ColorMyGraph: Student Proof Analysis and Verification

Adam Blank

1 Development of ColorMyGraph

ColorMyGraph is a course management web application that was written to support this research. It includes an interface for course staffs to create assignments, a submission system for LaTeX documents and programming assignments, a grading interface, and a feedback system for grades.

1.1 Technologies

ColorMyGraph is written in Python on top of the Django and Pinax frameworks. The front-end is mostly written in Javascript using the ExtJS framework. Many features of the application are time-based actions, such as asking the TAs to grade; ColorMyGraph uses Celery and ghetto-queue to support these features.

1.2 A Novel Solution for Grading Code

ColorMyGraph implements a novel type of autograder. To ensure that the student code does not harm the system, we use a `chroot` jail. First, the essential libraries to compile and run the code are put into a temporary folder along with a student tarball. Next, we use the command `chroot` to change the root directory to the temporary folder. Finally, we compile and run the students' code inside this jail and give the results via standard out. The solution is reasonably light-weight but still secure. It supports C++, Java, Python, and SML.

2 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.

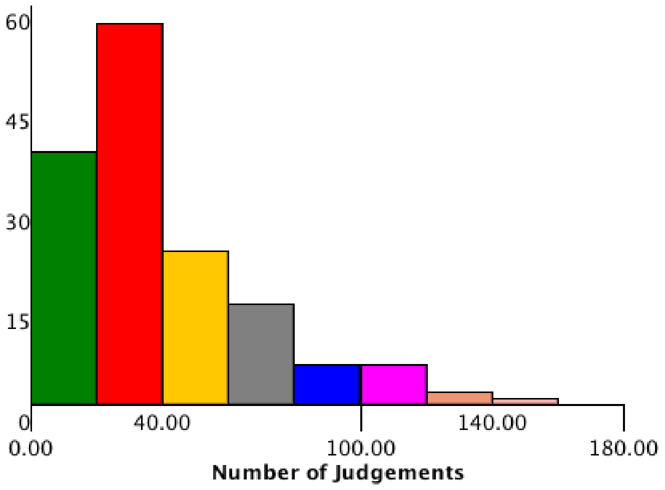


Figure 1: Histogram of Number of Judgements over Problems

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.

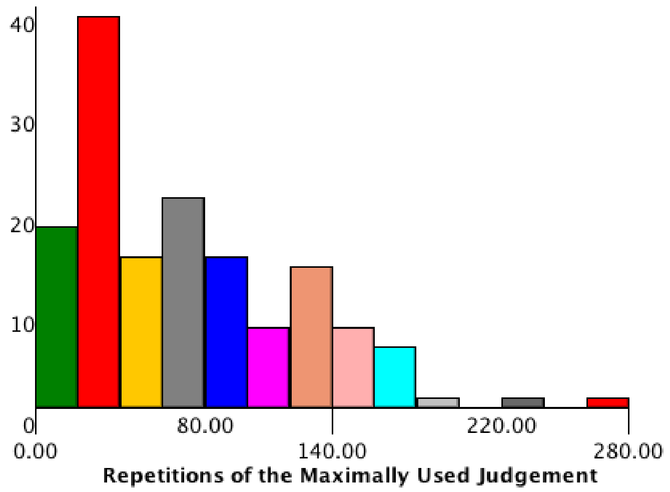


Figure 2: Histogram of Usage of Maximally Number of Judgements over Problems

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

3 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.

4 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.

5 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.

5.1 List Verifications

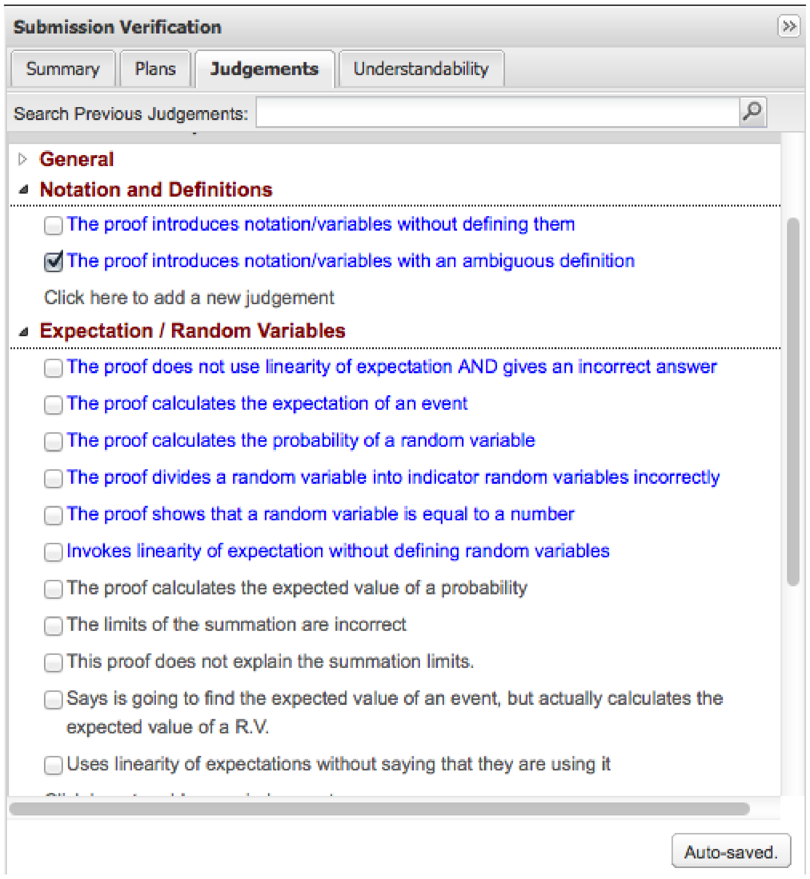
List verifications have four parts: summary, plans, judgements, and understandability. The ColorMyGraph UI for List Verifications can be seen in the figure below. An example problem with judgements after verification is in the first appendix.

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.



General

Notation and Definitions

- [ ] The proof introduces notation/variables without defining them
- [x] The proof introduces notation/variables with an ambiguous definition

Click here to add a new judgement

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

Figure 3: List Verifications on ColorMyGraph

5.2 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.

6 Preliminary Deployment of Verifications

Over the course of the last few semesters, we have changed several aspects of the system in important ways. For the first half of the very first semester with verifications, we gave the students two verifications for each question they were assigned as homework. Students felt overwhelmed by the number of distinct questions, because every few verifications they had to start over in their understanding. Furthermore, two students don't give accurate enough answers for it to be useful to the graders most of the time. Another issue with that system was that there are some questions that students don't gain anything by verifying; in particular, there are usually a few easy questions that don't involve very much variation. Those verifications turned into a boring waste of time for the students. Since then, we have switched to a system where we hand-pick the questions that the students verify. Furthermore, because we decreased the number of questions, we increased the amount of repetition. We usually give between 3 and 5 of each question to be verified. One other major problem that we found was that students didn't understand the reason they were doing the verifications. Some students thought that the scores they gave their peers were their actual grades; other students didn't like that they were being exposed to different solutions. Many of these misconceptions can be seen by reading through the second appendix of this document. More recently, we have been allocating a small amount of class time to explain the reasons for verifications which seems to have helped.

7 Other Improvements to Grading

7.1 Submission Ordering

Since all student submissions are digital, we can pre-analyze the submissions for length, verbosity, and group members. This can make grading go significantly faster. In particular, ordering the submissions so that all of the students who worked together are graded one after another, allows the grader to speed through those solutions more quickly; he or she should obviously still check all the details, but it can really help to know that several proofs in a row will be the same. Additionally, some students write their answers in a confusing (but correct) way; if the grader reads a more clear version of the same argument, it's easier to give students with confusing proofs more accurate scores.

7.2 Anonymizing

When grading, the grader can usually see identifying information about the student. This can often bias him or her into giving a better or worse grade because of personal reasons or past experience with the student. ColorMyGraph enforces that all grading be anonymous to avoid this problem.

8 Results and Statistics

There are several important factors that we have used to determine the effects that verifications have on students.

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 with all types of grades in the courses have identified that they find verifications helpful. We ran a survey in Fall 2011 and found that the majority of people had feelings about verifications that were either neutral or positive. Specifically, the break-down was:

Positive response	22/53	0.415094339622642
Neutral response	16/53	0.30188679245283
Negative response	15/53	0.283018867924528

8.2 Correlation to Grades

One, potentially surprising, result is the poor correlation between verifications and grade in the class. We found an R^2 value of 0.0446 which indicates a terrible fit. We discuss possible reasons this number is artificially low in the next part of this section.

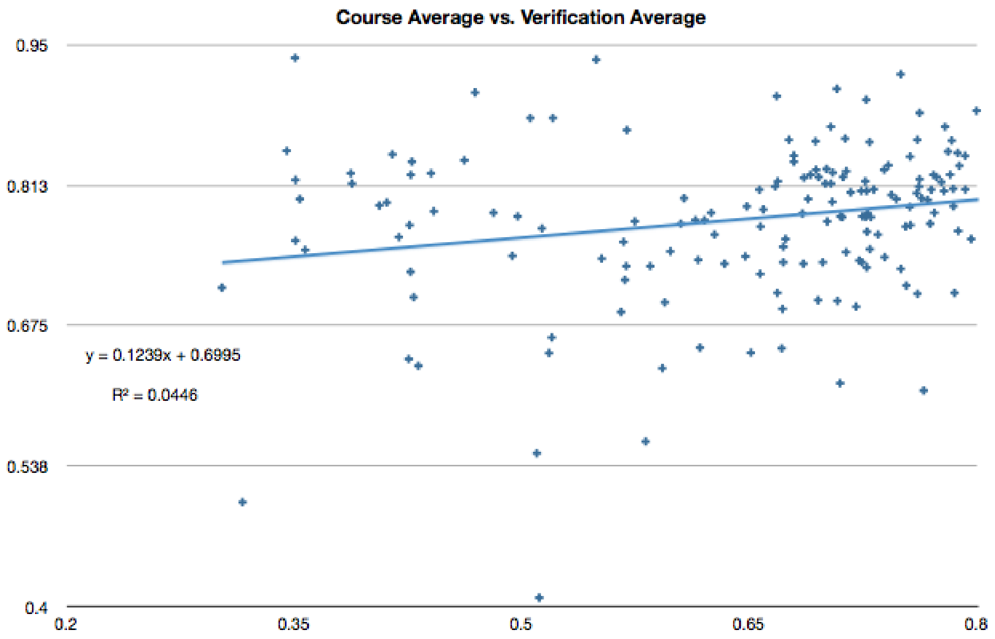


Figure 4: Correlation between student verification average and student course average

8.3 Possible Sources of Errors in Data

In the original system, we didn't filter out solutions that were blank; this caused some students' verification grades to be artificially inflated. Additionally, in the first semester that the system ran, one of the TAs ignored the students judgements and gave them all perfect scores rather than grading them. These two points could have caused the data to show fewer trends or correlations than it would have otherwise.

8.4 Writing Judgements is difficult

In several of the courses that used ColorMyGraph, we allowed the students to add their own judgements. This caused a large number of judgements that didn't make sense or weren't ever applicable to be in the list. We separated out these judgements by color-coding them, and the students fell into a pattern of never selecting the ones that were written by other students. Several of the people on course staff have also shown concerns about writing judgements; sometimes, it can be difficult to write a judgement general enough to apply to many proofs that's also specific enough to give good feedback.

9 Scope and Applicability

Thus far, we have restricted 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. We have tailored our techniques to 15-251 ("Great Theoretical Ideas in Computer Science") and 21-127 ("Concepts of Mathematics"), but there is very little that would change if we were verifying an essay or a chemistry solution instead. In fact, as long as the judgements are populated correctly, the current system should work as is.

10 Features and Ideas To Be Implemented

10.1 Pathway Verifications

Pathway verifications pose an interesting implementation challenge, because they have a much larger visual component than list verifications. In particular, since the solution space is represented as a graph, it must be shown as one to the user. Lists, on the other hand, have a very natural representation on a website. Additionally, to fully implement pathway verifications, the course staff needs a way to populate them. One of the positive aspects of pathway verifications is that once the common pathways are created, they can be re-used, but to get it running in the first place will take a significant effort.

10.2 Location-sensitive Judgements

A common complaint about judgement-based grading is that graders can't point out the location on the student paper that it was incorrect. While most judgements are clear without a location component, certain judgements like "The proof makes a minor arithmetic mistake" could be made a lot easier to understand with a location in the write-up. Additionally, location tags on judgements can help us better classify how correct the judgements the students enter are. If multiple students point out the same location, then we can be relatively sure that there is a problem in the proof *at that location.*

10.3 Collaborative Verifications

Another potential way to increase agreement among students is to add a collaboration aspect to the verifications. Students verifying the same proof would have a chat-like box available to them.

To avoid allowing people to just look at other conversations, there would be a time-out associated with conversations. Ideally, students would be in a "discussion phase", where they could talk to their peers, at first. Then, after they finished, they would signal to the system that they are ready to answer the verification.

10.4 Categorize Solutions Automatically via Machine Learning

It is undoubtedly more helpful for students to see varied proofs when they verify. If we could use machine learning to automatically classify proofs into several smaller groups, this could help us distribute them better. Thus far, ColorMyGraph has collected enough data to start attempting this with a small corpus.

10.5 Assign Proofs Based on Heuristics

As the semester continues, the students' grades and accuracy on verifications will give the system a good idea of which students are capable and which aren't. Importantly, as pointed out above, this has very little correlation to how good their grade is in the class. We could heuristically identify which proofs are "difficult"–both to read (i.e. common grammatical mistakes, bad typesetting, etc.) and conceptually (i.e. nobody else did the proof that way). This should help us get more accurate judgements from the students, and it can also help us make the random distribution more fair.

10.6 Proof Concept Inventory

A Concept Inventory is a multiple-choice exam that assesses conceptual knowledge of a very narrow topic. The majorly successful Force Concept Inventory, which was generated by Hestenes, Halloun, and Wells, has been used at many top-tier universities as a test for how successful introductory physics courses have been. The verification data over several semesters gives a large insight into how students view proofs. In particular, we can see what common errors they make, but also the common errors they miss. This could allow us to effectively generate a Proof Concept Inventory efficiently and quickly.

10.7 Supporting Large, Online Courses with Verifications

As we begin to offer real online courses to 50,000 people simultaneously, our paradigm for how to grade must change. Currently, the solution that has mostly been implemented is to move to assignments (like programming problems and multiple choice tests) which can be graded automatically. Unfortunately, via this scheme, you lose a lot of learning. In particular, if we look at an introductory math class where students learn how to write proofs, there is no way it can run with only these types of assignments. Verifications provide an outlet for these gigantic courses to use their size as an advantage. Using several of the techniques discussed above in this section, we can classify the proofs and find an optimal way to distribute them to students. Furthermore, because the course is free, we could set up a system where the students do verifications (potentially even a larger number than we ask for in courses at CMU) in exchange for the feedback on their assignments.

A Example List Verification: 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

A Full feedback for verifications in Fall 2012, 15-251

Do verifications help you to better understand the solutions and related concepts for the problems you're verifying?	**Do you think that the amount of time you spend on verifications gives you a reasonable amount of return? If not, do you have any specific suggestions to make them more useful to you?**
(six responses like this) Yes	Yes
Somewhat. I feel that they would help more if certain proofs were specifically selected, as I have verified several incomplete proofs.	No. Simply receiving which judgements I selected match/don't match is not very helpful. I would like to see more specific feedback, which requires additional time at office hours.
Sometimes.	huh?
Yes	Slightly less verifications please. :)
Absolutely.	I like verifications a lot as currently implemented. Requiring to read the correct and wrong solutions is great for constructing new solutions.
No, but they help show how to/how to not write good proofs.	I do not spend nearly as much time on them as Mark said I should (though I've done well on them so far).
Not really...	?idk? As in value of doing them, or having the homework returned on time? In theorey, I understand what verifications are for but in practice, I don't feel like I get much out of them except ... The most subtle things that...
Yes.	No. I am not sure I am verifying well. I wish there are guidline like solution of HW.
Yes.	Maybe have 3 versions of 3 problems instad of 5 versions of 2 problems.
To some degrees, see different ways of approaching problems.	Go over some in recitation.
A little	Yes
A little. I can also see how other people solve the same question. Kind of fun.	Yes.
Sometimes, but only in relation to seeing variations of my solution/the model solution	Yes

Occasionally they demonstrate a method of proof that I did not use/didn't occur to me then they are useful. Otherwise, I haven't noticed them helping.	If I could see one or two solutions like my own, and several that used a different method, that would be ideal (of course, that requires pre-screening all the proofs, which is probably infeasible) as it is, they just take away from 251 late day or recovery time.
Not really. I spend more time thinking about logical rigor than anything else.	I think my return is not in the intended area. I don't know the level of detail/rigor at which I'm supposed to verify. Am I supposed to catch every mistake in the proof? If so, verifications should be worth more points.
Yes. They're an excellent idea. I liked them. ...although, they lowered my grade :(	I'd rather spend the time sleeping, but short of that, it's fine.
A little.	I suppose.
Mmm...only sort of. Even though we are given a solution, it can be confusing what we are expected to say for verifications.	?
Not really.	Yes.
No, by the time the homework has been submitted, I understand the solutions.	No. A better use of the time would be reviewing several different correct solutions, with common pitfalls highlighted.
A little.	Somewhat.
No.	No. Lessen the question so we can read more in detail.
Definitely.	It's helpful in theory, and in most cases. However, I often get proofs that are just awfully written, and it takes me a long time just to figure out what people are saying.
No.	I think they're a complete waste of time, and would rather have some easy/medium proofs added to the homework.
No - I usually gain more from reading the actual solutions than the verifications. Verifications are needlessly tedious, can confuse my understanding of the problem.	No - I often find myself just mindlessly looking at the options for mistakes/corrections. VERIFICATIONS DON'T HELP ME.
No.	Yes.
Not really	
Yes.	It is reasonable.
no	
Kind of	yes
Not really, I usually can understand the posted solutions.	I think that doing verifications for more varied problems would help more than doing a lot of verifications for 2 problems.
The verifications don't help understand the solutions and concepts because reading wrong/confusing answers doesn't help.	Yes, my time spent has a reasonable return.

Yes	No, I think the TAs should show the different correct ways to do the problems that are possible (they should grade the proofs first.)
Sometimes, but not when the proof I'm verifying is really vague or confusing.	For the most part yes, sometimes I need to spend more time than I should.
Sometimes, although it can be very difficult to understand other people's proofs/thought processes.	No, I just find that I spend a lot of time trying to understand student's reasoning processes sometimes to no avail, I feel like going over proofs in class would help me to a greater extent.
Depends on the problem (hard ones, yes, simple ones, not so much)	Not always. It is useful if the person I'm grading uses a different technique.
Sometimes	No, I would rather have less verifications, but I would like them to have very specific mistakes that are important for us to be able to identify rather than random problems by students.
To a point. However, doing 5 verifications for a problem is a bit excessive, because by the second or third one I completely understand the solutions. Additionally, it can take a long time to verify/find mistakes in nearly correct solutions, when often times those mistakes are hard to find just because people are unclear with their proofs. The 1st verifications were the best.	See `<--`
No.	I think we should have fewer verifications; it gets very tedious
To some degree. Not when the problems we are verifying are just brute force problems, though.	Suuure?
To some extent....not too helpful if someone else did it completely differently.	Yes, it's reasonable.
Yes.	It depends on the problem. Some proofs seem hard to verify.
No, totally not !	Kind of !
No.	I think they would be more helpful if we did them for problems to which our solns. had already been graded, b/c sometimes you don't realize something is a major error just b/c it's a little different in the solutions.
Sometimes but sometimes they feel kind of pointless	Sometimes–I think they would be better if we did fewer of them but looked at them more in depth.
Yes	Yes, however, some submissions are hard to understand sometimes. I don't think the grading reflects that. There also don't need to be so many, after the 2nd or so submission I understand the problem completely.

No. I find when I spent more time on them, I end up doing worse. I guess I understand the problems better, but I have a big problem with the stringent "fixed-ness" of the verifications.	I suppose I learn well. I feel they should be worth less or be graded easier.
Yes	If I understood the proof yes, otherwise I end up more confused.