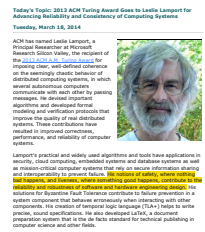
Graph Theory and Program Analysis

GRAPH-THEORETIC METHODS IN DATABASE THEORY

Mihalis Yannakakis

Secure programming via game-based synthesis

William Harris



Today's Topic: 2013 ACM Turing Award Goes to Leslie Lamport for Advancing Reliability and Consistency of Computing Systems

Tuesday, March 18, 2014

ACM has named Leslie Lamport, a Principal Researcher at Microsoft Research Silicon Valley, the recipient of the 2013 ACM A.M. Turing Award for imposing clear, well-defined coherence on the seemingly chaotic behavior of distributed computing systems, in which several autonomous computers communicate with each other by passing messages. He devised important algorithms and developed formal modeling and verification protocols that improve the quality of real distributed systems. These contributions have resulted in improved correctness, performance, and reliability of computer systems.

Lamport's practical and widely used algorithms and tools have applications in security, cloud computing, embedded systems and database systems as well as mission-critical computer systems that rely on secure information sharing and interoperability to prevent failure. His notions of safety, where nothing bad happens, and liveness, where something good happens, contribute to the reliability and robustness of software and hardware engineering design. His solutions for Byzantine Fault Tolerance contribute to failure prevention in a system component that behaves erroneously when interacting with other components. His creation of temporal logic language (TLA+) helps to write precise, sound specifications. He also developed LaTeX, a document preparation system that is the de facto standard for technical publishing in computer science and other fields.

Graph Theory

- Problems
  - transitive closure, shortest paths, bill of materials, critical paths, regular expressions
- Algorithms
  - Kleene's alg. for regular expressions
  - Floyd's alg. for shortest paths
  - Warshall's alg. for transitive closure

Matrix Multiply on Semiring

- A closed semiring
  - 0 and 1 exists
  - product x is associative, closed under finite products
  - sum + is associative/commutative, closed under finite sums
  - x distributes over +
- A graph
  - 0 is no path, 1 is null path
  - sum combines multiple paths
  - product extends a path

$$\textbf{for } k := 1 \textbf{ to } n \textbf{ do}$$
$$\quad \textbf{for all } i,\ j \textbf{ do}$$
$$\quad\quad A_{ij}^{k} := A_{ij}^{k-1} + A_{ik}^{k-1} \times (A_{kk}^{k-1})^{*} \times A_{kj}^{k-1}$$

Reachability

- Problem
  - Aij = 1 iff there is a path from i to j
- Solution
  - sum is ||, product is &&
  - initial Aij = 1 if ij connected; otherwise 0
- Explanation
  - connectivity including node k is the connectivity with node 1 to k-1, plus the connectivity from i to k, and plus the connectivity from k to j
- Transitive closure can then be computed easily

$$\textbf{for } k := 1 \textbf{ to } n \textbf{ do}$$
$$\quad \textbf{for all } i,\ j \textbf{ do}$$
$$\quad\quad A_{ij}^{k} := A_{ij}^{k-1} + A_{ik}^{k-1} \times (A_{kk}^{k-1})^{*} \times A_{kj}^{k-1}$$

Database Queries

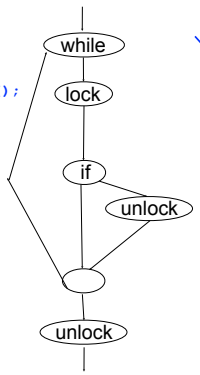
- Relational databases
  - "directed" hypergraphs, with labelled edges
  - a path "spells" a word
  - an L-path if the spelled word is in the language L
- Recursive queries
  - need to compute the transitive closure
  - not expressible in relational algebra/calculus
  - extensions in datalog and graph-oriented query languages
- Datalog language
  - example $p(X,Y) :- q_0(X,Z_1),\ q_1(Z_1,Z_2),\ldots,\ q_k(Z_k,Y),$
  - variables, predicates, recursion

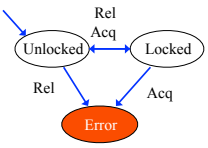
L-path

- A specification L
  - a regular expression
    - e.g. lock acquire/release/error DFA in SLAM
    - e.g. even-length paths
  - a context-free grammar
    - e.g. legal interprocedural paths in optimization
- A path satisfies L is an L-path
- SLAM
  - is there an L-path that reaches the error state anywhere in code?
    - is it an MOP problem?
- Interprocedural analysis
  - what is MOP invariance at every point?

Lock Safety

```
do {
   KeAcquireSpinLock();
   if(*){
        KeReleaseSpinLock();
   }
} while (*);
KeReleaseSpinLock();
```





RE specification?

Type State Analysis (one slide!)

languages. That is, we construct a graph H whose nodes are pairs (s,u) consisting of a state s of M and a node u of G, and which has an arc labelled a from a node (s,u) to another node (t,v) if M has a transition on letter a from state s to state t and G has an arc from u to v. (Actually, the labels in the product graph are not important.) Let s_0 be the initial state of M and F its set of accepting states. Then, the database graph G contains an L-path from a node x to a node y iff (s_0,x) can reach in the product graph a state (t,y) with $t \in F$.

- A state space H for lock-safety analysis
  - (prog point, lock status)
  - (s,u) -> (t,v) if s->t and u->v
  - problem: if (start, unlocked) can reach any (s, error) in H
  - single-source transitive closure

Program Analysis as Queries

- Queries
  - proposition logic
    - SQL
  - recursive
    - regular
      - e.g. lock safety spec
    - context-free
      - e.g. datalog
- Type of queries
  - existential: does something exist?
  - universal: does the property always hold?

UNIVERSITY OF COPENHAGEN — Department of Computer Scienc

Type interpretation

Definition (Type interpretation)

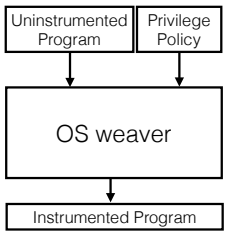
The *type interpretation* $\mathcal{T}[\![.]\!]$ compositionally maps a regular expression E to the corresponding simple type:

$\mathcal{T}[\![0]\!]$	$=$	$\emptyset$	empty type
$\mathcal{T}[\![1]\!]$	$=$	$\{()\}$	unit type
$\mathcal{T}[\![a]\!]$	$=$	$\{a\}$	singleton type
$\mathcal{T}[\![E+F]\!]$	$=$	$\mathcal{T}[\![E]\!]+\mathcal{T}[\![F]\!]$	sum type
$\mathcal{L}[\![E\times F]\!]$	$=$	$\mathcal{T}[\![E]\!]\times\mathcal{T}[\![F]\!]$	product type
$\mathcal{T}[\![E^*]\!]$	$=$	$\{[v_1,\ldots,v_n] \mid v_i \in \mathcal{T}[\![E]\!]\}$	list type

http://www.cs.ox.ac.uk/ralf.hinze/WG2.8/27/slides/fritz.pdf

Secure programming via game-based synthesis

William Harris



URCS Department Seminar, March 2014

Weaver **PL** results

- Automatically reproduced manually-written programs and automatically wrote new ones from **small, declarative policies that forbid known vulnerabilities**
- Reduced weaving to game-based synthesis, and proposed and evaluated **a novel symbolic game-solving algorithm [CAV '12]**

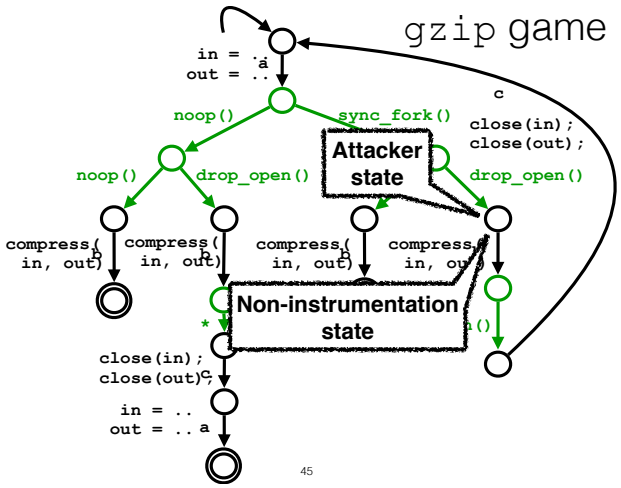
gzip on Capsicum

Server

```
gzip(argv):
   foreach (nm in argv):
IN:  in = open(nm);
OUT: out = open(nm + ".gz");
     sync_fork();
     drop_open();
     compress(in, out);
     sync_join();
     close(in);
     close(out);
```

At `compress`, `gzip` should have exactly the the last descriptors allocated at `IN` and `OUT`

gzip game



Weaver **systems** results

- Internal adoption: have engaged **both Capsicum and HiStar developers**
- External adoption: used by **DARPA** evaluation team to rewrite secure **PHP** interpreter as layer of prototype secure software stack
- Future goal: release Capsicum weaver to **`capsicum-dev`**

Today's Topic: 2013 ACM Turing Award Goes to Leslie Lamport for Advancing Reliability and Consistency of Computing Systems

Tuesday, March 18, 2014

ACM has named Leslie Lamport, a Principal Researcher at Microsoft Research Silicon Valley, the recipient of the 2013 ACM A.M. Turing Award for imposing clear, well-defined coherence on the seemingly chaotic behavior of distributed computing systems, in which several autonomous computers communicate with each other by passing messages. He devised important algorithms and developed formal modeling and verification protocols that improve the quality of real distributed systems. These contributions have resulted in improved correctness, performance, and reliability of computer systems.

Lamport's practical and widely used algorithms and tools have applications in security, cloud computing, embedded systems and database systems as well as mission-critical computer systems that rely on secure information sharing and interoperability to prevent failure. His notions of safety, where nothing bad happens, and liveness, where something good happens, contribute to the reliability and robustness of software and hardware engineering design. His solutions for Byzantine Fault Tolerance contribute to failure prevention in a system component that behaves erroneously when interacting with other components. His creation of temporal logic language (TLA+) helps to write precise, sound specifications. He also developed LaTeX, a document preparation system that is the de facto standard for technical publishing in computer science and other fields.

Program Security

- Security = Safety + Liveness
- OS weaver
  - safety
    - i.e. 'compress' cannot have certain privileges
  - liveness
    - i.e. 'compress' can still be used correctly
    - Xiaochen's question at the talk
- Reachability analysis
  - can check for safety
  - can it insert code as OS Weaver can?
  - what about liveness?