

EECS 598. Program Synthesis: Techniques and Applications

Lecture 1: Introduction

Xinyu Wang

Logistics: Course staff

- Instructor: Xinyu Wang
 - Assistant Professor in CSE
 - Research: Programming Languages, Formal Methods, Software Engineering
 - Email: xwangsd@umich.edu
 - Homepage: <https://web.eecs.umich.edu/~xwangsd/courses/f20/eecs598.html>
 - Canvas
- No TA

Introduce yourself

- Name?
- What program? What year?
- What areas in CS are you interested in?

Logistics: Course mode

- Mode: Hybrid but to COVID (some components in-person, other components online)
 - In-person lectures: 1014 DOW
 - Online lectures: Zoom (check course webpage!)
 - 75-90mins with 3mins break
 - Ask questions during lecture: unmute yourself and ask
 - Non-urgent questions/comments: type in chat
 - Recordings available after class
 - Mode will be available at least 1 week prior to class (check schedule online!)
 - Office hours: T/TH 4:30-5:30pm eastern time (Zoom)

Logistics: Course structure

- Research-oriented seminar class
 - Reading papers, presentations, discussions
 - Lectures on basics and general landscape

Logistics: Course structure (cont'd)

- Content: Program synthesis techniques & applications
- Four modules
 - Module 1: Programming-by-Example Techniques
 - Module 2: More techniques
 - Module 3: More applications
 - Module 4: Final Project Presentations

Logistics: Course structure (cont'd)

- Module 1: Programming-by-Example Techniques
 - A set of fundamental ideas/techniques underlying many program synthesizers
 - After this module, you should be familiar with all these techniques
- Module 2: More techniques
 - A set of more advanced techniques
 - You should be able to solve many problems using these techniques
- Module 3: Applications
 - Interesting applications that combine different techniques

Logistics: What do you need to do?

- Paper presentation(s): 1-2 papers/student, depending on how many students enrolled
- Paper reviews: at most 2 reviews per week
- Participation: discuss, ask questions, brainstorm new ideas, ...
- Final project: team (1-2 people), proposal, checkpoints, final report, final presentation

Logistics: Paper presentation

- Identify 1-2 papers you want to present
 - Send to instructor
 - If not, you may get any paper
- Prepare (e.g., slides, demo, thoughts, ideas, discuss with others)
- Present (45m talk + 30m QA)
 - Thorough (45m is quite a long time)
 - Give high-level ideas as well as important lower-level technical details
 - Introduce necessary background

Logistics: Paper reviews

- Write a review (template available on course page)
 - A short summary, pros, cons
 - Questions
 - Thoughts
- Send to instructor via email **by midnight the day before class**

Logistics: Participation

- Attend
- Ask questions (don't be shy!)
- Express your opinions
- Connect to your research
- Your ideas
- ...

Logistics: Final project

- Find a teammate (solo is also okay; but if more than 3, check with instructor)
 - Sooner than later!
- Generate ideas
 - Sooner than later!
- Write proposal
- Checkpoints: Progress report
- Final presentation
- Final report

Logistics: Final project (cont'd)

- Different kinds of final projects
 - Extend/improve a technique in a paper
 - Apply an existing synthesis framework to a new problem domain
 - Develop a new synthesis technique for an existing problem
 - Develop a new synthesis technique for a new problem
 - ...
- Grading of final project is based on: originality, completeness, scope

Logistics: Final project (cont'd)

- Proposal
 - 1-2 pages, like an introduction, also include a timeline and a sketch of solution
 - need to convince me your problem is worth solving and is technically challenging
 - also need to convince me you are able to solve it within 2 months (at least partially)

Logistics: Final project (cont'd)

- Checkpoints
 - Nothing but a progress report
 - A partial final report that is gradually more complete over time

Logistics: Final project (cont'd)

- Final project report
 - 6-8 pages, structured like a conference paper
 - Include:
 - Introduction — why this project
 - Motivating example — illustrate how your technique works concretely
 - Technical details — make sure to first give high-level idea before showing details
 - Evaluation — how it works in practice
 - Related work — how your idea relates to existing work

Logistics: Grading

- Paper presentation: 20%
- Paper reviews: 30% (2% x 15)
- Participation: 5%
- Final project: 45%
 - Proposal: 5%
 - Checkpoints: 16% (8% x 2)
 - Final project presentation: 12%
 - Final project report: 12%

What is this course about?

- This course is about program synthesis, including both techniques and applications
 - Techniques: general synthesis algorithms not necessarily tied to a specific application
 - Applications: novel application of program synthesis techniques
- Beyond acquiring knowledge about program synthesis, also:
 - PL thinking
 - Writing, presentation, ...

You should take this course

- .. if you are doing or plan to do research in program synthesis
- .. if you are interested in programming languages research
- .. if you plan to explore possibilities of applying PL in your own research
- .. if you just want to learn about the topic!

What is “program synthesis”?

- What is “program”?
 - C/C++/Java/Python...
 - Haskell/ML/OCaml/Lisp/...
 - SQL/Datalog/...
 - ...
- Synthesis from what?
 - Input-output examples
 - Natural language
 - Demonstrations
 - ...

Example 1: FlashFill [Gulwani et al. 11]

- Synthesize Excel macros for string processing from input-output examples ([video](#))



	A	B
1	Names	Initials
2	Neil Lieber	N L I
3	Mathew Prisco	
4	Althea Bertin	
5	Kelly Gamblin	
6	Chandra Valenzula	
7	Cody Castillon	
8	Tyrone Brazier	
9	Althea Buhl	
10	Dollie Munsey	
11	Allyson Phou	



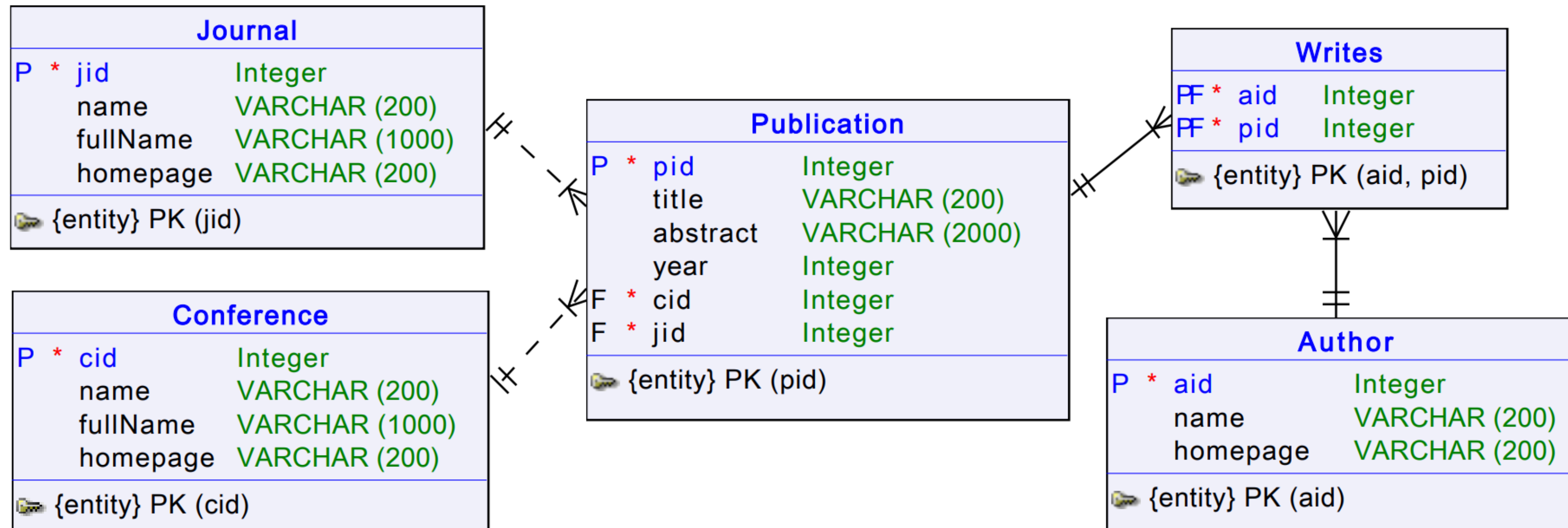
	A	B
1	Names	Initials
2	Neil Lieber	N L
3	Mathew Prisco	M P
4	Althea Bertin	A B
5	Kelly Gamblin	K G
6	Chandra Valenzula	C V
7	Cody Castillon	C C
8	Tyrone Brazier	T B
9	Althea Buhl	A B
10	Dollie Munsey	D M
11	Allyson Phou	A P

Example 2: SQLizer [Yaghmazadeh et al. 17]

- Synthesize SQL queries from natural language (given schema)

NL: *“Find the number of papers in OOPSLA 2010”*

Schema:



SQL query:

```
SELECT count(Publication.pid)
FROM Publication JOIN Conference ON Publication.cid = Conference.cid
WHERE Conference.name = "OOPSLA" AND Publication.year = 2010
```

Example 3: Rousillon [Chasins et al. 18]

- Synthesize web scraping scripts from example demonstrations ([video](#))

The screenshot displays the Google Scholar interface. At the top, the search bar contains the query 'label:computer_science'. Below the search bar, a list of profiles is shown, including Vapnik, Geoffrey Hinton, DEYWIS MORENO, David S. Johnson, and David Haussler. The profile for Vapnik is expanded, showing his title as 'Professor of Columbia, Fellow of NEC Labs America' and his research interests in 'machine learning', 'statistics', and 'computer science'. A table lists his works with their citation counts and years. To the right, a 'Cited by' section shows a bar chart of citations from 2011 to 2018 and a table of citation statistics.

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	77066 *	1995
Statistical Learning Theory VN Vapnik Wiley-Interscience	76088 *	1998
Support-vector networks C Cortes, V Vapnik Machine learning 20 (3), 273-297	31294	1995
Pattern classification and scene analysis RO Duda, PE Hart A Wiley-Interscience Publication, New York: Wiley, 1973	20912	1973

	All	Since 2013
Citations	257774	105623
h-index	124	81
i10-index	465	347

Year	Citations
2011	~15000
2012	~16000
2013	~16500
2014	~17000
2015	~17500
2016	~18000
2017	~19000
2018	~14000

Example 3: Rousillon [Chasins et al. 18] (cont'd)

- Synthesize web scraping scripts from example demonstrations (video)

The image shows a split-screen view. On the left is a web scraping tool interface with a 'Current Script' tab. It contains instructions: 'Awesome. We're recording, so go ahead and collect that first row of data. When you're ready to add a new cell to the first row of your dataset, just hover over the text you want to add, then press **ALT** + click. We'll show the data you've collected right here:'. Below this is a text box containing the text: 'vapnik The Nature of Statistical Learning Theory 77066 1995'. At the bottom of the tool is a red square button with a white square inside, labeled 'Stop Recording'. On the right is a browser window showing a Google Scholar profile for 'vapnik'. The profile includes a circular profile picture, the name 'vapnik', and a 'FOLLOW' button. Below the profile is a list of articles with columns for 'TITLE', 'CITED BY', and 'YEAR'. The first article is 'The Nature of Statistical Learning Theory' by V Vapnik, with 77066 citations and published in 1995. The second is 'Statistical Learning Theory' by VN Vapnik, with 76088 citations and published in 1998. The third is 'Support-vector networks' by C Cortes and V Vapnik, with 31294 citations and published in 1995. The fourth is 'The Nature of Statistical Learning Theory' by V Vapnik, with 20912 citations and published in 1973. A purple banner at the bottom of the browser window reads 'demonstrate how to collect the first row'.

TITLE	CITED BY	YEAR
The Nature of Statistical Learning Theory V Vapnik Data mining and knowledge discovery	77066 *	1995
Statistical Learning Theory VN Vapnik Wiley-Interscience	76088 *	1998
Support-vector networks C Cortes, V Vapnik Machine learning 20 (3), 273-297	31294	1995
The Nature of Statistical Learning Theory V Vapnik A Wiley-Interscience Publication, New York: Wiley, 1973	20912	1973

demonstrate how to collect the first row

What is “program synthesis”?

- “Program Synthesis correspond to a class of techniques that are able to generate a program from a collection of artifacts that establish semantic and syntactic requirements for the generated code.”¹

High-level intent
Specification

Program synthesis
→

Lower-level code
Program

Program Synthesis vs. Machine Learning/Deep Learning

- ML/DL is also program synthesis?
 - ML/DL: data is spec, model is program, try to learn a model that matches data
 - At a high-level, yes
 - But in this class, no, at least not the focus
 - Definitions of “programs” are very different (e.g., grammar vs. neural nets)
 - Data is noisy whereas spec is less noisy (but there is a trend in program synthesis to tolerate noise in spec)
 - Typically continuous in ML/DL vs. discrete search space in program synthesis
 - The line is getting blurry

Program Synthesis vs. Compilers

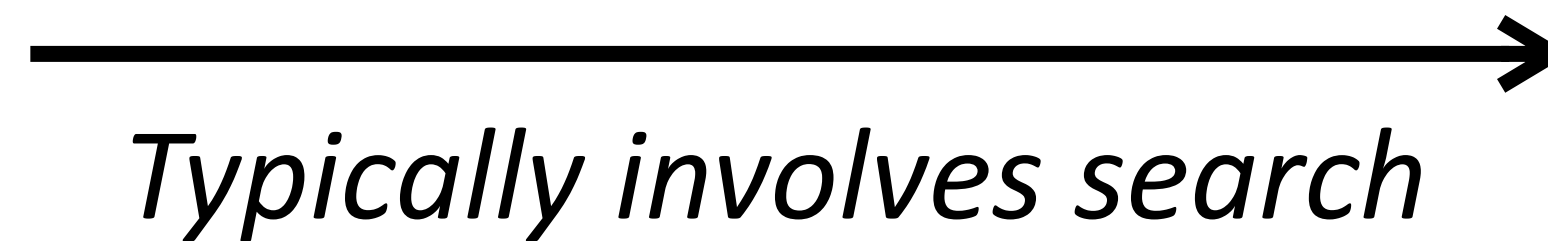
- Program synthesizers are compilers? Compilers are synthesizers?
 - Compilers also convert high-level intent (code) to lower-level code
 - At a high-level, yes
 - But in this class, no, at least not the focus
 - Compilers translate (well, not really nowadays) whereas synthesizers discover
 - Compilers apply predefined transformations (again, not really nowadays) whereas synthesizers perform search
 - The line is getting blurry

Working definition of program synthesis in this course

High-level intent

Specification

Program synthesis



Lower-level code

Program

*I/O examples, demonstrations,
natural language, reference
implementation, etc.*

*In some programming language
(grammar + semantics)*

Why program synthesis?

- Many useful applications
 - E.g., FlashFill in Excel
- Technically challenging
 - Exponential search space (or even undecidable)
- Cool
 - Intersection of many areas: PL, AI, FM, systems, logics, ...

Three pillars of program synthesis [Gottschlich et al. 18]

- Intention
 - How do users specify their goals?
 - Examples, demonstrations, NL, ..., or their combinations!
 - Challenges: under-specified, ambiguous, unstructured
- Invention
 - How to find the right solution?
 - Search-based, representation-based, learning-based, ..., and their combinations!
 - Challenges: scalability, ambiguity
- Adaptation
 - How to find the right solution, not starting from scratch?
 - Bug fixes, patches, extension to new hardwares, ...
 - Challenges: analyzing, learning, scalability

This course

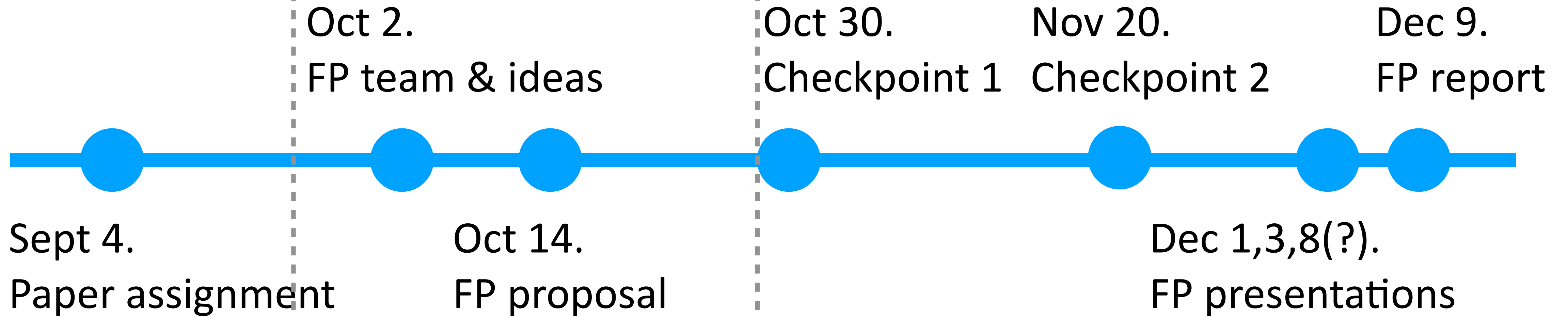
- Module 1: Techniques for example-based specs
 - Representation-based techniques (both top-down and bottom-up)
 - Search-based techniques (both top-down and bottom-up)
 - Using deduction to guide search and prune search space
 - Module 2: Techniques for specs beyond just examples
 - Specs: reference implementation, types, NL, multi-modal
 - Techniques: CEGIS, ML/DL-based, combinations, interactive
 - Module 3: Applications
 - Super-optimizations, SE, web, DB, security, graphics, arch, ...
-
- Invention
- Intention
Invention
- Intention
Invention
adaptation

Timeline (still tentative)

Module 1

Module 2

Module 3



Summary of this lecture

- Program synthesis is cool
- You should take this class and learn about it
- You will learn a lot from this class

Next lecture (Sept 3)

- Syntax-guided synthesis
 - Popular framework for program synthesis
- Representation-based techniques
 - Top-down: FlashFill [Gulwani11], Sept 8
 - Bottom-up: Dace [Wang17], Sept 10
- Search-based techniques
 - Top-down: L2 [Feser15], Sept 15
 - Bottom-up: Optional readings of Sept 15 — [Udupa13], [Albarghouthi13]

Survey (optional)

- Send me a brief email with:
 - Name
 - I am a [CS/___] [PhD/Masters/undergrad] in year [1/2/3/4/5/...]
 - Write one reason why you are taking this class or one thing you want to get out of it
 - One thing you would like the instructor to do in this class
 - One fun fact about you, or what you like to do in your spare time, or whatever