Equality Saturation: A New Approach to Optimization

Ross Tate, Mike Stepp, Zach Tatlock, Sorin Lerner
University of California, San Diego
Traditional Optimization

Phase Ordering Problem

Original Program

Optimized Program

Local Profitability Heuristics

Optimizations
Traditional Optimization
Exploring Equivalences
Exploring Equivalences
Our Approach

CFG → PEG
Conversion

Program Expression Graph
Our Approach

CFG → PEG Conversion

Equality Saturation
Our Approach

Global Profitability Heuristic

CFG → PEG Conversion
Equality Saturation
Our Approach

PEG → CFG Conversion

CFG → PEG Conversion
Equality Saturation
Global Profitability Heuristic
Benefits

- Mitigates Phase Ordering Problem
  - Non-destructive updates allow exponential search
- Global Profitability Heuristic
  - Explore first, then decide
- Translation Validation
  - Verify translations using equality saturation
Example

```c
sum = 0;
for (i = 0; i < 10; i++)
    sum += 4 * i;
return sum;
```
Representing Loops

\[
\text{sum} = 0; \\
\text{for} \ (i = 0; i < 10; i++) \\
\text{sum} += 4 \times i; \\
\text{return sum;}
\]

- Complete Representation
- Referentially Transparent
- No Intermediate Variables
Equality Analyses

- Identify Equalities
- PEG Node Granularity
- Equality Axioms
  - $\forall X. \ 4 \cdot X = (X \ll 2)$

"CFG $\rightarrow$ PEG Conversion"  "Equality Saturation"  "Global Profitability Heuristic"  "PEG $\rightarrow$ CFG Conversion"
Equality Inference

∀X. 4*X = 4*X

∀X, Y. 4*[θ(X, Y)] = θ(4*X, 4*Y)

∀ X, Y, Z. X*(Y + Z) = X*Y + X*Z

∀X. X*0 = 0

∀X. X*1 = X

CFG → PEG Conversion

Equality Saturation

Global Profitability Heuristic

PEG → CFG Conversion
E–PEG

Conversion

Equality Saturation

Global Profitability Heuristic

PEG→CFG Conversion

CFG→PEG Conversion
PEG Selection

Global Profitability Heuristic

CFG → PEG Conversion

Equality Saturation

Global Profitability Heuristic

PEG → CFG Conversion
Optimized PEG

\[ \theta_1 \]

0 +

4

CFG → PEG Conversion

Equality Saturation

Global Profitability Heuristic

PEG → CFG Conversion
sum = 0;
for(j = 0; j < 40; j += 4)
  sum += j;
return sum;
sum = 0;
for(i = 0; i < 10; i++)
    sum += 4 * i;
return sum;

sum = 0;
for(j = 0; j < 40; j += 4)
    sum += j;
return sum;

Loop Induction Variable Strength Reduction
Emergent Optimizations

Optimizations composed from simple rules

- Loop Induction Variable Strength Reduction
- Loop–Operation Factoring
- Loop–Operation Distributing
- Inter–Loop Strength Reduction
- Temporary Object Removal
- Partial Inlining
Implementation
Algorithm provided in the Technical Report
Model heap with values having linear types
Tarjan’s Union–Find Algorithm
  ◦ tracks equivalence classes in the E–PEG

Rete Pattern Matching Algorithm
  ◦ incrementally finds significant nodes in the E–PEG

Equality Analyses:
  ◦ PEG Operator Axioms
  ◦ Language–Specific Axioms
  ◦ Domain–Specific Axioms
Pseudo-Boolean Solver

- Assign a cost to each operation in the E-PEG
- Impose constraints for a well-formed PEG
- Minimize the cost of the selected PEG
Observed Emergent Optimizations
- Traditionally need to be explicitly implemented

Domain–Specific Analyses:
- 7% runtime improvement on Java ray tracer

Compilation of SpecJVM (per method):
- $10^{30}$ programs found in less than 200MB memory
- Average compilation time per stage:
Translation Validator

- Validation of Soot optimizer on SpecJVM:
  - 98% of optimized methods successfully validated
  - Optimization bug found within remaining 2%
Conclusions

- **Powerful**
  - Simultaneous Exponential Search
  - Emergent Optimizations

- **Extensible**
  - Cooperative Equality Analyses
  - Domain–Specific Axioms

- **General**
  - Optimization
  - Translation Validation
Related Work

- **E-Graphs**
  - Denali: Basic Block Assembly Superoptimizer
  - Simplify: Theorem Prover

- **Representations**
  - Thinned–Gated SSA
  - Lucid programming language
  - Value Dependence Graph
  - Dependence Flow Graph
  - Program Dependence Graph/Web

- **Rewrite–Based Optimizers**
  - TAMPR
  - ASF+SDF
  - Stratego