Synthesis

Reading Reflection

Discuss in groups

- So far, do you find enumerative search (tracking the subset of the program space explored so far) more natural? Or symbolic search (representing the space of all valid programs)?
- How are you feeling about viewing programs as manipulable objects rather than text?
- Can you run this program in DrRacket? Please try to help each other debug if you can't!
  - ```
    #lang rosette/safe

    (require rosette/lib/synthax)
    (current-bitwidth #f)
    ```
    ```

Reading Key Takeaways

- Inductive reasoning makes broad generalizations from specific observations
  - → inductive synthesis is about generalizing from ambiguous specifications
- The difference between finding *a* program that satisfies the spec and finding the program the user actually wants
- A nice review of ASTs and the importance of the DSL design for the program space size (although we touched on these in last class)
- A friendly introduction to symbolic search (representing the space of all valid programs instead of tracking the subset of programs explored so far)

Enumerative → Symbolic (Constraint-Based)

The Rosette Language

ABOUT DOWNLOAD DOCS APPS COURSES PAPERS

A brilliant language from Emina Torlak

About Rosette

Rosette is a solver-aided programming language that extends Racket with language constructs for program synthesis, verification, and more. To verify or synthesize code, Rosette compiles it to logical constraints solved with off-the-shelf SMT solvers. By combining virtualized access to solvers with Racket's metaprogramming, Rosette makes it easy to develop synthesis and verification tools for new languages. You simply write an interpreter for your language in Rosette, and you get the tools for free!

```
#lang rosette

(define (interpret formula)
  (match formula
    [`(∧ ,expr ...) (apply && (map interpret expr))]
    [`(∨ ,expr ...) (apply || (map interpret expr))]
    [`(¬ ,expr)     (! (interpret expr))]
    [lit            (constant lit boolean?)]))
```

If you want to get *really* into Rosette, I recommend…

- https://courses.cs.washington.edu/courses/cse507/19au/index.html


Here be
dragons
Calandra.dua

To think about for next reading

- You do *not* need to memorize or deeply understand details of these approaches!
  - I want you to recognize the key terms and know where to turn for a high-level overview of key techniques. Also, this chapter offers excellent pointers to examples of synthesis work, which you might find useful if you start tackling a synthesis project.
- Think about
  - How these different approaches would or wouldn't apply to the synthesis ideas you brainstormed on Tuesday
  - How these different approaches shape the user interaction