Compilers (Spring 2020 202030)

Instructor: **Lerner, Benjamin**	Enrollment: **27**
Subject: **CS**	Responses Incl Declines: **19**
Catalog & Section: **6410 01**	Declines: **0**
Course ID: **33140**	
Objectives:	

Course Related Questions (16 comments)

Q: Please comment on the strength and/or weakness of the required textbook/course materials.

1 Absolutely incredible lecture notes and assignment specifications -- anything not clearly stated or explicitly clarified in the notes was fixed as quickly as possible

2 Lecture notes are super helpful. Textbook isn't really required.

3 The lecture notes were great!

If they could've always been posted before the lecture began, that would've been even better. But with the class being converted from 32 to 64 bit, plus the covid craziness, it's understandable that they were not.

4 The assignments along with in-class instruction were very descriptive and it never felt like we did not possess enough information to complete the assignments.

5 The textbook was mostly irrelevant for me because the lectures were supplemental in their entirety minus garbage collection. Plus wrapping my head around lectures mostly did not leave me time to go looking in the textbook. However, once I understood the lectures I had the tools to be successful in the course.

6 The textbooks for this course were largely optional as we mostly used the lecture notes, which were a great resource and helped me see the application of the things we talked about in lecture. That said, I think the starter code for assignments could be improved a bit as towards the end of the semester I felt I was spending a significant amount of my assignment working time just porting my code to work with the new starter code. That was tedious, but things got interesting and fun again once we moved onto the actual meat of the assignment.

7 The lecture notes were incredibly useful throughout the course. Not having notes on garbage collection made the final assignment unnecessarily more difficult.

8 No required materials. The lecture notes were in transition this year between 32-bit and 64-bit but were mostly free of inconsistencies. They were very helpful for completing the homework assignments.

9 No textbook required, all course notes are free online.

10 The lecture notes were very thorough and extremely useful in understanding course materials. They were very critical supplements to the lectures and assignments.

11 Starter code can be improved a lot. Though, this is the first year using amd64 so there are bound to be issues.

12 We use OCaml, NASM, and CLANG for our compilers. They work well and are well-suited for the course

13 online lecture notes, although did not always track 1:1 with lecture, were helpful! I would say the most important part of the online lecture that were missing was GC. Further sometimes with online lecture notes they do not always explain `magic` numbers used in code examples.

14 There was not a required textbook for this course to learn from. However, lecture notes were written online for each lecture, which were very helpful references. My only criticism of these notes is that notes related to the final homework assignment were never posted, and the assignment was based on one unrecorded lecture about a month before. It would have been helpful to have reminder notes for that assignment.

15 There isn't really a textbook for this course, but Professor Lerner's lecture notes are second to none. Any material that would be useful to better grasp the content of the course is readily available.

16 This class takes a lot of time, but it's definitely manageable. For example, my partner refused to do more than 5 hours of work over the entire duration of the semester, but I only needed to take 2/4 late days.

Learning Related Questions (15 comments)

Q: Please comment on the strengths of this course and/or ways to improve this course.

1 If you are at Northeastern studying CS, you should take this course. You will end up learning so much, and it will bring together what you learned in Fundies and OOD. There is no playing around though. It's super intense, and it really pushes you to learn as much as possible. I am so ready to build my next compiler:)

Ways for improvement:

1. It would be great if we spent more time on type inference. I didn't fully understand it in lecture, so I looked into the topic on my own and it is eye-opening.

2. Allow students to build on top of their previous work, instead of giving starter code for every assignment. I think only the parser config and changes to AST need to be provided. I would've preferred to fill in other changes to the existing code base on my own.

2 I wish there was a little more showing of the "correct answers" after each assignment. I get that it's supposed to be an iterative project, but there were things my partner and I were doing wrong every single week, and ways of implementing features we didn't know about, because we never saw our mis-guided coding attempts corrected.

3 I feel like a lot of time was spent on edgy constructs like tail calls when (to me) it would have been more interesting to spend more time on typechecking and building record types (not lazy record types that are syntax sugar for the heap tuples we did, but like actual flat records).

4 This course teaches you how a program really interacts with its runtime environment, and how you create functions down to assembly. One thing I would have liked to do as an undergrad, was type inference and tail-recursive functions. There was not enough time in 4 weeks to learn all that was necessary get that two pieces of material in as well. However, those two concepts I thought were really relevant and wish I could have spent more time on it during class. In general, this class is pretty much perfect, it is easy to succeed and learn if you just pay attention and put in effort.

5 The course content was great. Tied together a lot of things I learned in other classes into one project.

6 This was a very interesting and rewarding course, I especially liked the "middle-out" approach of having a working compiler after each assignment. On the other hand, there were several aspects of the assignments that caused a lot of frustration. Requiring old tests to be ported led to a fair amount of busy work, especially when there were subtle changes to the starter code which essentially required us to rewrite all our tests to pass, defeating the purpose of porting them in the first place. Some of these changes had impacts on porting existing functions in the pipeline as well. For example, changing our environment variable to be a map from a list in the final assignment. This added a fair amount of seemingly unnecessary busy work. The final assignment also had very little instructions and description compared to previous assignments, and since there were no corresponding lecture notes, a large amount of time was spent inferring what we had to do based on the starter code, which had many lines confusingly commented out without documentation,

Q: Please comment on the strengths of this course and/or ways to improve this course.

7 In this compilers course, you start with a working compiler for a very simple two-operation programming language, and extend the language week by week while also introducing new functionality to the compiler internals as it becomes relevant (A-normal form conversion, typechecking, desugaring, etc). The lectures and material build off of each other in a nice progression, and I felt that I was always given the tools I needed to understand the topics presented. It feels really cool at the end when you've developed a compiler for a language that is powerful enough to write programs that have bugs (and it also supports debugging with print statements).

There's a lot of starter code to read, especially towards the end of the semester. Sometimes the starter code you're given implements functionality that was part of the requirements in the previous week, and sometimes it's more difficult than it needs to be to make it work with the rest of your implementation and the unit tests you wrote.

This is a class where it is worth starting the work early and working on it a bit every day. There are some really tricky concepts, like implementing tail recursion support and typechecking, that you need to just sit and struggle with for a while before they really make sense.

8 Well-designed curriculum. Ben Lerner says he'd like to teach a Compilers 2 course, which I'd very much support.

9 A major strength of the course was how closely tied the assignments were to the course material. Assignments followed the progression of the course and built upon old and new material.

10 I think the schedule at the beginning can be squished a little bit to redistribute the time to the later assignments. There were a few assignments where we started it the day after its release and worked on it daily for at minimum 2 hours and still had to take a late day.

Actually this is the only course I've ever had to take late days on. It's a hard course and I think it should remain that way, but I think the initial assignments could probably be compressed into 3-4 day time spans.

While it is the year of the pandemic, I wish we would get our grades quicker, especially if we are subject to regression tests. It's not very fair to not know what you missed and worry about getting those points deducted multiple times.

11 Strengths:
- Assignments were difficult but rewarding. End course with a compiler that has type checking, lambdas, and garbage collection which feels good.
- Reinforced what I'd learned in PL, except this time with assembly.
- Each assignment ends with a working compiler. Iteration vs. building a compiler in steps felt better for learning.

Weaknesses:
- Each assignment had extensive starter code. So, we had to port over code from the previous assignment into the new starter code. There were always several tiny changes to things like function signatures which made porting over a painful experience. It wasn't particularly fun to port things over and we didn't learn much while doing it, but it still ate up a lot of time.
- Felt like we were always a few assignments ahead of our last graded assignment.

12 The homeworks and lectures were amazing. The lectures taught us exactly what we needed to know and the homeworks reinforced what we learned and let us put it into action. I greatly enjoyed the course content and how it was presented. We spent the perfect amount of time on everything and covered a lot, despite frequently going on some interesting tangents. I think the homework was a good level of difficulty. It was always challenging, but never undoable.

There are a few ways I would suggest to improve the course. A few homeworks had buggy/incomplete starter code, errors in the description, or other problems. These problems were fixed in the first few days of the assignments, but it was annoying to have to frequently check for new commits in the starter code repo and merge them. Also, there was no easy way for us to detect changes in the assignment description. Some kind of change history of the assignment page would be helpful. I understand there were a lot of difficulties switching from 32-bit to 64-bit and there were some new assignments created that didn't exist in previous semesters, but there should be some more polish before publishing assignments.
One problem I frequently ran into was the issue getting a fresh starter code repo every assignment. If we decided to change the expr type, for example, we'd have to make those exact changes to all of the starter code every single assignment. This led us to work around that in a way that felt limiting. We said to ourselves "if we do that, we're going to have to do all those changes to the starter code every time, let's do something else." I don't know how to fix that, but it is a problem that the way we get starter code affects what we consider to be a viable/maintainable design.
One of the things I was most fascinated by was type inference. However, I was not able to do type inference since it took my partner and I long enough to just get type checking working. After that, we were stuck with just type checking for the rest of the semester since we never had time to add type inference. This made testing new features' interactions with the type system very annoying because of all the annotations that are necessary. Additionally, we'd have to change the concrete syntax of the language to allow type checking of lambda expressions and arbitrary-expression applications. I think it would be a good idea to have something in the middle of checking and inference like a bidirectional type system and have that be mandatory for both levels. That way, adding and testing new features in the type system would be much easier and all students would get to implement something more advanced than just type checking.

13 Overall this course was a very interesting introduction to compilers. I learned a lot in it, and I appreciated the design approach that the course took, rather than just throwing a bunch of miscellaneous compiler concepts at us. The capstone, cumulative nature of this course made later homeworks quite difficult, but it really was the most effective way to learn the concepts in a robust way.

Fortunately, moving online worked fairly well for this course. While asking questions and creating discussions were naturally more difficult, my ability to learn the content was mostly unimpacted. My only suggestion for this course would be to fully address the issues with regard to moving to 64-bit compilers before teaching the next course. Some assignments were made unnecessarily difficult by issues in the starter code that were fixed through pull requests days after the assignment started.

14 This is a fantastic course. It is one of my favorite courses that I have ever taken. Perfect for anyone who is deeply curious about the core concepts of computer science and how they connect: it is a commitment but beyond worth it.

15 This class was a fantastic experience. It was a great way to tie everything I've learned at NEU back together in a big way.

Instructor Related Questions: Benjamin Lerner (17 comments)

Q: Describe instructor's strengths, areas for improvement, and any additional comments.

1 Incredible professor. Lerner always goes out of his way to help other students understand the material and is always passionate about the material he covers.

2 Never regretted taking his class. One of the best professors I've had at Northeastern. He is humorous, helpful, knowledgeable, and articulate. There is not much to say. Professor Lerner plus a compiler course. It just can't get any better.

3 Best instructor I've had at Northeastern, by far. He was completely devoted to each and every student and their learning, and has an almost spookily-accurate ability to immediately identify the root cause of any student misconceptions and problems. Thank you, professor Lerner

4 Lerner is a really good professor, he communicates the course material very effectively. He clearly really enjoys teaching and it's a very fun class to take.

I feel like he spent a lot of time in class proposing a question, getting a lot of bad answers from students that didn't know what they were talking about, then entertaining those answers for far too long instead of just saying "that's a bad solution because ___, here's the actual correct solution".

I would have enjoyed using Racket rather than OCaml though. I'm not sure if that's because of the style guide that was suggested or not (I don't know OCaml very well) but OCaml feels like syntax soup and Racket (not fundies Racket, real Racket) is much easier to read, for me at least.

5 Prof. Lerner is the best computer science professor I have had at NEU. The best part about him is that he genuinely tries to empathize with students and puts them in the students shoes. This makes it easy to talk to him about course material, and overall just enjoyable to work with him. What's more, he has an incredibly deep knowledge bank; it was valuable to pick at his brain.

6 I've never actually had a class with Professor Lerner before, which is surprising given that he frequently teaches a lot of the introductory level courses, but now I wish I had. He's a very engaging lecturer and pushed us to explore the course material without feeling overbearing. Definitely a candidate for best professor I've had in my years at NU .

Q: Describe instructor's strengths, areas for improvement, and any additional comments.

7	Lerner is a fantastic professor. His lectures are very engaging and informative, and he puts a lot of effort making sure students have what they need to get the most out of the course and succeed, especially when working with him one-on-one.
8	As most of the other reviews probably state, Ben Lerner is a fantastic lecturer and professor. He can go a little fast during lectures sometimes, but is always eager to answer questions and take the time to re-explain examples and ideas that weren't clear. He is responsive on Piazza, Teams, email, and in office hours. He even took the time to meet with my team outside of class to explain a concept that had gone over our heads. Compilers are a topic that he clearly has a deep passion for, and that passion comes through in class and in how he designs and runs this course.
9	Since the class is without a textbook, I was regularly dependent on the lecture notes. These notes unfortunately weren't always posted at the time of lectures which made it hard for me to follow lectures often (notes tend to make it easier for me to follow a lecture where my mind can easily drift).
10	Ben Lerner always does an amazing job teaching his classes. Although needing to work remotely is always an extra challenge, his lectures were still engaging and interesing.
11	The instructor's enthusiasm and anecdotes related to course material made lectures very enjoyable and offered a greater perspective than what was described on the syllabus. Instructor was also very responsive to students' questions and conveyed the course material clearly and thoroughly in both lectures and online notes.
12	<3 blerner
13	Strengths: - Engaging lectures. - Lot of personal experience building compilers and he has plenty of battle scars that inform his teaching. - Lecture notes are extensive and easy to understand. - Answers questions online extremely quickly Weaknesses: - This semester there was always a bit of confusion between 32-bit and 64-bit specifics in lectures and posted lecture notes. This specific iteration of the course there was a transition from 32 to 64-bit.
14	Professor Lerner is the best teacher I've had at Northeastern University. He explains things very well and is very interactive with students. He has a very good understanding of how a student thinks and learns. The way he teaches, he often guides you down a path of naturally discovering something, rather than explaining it directly. This is a great way to instill intuition and make an idea unforgettable. There were certain concepts in this course that I thought were totally beyond me, but professor Lerner taught them so well and they felt so accessible. When I ask him a question, he always knows exactly what I mean. And if my question doesn't make sense because it's founded on some misconception, he spots it and corrects it, rather than saying "that question makes no sense." I feel very comfortable asking questions in his class. He encourages good questions and they often lead to very informative discussions. Professor Lerner is also very enthusiastic and passionate about the material, which makes his lectures more engaging.
15	This is clearly a course that Professor Lerner loves teaching, and that enthusiasm that he showed made the course very interesting to attend each day. I don't really have any criticisms of him as a professor, thank you for teaching all of us a very interesting topic, and going above and beyond in teaching it.
16	Professor Lerner is one of the best professors I have had in my undergraduate education. He demonstrates an infectious passion for every course he teaches, and it is clear that he deeply cares about his students' learning. I have never been bored in lecture, and always look forward to class with him. Cannot recommend him highly enough.
17	Honestly, couldn't ask for a better Prof than Lerner. If I could rate him higher in these categories, I would.