Relational String Verification Using Multi-track Automata

Relational String Analysis

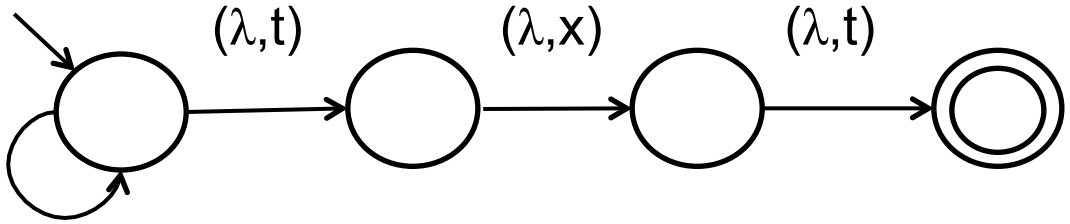
- Earlier work on string analysis use multiple single-track DFAs during symbolic reachability analysis
  - One DFA per variable per program location
- Alternative approach: Use one multi-track DFA per program location
  - Each track represents the values of one string variable

- Using multi-track DFAs:
  - Identifies the relations among string variables
  - Improves the precision of the path-sensitive analysis
  - Can be used to prove properties that depend on relations among string variables, e.g., $file = $usr.txt

Multi-track Automata

- Let X (the first track), Y (the second track), be two string variables
- λ is the padding symbol
- A multi-track automaton that encodes the word equation:

Y = X.txt



(a,a), (b,b) …

Alignment

- To conduct relational string analysis, we need to compute union or intersection of multi-track automata
  - Intersection is closed under aligned multi-track automata
    - In an aligned multi-track automaton λs are right justified in all tracks, e.g., abλλ instead of aλbλ

- However, there exist unaligned multi-track automata that are not equivalent to any aligned multi-track automata
  - Use an alignment algorithm that constructs aligned automata which over or under approximates unaligned ones
    - Over approximation: Generates an aligned multi-track automaton that accepts a super set of the language recognized by the unaligned multi-track automaton
    - Under approximation: Generates an aligned multi-track automaton that accepts a subset of the language recognized by the unaligned multi-track automaton

Symbolic Reachability Analysis

- Transitions and configurations of a string system can be represented using word equations
- Word equations can be represented/approximated using aligned multi-track automata which are closed under intersection, union, complement and projection
- Operations required for reachability analysis (such as equivalence checking) can be computed on DFAs

Word Equations

- Word equations: Equality of two expressions that consist of concatenation of a set of variables and constants
  - Example: X = Y . txt
- Word equations and their combinations (using Boolean connectives) can be expressed using only equations of the form X = Y . c, X = c . Y, c = X . Y, X = Y. Z, Boolean connectives and existential quantification
- Our goal:
  - Construct multi-track automata from basic word equations
    - The automata should accept tuples of strings that satisfy the equation
  - Boolean connectives can be handled using intersection, union and complement
  - Existential quantification can be handled using projection

Word Equations to Automata

- Basic equations X = Y . c, X = c . Y, c = X . Y and their Boolean combinations can be represented precisely using multi-track automata
- The size of the aligned multi-track automaton for X = c . Y is exponential in the length of c
- The nonlinear equation X = Y . Z cannot be represented precisely using an aligned multi-track automaton

Word Equations to Automata

- When we cannot represent an equation precisely, we can generate an over or under-approximation of it
  - Over-approximation: The automaton accepts all string tuples that satisfy the equation and possibly more
  - Under-approximation: The automaton accepts only the string tuples that satify the equation but possibly not all of them
- We can implement a function CONSTRUCT(equation, sign)
  - Which takes a word equation and a sign and creates a multi-track automata that over or under-approximation of the equation based on the input sign

Post condition computation

- During symbolic reachability analysis we compute the post-conditions of statements using the function CONSTRUCT

  Given a multi-track automata M and

  an assignment statement: X := sexp

  Post(M, X := sexp) denotes the post-condition of X := sexp with respect to M

  Post(M, X := sexp)

  = (∃ X , M ∩ CONSTRUCT(X’ = sexp, +))[X/X’]