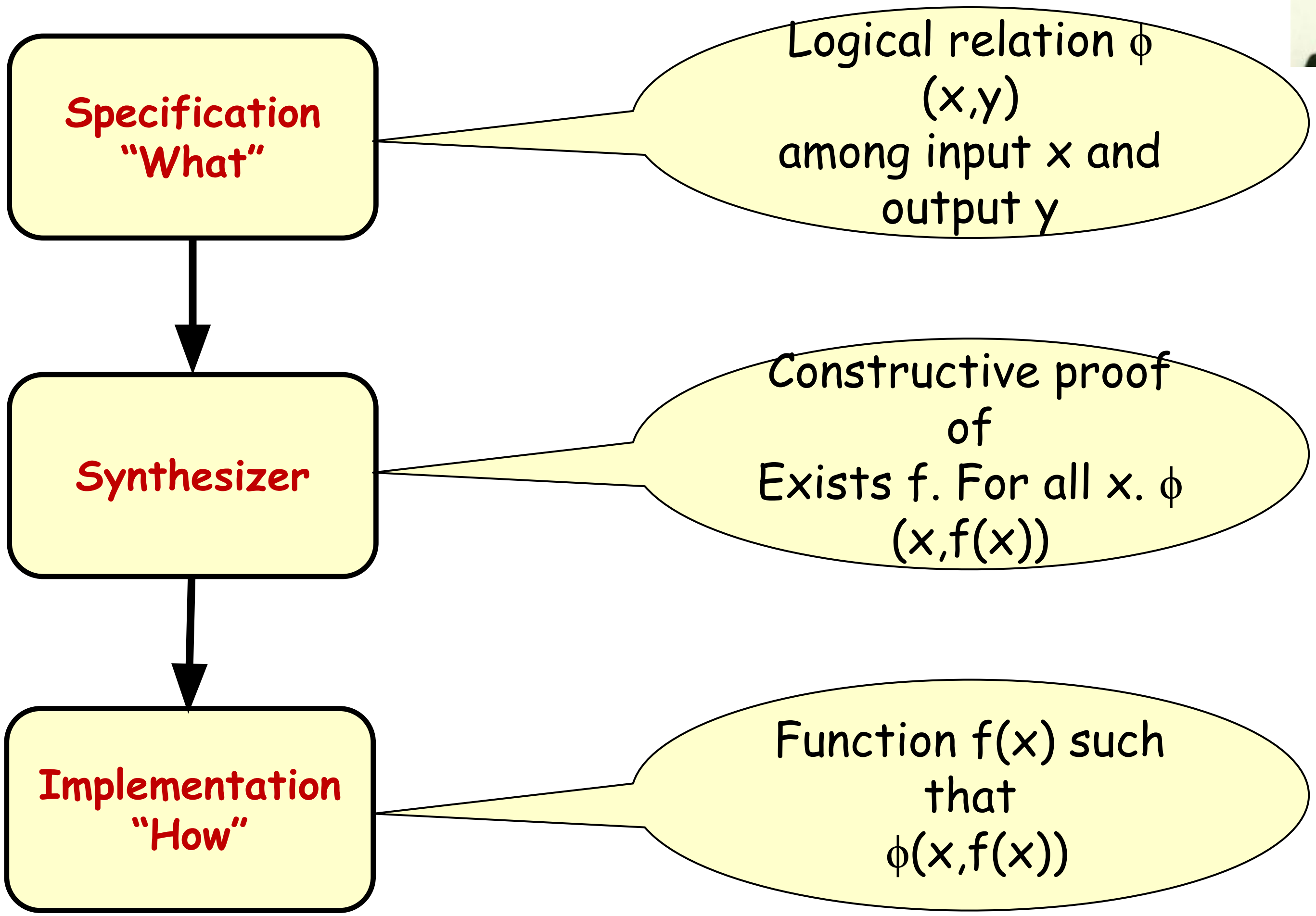
Syntax-Guided Program Synthesis

Rajeev Alur

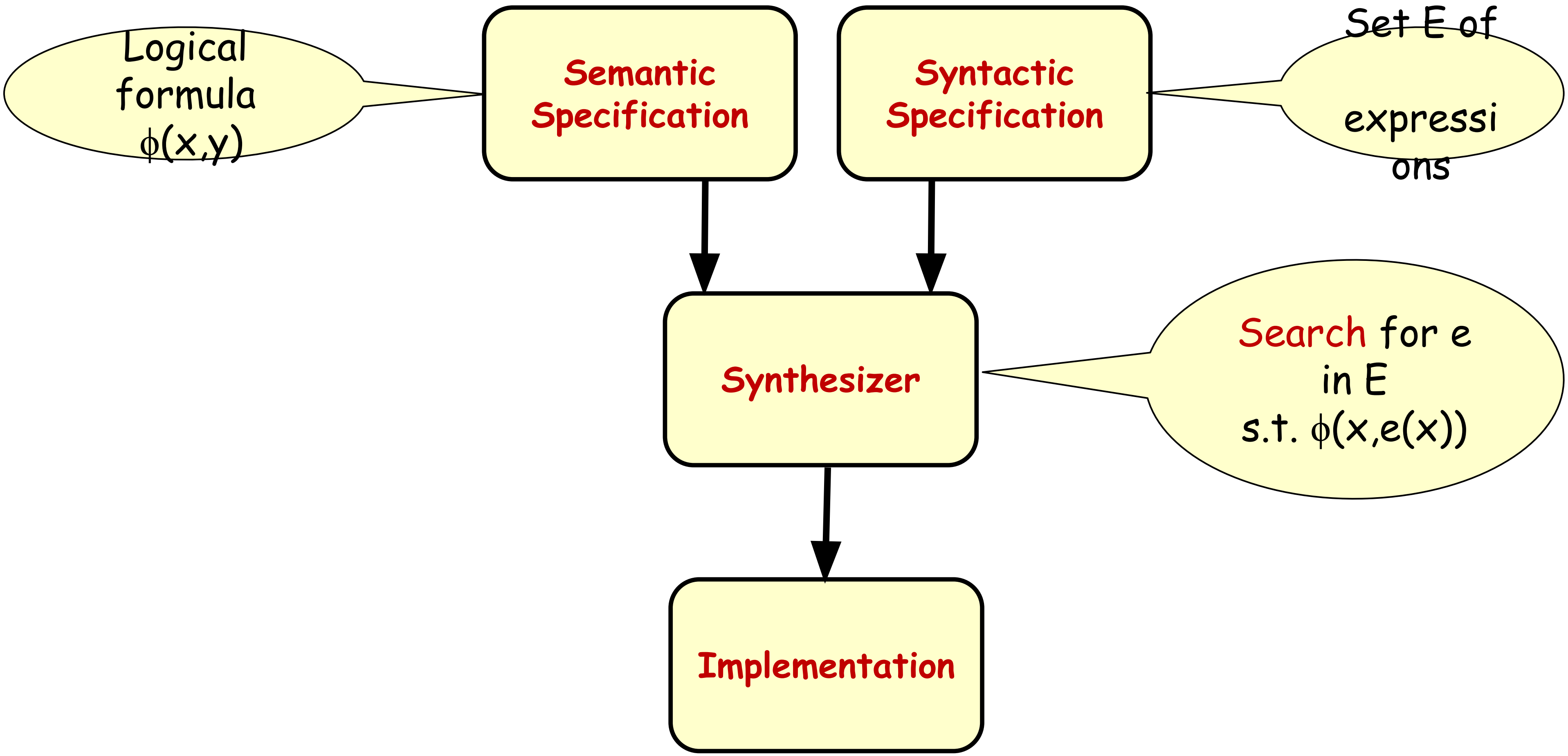
Workshop on Program Synthesis for Scientific Computing, August 2020

Classical Program Synthesis

Church (1957)



Syntax-Guided Search-based Program Synthesis



Motivating Applications

- Superoptimizing compilers: Given a program fragment P, find a functionally equivalent program with resource constraints (e.g. fewer instructions, or avoid certain expensive instructions)

- Program repair: Automatically edit a program locally to fix a bug (particularly helpful to students in Intro Programming courses)

- Proof objects for verification: template-guided synthesis of inductive invariants, ranking functions, program analysis rules, …

- Programming by examples / demonstration: Can non-programmers communicate intent intuitively?

Syntax-Guided Synthesis (SyGuS)

SyGuS
www.sygus.org

- Fix a background theory T: fixes types and operations

- Function to be synthesized: name f along with its type
  - General case: multiple functions to be synthesized

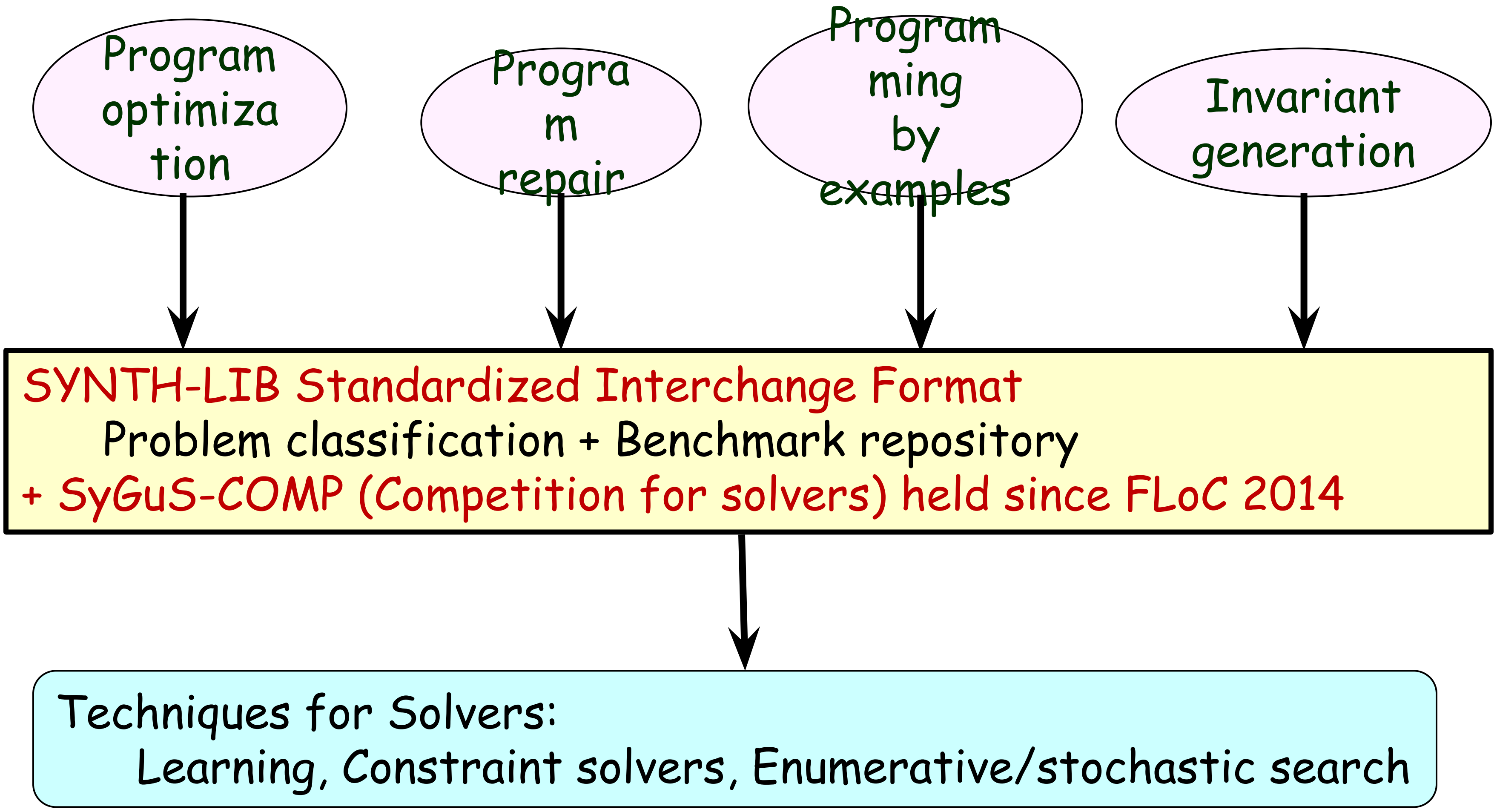
- Inputs to SyGuS problem:
  - Specification $\phi(x, f(x))$
    Typed formula using symbols in T + symbol f
  - Set E of expressions given by a context-free grammar
    Set of candidate expressions that use symbols in T

- Computational problem:
  Output e in E such that $\phi[f/e]$ is valid (in theory T)

Syntax-guided synthesis; FMCAD'13

with Bodik, Juniwal, Martin, Raghothaman, Seshia, Singh, Solar-Lezama, Torlak, Udupa

SyGuS Competition

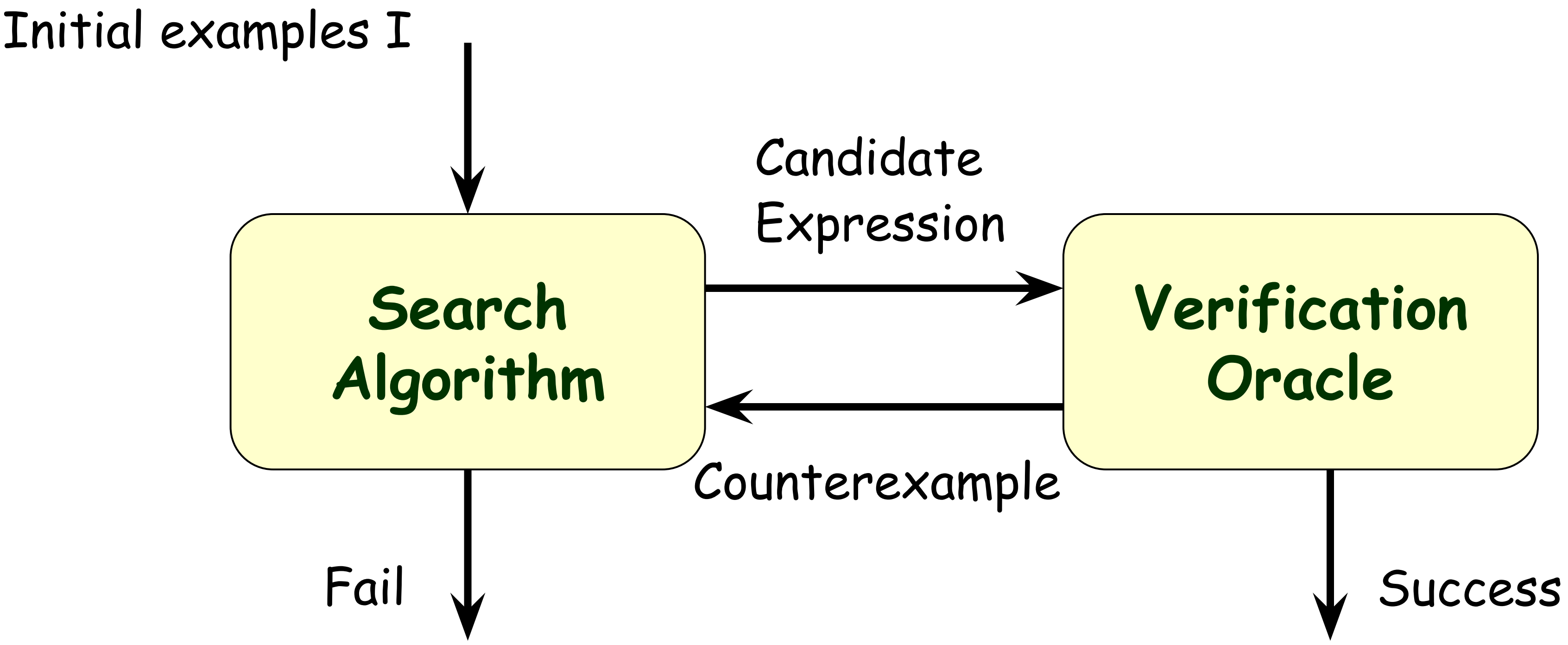


SyGuS Progress

www.sygus.org

- Over 2800 benchmarks
  - Hacker's delight
  - Invariant generation (based on verification competition SV-Comp)
  - FlashFill (programming by examples system from Microsoft)
  - Synthesis of attack-resilient crypto circuits
  - Program repair
  - Motion planning
  - ICFP programming competition
- Special tracks for competition
  - Invariant generation
  - Programming by examples
  - Conditional linear arithmetic
- Current winner: CVC4 (Reynolds et al) search integrated in constraint solving

Search and Verify



Concept class: Set E of expressions

Examples: Concrete input values

Counterexample-guided Inductive Synthesis (CEGIS)

Goal: Find f such that for all x in D, $\phi(x, f)$ holds

I = { }; /* Interesting set of inputs */

Repeat

    Learn: Find f such that for all x in I, $\phi(f, x)$ holds

    Verify: Check if for all x in D, $\phi(f, x)$ holds

        If so, return f

        If not, find x such that ~ $\phi(f, x)$ holds, and add x to I

Implementing Search

- Given:
  Specification $\phi(x, f(x))$
  Grammar for set E of candidate implementations
  Finite set I of inputs
  Find an expression e(x) in E s.t. $\phi(x,e(x))$ holds for all x in I

- Enumerative search with lots of optimizations for pruning
- Symbolic constraints over variables encoding desired expression tree
- Stochastic search in spirit of genetic programming
- Divide and conquer strategies to build sub-expressions
- Partial evaluation to rule candidates before fully expanding them
- Establishing unrealizability of synthesis
- Type-directed enumeration

Acceleration Using Learned Probabilistic Models

- Can we bias the search towards likely programs?

- Step 1: Mine existing solutions to convert given grammar into a probabilistic higher-order grammar
  - Weighted production rules
  - Conditioned on parent and sibling context
  - Transfer learning used to avoid overfitting

- Step 2: Enumerative search to generate expressions in decreasing likelihood
  - Use A* with cost estimation heuristic
  - Integrated with previous optimizations (equivalence-based pruning…)

With W. Lee, K. Heo, and M. Naik (PLDI 2018)

Future Directions

- Beyond SMT Solvers: SyGuS-like back-end focused on efficient search, but decoupled from SMT solvers so as to allow interface with alternative testing / verification tools
- More theories, benchmarks, and applications: tables and relational queries, floating point arithmetic
- Quantitative synthesis and optimization
- Applications in scientific computing ??