Unit 0

Syllabus

Course: Computational Linguistics 2	**Name:** Thomas Graf
Course#: LIN 637	**Email:** lin637@thomasgraf.net
Time: MW 2:30–3:50?	**Office hours:** tba
Location: CompLab SBS N250	**Office:** SBS N249
Course Website: lin637.thomasgraf.net	**Personal Website:** thomasgraf.net

1 Course Outline

1.1 Bulletin Description

An introduction to the theoretical foundation of computational linguistics. The course emphasizes the importance of algorithms, algebra, logic, and formal language theory in the development of new tools and software applications. Empirical phenomena in phonology and syntax are sampled from a variety of languages to motivate and illustrate the use of concepts such as strictly local string languages, tree transducers, and semirings. Students will develop familiarity with the literature and tools of the field.

1.2 Full Description

This course serves a specific purpose in our program (see Fig. 1 on page III): it acts as the bridge from introductory courses in linguistics (Syntax 1, Phonology 1, Phonetics) and computational methods (Statistics, Mathematical Methods in Linguistics, Computational Linguistics 1) to advanced courses and seminars in computational/mathematical linguistics. In contrast to the NLP courses offered by the department of computer science, this course focuses on studying the properties of natural language from a computationally informed perspective. The question is not how computers can solve language-related tasks, but how language can be conceptualized as a computational problem. This emphasis is also reflected in the selection of topics for this course.

- **What this course is not about**
  - Computer-assisted research methods in linguistics
  - Software development for natural language tasks
- **What is not covered but benefits from what is covered**

  - Speech recognition
  - OCR
  - Text generation
  - Parsing
  - Semantic analysis
  - Machine translation

- **List of topics**
  - *Phonology and Morphology*

    The role of formalization
    String languages
    Subregular hierarchy
    Regular languages
    Generative capacity of phonology
    String transductions
    2-level morphology
    Equivalence of SPE and OT
  - *Syntax*

    Tree languages
    Syntax is more complex than phonology
    Mildly context-sensitive formalisms (TAG, MGs)
    Tree transductions
    Regular representations of MCS formalisms
    Reinterpreting the T-model

A rough outline of the course progression is given in Tab. 1 on page III. Many of the topics we cover draw from very specialized areas of formal language theory that even most mathematicians and computer scientists do not know about, e.g. the correspondence between finite-state machines and monadic second-order logic, or the logical characterization of tree transductions. So this course might be of value to you even if you do not particularly care about natural language. Make no mistake, though, we'll talk a lot about language and linguistics — this is not a math class.

1.3 Prerequisites

The only official prerequisite is Computational Linguistics 1 (Lin 537) or comparable programming skills in Python. Python will be used to illustrate formal concepts, and some of the homeworks will require you to implement an algorithm or procedure in Python. Prior experience with git and markdown is useful for the homeworks but not required.

It is also helpful to have some basic familiarity with linguistics (phonemes, phrase structure rules, syntactic trees) and mathematics (sets, functions, relations, and first-order logic as covered in Semantics 1, for instance). You can take an online survey to identify weaknesses, and several introductory readings on these topics are available on the course website.

Survey URL: https://testmoz.com/432409

Wk	*Formal*	*Linguistics*
1	What is computation?	Marr's Three Levels
2	Formalizing phonology	Why formalize?
3	Strictly local languages	Local dependencies
4	Subregular hierarchy	How powerful is phonology?
5	Regular languages	Abstractness
6	String transductions	SPE-OT equivalence
7	Two-level morphology	Null morphemes
8	(Spring Break)	
9	Weak Generative Capacity	Phonology < Syntax
10	Tree languages	Headedness, feature percolation
11	Local tree languages	GPSG
12	Recognizable tree languages	GB
13	TAG and MGs	Minimalist syntax
14	Tree transductions	Reinterpreting the T-model
15	Unification via first-order logic	Strictly Derivational Minimalism

Table 1: Tentative course outline

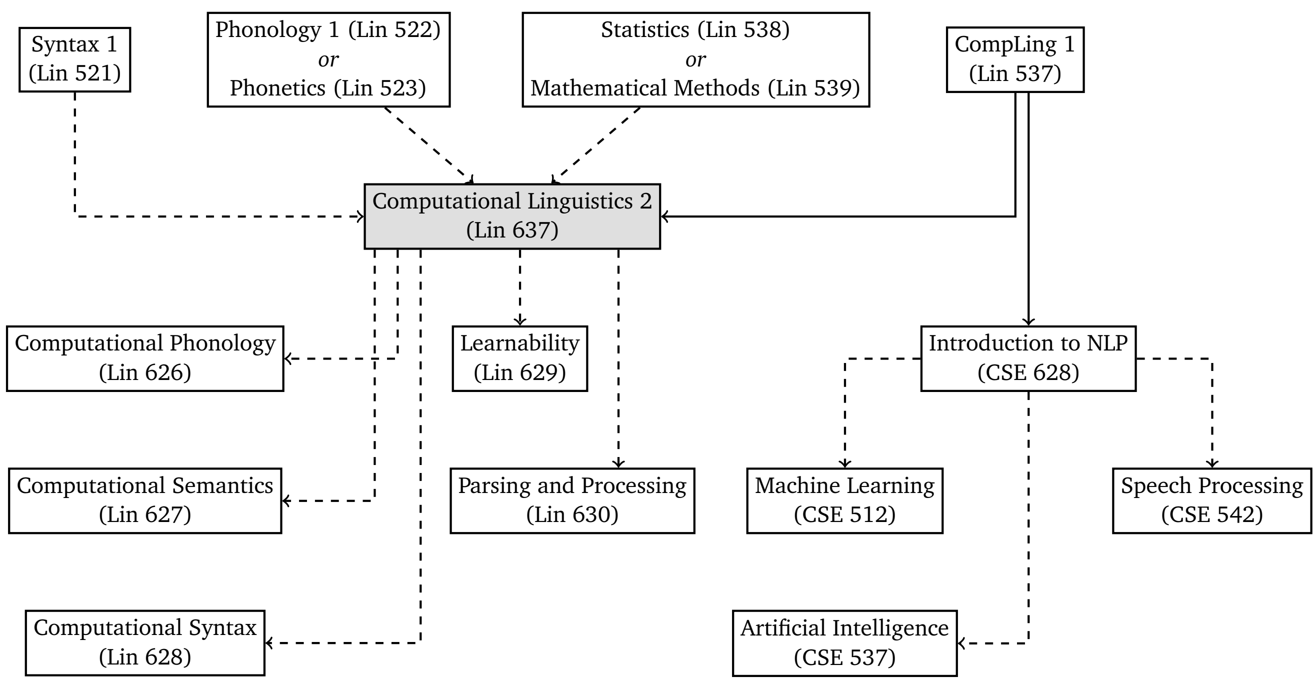

Figure 1: Computational Linguistics 2 in the curriculum (dashed lines indicate recommendations rather than prerequisites)

2 Teaching Goals

- **Practical Skills**
  - conceptualize a problem in mathematical terms
  - optimize your programs through the use of adequate algorithms and data structures (dynamic programming techniques, hash tables, etc.)
  - a more abstract and theoretically informed perspective on current tools and techniques in NLP
  - an understanding for how linguistic insights can be invoked to simplify NLP tasks
- **Research Skills**
  - assess linguistic phenomena from a computational perspective
  - evaluate linguists' claims about computational efficiency
  - basic overview of current research in theoretical computational linguistics
  - use computational concepts to identify new empirical generalizations
  - bring linguistic data to bear on computational claims
  - mathematically informed understanding of linguistic theories

3 Grading

- **Homework**
  - exercises, programming assignments, or critical evaluations of assigned readings
  - Homework submission and grading is done via github.
  - No late hand-ins!
  - Collaboration on homework problems is encouraged as long as you write up the solutions by yourself, using your own words, examples, notation, and code.
- **Readings**
  - at most two readings per week
  - It is presupposed in the lectures that you have done the required readings.
  - Reading comprehension may be tested as parts of the homeworks.
- **Lecture Note Feedback**
  - I plan to publish the lecture notes as an open-access textbook with *Language Science Press*.
  - This is the last time I teach the course before the submission deadline, so I want feedback.
  - Every week, you should file issues on Github for the relevant units, where you spot typos, suggest exercises, pictures, examples, etc.

  - This is a collaborative enterprise: comment on other student's suggestions if you (dis)approve, expand their ideas, and so on.

- **Workload per Credits**
  - *0 credits*: none, but I highly recommend that you at least read the assigned papers as they will be important for following the lectures
  - *1 credit*: readings
  - *2 credits*: readings, feedback
  - *3 credits*: homework, readings, feedback

4 Online Component

This class uses some online tools to facilitate homework collaboration and submission, student discussions, and dynamic lecture evaluation.

- **Homework submission**
  How it works: Homeworks will distributed via a github repository. You can fork this repo and upload your own code, or checkout other students' forks to see how they dealt with the problem. In order to submit a homework you upload your solution to your fork and issue a pull request. After the due date, I'll upload my solution to the repository.

  Why we do it: This setup mimics the modern workflow in collaborative development projects. Git is one of the best-known version control systems, and github is the biggest online service for hosting git repositories. Familiarity with version control systems is an essential job requirement for computational linguists, and it is also very helpful for academic work. See this discussion on Stackflow for some ideas how git can be used in conjunction with Latex: http://stackoverflow.com/questions/6188780/git-latex-workflow

  What you'll need: A github account (the free tier is enough) and a way of uploading your code to a github repository. Linux users can install git via the command line, whereas Windows and Mac users should download and install the github app, which comes with a nice GUI.

- **Homework Feedback and Discussion**
  How it works: Every github repository comes with an issue (= ticket) tracker. If you have a question or wish to discuss a topic, you can open a ticket on the main repository (note: tickets support markdown). I may also use tickets to leave comments on your homeworks.

  Why we do it: Once again this is an essential part of modern software development. And it is also a lot more convenient than anything Blackboard has to offer.

  What you'll need: Not much beyond the ability to navigate the github repos.

- **Class announcements**
  How it works: Normal announcements (readings, due dates) are put on the course website, time-critical ones (i.e. class cancellations) are emailed out via Blackboard.

What you'll need: If you're not officially enrolled in the course for at least 0 credits, send me a message so I can add you to Blackboard.

- **Lecture Notes**
  How it works: The lecture notes are made available online Monday and Wednesday before 3pm. You can look at them on your laptop/tablet or make a hardcopy before class. I will not bring handouts to class unless I could not upload them on time the day before.

  Why we do it: Mostly because I just don't like paper. Also keep in mind that you can fork the lecture notes repository and include your notes directly in the Latex source files. Then you can compile a version of the lecture notes with your own notes already included.

5 Policies

5.1 Contacting me

- Emails should be sent to lin637@thomasgraf.net to make sure they go to my high priority inbox. Disregarding this policy means late replies and is a sure-fire way to get on my bad side.
- Reply time < 24h in simple cases, possibly more if meddling with bureaucracy is involved.
- If you want to come to my office hours and anticipate a longer meeting, please email me so that we can set apart enough time and avoid collisions with other students.

5.2 Disability Support Services

If you have a physical, psychological, medical or learning disability that may impact your course work, please contact Disability Support Services, ECC (Educational Communications Center) Building, Room 128, (631) 632-6748. They will determine with you what accommodations, if any, are necessary and appropriate. All information and documentation is confidential.

Students who require assistance during emergency evacuation are encouraged to discuss their needs with their professors and Disability Support Services. For procedures and information go to the following website: `http://www.stonybrook.edu/ehs/fire/disabilities`

5.3 Academic Integrity

Each student must pursue his or her academic goals honestly and be personally accountable for all submitted work. Representing another person's work as your own is always wrong. Faculty are required to report any suspected instances of academic dishonesty to the Academic Judiciary. Faculty in the Health Sciences Center (School of Health Technology & Management, Nursing, Social Welfare, Dental Medicine) and School of Medicine are required to follow their school-specific procedures. For more comprehensive information on academic integrity, including categories of academic dishonesty, please refer to the academic judiciary website at http://www.stonybrook.edu/uaa/academicjudiciary/

5.4 Critical Incident Management

Stony Brook University expects students to respect the rights, privileges, and property of other people. Faculty are required to report to the Office of Judicial Affairs any disruptive behavior that interrupts their ability to teach, compromises the safety of the learning environment, or inhibits students' ability to learn. Faculty in the HSC Schools and the School of Medicine are required to follow their school-specific procedures.