Thoughts on the Computer Science Ph.D.
Steven P. Reiss

1. First Principles

Any reevaluation or assessment of our Ph.D. program should start by considering what its goals are and how we view the intent of the program. As I see the program, it has several goals (with differing priorities based on both the faculty member or the student):

- **Train professors:** One of the principle reasons for getting a Ph.D. is to become a faculty member. As such, one of the goals of the Ph.D. program should be to get our students ready for such a career.

- **Teach how to do research:** The Ph.D. degree is the certificate that says its holder is capable of doing independent and original research. Hence a goal of the Ph.D. program should be to ensure that the certificate means what it says.

- **Provide a technical foundation for doing research in a specific area:** When we produce a student who has specialized in an area such as software engineering, programming languages, natural language processing, combinatorial optimization, etc., we are saying that the student has a sufficient background for doing original work in that particular area.

- **Provide a foundation for lifelong research in Computer Science:** We want our students to be able not only to do research right away once they graduate, but also to be able to do research for the rest of their lives, i.e. to be able to adapt and extend their work as the field changes.

In addition, any changes to our program have to take into account both University regulations and practical considerations. The University requirements for a Ph.D. are minimal, essentially stating there has to be a point where the student is admitted to candidacy, they have to file a dissertation, and they have to complete (or at least pay for) 3 years of courses. The practical considerations are a bit trickier. We, and most other schools, have noted that only about half of those starting a Ph.D. program will actually finish the program with a Ph.D. for a wide variety of differing reasons. To best serve these students (and to husband our monies), it is best to identify students who will not succeed as early as possible, encouraging them to find other careers or career paths.

2. Objectives

Each of the goals cited above requires the program to address slightly different issues in different ways. We need to have an understanding of what the program’s objectives should be for each of the goals.

If we want to train professors, we first need to teach our students how to teach. This involves the ability to give lectures, advise students, organize courses, and prepare homeworks and exams. Next we should be training them to be able to work in a Computer Science (or related) department. This means that they should feel comfortable with all aspects of the field so that they can talk intelligently with other faculty and students, so that they can participate in intra- and interdisciplinary work, so they can teach courses as needed by their department, and so that they can participate in evaluating students, faculty candidates, and eventually other faculty members. Professors also should have an in-depth knowledge of a particular subdiscipline of the field, both to provide a basis for doing appropriate research and so that they can teach appropriate advanced courses.
This last objective brings us to what we should be doing to teach our students how to do research. This involves making sure that they can grasp the fundamentals of research, i.e. being able to find and absorb the body of related prior and current work for a field, understanding the tools and techniques that are appropriate, and being able to identify and define interesting, open, and potentially solvable problems. These particular objectives, when applied to a specific field or subfield of Computer Science, should also provide the student with the necessary basis for doing research in that subfield.

Finally, to ensure that our students are prepared to do research for an extended period, we should ensure that they have a deep and broad knowledge of the field. Have a good understanding of the basic tools (e.g. mathematics and programming skills) as well as exposure to other aspects of Computer Science is one way of doing this. Providing applications knowledge, such as exposure to nanotechnology, quantum physics, or biology, could also be considered a plus here.

From these discussions, we can draw up a list of objectives that we should be considering for our Ph.D. program. These would include:

- **Breadth in Computer Science.** Students should have an exposure to most if not all of the field at some level of detail.

- **Depth in a particular area.** Students leave as “experts” in their particular subarea.

- **Teaching and advising experience.** Students who want to become faculty members should have some experience in order to learn the skills that will be needed for teaching.

- **Knowledge of tool areas.** Students should be able to effectively use the underlying tools we use for research, e.g. mathematics in various forms & programming, to help solve their particular problems.

- **Knowledge of application areas.** Since Computer Science is becoming more interdisciplinary and much research is done with particular application domains in mind, having knowledge of some application areas is becoming more necessary.

- **Research experience.** Experience in successfully doing real research is essential to demonstrate that the student can do it. This is what the dissertation is all about.

Each of these objectives can be achieved in various ways. In considering these, we should think about whether our current Ph.D. program addresses them adequately or whether we should change our requirements appropriately.

### 3. Achieving Breadth

Achieving the objective of breadth has probably been the most controversial part of our Ph.D. program over the years. For a long time we had comprehensive exams where the student were expected to show their breadth by passing exams in six different areas. The exams happened to be tied pretty closely to undergraduate courses in those areas. In addition to putting a lot of pressure on the students, the existence of the exams tended to make students take our undergraduate courses even if they had a similar course before coming here, which we saw as a waste of their time and education.

To remedy both these problems, we shifted to a course-based breadth requirement. This shared the dual objectives of ensuring that the students had breadth and in getting them to take graduate level courses. The requirement, as eventually ended up, requires that students take an appropriate number of 100 and 200-level courses in n out of k areas. Students who wanted to take a 200 course were first required to demonstrate they had the knowledge embodied in the 100 level course in the area, which is the stated purpose of the entry interviews. Since the course require-
ment first started, we have introduced many new courses, new areas of the field have arisen, other
courses have disappeared, and in general the field has changed. There is also substantial confu-
sion with the areas since the areas involved in the Sc.M. and Ph.D. are different.

The additional exam requirement we had was the programming comps. We retained this
portion of the comprehensive exams because the courses that covered the expected level of exper-
tise were sub-100 level (i.e. cs15, cs31, and cs32) and hence could not be taken for graduate
credit.

The question we should be asking regarding the breadth requirement today is whether the
current requirements are appropriate and if not, how they should be changed. Simple changes
might involve rethinking the areas and the making the requirement more adaptable to future
changes in our courses or the field. We could go back to comprehensive exams, trying to ensure
that they cover general knowledge of a particular area rather than our specific courses. We could
move to oral exams in a specific number of areas. We could drop the breadth requirement alto-
gether, replacing it with a simple requirement that students must complete an approved program
of 8 courses in their first two years (5 of which are 200-level) that demonstrates breadth in Com-
puter Science.

The programming exam is another issue that has raised some controversy. Part of breadth
(and also part of understanding the tools of the field) involves being able to design and implement
programs. While the exam does put pressure on our students, it also serves a valid purpose that is
difficult to achieve with graduate-level courses. If we decide to eliminate it, which might not be
the right choice, we could replace it with an advanced course that involves significant program-
ing along with the requirement that the student present their design and program to a committee
of the faculty (as has been done recently with the programming exam). Alternatively, we could
keep the current exam and make it clear that its purpose is to ensure breadth, not eliminate stu-
dents. This would involve developing specific alternatives for failure, which could range from
taking CS32 or CS36 or an appropriate advance course, or undertaking a programming effort
within their research and presenting the corresponding program to a programming exam commit-
tee.

4. Achieving Depth

We currently have two requirements aimed at achieving depth. One is a course requirement of
a major sequence both involving two 200-level courses. The second is the thesis proposal in
which the student can demonstrate their knowledge of the subfield in which they are working.
The former doesn’t really ensure depth since the courses vary substantially and do not necessarily
cover a substantial body of past or current work. The latter often is done too late in the process
and is focused on the particular research of the student rather than overall knowledge in the field.

Because of this, it might make sense to add another requirement for the depth, a requirement
that would be both beneficial for the student and complimentary to what we currently have. This
might take the form, for example, of an area exam, either oral or written, in which the student is
asked to demonstrate their knowledge of prior work, appropriate tools and techniques, and other
knowledge in their particular area as a prelude to doing their dissertation. Such a requirement is
relatively common at other schools.

5. Teaching Experience

While one of our objectives is to train our students to teach, our current requirements do not
reflect this in any way. Student may TA, but generally here that involves mostly grading and pro-
viding help sessions to students, knowledge that is helpful but doesn’t really address being a good teacher. Note that this is due in part to the heavy emphasis on undergraduate education at Brown and in the fact that we are quite proud that our courses are taught by faculty members and not by graduate students.

To address this potential deficiency we could do any of several things. We could require students to TA. However, as noted, this might not really address the problem. We could add and require a ‘teaching’ seminar in which students were required to prepare and give a sequence of lectures, were graded on this, and were actively involved in helping and critiquing the others in the course. This could be done independently or in conjunction with some of our graduate courses. We could have a separate requirement that Ph.D. students must give at least 2 presentations (of 30 or 60 minutes each) during each semester either as part of a course, as a TA for a course, or in a seminar and then ensure we have an appropriate seminar series covering current research.

6. Knowledge of Tool and Application Areas

Our current requirements to achieve these objectives consist of an inside minor (2 courses), an outside minor (2 courses approved by the advisor), and, as previously mentioned, the programming exam. With the increasing importance of mathematics in all aspects of Computer Science and the increasing push for interdisciplinary research, one issue that comes up is whether these requirements are the most appropriate ones.

One alternative here would be a minor area exam (similar to the area exam, but not in as much detail or depth), either written or oral. This could also mean achieving a certain level of proficiency on the major exam in that area if we didn’t want to create multiple exams. Another alternative would be to require two outside minors, one in a mathematical area and one in an applications area. If this is too many outside courses, we could instead require 3 (or some number of) outside courses that are preapproved by the advisor (or some subcommittee) as part of a particular course of study.

7. Research Experience

Research experience in the Ph.D. program is reflected both in the research comps and in the dissertation. The research comps provide the means for the student to learn what research is all about and to demonstrate their talents for undertaking research at an early point in their tenure as a student. This is helpful both to the student and to the faculty in evaluating the student. The issues that we need to address here include what to do if the student has problems either finding or maintaining an advisor and developing a common standard for what it means to pass the exam. The latter we have started to do by insisting on committees for each student rather than just the advisor. The former we have been addressing on a case to case basis. One possibility is to create a rule that says that a student may, with the consultation and approval of the graduate advisor, change their research advisor the end of their third semester. If the case warrants, they will then be given an additional 3 months to complete their research requirement.

The thesis and thesis requirements are relatively straightforward. The only aspect that we might want to consider changing or formalizing is the point in the research where the proposal should be done. This becomes less important if there is an area exam then in the current case where the thesis proposal substitutes for an area exam.