Program synthesis using smoothed numerical search

Swarat Chaudhuri

[Joint work with Armando Solar-Lezama and Martin Clochard]
Smoothed numerical search

• A new family of algorithms for program synthesis

• Combines
  – Numerical optimization
  – Proof search (abstract interpretation)

• Applies to
  – Approximate synthesis
    • “Generate program that meets a spec *approximately*”
  – Synthesis with optimality goals
    • “Generate program with *optimal* performance”
  – Stochastic synthesis
    • “Generate program that meets a *probabilistic* guarantee”
**Synthesis vs. supervised learning**

**Regression**
- ✓ Find a curve that optimally fits a dataset
- ✓ Curve from parameterized family of functions (e.g., polynomials with missing coefficients)
- ✓ Overfitting is bad
- ✗ No notion of proofs

**Synthesis from examples and templates [SRBE05, SGF10, Gul11]**
- ✓ Find a program that fits a set of input/output examples
- ✓ Program from a parameterized family of functions (templates/sketches)
- ✗ Insists on exact fit
- ✓ Additional correctness requirements
But…

• Often, examples need not be followed exactly
  – Quality of output can be negotiable
    • Ok to go to location \( x + \delta \) if asked to go to \( x \)
  – Examples can have uncertainty

• What’s more important:
  – Resulting program fits a simple structural hypothesis

Approximate synthesis, or
Regression with program-shaped models
Problem #1-a: Approximate synthesis
[CS10, CS12, CCS13]

- **Problem instance:**
  - A set of input-output examples \((x_i, y_i)\)
  - Program P with missing parameters c
    - Infinite-state
    - In work so far, functional rather than reactive

- **Goal:**
  - Find parameter value \(v^*\) such that
    \[
    v^* = \arg \min_v \sum_i \|P_{c \rightarrow v}(x_i) - y_i\|
    \]

More generally, minimization of aggregate error

\[
\begin{align*}
  v^* &= \arg \min_v \quad \oplus_x \|P_{c \rightarrow v}(x) - \text{IdealOutput}(x)\|
\end{align*}
\]
**Example: thermostat**

```java
double[] thermostat(double lin, double ltarget) {
    double h = ???(0,10);
    double tOn = ltarget + ???(-10,0);
    double tOff = ltarget + ???(0,10);
    boolean isOn = false;
    ...
    double curr = lin;
    double[] Traj;
    for(int i=0; i < 40; i=i+1){
        if(isOn) {
            curr = curr + (h – K * (curr – lin));
            if(curr > tOff) { isOn = 0.0; }
        } else {
            curr = curr – K * (curr – lin);
            if(curr < tOn) { isOn = 1.0; }
        }
    }
    Traj = curr::Traj;
}
return Traj;
}
```
Examples (init\textsubscript{i}, Θ)

- init\textsubscript{i} is a pair <initial temp, target temp>
- Θ: temperature sequence <75, ..., 75>
- Traj\textsubscript{i}: actual temperature sequence when system starts at init\textsubscript{i}

\[
\text{argmin}_{t_{\text{off}}, t_{\text{on}}, h} \sum_i \| Traj_i - \Theta \|
\]
Problem #1-b: Optimal synthesis
[BCHJ09, CS10, CS12]

- Applicable beyond synthesis from examples
- **Synthesis with quantitative objective** $\Omega$
  
  $$v^* = \arg\min_v \max_x \|\Omega(P_{c \rightarrow v}(x))\|$$

  Potentially infinite, possibly uncountable input space

- **Example**: $\Omega$ could quantify resource consumption
  - Symbolic complexity, energy use
Generalization to \textit{probabilistic} program inputs

\begin{itemize}
  \item \textbf{Problem instance:} \\
    \begin{itemize}
      \item Probabilistic assumption on program inputs $x$ 
        \begin{itemize}
          \item Discrete or continuous distribution 
        \end{itemize}
      \item Program with missing parameters 
    \end{itemize}
  \item \textbf{Goal:} \\
    \begin{itemize}
      \item Find parameter value $v^*$ that minimizes expected objective
    \end{itemize}
\end{itemize}

$$v^* = \arg\min_v \left\| \mathbb{E}[\Omega(P_{c\to v}(x))] \right\|$$
Adding proof goals

“Theorems are forever”
— Moshe Vardi’s webpage

• Two kinds of proofs
  – Proof of \textit{sanity}
    • “Synthesized program meets boolean assertions, postconditions”
  – Proof of \textit{optimality}
    • “Synthesized program is provably optimal”

• Compromise
  – Guarantee of sanity
  – \textit{Interval bounds} on error
Problem #2-c: Stochastic synthesis with proofs [CCS13]

• Problem instance:
  – Probabilistic assumption on program inputs x
    • Discrete or continuous distribution
  – Program with missing parameters
  – Probabilistic safety property $\Phi$
    • “At location L, ($y > 0$) with probability > 0.9”

• Goal: Find parameter value $v^*$ that
  – Minimizes expectation of objective
    \[
    v^* = \arg\min_v ||\mathbb{E}[\Omega(P_{c \rightarrow v}(x))]||
    \]
  – $P_{c \rightarrow v^*}(x)$ satisfies $\Phi$
Example: thermostat

- **Input assumption:**
  - Initial temperature:
    - trimodal, bell-shaped with modes 30, 35, 50; Spread ±3
  - Target temperature:
    - unimodal, bell-shaped: mode 75; Spread ±1

- **Proof goal:**
  - \( \Pr[\text{curr} < 120] > 0.9 \) is an invariant

- **Optimization goal:**
  \[
  \arg\min_{t_{\text{Off}}, t_{\text{On}}, h} \left( \mathbb{E}_x [|| \text{Traj} - \Theta(\text{lin}) ||] \right)
  \]
The rest of the talk: Algorithms

1. Program smoothing

2. An abstract-interpretation-based approach to smoothing [CCS13]


Starting point: local optimization

$x_0, x_1, x_2, \ldots$ $\rightarrow$ $\text{Error}(\vec{x})$

argmin($\text{Error}(x_i)$)

Local Optimizer

- Gradient descent
- Conjugate descent
- Nelder-mead simplex search
- ...

$F(x, y)$
Local search is difficult for programs

**Reason:** Plateaus and discontinuities

Plot of error function for Thermostat for fixed $h$
Symbolic program smoothing
[CS10,CCS13]

Approximate a program by a smooth function
Using symbolic program analysis techniques
Key idea: program smoothing

- Add probabilistic perturbation to program inputs
- Execute on a distribution instead of a value
- Taking expectation smooths discontinuities

Program

\[ P(x) = \begin{cases} 1 & \text{if } x > 2 \\ 0 & \text{otherwise} \end{cases} \]

The spread \( \beta \) of the noise controls the extent of smoothing

“Smoothing parameter”

\[ P(x) = \begin{cases} \text{if } x > 2 \text{ then } 1 \text{ else } 0 \end{cases} \]
**Smoothed numerical search: Outer loop**

Progressively smaller \( \beta_0, \beta_1, \beta_2, \ldots \)

\( \vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots \)

Local optimization

Smooth approximation of program

Smooth(\( \beta, P \))

Note: no need for a closed-form representation of Smooth(\( \beta, P \))

Being able to evaluate is enough
Parameter search: Example
Parameter search
Parameter search
Parameter search
Parameter search
Smoothed search frameworks

• Smoothed search is a **family** of algorithms
• Many parameters you can instantiate
  – The local search method
    • Gradient descent, conjugate descent, Nelder-Mead...
  – The specific smoothing transform
    • Different noise distributions

• **What about proof search?**
  – Proofs explored via local search too!
  – Landscape consists of all possible...
  – To aid exploration, smooth...
The rest of the talk: Algorithms

1. Program smoothing
2. An abstract-interpretation-based approach to smoothing [CCS13]

...in the context of stochastic synthesis of real-number programs with proof goals

High-level view

\[ \beta \text{ multiplied by a constant } < 1 \text{ in each iteration} \]

\[ \beta_0, \beta_1, \beta_2, \ldots \]

\[ \vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots \]

Local optimization

Nelder-Mead search

Smooth(\( \beta, P \))

Probabilistic abstract interpretation

Progressively smaller
Running a program on distributions

if (x > 0)

Gaussian

Truncated Gaussian

Approximate using probabilistic abstract interpretation

Tracks sets of distributions
Abstracting probability distributions: Attempt #1 [Monn00]

• **Abstract state** = Set of components \((E_i, w_i)\) where
  – \(E_i\) is a support set
  – \(w_i\) is an upper bound on total measure

• We use *ellipsoidal supports*
  – Could use polytopes too

• **Concretization** = set of finite positive measures
Probabilistic abstract interpretation: Branch

Minimum volume enclosing ellipsoid

if ( B )

true

false
Probabilistic abstract interpretation: Join

Merging needed for scalability
Widening for convergence
Probabilistic abstract interpretation

On termination, sound bounds on
- Expected output
- Probability of assertion failure
...in addition to smoothing?

Appeal: Proofs for free
Problem: Nonsmoothness

Before perturbation:

Discontinuous change to expected output

Culprit: measures generated at branches

After perturbation:
Attempt #2: Tracking “proportions” [CCS13]

- **Abstract state:** Components numbered 1,...,k
- For each component, tuple \((E_i, w_i, p_i)\)
  - \(E_i\) is ellipsoidal support
  - \(\sum_i w_i = 1\)
  - **Proportion** \(0 \leq p_i \leq 1\)
    - When 0, component has measure 0
    - When 1, component has measure \(w_i\)
    - Otherwise, between 0 and \(w_i\)
Proportion-based abstract domain: Branch

\[
\begin{align*}
E_1 & \quad w_1 \\
E_2 & \quad p_2/2 \\
E_3 & \quad w_3 \\
\end{align*}
\]

\[
\begin{align*}
\neg B & \quad p_1 \\
B & \quad w_2 \\
\end{align*}
\]

if( B )

true

false
Proportion-based abstract domain: Join

Proportions track depth of branches

Theorem: soundness
The case of the disappearing ellipsoid

Before perturbation, along branch $B = \text{true}$:

After perturbation, along branch $B = \text{true}$:

Result: discontinuity
Solution: Track “Distance from disappearance” [CCS13]

Additional information  $0 \leq \alpha_i \leq 1$ for each ellipsoid
- $\alpha_i$ a smooth, monotone function of ellipsoid volume
- $p_i = 0$ if and only if $\alpha_i = 0$
\[ \alpha_1' = \alpha_1 \cdot 1 \]
\[ \alpha_3' = \alpha_3 \cdot 0 \]
\[ \alpha_2' = \alpha_2 \cdot \rho (Volume(E_2|B)) \]
Smooth join

if( B )

true

false

$\alpha_1 = 0.5$

$E_1$

$\alpha_1 = 1.0$

$E_1$

Center of gravity
Smooth join

$E_1$  \hspace{1cm} \alpha_1 = 0.5$

if ($B$)

true

false

$\alpha_1 = 1.0$

$E_1$

Adjust volume and position
Smooth join

$E_1$, $\alpha_1 = 0.5$

if (B)
  true
  $E_1$
  $\alpha_1 = 1.0$
  $E_1$
  false

But also, unsoundness!

Theorem: Smoothness of abstract interpretation
- Requires distance measure over abstract states

Min. volume enclosing ellipsoid
Achieving soundness: Note the $\beta$ in $\rho_\beta$

- $\beta$ is the smoothing parameter!
- $\rho_\beta$ is a smooth function of $\beta$
- As $\beta$ approaches 0, $\rho_\beta$ approaches a *discrete switch*
  - 1 if ellipsoid has any volume at all
  - 0 otherwise
- Hence, at limit, the proportion-based abstract domain!
  - Soundness!

“Start smooth, end sound”
“Local search over proof-like structures”
Thermostat: Errors from different starting points

(a) Nelder-Mead + smoothing

(b) Nelder-Mead (no guarantees)
Why “dovetail” proof and optimization

(a) Our algorithm;
(b) “Guess and check”: find parameters using Nelder-Mead, then verify using proportion-based analysis
Why dovetail proof and optimization (contd.)

“Guess” using smoothing without proofs [CS12]; Check using proportion-tracking sound abstraction
Related work

• Quantitative synthesis of finite-state programs
  – Chatterjee et al [CHJS10], Bloem et al [BCHJ09]

• Invariant generation using machine learning
  – Guess-and-check
    • Nori, Aiken, Sharma et al [SGH+13]
    • Even standard optimization approaches can go a long way
  – Program verification by probabilistic inference
    • Gulwani and Jojic, 07

• Probabilistic program analysis
  – [Monn00], [CM12]
Conclusion

• **Local optimization** can be a powerful tool in synthesis/verification

• **Challenge**: How to combine optimization with program analysis and proof search

• Simpler strategies may work in some cases
  – Off-the-shelf optimization/learning techniques
  – “Guess and check”

• But for at least some domains, you need more
  – Close coupling between optimization + proof search

This is only the first step!
Thank you!
Questions?