C Code Conversion to One Counter Automata for Reachability Analysis

Lars Van Roy
dept. of Mathematics and Computer Science
University of Antwerp
lars.vanroy@student.uantwerpen.be

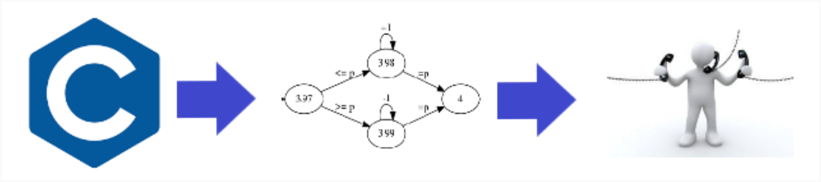
April 2020

Overview

Problem Overview

C code conversion

- Analyse code reachability
- Convert C code to counter automaton
- Code restrictions



Related Work

Related work I

On parametric timed automata and one-counter machines
Daniel Bundala, Joel Ouaknine, 2017
Information and Computation

Programs with Lists Are Counter Automata
Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, TomÃąÅą Vojnar, 2006
Computer Aided Verification

Flat Counter Automata Almost Everywhere!
JÃľrÃťme Leroux, GrÃľgoie Sutre, 2005
Automated Technology for Verification and Analysis

Related work II

Quantum Finite One-Counter Automata
Kravtsev, Makism, 1999
SOFSEM'99: Theory and Practice of Informatics

Motivation

Motivation

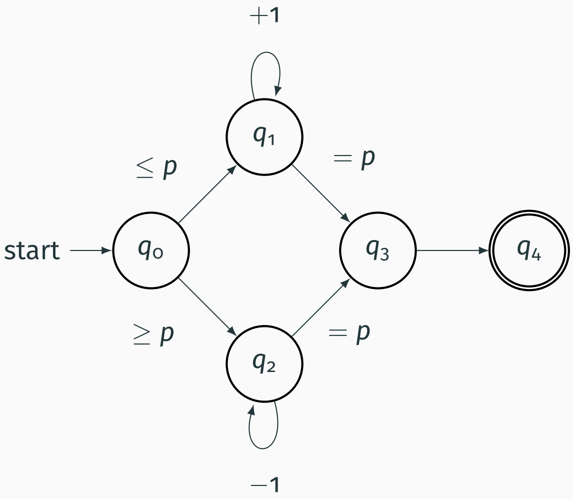
- Resource constrained environments
- Code efficiency
- Development time

One Counter Automata

Counter automata I

- Non-Deterministic Finite Automata with counter(s)
- limited set of operations
    - increment the counter with parameter/constant
    - compare the counter to parameter/constant
- transition relation $\sigma \subseteq Q \times op \times Q$

Counter automata II

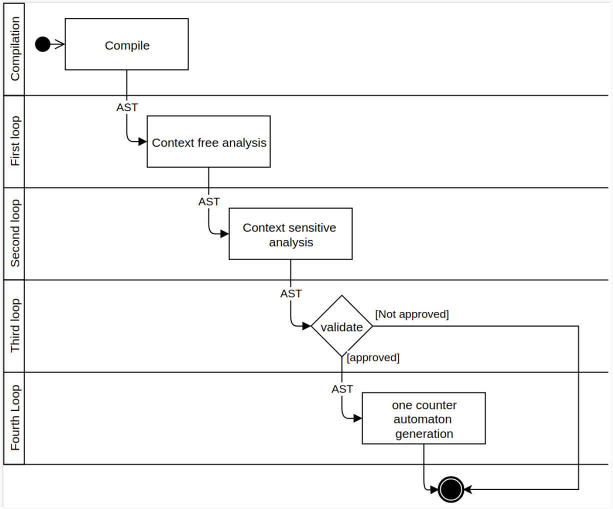


Counter automata III

- Advantages
    - Simple conversion
    - Known proven reachability solution
        - PSPACE-complete
        - 2NEXP time
- Disadvantages
    - Enforced restrictions
        - Counters
        - Operations

C code converter Tool

Tool overview



Compilation

- ANTLR compilation
- Generates AST
- Boolean extension

First loop

- Context free optimization
- Remove redundant chains of nodes
- Unify node structures
- Enables improvements in further loops

Second loop

- Main cleaner loop
- Symbol table
- Constant folding/substitution
- Dead code elimination
- Counter tracking
- No folding and/or substitution in conditional branches

Third loop

- validation loop
- checks counter generation conditions
    - Counter overlap
    - Counter type(s)
    - Return type
    - Parameter types
    - Parameter modification
    - Used operations on counter
    - Used conditional statements with counter

Fourth Loop

- Counter automaton generation
- Simple loop
- One node per line
- Transitions
    - Empty if no counter related operation occurs
    - Condition if a conditional statement with counter occurs
    - Operation if an operation on the counter occurs

Conclusion

Conclusion

- Improvement to current algorithms
- Cost in execution time
- Need for additional operation support
- Suited for resource constrained environments
- $(l + 3) * m$ loops
    - m = number of nodes in AST
    - l = number of lines in c code

Future work

Future work

- The reachability algorithm for the counter automata
- Additional supported operations
- Comparisons with other reachability algorithms