mistakes: Using students' incorrect proofs as a pedagogical tool. *PRIMUS: Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 21(6):530–544, 2011.