THE UNIVERSITY OF TEXAS AT AUSTIN

Department of Computer Sciences

Using Data-flow Analysis to Improve the Scalability of Model Checking

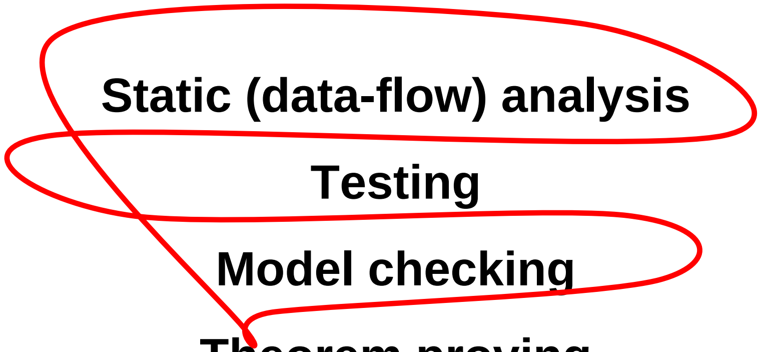
Adam Brown, James C. Browne and Calvin Lin

The University of Texas at Austin

This work has been supported by the National Science Foundation under grant number 0509354 , “Collaborative Research: CSR–AES: Unification of Verification and Validation Methods.”

Project Goal

Unification of Verification and Validation Methods


Static (data-flow) analysis

Testing

Model checking

Theorem proving

Runtime Verification

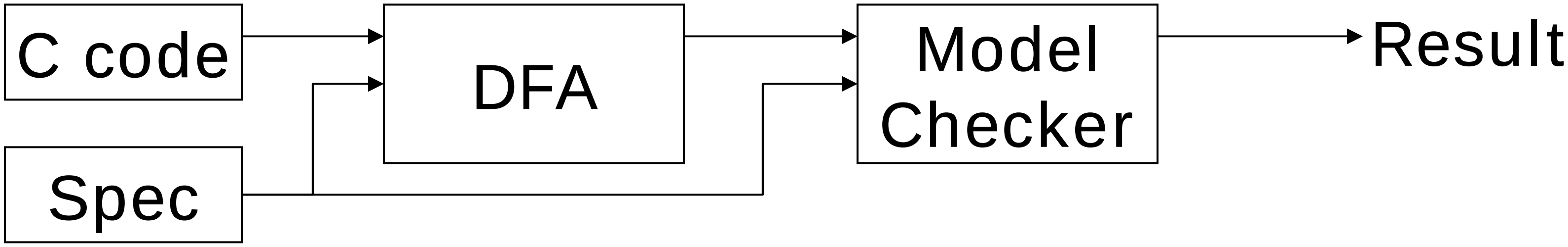
Model checking in one slide

- Method for proving properties of systems
- Suffers from state-space explosion
  - State-space grows exponentially
  - Does not scale for large software systems
- How do you reduce the state-space?

Use data-flow analysis (DFA)

□ Less precise but more efficient method for proving properties

□ We use DFA to improve model checking



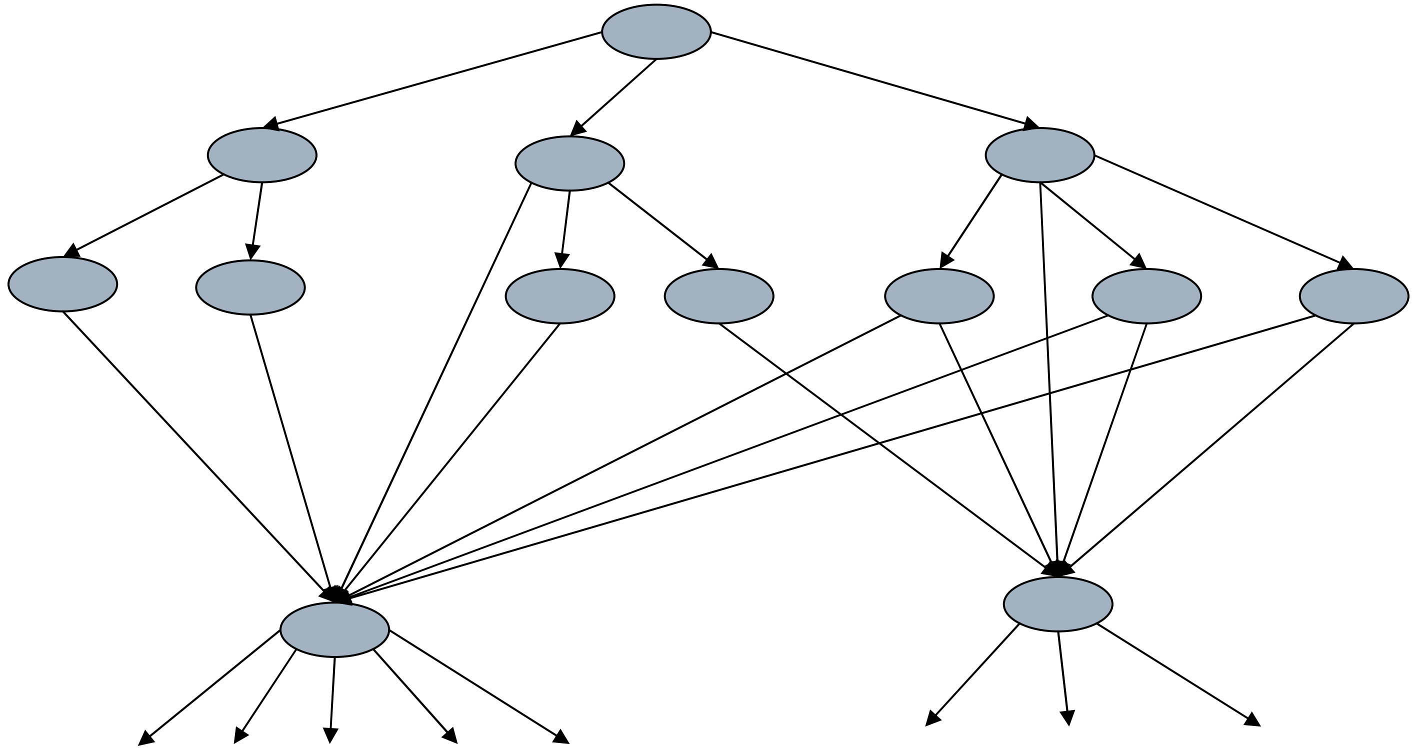
Improving model checking with DFA

- DFA has a number of approximation techniques
  - Context-insensitive analysis
  - Flow-insensitive analysis
  - Path-insensitive analysis
- Approximations improve performance of an analysis

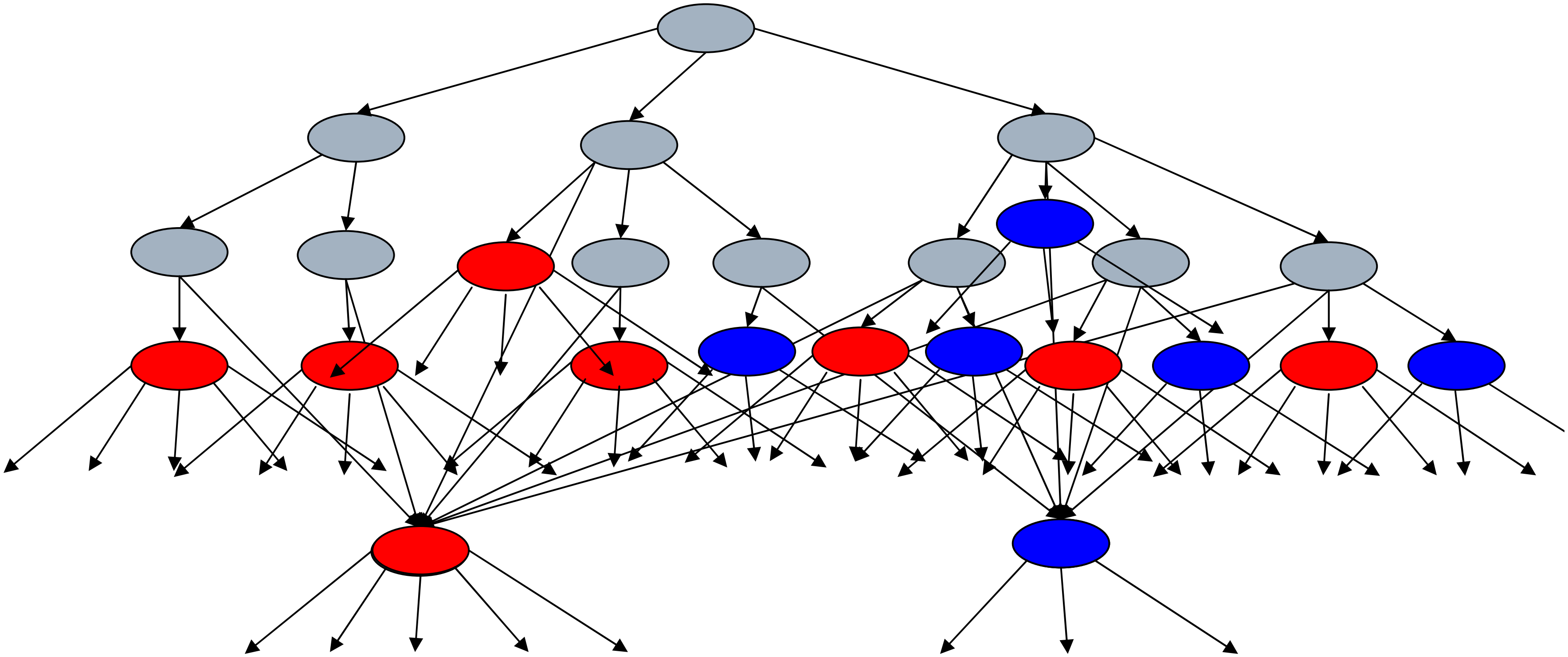
Context-sensitivity

- Context-sensitive
  - Analyze procedure for each call
- Context-insensitive
  - Analyze each procedure once
  - Merge info from all procedure calls

A simple callgraph



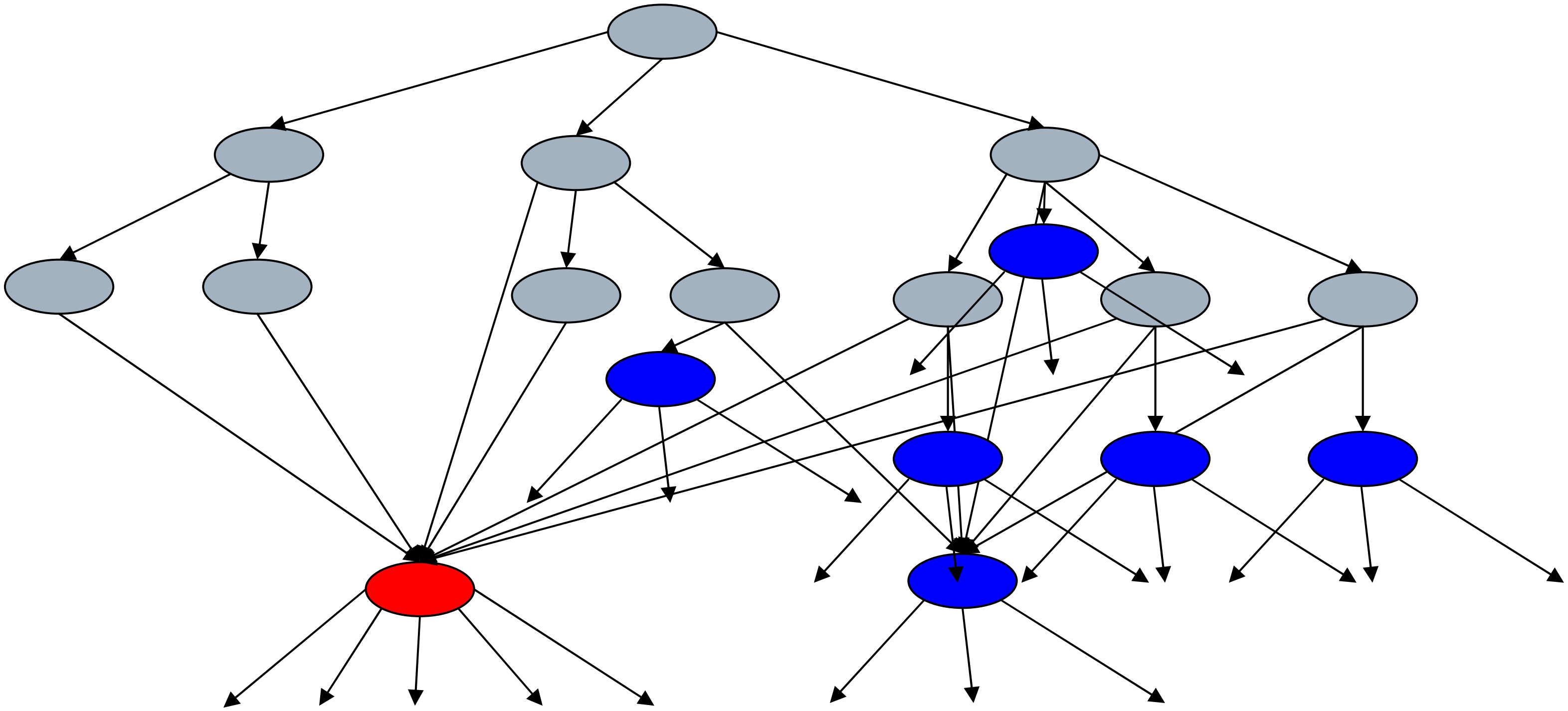
Context-sensitive



Key idea

- □ Identify when context-sensitivity is needed
- □ Use adaptive analysis [Guyer & Lin 2003]
  - ■ Quick, imprecise analysis
  - ■ Track where precision is necessary

With adaptive analysis



Evaluation

- Measured size of invocation graph
  - Indication of resulting model size
- C code programs
  - ~10Ks lines of code
  - 41 to 959 procedures and library routines
- Results from one security analysis
  - FTP behavior analysis

Motivating results

	Context-Insensitive	Context-Sensitive	Adaptive	Reduction
pfingerd	43			
muh	41			
blackhole	959			
named	311			

Insight

- Two-orders reduction in invocation graph size
  - Upper-bound because DFA adds unrealizable paths

Status & Future Work

- ☑ Translating abstracted program to model checker
- □ Understand relationship with other control abstractions
  - Partial-order reductions
  - Slicing
- □ Comprehensive integration between DFA and model checking

Questions

Model checking in one slide

- Completely determines satisfiability
- Performs an exhaustive search
- Can generate huge state-space from simple model

- Research focused on reducing state-space by over-abstracting the model

Data-flow analysis in one slide

- Determines safe, but incomplete, solution
- Iteratively solves flow equations
- Quickly converges in practice

- Adaptive analysis can identify where effort should be exerted

Best of both worlds

- Model checking and DFA are two sides of the same coin
- Completeness of model checking
- Scalability of data-flow analysis

Statement location results

	context-insensitive	adaptive	context-sensitive	reduction
pfinger	150	150	24361	162x
muh205	157	157	30114	191x
bind	1273	2061	>1449996	>703x
blackhole	3865	5265	>819997	>155x

Current status

- Implementing analysis for C programs
  - Using Broadway/C-Breeze compiler
  - Initial phase limited to typestate problems
- Output model to SPIN model checker
- Handles recursion

Conclusion

- Reduce by two orders of magnitude
  - Without other reduction techniques
- No loss of accuracy in the model check result