Program Synthesis

Emina Torlak
emina@cs.washington.edu
Today

Last lecture
- Angelic nondeterminism and execution

Today
- Program synthesis: computers programming computers

Announcements
- Please fill out the course evaluation form (Dec 02-08)
“Information technology has been praised as a labor saver and cursed as a destroyer of obsolete jobs. But the entire edifice of modern computing rests on a fundamental irony: the software that makes it all possible is, in a very real sense, handmade. Every miraculous thing computers can accomplish begins with a human programmer entering lines of code by hand, character by character.”

Interview with Moshe Vardi
The program synthesis problem

\[ \exists P . \forall x. \varphi(x, P(x)) \]

- \( \varphi \) may be a formula, a reference implementation, input/output pairs, traces, demonstrations, etc.

Synthesis improves

- Productivity (when writing \( \varphi \) is easier than writing \( P \)).
- Correctness (when verifying \( \varphi \) is easier than verifying \( P \)).

Find a program \( P \) that meets the input/output specification \( \varphi \).
Two kinds of program synthesis

- **Inductive (syntax-guided) synthesis**
  - Discover the program $P$ by searching a restricted space of candidate programs for one that meets $\varphi$ on all inputs.

- **Deductive (classic) synthesis**
  - Derive the program $P$ from the constructive proof of the theorem $\forall x. \exists y. \varphi(y, x)$.

**Examples**

- SPIRAL
- FlashFill

**Formal Expression**

$$\exists P. \forall x. \varphi(x, P(x))$$
Deductive synthesis with axioms and E-graphs

Complete specification $\varphi$ of the desired program (a reference implementation in an ISA).

$\forall k, n. 2^n = 2^{\ast\ast}n$
$\forall k, n. k \ast 2^n = k \ll n$
$\forall k, n. k \ast 4 + n = \text{s4addl}(k, n)$
...

1. Construct an E-graph.
2. Use a SAT solver to search the E-graph for a K-cycle program.

Optimal (lowest cost) program $P$ that is equivalent to $\varphi$ on all inputs (values of $\text{reg6}$).

Denali Superoptimizer [Joshi, Nelson, Randall, PLDI’02]

$\text{reg6} \ast 4 + 1 \rightarrow \text{s4addl}(\text{reg6}, 1)$

Two kinds of axioms:
- Instruction semantics.
- Algebraic properties of functions and relations used for specifying instruction semantics.
Denali by example

\[
\forall k, n. 2^n = 2^{**n}
\]
\[
\forall k, n. k\times 2^n = k \ll n
\]
\[
\forall k, n. k\times 4 + n = \text{s4addl}(k, n)
\]

E-graph matching

\[
s4addl \quad \ll
\]
\[
2 \quad + \quad 1
\]
\[
\star \\
reg6 \quad 4
\]
\[
\star \\
2 \quad 2
\]

\[
\text{SAT}
\]
\[
\text{s4addl}(\text{reg6}, 1)
\]
Deductive synthesis versus compilation

**Deductive synthesizer**
- Non-deterministic.
- *Searches* all correct rewrite sequences (proofs) for one that yields an optimal program.

**Compiler**
- Deterministic.
- Lowers a source program into a target program using a fixed sequence of rewrites.
Deductive synthesis versus inductive synthesis

**Deductive synthesis**
- Efficient and provably correct: thanks to the semantics-preserving rules, only correct programs are explored.
- Requires complete specifications to seed the derivation.
- Requires sufficient axiomatization of the domain.

**Inductive synthesis**
- Works with multi-modal and partial specifications.
- Requires no axioms.
- But often at the cost of lower efficiency and weaker (bounded) guarantees on the correctness/optimality of synthesized code.

\[
\exists P. \forall x. \varphi(x, P(x))
\]
Inductive syntax-guided synthesis

A partial or multimodal specification $\phi$ of the desired program (e.g., assertions, i/o pairs).

Solves $\exists P. \varphi(x_1, P(x_1)) \land \ldots \land \varphi(x_n, P(x_n))$ for representative inputs $x_1, \ldots, x_n$.

A program $P$ from the given space of candidates that satisfies $\varphi$ on all (usually bounded) inputs.

$\text{expr} := \text{const} \mid \text{reg6} \mid \text{s4addl}(\text{expr}, \text{expr}) \mid \ldots$

$\text{reg6} \times 4 + 1$

$\text{s4addl}(\text{reg6}, 1)$

CEGIS: Counterexample-Guided Inductive Synthesis

[Solar-Lezama et al, ASPLOS'06]

A syntactic sketch (e.g., a grammar) describing the shape of the desired program $P$.

This defines the space of candidate programs to search. Can be fine-tuned for better performance.
Overview of CEGIS

Searches for an input \( x^i \) on which \( P \) violates \( \varphi \).

Usually a solver, but can be a test suite, end-user, etc.

Any search algorithm: e.g., a solver, enumerative search, stochastic search.

Specification \( \varphi \)

Sketch \( S \)

Form of active learning (a special case of machine learning).

\[ P \in S \text{ s.t. } \wedge_i \varphi(x^i, P(x^i)) \]

\( x^{i+1} \)

Fail

no counterexample

\( P \)
Inductive synthesis with a solver

- Replace each ?? with fresh symbolic constant.
- Translate the resulting problem to constraints w.r.t. the current inputs.
- If SAT, convert the model to a c program P.

Logical encoding of the synthesis problem for the inputs 0, 1, 2.

[Solar-Lezama et al, ASPLOS'06]
Inductive synthesis with enumerative search

- Iteratively construct all programs of size $K$ until one is consistent with the current inputs.
- If two programs produce the same output on all current inputs, keep just one of the two.

$\text{expr ::= } 0 \mid 1 \mid 2 \mid x \mid \text{expr} \ll \text{expr}$

$K=1: 0, 1, 2, x$
$K=2: 1 \ll 2, 2 \ll 2, x \ll 1, x \ll 2$

[Udupa et al, PLDI'13]
Inductive synthesis with stochastic search

- Use Metropolis-Hastings to sample expressions.
- Mutate the current candidate program and keep the mutation with probability proportional to its correctness w.r.t. the current inputs.

A candidate program consistent with current inputs.

[Schkuftza et al, ASPLOS'13]

\[
\text{expr} := 0 | 1 | 2 | x | \text{expr} \ll \text{expr}
\]

\[0, 1, 2\]

\[x * 4\]
Summary

Today

• Deductive synthesis with axioms and E-graphs
• Inductive synthesis with solvers, enumeration, and stochastic search

Next (and final) lecture

• Solver-aided languages