Ling209A: Computational Linguistics 1
Spring 2019

Instructor  
Tim Hunter  
3103K Campbell Hall  
timhunter@ucla.edu

Classes  
Mon & Wed, 12:00pm – 1:50pm, Bunche 1265

Office Hours  
Tue, 10:00am – 12:00pm; or by appointment

Prerequisites  
This course is open to linguistics graduate students, and others with permission of the instructor. Permission of the instructor is unlikely to be granted to students who have not completed at least two of Ling200A, Ling200B and Ling200C.

Course overview

The aim of this course is to provide a deeper understanding of the formal machinery that is standardly put to use in linguistic theory — essentially, grammars — in order to improve students’ ability to assess, reason about and develop theoretical proposals. Building from the ground up, we will work from the simplest imaginable finite systems that express unboundedness, up towards systems that resemble the tools of contemporary linguistic theory. We will emphasize the commonalities that the familiar, more powerful systems share with their simpler counterparts, and the useful “reasoning patterns” that can therefore be applied to them.

The course is intended to be suitable and useful for open-minded students who don’t necessarily have any background in computational work and/or who don’t consider themselves to be primarily interested in computational linguistics as a research area. The aim is to provide you with concepts and skills that will benefit your studies of linguistics in general.

A few details on what to expect:

- Some technical concepts that will come up a lot include:
  - recursion
  - compositional interpretation
– type theory as an organizing perspective on how information is processed
– abstraction, or insensitivity to certain distinctions, as the basis of generalization

We will work with a functional programming language (specifically, Haskell), since this is a style of programming that naturally highlights many of these same ideas.

• Some of the linguistic questions that we will gain insight into include:
  – How can a finite device characterize an infinite array of linguistic expressions?
  – How does recursion relate to compositionality?
  – How do probabilistic models relate to grammars?
  – How do the tree structures we see in syntax and semantics relate to the linear structures we see in phonology?
  – Why are lambdas often used to compose meanings, but less often used to compose sounds?
  – What is meant by “interpretation at the interfaces”?
  – How do linguistic observations — both “traditional” and “psycholinguistic” — bear on hypotheses about mental representations?

In many cases, rather than trying to provide specific answers to these questions, the goal will be to make them appear less mysterious than they otherwise might.

Schedule

This schedule, particularly the latter parts, is somewhat tentative and may be adjusted according to the progress of class discussions and student interests.

<table>
<thead>
<tr>
<th>Week of</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Apr. 1</td>
<td>Core functional programming: Types, terms and evaluation</td>
</tr>
<tr>
<td>2 Apr. 8</td>
<td>Introducing recursion</td>
</tr>
<tr>
<td>3 Apr. 15</td>
<td>Finite-state string automata (FSAs)</td>
</tr>
<tr>
<td>4 Apr. 22</td>
<td>Variants of FSAs: strictly local, probabilistic, transductions</td>
</tr>
<tr>
<td>5 Apr. 29</td>
<td>Context-free grammars (CFGs)</td>
</tr>
<tr>
<td>6 May. 6</td>
<td>Stack-based CFG parsing</td>
</tr>
<tr>
<td>7 May. 13</td>
<td>Finite-state tree automata (FSTAs)</td>
</tr>
<tr>
<td>8 May. 20</td>
<td>Long-distance word-order dependencies: “discontinuous constituents”</td>
</tr>
<tr>
<td>9 May. 27</td>
<td>Long-distance meaning dependencies: scope-taking</td>
</tr>
<tr>
<td>10 Jun. 3</td>
<td>Spillover / wrap-up</td>
</tr>
</tbody>
</table>
Textbooks

There is no required textbook for this course.

Readings, assignments and notes will be distributed through the course website on CCLE.

Software

Most of the homework exercises will involve writing or modifying small programs. The programming language we will use is Haskell. You will need access to a computer with a text editor and the Haskell Platform.

You can download and install the Haskell Platform here: https://www.haskell.org/platform/

On the role of Haskell and programming

Our use of Haskell in this course is a means to an end: it’s a tool for helping to sharpen our understanding of certain concepts, and those concepts, rather than Haskell itself, are the real focus of the course. Completing this course will not necessarily make you a proficient Haskell programmer, although you’ll probably be off to a good — but “non-standard” — start towards that goal.

Because of the secondary role of Haskell itself, I do not recommend consulting programming textbooks or tutorials that aim to teach you the language, at least until after the first few weeks, and my aim is to provide enough information for it not to be necessary at all. Haskell is a very “big” language and some of these other sources will begin with things that will not be relevant for us.

If and when you really want one, a good textbook for many purposes is Programming in Haskell by Graham Hutton (Cambridge University Press). The best way to use this would be as an occasional reference throughout the second half of the course, rather than as an initial learning aid.

Requirements and Grading

- **Homework assignments (70%).** There will be roughly one per week, usually distributed soon after Wednesday’s class and due the following Wednesday. In addition to grading and returning these promptly, I may decide to require individual meetings with students soon after the due dates of the first two or three assignments to discuss your work — this is to ensure that everyone gets up and running properly.

Late assignments will not generally receive course credit (except in cases of legitimate disruptions, e.g. verified illness, jury duty, bereavement, religious observances). If you fall behind in this course, it may be very difficult to catch up!
Discussing the homework assignments with classmates will often be helpful, and is encouraged, but the work you turn in must be reflective of your own understanding, not something copied from someone else. To be concrete, your discussions should abide by (both the letter and spirit of) the “whiteboard policy”: you may work together on a whiteboard and discuss things for as long as you wish and in as much detail as you wish, but then erase the whiteboard and do not take any written notes away from this discussion. The idea is that being able to write up your solution individually establishes that you understand what you submit.

- **Class participation (15%).** This means coming to class having thought about any relevant readings and homework assignments, asking questions when something is unclear, engaging with other students’ questions and comments, and actively looking for connections to your own personal topics of interest.

- **Final mini-project or paper (15%).** This is pretty flexible and open-ended. You’re encouraged to try to find some small piece of work to set yourself, related to a research topic that interests you, that (broadly speaking) draws on the kinds of tools and techniques that are being introduced in this course. If you’re unsure, a good place to start is to think about extending one of the homework assignments; I’ll try to make a few suggestions about these possibilities as we go. The tentative due date for this is the end of the exam week, i.e. Friday 6/14/2019.

<table>
<thead>
<tr>
<th>Letter grades</th>
</tr>
</thead>
</table>