Expectation-Oriented Framework for Automating Approximate Programming

Jongse Park, Kangqi Ni, Xin Zhang, Hadi Esmaeilzadeh, Mayur Naik
Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology
Approximate Programming

Programmer’s manual/explicit specification

[ErerJ PLDI’11, Rely OOPSLA’13]

AUTOMATE approximate programming

Where? How much?
ExpAX Overview

Source Code

Approximation Safety Analysis

Approximate Operations Selector

Error and Energy Analyzer

Expectation Checker

Source Code

Expectation

programming Analysis Optimization

Approximate Program
1. accept rate(v) < c
e.g. rate(v) < 0.2
2. accept magnitude(v) < c using f
   e.g. magnitude(v) < 0.1
3. accept magnitude(v) > c using f with rate < c'
   e.g. magnitude(v) > 0.9 with rate < 0.3

Programmer’s Annotations with Expectation
Approximation Safety Analysis

Find possible **safe-to-approximate variables**

**Unsafe-to-approximate variables**
1. Variables violating memory safety
2. Variables violating functional correctness
Approximation Safety Analysis

Backslicing Analysis

For each variable $v$ in program, find all variables contributing to the variable $v$

unsafe-to-approximate variables should be precise

Everything else should be precise variables
```c
void edgeDetection(Image &src, Image &dst) {
    grayscale(src);
    for (int y = ...)
        for (int x = ...)
            dst[x][y] = sobel(window(src, x, y));
    accept rate(dst) < 0.1;
}

Float sobel (float[3][3] p) {
    float x, y, gradient;
    x = (p[0][0] + 2 * p[0][1] + p[0][2]);
    x += (p[2][0] + 2 * p[0][1] + p[2][2]);
    y = (p[0][2] + 2 * p[1][2] + p[2][2]);
    y += (p[0][0] + 2 * p[1][1] + p[2][0]);
    gradient = sqrt(x * x + y * y);
    ...
    return gradient;
}
```

Example
Optimization

Find a subset of safe-to-approximate operations
• Minimize error
• Maximize energy saving

Objective function

\[ f(subset) = \left( \alpha \times error + \beta \times energy \right)^{-1} \]

Genetic algorithm

*phenotype*: a bitvector representing a subset (approximate(‘0’) or precise(‘1’))
Statistical Guarantee

Find a subset of best phenotype!

For each eval in genetic algorithm:
calculate a score for each operation

\[ f(\text{operation}) = \sum_{\text{eval} \in \text{Eval}} \left( \frac{\alpha \times \text{error} + \beta \times \text{energy}}{n(\text{approx})} \right) / n(\text{Eval}) \]
Space Exploration with Transformed Best Phenotype

sort w.r.t scores

Best phenotype

evaluate test inputs
Evaluation

Benchmarks:
scimark2 – FFT, LU, SOR, MonteCarlo, SMM Imagefill, raytracer, jmeint, zxing

Simulator:
Open-source simulator provided by EnerJ
## Analysis Result

<table>
<thead>
<tr>
<th>BenchName</th>
<th>Enerj: # of Annotations</th>
<th>ExpAX: # of Expectations</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>LU</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>SOR</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>MonteCarlo</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SMM</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>imagefill</td>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>RayTracer</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>jmeint</td>
<td>113</td>
<td>1</td>
</tr>
<tr>
<td>zxing</td>
<td>172</td>
<td>15</td>
</tr>
</tbody>
</table>
Genetic Algorithm Results

LU on aggressive system specification
Error

- fft
- sor
- mc
- smm
- lu
- zxing
- jmeint
- imagefill
- raytracer

Exp=0.00
Exp=0.03
Exp=0.05
Exp=0.10
Exp=0.50
Exp=0.75
All Approx
Conclusion

Expax: an expectation-oriented framework for automating approximate programming

1. Programming model with a new program specification
2. Approximation safety analysis
3. Optimization framework with heuristics for statistical guarantee