Universal Code Generator

Tim Richards, Eliot Moss, and Chip Weems

University of Massachusetts Amherst
Problems Building Code Generators

- Dependencies
  - Compiler
  - Target

- Hard to Implement
  - Special expertise required
  - Correctness

- Time Consuming & Error Prone
Code Generator Approaches

- **Manual**
  - You write the code

- **Write CG Specification**
  - CG generated from pattern specification

- **Discover CG Specification**
  - Find the pattern specification
  - Generate the CG from the specification

- **Problems**
  - *Still target and compiler dependent*
  - Correctness, solution expert, suboptimal, etc.
Solution

• Target and Compiler Independent
• Description Driven
  • Target instruction semantics
  • Compiler IR semantics
• Automate: as much as possible
  • Search for code generator patterns
  • Ensure correct patterns
• No Special Expertise Required
Solution

• Target and Compiler Independent

• Description Driven
  • Target instruction semantics
  • Compiler IR semantics

• Automate: as much as possible
  • Search for code generator patterns
  • Ensure correct patterns

• No Special Expertise Required

Universal Code Generator Generation
• **Goal:** Given a description of the IR and target semantics, **Find** target instruction sequences that **correctly** implement the source IR.

• **Approach:**
  - **FAST** heuristic search
    - Potential target sequences
  - **FAST** semantic interpretation
    - Candidate target sequences
  - Theorem prover
    - Valid target sequences
• **Goal:**
  
  Given a description of the IR and target semantics, *Find* target instruction sequences that *correctly* implement the source IR.

• **Approach:**
  
  - **FAST** heuristic search
    - Potential target sequences
  - **FAST** semantic interpretation
    - Candidate target sequences
  - Theorem prover
    - Valid target sequences

*Filter obvious bad sequences*
Goal:
Given a description of the IR and target semantics, Find target instruction sequences that **correctly** implement the source IR.

Approach:
- **FAST** heuristic search
  - Potential target sequences
- **FAST** semantic interpretation
  - Candidate target sequences
- Theorem prover
  - Valid target sequences

Slow, OK for small # of sequences
Heuristic Search

• Given:
  • Compiler IR description
  • Target instruction description
• Extract *Features*
  • Input/output
  • Operators
• Guess Target Instructions
  • *Does IR and target do the same thing?*
    • Leverage features
• Result:
  • *Similar features* \(\Rightarrow\) *potential sequence*
Heuristic Search

• Given:
  • Compiler IR description
  • Target instruction description

• Extract Features
  • Input/output
  • Operators

• Guess Target Instructions
  • Does IR and target do the same thing?
    • Leverage features

• Result:
  • Similar features == potential sequence
Heuristic Search

- Given:
  - Compiler IR description
  - Target instruction description
- Extract Features
  - Input/output
  - Operators
- Guess Target Instructions
  - Does IR and target do the same thing?
    - Leverage features
- Result:
  - Similar features == potential sequence
Heuristic Search

• Given:
  • Compiler IR description
  • Target instruction description

• Extract Features
  • Input/output
  • Operators

• Guess Target Instructions
  • Does IR and target do the same thing?
    • Leverage features

• Result:
  • Similar features == potential sequence
Semantic Interpretation
Semantic Interpretation

- Given:
  - Potential target sequence
  - IR description
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description

- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector

Diagram:
- Test Vector → Semantic Interpreter
- IR → Equivalent States?
- Target Sequence
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description

- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector

Test Vector

$S(IR)^0$

IR

Target Sequence
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description
- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector
- **Map Memories**
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description

- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector

- **Map Memories**

- **Initialize States**

Test Vector

\[ \text{map} \]

\[ S(\text{IR})^0 \rightarrow S(\text{Target})^0 \]
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description
- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector
- **Map Memories**
- **Initialize States**
Semantic Interpretation

- **Given:**
  - Potential target sequence
  - IR description

- **Semantic Interpreter**
  - IR/Target semantics
  - Test vector

- **Map Memories**
- **Initialize States**

![Diagram](image)
Semantic Interpretation

- Given:
  - Potential target sequence
  - IR description
- Semantic Interpreter
  - IR/Target semantics
  - Test vector
- Map Memories
- Initialize States
- Compare Resulting States
Semantic Interpretation

- Given:
  - Potential target sequence
  - IR description
- Semantic Interpreter
  - IR/Target semantics
  - Test vector
- Map Memories
- Initialize States
- Compare Resulting States
- Result:

\[ S(IR)^0 \xrightarrow{map^{-1}} S(Target)^0 \]
\[ S(IR)^1 \xrightarrow{=?} S(Target)^1 \]
\[ S(IR)^{map} \xrightarrow{map} S(Target)^0 \]
Semantic Interpretation

- Given:
  - Potential target sequence
  - IR description
- Semantic Interpreter
  - IR/Target semantics
  - Test vector
- Map Memories
- Initialize States
- Compare Resulting States
- Result:
  - Identical states $\Rightarrow$ Candidate Sequence
Theorem Prover

- Given:
  - Candidate sequence
  - IR

- Prove
  - The target correctly implements the source

- Rigorous Methods
  - Combination
  - Term rewriting systems, boolean satisfiability, etc.

- Result:
  - identical states == valid sequence
Theorem Prover

- **Given:**
  - Candidate sequence
  - IR

- **Prove**
  - *The target correctly implements the source*

- **Rigorous Methods**
  - *Combination*
  - Term rewriting systems, boolean satisfiability, etc.

- **Result:**
  - identical states $\equiv$
    valid sequence

---

TRS, SAT, etc.  

$\downarrow$

Equivalent Semantics?
Theorem Prover

- **Given:**
  - Candidate sequence
  - IR
- **Prove**
  - The target correctly implements the source
- **Rigorous Methods**
  - Combination
  - Term rewriting systems, boolean satisfiability, etc.
- **Result:**
  - identical states == valid sequence
Universal Code Generator Generation

Source

Front-end

IR

Code Generator

Target Code
Universal Code Generator Generation

- IR/Target Descriptions
- Search
- Interpret Semantics
- Prove
- Code Generator Patterns
- Generate Code Generator

Diagram:
- Source
- Front-end
- IR
- Code Generator
- Target Code

Steps:
1. IR Description
2. Target Description
3. Universal Code Generator Generator
4. Search
   - Potential Sequences
5. Interpret
   - Candidate Sequences
6. Theorem Prover
   - Valid Sequences
   - CG Patterns

Test Vectors

Flow:
- IR Description → Target Description → Universal Code Generator Generator → Search → Potential Sequences → Interpret → Candidate Sequences → Theorem Prover → Valid Sequences → CG Patterns → Code Generator → Target Code
CoGenT Project

CoGenT

Compiler Tools

Simulator Tools

Universal Code Generator Generation
CoGenT Project

CoGenT

Compiler Tools

Simulator Tools

Universal Code Generator Generation
Conclusion

- **Universal Code Generator Generation**
  - IR and Target Independent
  - Generates Correct CG
  - No Special Expertise Required
  - Automates CG Construction
Status

- Semantic Language Completed
- Description Compiler Completed
- Heuristic Search Prototype
- Semantic Interpreter Prototype
- Theorem Prover
  - TRS library implemented
  - SAT Generator library implemented
- Passed Thesis Proposal Exam
- *More to Come Soon!*
Thank You

Questions?

Tim Richards
richards@cs.umass.edu
http://www.cs.umass.edu/~richards