1. First Principles

The current Brown Computer Science curriculum hasn’t changed significantly over the last
decade while the field itself has both expanded and changed direction. It is time to rethink our
current curriculum. We must redesign it to better fit the interests and demands of 21st century
Brown students and to reflect changes in the field.

The “New Curriculum” has been used successfully at Brown for 35 years. It has provided the
University with a curricular philosophy that makes Brown stand out and that has been quite suc-
cessful at attracting students and producing high-quality graduates. Consistency with the Brown
curricular philosophy should be central to the redesign of our departmental curriculum and con-
centration requirements.

There are several basic principles that underlie the Brown curriculum. First and foremost of
these is the principle of student choice and individuality. Students should be free to choose, with
appropriate guidance, the set of courses that best meets their needs. Students are assumed to have
an understanding (again with appropriate guidance) of what these needs are and how they can best
be fulfilled.

Central to this principle is the notion of appropriate guidance. This is assumed to be provided
by the wide range of advising that is offered to Brown students in the form of individual faculty
advisors, undergraduate advisors (Meiklejohns), advising fairs, departmental undergraduate
groups, etc. It is also seen in the various published example programs.

The new curriculum has taught us that while flexibility can be a bane, it can also be a blessing.
Brown tends to require the minimum from its students, but expect the maximum. While Brown
has no distribution requirements, it is still the case that a large fraction (90+%) of the student’s
programs would meet such requirements. At the same time, the flexibility both empowers the stu-
dents and lets them orient their program to their particular interests. Students are attracted to
Brown by this empowerment and flexibility; our student population largely consists of students
who are looking for such flexibility throughout their time here both in terms of basic requirements
and in terms of their concentration. We need to design our curriculum in the spirit of Brown and
its current students.

In addition, computer science is becoming more central to a wide range of disciplines, espe-
cially in the biological and social sciences. Understanding how to effectively use computers, their
limits and their strengths, is useful in fields ranging from biology to politics. One way of encour-
aging computer science at Brown is to make it more amenable to these new roles, providing our
students with the background needed to work in these various areas. This, however, requires flex-
ibility. What is needed in one discipline can be very different from what is stressed in another.
Flexibility, with appropriate guidance, will let our students best address these needs and will
encourage more students to enter our field. Moreover, even within our discipline, new subfields
and applications are emerging. We need to prepare our students to handle these new areas as well.

2. Outside Factors

Any Brown concentration is subject to the restraints imposed by the University. Moreover, we
need to consider how any revised concentration fits into changes that are being made at Brown
and to the way the field itself is changing.
The limitations imposed by Brown are written in the faculty rules. An AB degree can only require 10 courses, although there can be a “reasonable” number of prerequisites which are not included in this number. An ScB degree can require 20 courses, only 10 of which can be in a single department. Again, there can be additional prerequisites.

The Brown curriculum is constantly changing. Currently, the University is putting an emphasis on freshman seminars. The administration wants these to be integrated into departmental curricula. Once this is done, there will be a push to consider sophomore seminars. We need to consider how these potential changes will fit into our curriculum as we revise it, asking such questions as can and should we make use of these new course types as part of our core concentration.

At the same time, our field itself is changing. It continues to expand, adding more topics and requiring more knowledge all the time. New areas such as quantum computing, web development, or nanoscale technologies are going to be of interest to students. There are always new and broader applications. We need to consider how do we design our curriculum so that students can learn of these new areas in addition to obtaining the necessary CS background to work in these and other areas.

3. Our Goals

The first step in designing a curriculum that addresses first principles and fits into Brown should be understanding what the goals should be. We need to do this separately for our two degree programs, the AB and the ScB, since they tend to address different needs and different classes of students.

The ScB should be a strong computer science degree. This is the program that students with a serious interest in computer science will probably be taking and the degree that is designed for those looking for computer science-specific careers. The primary goals here should be:

1. Prepare students for a career in computer science. Our ScB graduates should have enough of a background to pursue serious computer science related jobs or to undertake top-level graduate study in the field.
2. Address the needs of working computer scientists. ScB graduates should have a broad and deep background that will enable them to work throughout the field.
3. Provide a foundation for future learning in the field, both through graduate school and through the ability to learn on ones own. While this goal is shared with the AB program, here we should be aiming more toward students who will be working purely within CS rather than using CS in other disciplines.

The basic AB should be our degree for students interested in applying computer science to other disciplines or for those interested in getting a liberal arts education that includes a computer science component. Students with a deeper interest in computer science should be advised to either take the ScB or to take additional courses beyond the basic AB. The primary goals for the AB degree should be:

1. Prepare students for future learning in Computer Science. We cannot hope to provide our students with all the knowledge they need. Instead, we should provide them with a foundation that will enable them to become life-long learners in the field.
2. Prepare students to use computer science in a particular discipline. The discipline here might be either inside or separate from what we consider the core areas of CS (e.g. natural language recognition or fluid mechanics).
3. Provide an appropriate concentration that fits into Brown’s notion of a liberal education. We should have a degree program that fits with the Brown curriculum and that more Brown students are comfortable taking.

4. Attract more students to computer science. It is important both for the department and for the University overall (i.e. other disciplines) that more students have an understanding of computer science.

These various goals for the concentration programs can be met by choosing the appropriate specific objectives that the programs need to achieve. These can be divided into four categories: fundamentals, breadth, depth, and experience.

4. Fundamentals

We expect all our students to understand the fundamentals of Computer Science. This is the knowledge that is essential for undertaking any work in the field and constitutes the basic prerequisites that any more advanced CS course would assume. In order to provide flexibility, to make room for the other requirements, and to make it easier to take advanced courses, what is considered fundamental should be kept to a minimum. In particular, there are really only two basic notions that we rely upon here: basic programming skills and mathematics. The ability to control a computer is central both to practical and theoretical computer science. You really can’t understand anything in the field until you understand what programming is about and have a basic notion of how it is done. Similarly, some fundamental notions of mathematics including proofs, mathematical modeling, and complexity, are used throughout all CS courses and need to be understood by all of our students. These topics should be covered through appropriate introductory courses that are required of all majors, both AB and ScB.

Several current and proposed alternatives exist for covering basic programming concepts and skills. These include:

- A strong, traditional computer science introductory programming sequence as offered in CS15 and CS16.
- An alternative introductory sequence that is a bit more mathematical and covers other programming models offered in CS17 and CS18.
- A one-semester course (CS19?) that combines the topics of CS15 and CS16 for those students who have a prior programming background either from high school or from other courses such as CS4.
- Freshman seminars that introduce basic programming from a particular point of view or by addressing a particular problem, possibly combined with CS16 or CS19.

For foundational mathematics, other alternatives exist:

- Calculus (Math 9,10) combined with a traditional discrete mathematics course such as CS22. For the AB degree, we might even want to relax this to just be CS22 or replace the calculus requirement with a more general science/math requirement of a 2 course sequence in math, applied math, or a science.
- An appropriate high-level mathematics course that involves proofs but covers other mathematical topics that are relevant to computer science such as linear algebra, abstract algebra, or statistics. (Note that this assumes the appropriate prerequisites which might depend on the course.)

In considering our concentrations, we need to include courses that cover these foundations. In doing so we should attempt to attract as many students to the field as possible and to provide the students with the knowledge and insights needed to take more advanced courses. These courses have to provide a solid base for students with a known interest in computer science and at the
same time attempt to attract students who have not been exposed to the field in high school and show them what computer science is all about. These possibly conflicting goals imply that we want to be flexible here, offering a variety of alternatives each of which will be attractive to some population of students. It might also mean that we want to do more to emphasize that computer science is not just is fundamentals, for example seminars that introduces advanced topics at a freshman level or sophomore seminars that address different domains using computer science techniques.

5. Breadth

Breadth is a way of ensuring that students are exposed to a variety of areas of computer science. Going back to first principles, the purpose of breadth should be to provide what is needed to prepare the students for future learning, for understanding how to use CS in a particular discipline (which might be CS itself) and within the bounds of what is appropriate at Brown. There are a lot of topics that are classically considered computer science including hardware, software, artificial intelligence, information technology, graphics, algorithms, and theory of computation. Applying CS to a particular discipline typically requires a subset of these. One needs to consider what are needs for students interested in computational biology, natural language processing, software engineering, mathematics, linguistics, or scientific computing and how these needs differ by discipline. While many computer science topics are going to be widely applicable, any particular application or discipline (outside of computer science itself) should probably concentrate on only a subset of them. The objective then in conveying breadth should be to ensure that students have covered enough of the topics of CS so that they can understand and apply the methods of our field to their particular discipline. In doing this, we should follow the Brown philosophy in using advising and example curricula to most appropriately direct students.

At a minimum, breadth in computer science should expose students to the way that computer science works. This is two fold. First, it should show the students how systems are built. This can be hardware systems, software systems (applications or systems programming), software engineering, operating systems, database system, etc. Second, it should show the students how foundational or theoretical topics are created and used. This would include theory of computation, algorithms, logic, and the mathematical fundamentals of AI.

We currently have a number of courses that are above the introductory sequence and still not particularly specialized. These courses tend to offer students breadth (but not depth) in a particular area. For example, CS31 offers breadth in computer systems, CS32 offers breadth in software development and engineering, and CS51 has offered breadth in the theory of computation. Other proposed courses at this level include CS41, introducing the fundamental mathematics and principles of AI, an expanded CS34 providing systems and low-level programming background, and an algorithms course.

Attempting to balance minimum requirements (and hence more flexibility) with the overall desire for breadth, we propose that we ensure that each student get breadth courses on both systems building and foundations. For the AB degree, this would mean:

• Students would be required to take at least one systems course from (CS31,CS32,CS34,...) and one foundations course (CS41,CS51,...). Students would be advised to take additional courses from this set based on their areas of interest.

The ScB degree is designed more for students who are interested in computer science itself. In order to have a strong understanding of the field, most such students should be at least familiar
with all the areas. Given that there is some overlap among the second-tier courses, we could reason-
ably specify for the ScB that:

• Students would be required to take at least two systems courses from (CS31,CS32,CS34,...) and two
foundations courses from (CS41,CS51,...). Again additional courses might be recommended based on
their interests.

We note that the courses at this level will most likely be prerequisites for more advanced
(100-level) courses and thus students would need to take them based on the 100 level courses they
would want to take. Predicting this and appropriately steering the students based on their interests
is one of the roles we should do through advising.

An interesting and viable alternative here would be to develop a series of sophomore-level
seminars each of which provides breadth in either systems or foundations through a focus on a
particular topic or application (similar to the BIO 19 model). These seminars might be of more
interest to some students than the more general, lecture-oriented courses we currently have. More-
over, they could easily fit into a breadth requirement as stated above (although we would then
have to worry about prerequisites for advanced courses). Accepting flexibility in the curriculum
will let us experiment with such courses at this level.

6. Depth

Depth involves ensuring that students explore some area of computer science in detail. This is
needed both as a part of a successful liberal education and to ensure that students actually under-
stand what computer science is all about beneath the surface. Depth again prepares students for
future learning by showing them how to study things in more detail. It provides the actual details
from the field that will be needed to apply CS to a particular discipline. Depth involves having the
students understand some problem or area of computer science in enough detail to see how the
field works, how different areas and techniques interact, how mathematics is combined with pro-
gramming and the various basic areas of computer science to yield useful results. What is a rele-
vant problem or area here however is going to change with each student depending on the
student’s current and future interests. However, we want to ensure that in studying the problem,
students achieve a solid understanding of basic principles that can be transferred to other areas
and applications of computer science.

While some of this will come from second-tier courses, it really can’t be appreciated until the
student considers topics in more detail and from different perspectives. This is typically done in
advanced undergraduate (100-level) and graduate (200-level) courses. What we propose here is to
define course pairs. These pairs would be either two 100-level courses or a 100-level course and a
200-level follow-on that share a common theme or problem but view that problem from different
perspectives. They should also be chosen so that together they provide a basic understanding of
how computer science is done.

Rather than attempt to predefine such sequences, we note that just requiring approved course
pairs as part of the concentration is probably sufficient. We need to retain a degree of flexibility
here. First, our course offerings at this level are changing, with new courses being offered each
year and with some old ones being phased out or offered in alternate years. Second, this require-
ment should allow for integrating appropriate research or group research courses and senior semi-
nars into the concentration requirements. Third, as our field becomes more interdisciplinary, it
might make sense to have some of these pairing reflect the use of computer science in other areas.
Finally, because we can’t anticipate all the differing interests of our students, we should allow stu-
Students to propose course pairs that might make sense to their particular interests and consider those for approval.

For the AB degree, depth should focus on the student’s interest, whether that is completely in computer science or in the application of computer science to another discipline. To achieve this within the confines of the AB limitations, we would require:

- Students complete one approved pair of courses.

The ScB degree is focused more on computer science itself. Here it is important that the students get depth as well as breadth and cover a broader range of topics. One way of achieving this would be to require:

- Students complete two approved pairs of courses. (We may or may not want to allow overlap within these two pairs of courses. Moreover, we might want to ensure that one or both of these pairs is exclusively within computer science.)

Note that this is quite different to our current ScB degree which essentially has a 100-level breadth requirement and no depth requirement.

7. Experience

Providing students with experience in using and understanding the tools and techniques of computer science should be the final specific objective for both concentrations. Students need experience in order to be prepared to use computer science for their careers, in their particular disciplines, and for their future education. Experience can be provided through additional courses or through a capstone experience.

The simplest approach here is to require additional elective courses. These should be chosen by the student, with help from an advisor, to provide the student with the additional breadth, tools, or depth that they will need in the particular area or areas that they are interested in. These would typically be an appropriate mixture of 100, 200 and sub-100 level courses that were not counted as part of other requirements.

For the AB concentration, we would recommend:

- Students complete an approved program that includes 8 computer science or related courses (not including CS1, CS4, ...).

If we assume that three courses are taken as part of foundations, two as part of breadth, two as part of depth, this would leave the students with one elective. We should provide the majors and potential majors with guidance both through direct advising and published example degree programs. We would expect that this course would often be an additional sub-100 level breadth-type courses and would advise most students accordingly. Moreover, we will probably be advising students to take additional courses based on their particular interests.

For the ScB, it is important to ensure that the students have design experience, either in terms of building a system or in terms of mathematical investigations. This can be achieved through an appropriate capstone experience either in the form of a project course such as CS190, an independent study or research course, or even senior seminars on particular topics as is done in other departments.

The particular proposal we recommend for the ScB concentration would be:

- Students complete an approved program that includes 13 courses in computer science or related areas. One of these courses must reflect a design project or capstone experience.
8. Summary

For illustrative purposes, the complete requirements for the AB degree (assuming only our current courses) might then be:

- Prerequisite: Calculus, Writing course
- CS15, CS16 or CS17, CS18
- CS22
- One of CS31, CS32
- One of CS51, CS141 (as CS41)
- An approved pair of courses in computer science or computer science and an application area
- One additional approved course in computer science (numbered 30 or above) or a related area (numbered 100 or above).

A sample complete requirements for the ScB degree might then be:

- Prerequisite: Calculus, Writing course, Outside science sequence
- CS15, CS16 or CS17, CS18
- CS22
- Two of CS31, CS32
- Two of CS51, CS141 (as CS41)
- Two disjoint approved pairs of courses, one of which is entirely in computer science.
- A design course, either CS190 or an independent study
- One additional approved courses in computer science (numbered 30 or above) or a related area (numbered 100 or above). If the design course is part of the pairs, then an additional elective is required.

9. Conclusion

The proposals in this document represent a more flexible approach to computer science that will let us adapt our curriculum to a broader range of students and the changing needs of the field and the university population for the next ten years. They address the needs of the field as well as the requirements and changing curriculum of Brown.