First Day Stuff

Introductions

Adi, Eli, Aldrin, Clara, Jenny, Hannah, Sofiane, Diego, Max, Shaheen, Abtin, Ben

- Name, school, major, year
- Fun thing looking forward to this summer
- X that you'd recommend to somebody else?

Scheduling

Overall summer plan

- Ongoing (stay tuned)
    - "ETC" talks (Extraneous Topics Colloquium) where people present for 15 minutes about anything as long as not related to work. Topics range from how to juggle, bake bread, or pick locks to stuff about ancient history, anime, and the thermodynamic arrow of time.
    - Weekly (trivia?) competition
    - Reading groups
- First few weeks
    - Meet pretty often (9-5 ish, but flexible)
    - Explore your problem, come up with research questions, hypotheses, brainstorm
    - Explore related work and publication venues
    - Presentations from profs
- Middle weeks
    - Meet less often, but also anytime as necessary
    - Projects ongoing
    - Student presentations
        - Early on: 1 minute, 1 slide
        - Later: 10-30 minutes depending on project
- Later weeks
    - Wrapping up experiments, coding, designs
    - Writing up results
    - Final presentations (possibly videos)
    - (Group) tech report due by your last day of summer research

Week 1 (soft opening)

- Today (Monday)
    - 9:30-?? initial meeting
    - ??-2:30 go explore! Please chat with your research subgroup.
    - 2:30 - 5:00 subgroup meetings
        - 2:30 Clara Jenny

        - 3:00 Ben Aldrin
        - 3:30 Safiane Eli Adi
        - 4:00 Shaheen Hannah Max
        - 4:30
- Tuesday
    - 9:00 (less than 1 hour) big group check-in
    - Afternoon: up to you! you're on your own! But not alone! Please work together and check in together!
- Wednesday: all check in during the afternoon
- Thursday
    - 9:30 check-in
    - Afternoon?
- Friday
    - 9:30 Check in
    - Afternoon: game? suggestions?

Week 2 (grand opening)

Monday: Meet and plan for the week
Tuesday: Big kick-off meeting (all summer CS students and advisors)

Communication

- Summer of CS Slack
    - https://join.slack.com/t/hmcsummerofcs2021/shared_invite/zt-qhxnwetd-4UApiUfA5F56v3pb0lfYMQ
- ALPAQA Slack
    - Change the name from fall 2020??
    - Add all the new folks: https://join.slack.com/t/hmc-alpaqa/shared_invite/zt-q9qtybiv-IDELdmCR__mAkmbLNnzF0w

Access to resources

- Github
    - https://github.com/hmc-alpaqa
- Google Drive
    - https://drive.google.com/drive/folders/1uFLxG1c0hGmQZihG12R3TIrsFSYmgojp?usp=sharing

Getting paid

- Who has and has not heard from HMC Human Resources?

Diego and Sofiane -- Path Complexity

Path complexity analysis: counting the number of paths of length (# “steps”) of length n through a program of interest. Asymptotic Path Complexity is APC

- Background
  - Paper 1: https://www.cs.hmc.edu/~bang/research/fse15.pdf
  - Slides 1: https://www.cs.hmc.edu/~bang/research/fse15slides.pdf
  - Paper 2: https://www.overleaf.com/read/rbqbdznsmkxt
  - Video 2: https://www.youtube.com/watch?v=qu2n-QYV6-w

- Repo:
  - https://github.com/hmc-alpaqa/metrinome

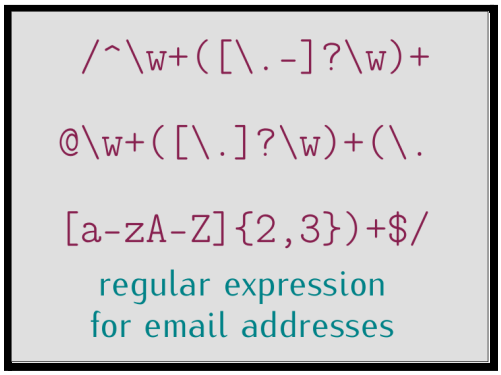
- Suggested case studies
  - Is APC related to “cognitive complexity”
    - Get some cognitive complexity data for come code, run APC on the code, see if correlation
    - https://www.tu-chemnitz.de/informatik/ST/publications/papers/ICSE21.pdf
    - https://github.com/brains-on-code/fMRI-complexity-metrics-icse2021/tree/master/data
    - https://arxiv.org/pdf/2102.12376.pdf

  - Is the number of tests related to APC?
    - Find a Java or C project with tests
    - Compute APCs for functions
    - Count number of tests that call each function
    - Correlate?
  - Inverse correlation between APC and time complexity?
    - We did a quick study with sorting algs, didn’t get far
    - Let’s look at something with very obvious differences:
      - O(n^2) closest pair algorithm
      - O(n log n) closest pair algorithm
      - https://www.youtube.com/watch?v=6u_hWxbOc7E
      - https://rosettacode.org/wiki/Closest-pair_problem
  - Identify code “hot spots” faster than profiling?
    - APC computes number of paths from entrance to exit of a function, but we can also compute number of paths for any pair of nodes. Does high APC predict places where lots of computation might occur?
    - *Prof Bang has some old notes on this if he can find them!*

Jenny and Clara -- Semantic Density

Programming languages vary in their verbosity. Some languages require writing a lot of code (think Java) and some languages require writing very little code (APL and J are extreme examples of terse languages). The amount of meaning that is encoded into the symbols of a program determines the semantic density of that program. In this project we will borrow techniques from natural language processing and information theory to quantify the semantic density of a programming language's syntax. We will design an experiment to compare semantic density with human performance on programming tasks in a custom domain specific language.

- https://drive.google.com/file/d/134kY1TMlvegoRIO4ha6w5-JLUxZEzcW4/view?usp=sharing
- A usually terse domain specific language: **Regular Expressions (regex)**



  - Tutorials
    - https://www.youtube.com/watch?v=sa-TUpSx1JA
    - https://towardsdatascience.com/nlp-basics-understanding-regular-expressions-fc7c7746bc70
  - A game
    - https://alf.nu/RegexGolf
  - Papers, articles, projects about Natural Language RegEx
    - https://simple-regex.com/examples
    - https://github.com/SimpleRegex/SRL-Python
    - http://taoxie.cs.illinois.edu/publications/nl4se18-regex.pdf
    - https://dev.to/narendersaini32/how-to-write-regex-in-natural-language-158j
    - https://github.com/mbasso/natural-regex
    - https://github.com/mbasso/natural-regex/wiki/Examples
  - Regex Syntax in Python
    - https://docs.python.org/3/library/re.html
- How much information is in some text?
  - Paper in the google drive

Adi and Eli (and Sofiane)-- Difficulty of Automated Testing

Given some code, can we predict how hard it is to test, but without testing it!?

Path complexity: measures how the number of paths in the program grows as we let the execution length grow.

- Background
  - Paper 1: https://www.cs.hmc.edu/~bang/research/fse15.pdf
  - Slides 1: https://www.cs.hmc.edu/~bang/research/fse15slides.pdf
  - Paper 2: https://www.overleaf.com/read/rbqbdznsmkxt
  - Video 2: https://www.youtube.com/watch?v=qu2n-QYV6-w

- Repo: https://github.com/hmc-alpaqa/metrinome

Branch selectivity: proportion of inputs that follow different execution paths after a branch. E.g if(x == 0) has one path that can only be executed with a single input (x = 0) and one path that can be executed with (2^32 - 1) inputs, whereas (if x % 2 == 0) has a 50-50 split.

- Background
  - Probabilistic reachability using branch selectivity
    - Paper in the google drive (DO NOT DISTRIBUTE)

Idea for the summer:

- We have some data from an ML model used to predict bugs
  - Training set: code files + bug labels
  - Predictor
    - input some code features (LOC, # branches, # api calls, etc.)
    - output error “proneness”
    - output quality of prediction on training set
  - The features are very basic
  - Will provide link to paper
- Can we improve prediction by including other more sensitive features?
  - Path complexity (HMC)
  - Branch selectivities (UCSB)

First steps:

- Be able to run Metrinome
  - Get help from Sofiane :)
- Be able to output a feature vector instead of a function
  - E.g.
    - f(n) = n^2 * 6 * n ^ 1.37 → [6, 2, 1.37, 0, 0]
    - f(n) = 5.3 n^3 → [0, 0, 0, 5, 3]
- Once we can do this, check with Mara at UCSB on exact format

Aldrin and Abtin -- Domain Specific Language for Group Theory

The student's problem: in intro group theory, you are given some rules and axioms and asked to prove some properties of groups or elements of groups. It is often hard to know which things you are allowed to do, and previous experience in math can lead you astray. E.g. when can you multiply on the left and the right?

The grader's problem: grading incorrect group theory proofs is really hard!
A not so great solution: proof assistants (LEAN) use arcane syntax and do too much.

The goal: Create an interactive language for building group theory proofs.

- Appeal to intuition
- Easily map to how proofs in group theory actually look

Some first steps

- Explore some existing interactive proof tools for logic
    - This one from UCSD:
        - Play the game for a while
        - Try with logic syntax turned on or off
        - https://cseweb.ucsd.edu/~lerner/proof-game/
        - Check out the paper
    - This one for natural deduction (used in CS81?)
        - https://proofs.openlogicproject.org/
    - Check out this video about writing an interactive proof assistant in Racket
        - https://www.youtube.com/watch?v=LQhn71FSLOw
        - The paper: https://arxiv.org/pdf/1611.09473.pdf
    - This talk on "the future of mathematics" requiring interactive theorem provers
        - https://www.youtube.com/watch?v=Dp-mQ3HxgDE
    - A video on using LEAN for a simple proof

- Think about what an interactive proof checker could be like when proving "if xx = e for all x in G, then G is commutative"
    - What happens when things go right?
    - What happens when things go wrong?
    - What does the user do?
    - What does the output look like?
    - How do you know you are done?
    - What other questions can help guide the design?;;

- Explore Mathematica's Proof Generator
    - https://reference.wolfram.com/language/ref/FindEquationalProof.html
        - Scroll down → Applications → Abstract Algebra

Resources

- Some chapters on theorem proving for equality logic in the google drive

Hannah, Shaheen, Max, (Abtin) -- Linear Algebraic Parallel Symbolic Execution

- Current status of project

- Symbolic Execution
  - Video Tutorials
    - Mike Hicks
    - Armando Solar-Lezama
  - Papers
    - https://people.eecs.berkeley.edu/~ksen/papers/cacm13.pdf

- Basic Algebraic Graph Theory
  - Matrix Multiplication = Path counting
    - https://quickmathintuitions.org/finding-paths-length-n-graph/
  - Other graph algorithms using linear algebra
    - Chapters in the google drive
- Parallel Programming with MPI in Python
  - https://www.youtube.com/watch?v=onwBKVFdXzc
  - https://www.youtube.com/watch?v=Udn9wmmb9YY
  - https://www.youtube.com/watch?v=36nCgG40DJo

- Our notes:
  - https://www.overleaf.com/6571987337wskkndhpqhjw

- Demos from Shaheen and Max on ??