CMG: Collaborative Machine-facilitated Grading

Adam Blank

1 Development of ColorMyGraph

This section will discuss the development of and technologies used for programming ColorMyGraph.

2 Scope and Applicability

Throughout this paper, we restrict our attention to introductory proof-based mathematics. Many of the techniques and results generalize to other disciplines naturally, and indeed, we plan to test this experimentally in the near future. Thus far, we have tailored our techniques to 15-251 ("Great Theoretical Ideas in Computer Science") and 21-127 ("Concepts of Mathematics") at Carnegie Mellon.

3 Judgement-Based Grading

Traditionally, we grade student proofs by printing them out (or accepting handwritten submissions) and writing down the places in which students make mistakes on the physical student paper. Ideally, in the interest of being consistent among students, we would like to have one grader grade each problem; the larger the number of people grading a problem, the more likely that two of them will disagree on what points should be allocated for. If a particular grader grades every student, the students are all held to the same standard.

If we want problems to be assigned to graders, the first most obvious issue that graders have is that only one grader can physically have the paper at any given time. Some course staffs rectify this problem by asking the students to print out each question on a separate page, but this creates a logistical nightmare for collection and distribution of homework. Every problem must be put in the right pile, filtered to the right grader, recombined to enter the grades, and finally distributed back to the student. Our proposed solution requires that the students learn a minimal amount of LaTeX to typeset their proofs. ColorMyGraph, our course management software, can automatically parse our auto-generated LaTeX template into individual questions if the course staff decides that it still wants to grade on paper. ColorMyGraph also provides an interface for entering individual problem grades if necessary as well.

Paper grading, even enhanced by ColorMyGraph, still has several unavoidable issues: consistency in scores, redundancy in comments, and bias toward certain students. Our solution, which ColorMyGraph implements, is called Judgement-Based Grading.

A *judgement* is a comment about a proof that a grader might want to write down on a student's paper to help them understand what they did wrong. Usually, judgements are written so as to potentially apply to multiple students' proofs. For instance, if the proof needs to calculate $\frac{10}{5}$ but writes 3 or 4, these could both be encapsulated in the judgement "The proof incorrectly calculates $\frac{10}{5}$."

The largest difference between traditional grading and Judgement-Based Grading is that when grading a submission with Judgement-Based Grading, any time the grader would have normally written a comment on the student's paper, he or she creates a new judgement instead. The judgements are saved and listed next to every proof as they are graded. The novel thing here is

that the grader can spend a long time crafting the wording of that single judgement, because he or she won't have to write it on all 200 submissions. One concern about this approach is that the list of judgements might often be unmanageable. Of the 148 problems graded using Judgement-Based Grading on ColorMyGraph, 9 have 100 or more judgements, and the average number of judgements is 40.574 as can be seen in Figure 3 below.

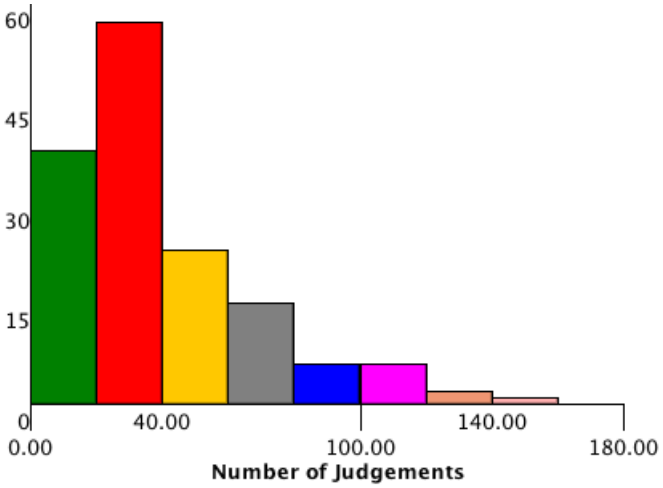

Figure 1: Histogram of Number of Judgements over Problems

One other major change with Judgement-Based Grading is that writing a comment on a student's paper is separated from taking off points for that mistake. With Judgement-Based Grading, a grader will only consider point values once he or she is done creating all of the judgements for a problem. The implementation on ColorMyGraph supports several common grading schemes such as adding/subtracting for a comment and giving a minimum/maximum score based on a comment. This forces the grader to be specific enough to differentiate between to different mistakes that lose the same number of points, because if the same judgement is given twice, it will always be worth the same amount. Furthermore, this allows the grader to go back and change his or her rubric as necessary without wasting a large amount of time going back through all the graded submissions. Of the same 148 problems, graded using Judgement-Based Grading on ColorMyGraph, 91 problems have a judgement that was used at least 40 times as seen in Figure 2 below.

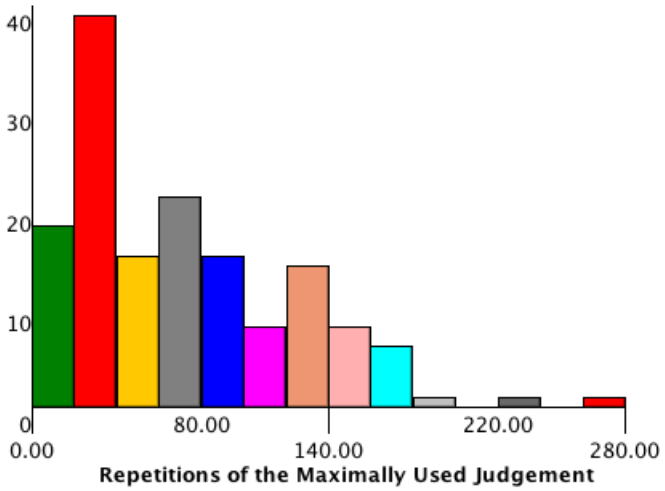

Figure 2: Histogram of Usage of Maximally Number of Judgements over Problems

Finally, because the submissions are electronic, all identifying information is stripped from the submissions before the grader sees them.

4 Verification Assignments

The core of this paper is about the new type of homework assignment we created which we coined a "Verification Assignment." The idea is to distribute some number of submissions to each student and ask them to explain what is correct and incorrect about each submission. Before distributing proofs to students, the grader of the problem being verified grades some small number of submissions (usually around 10-15) using Judgement-Based Grading. This populates the vast majority of judgements for the problem. We've found that usually very few additional judgements are necessary to grade the submissions. This scaffolding makes it possible for the students, who just learned the topic they are verifying, to actually do accurate verifications.

ColorMyGraph implements the distribution piece by creating n perfect matchings (one for each verification we want the students to do) from verifiers (students) to submitters (students). Upon submission of their assignments, the students are required to write down the students they collaborated with. Before finding perfect matchings, ColorMyGraph removes all edges between students who are *connected* in the collaboration graph. This reduces the chances that students will know the person they are verifying.

Once the judgements are pre-populated, and the submissions are distributed, the students are given several days to complete the verifications.

5 The Need for Verifications

Verifications provide several benefits that are not traditionally present in courses such as displaying all the possible mistakes a student could have made and training them to get better at checking their own work. This section details several of the most important reasons that we should use verifications in our courses.

In a traditional course without verification assignments, assessments like exams and homework test students' ability to solve problems and write down their ideas, but students are never assessed

on their ability to check their own work. This can be particularly damning when students regularly turn in proofs that they believe are completely correct that miss key points and get failing grades. Verifications fill this hole in their education, because they train the students to get better at judging the correctness of their own (and their fellow students') proofs.

It is commonly accepted that being a teaching assistant for a course allows you to really internalize the material for the first time. One major reason for this phenomenon is that TAs have to read a ludicrous number of submissions and, as a result, get exposed to many different ways of thinking about material in the course. Verifications allow the students to get some of this exposure in a controlled way that will benefit them over time. In particular, by pre-populating most of the judgements, identification of errors is reduced to recognition of errors, which is a more reasonable task to ask the students to do.

Furthermore, asking the students to figure out if each error occurs in submissions, forces them to understand *all* of the possible errors they could have made that the course staff deemed to be important. In particular, reading and internalizing a list of possible mistakes makes the students less likely to make similar mistakes in the future. This can be even more beneficial when a problem has multiple solutions. Verifying proofs exposes the students to many different solutions that they otherwise never would have seen. Students can pick up tricks and ideas that will help them later in the course in this way.

Finally, in introductory courses, students are often introduced to new ways of writing their solutions. In particular, it's really hard for students to understand what exactly constitutes a proof in the beginning. Verifications allow the students to see lots of examples and styles (some of which are good and some of which aren't) that they usually wouldn't see in a large volume until they went through several courses. This aspect of not knowing what is expected is certainly not unique to proofs; one topic that has a similar start-up cost is programming style.

6 Verification Implementations

We have explored several different implementations of verifications. All of the testing has been done with List Verifications, but we plan to implement Pathway Verifications as a solution to many of the deficiencies of List Verifications.

6.1 List Verifications

6.1.1 Description

List verifications have four parts: summary, plans, judgements, and understandability. The ColorMyGraph UI for List Verifications can be seen in the figure below.

Summary. A summary is a short description of a proof that details what the proof does that might be different than another similar one. It ultimately has two purposes. First, it allows the student to put down in his/her own words what the argument is doing. Second, it allows a TA to quickly figure out if the student actually understood the proof.

Plans. Plans are high-level, one sentence explanations of possible approaches students could make to a problem. These should be entirely populated by course staff, and serve almost exclusively as a way of grouping together student responses.

Judgements. Judgements are descriptions of individual mistakes and successes that are distilled from student submissions. These make up the core of the system. Since a large percentage of

students make the same mistakes in their responses, a list can be created and maintained that students can select from to explain what went right or wrong. The system would ask course staff to pre-populate many of these based on doing a subset of the grading before students begin.

Understandability. The understandability section is about ascertaining information about both how well the student reading the proof was able to understand it, and also information about how well the argument was conveyed in the writing. This would be a static set of check-boxes that should uncover these fundamental things.

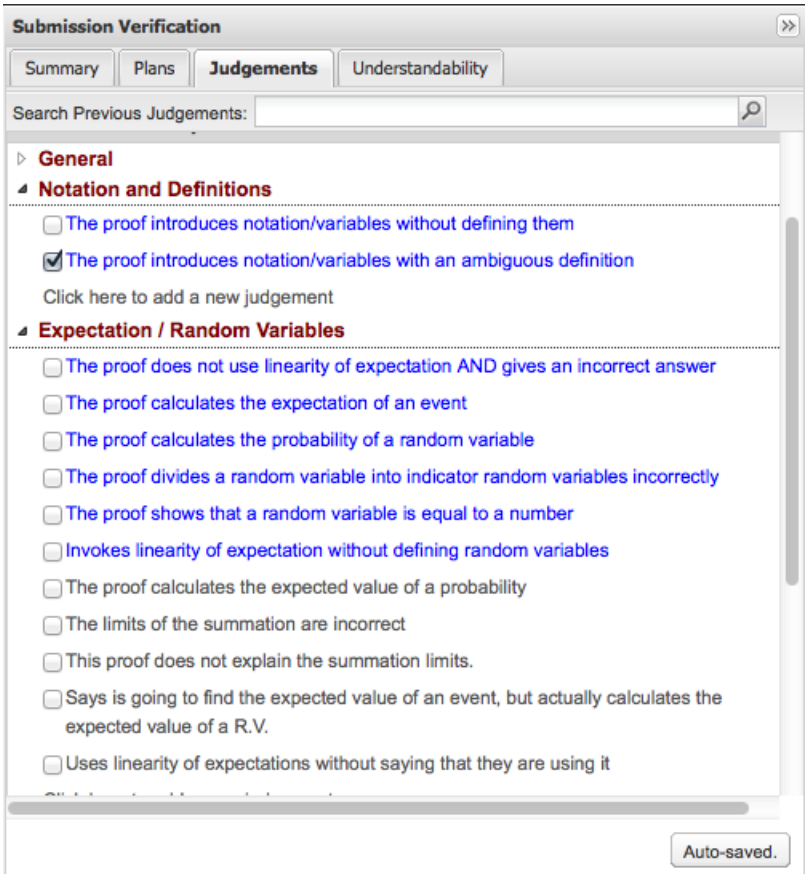

Figure 3: List Verifications on ColorMyGraph

6.1.2 Example: Bunny Duck and the Chamberer of Musicer

The following is a sample problem with judgements after students verified it.

> While on his morning run, Bunny accidentally dropped his duckPod into a lake, frying it. So he upgraded to the duckPod 2.0. After one of the prototype duckPod 2.0s played the Ma¢k€y song "We R(5,5) Who We R(5,5)" five times in a row, the algorithm was changed to disallow the same song to be played twice. Bunny starts with the first song and listens to all 1000 songs. Bunny's music collection contains 100 different artists, each of whom wrote 10 songs. What is the expected number of times Bunny will hear the same artist play three songs in a row. (If four Luis Gaga songs are played in a row, that counts as three in a row occurring twice).

- General
  - The proof is blank
  - The proof is not blank, but makes no progress towards a solution
  - The proof is 100% correct
  - The proof gives the correct answer, 8/111.
  - The proof gives an incorrect answer or no answer.
  - The proof makes an algebraic error in calculating the answer.
  - The proof keeps the final answer in the form of a summation
  - This proof rounds the final answer in decimal form.
  - Proof claims the answer as an upper bound
  - Claims the expected value is less than 1 so it is 0
  - Proof assumes that songs are replaced after each triple
- Expectation / Random Variables
  - The proof does not use linearity of expectation AND gives an incorrect answer
  - The proof calculates the expectation of an event
  - The proof calculates the probability of a random variable
  - The proof divides a random variable into indicator random variables incorrectly
  - The proof shows that a random variable is equal to a number
  - Invokes linearity of expectation without defining random variables
  - The proof calculates the expected value of a probability
  - The limits of the summation are incorrect
  - This proof does not explain the summation limits.
  - Says is going to find the expected value of an event, but actually calculates the expected value of a R.V.
  - Uses linearity of expectations without saying that they are using it
- Probability / Events
  - The proof incorrectly calculates the probability of three adjacent songs being played by the smae artist

- The probability of three adjacent songs being played by the same artist is given without proof
- Claims the probability of two songs being played by the same artist is independent of what was played before
- The proof does not calculate the probability of three adjacent songs being played by the same artist

7 Pathway Verifications

Pathway verifications eliminate the four sections of list verifications in favor of a directed graph representation of the solution space. There are two types of pathways which together compose pathway verifications: Technique pathways and lemma pathways. A technique pathways splits a claim up into pieces (usually via a template that describes the technique); each of these pieces is a list of judgement nodes and a brief description of the piece of the proof being verified. Technique pathways only have to be written once, although the individual judgement nodes may change, because techniques are often duplicated across proofs. Lemma pathways are for proofs that use several sub-claims that are independent; they are made up of option nodes, lemma nodes, and judgement nodes. Option nodes allow the verifier to point out which of several ways the prover used to show a particular necessary part of the proof. Lemma nodes make several requirements to proof a claim, each of which has their own pathway graph.

The major conceptual difference between list verifications and pathway verifications is that the structure of the solution space of a problem is pre-built into pathway verifications, whereas verifiers would have had to derive it from very little context in list verifications. One other major improvement is that ColorMyGraph can more easily figure out if a verification is completely bogus with pathway verifications, because they provide more structure.

8 Results and Statistics

There are several important factors that we have used to determine the effects that verifications have on students. This section will have more results and actual data in the final draft.

8.1 Student Reception

Verifications have typically deployed in courses that already have a large amount of work every week. One constant complaint that we had was that the verifications were "extra work" that the students were not fond of. Despite this, students from all types of grades in the courses have identified that they find verifications helpful. We ran a survey once and found that the majority of people had feelings about verifications that were either neutral or positive.

8.2 Correlation to Grades

One, potentially surprising, result is the poor correlation between verifications and grade in the class. Students who are among the very best often do a poor job verifying other students' proofs.

8.3 Improvements in Verifying Over Time

Finally, we've found that most students do get better at verifying over time. In particular, most students do poorly on their first few verifications, and then start doing significantly better on the

remaining ones.