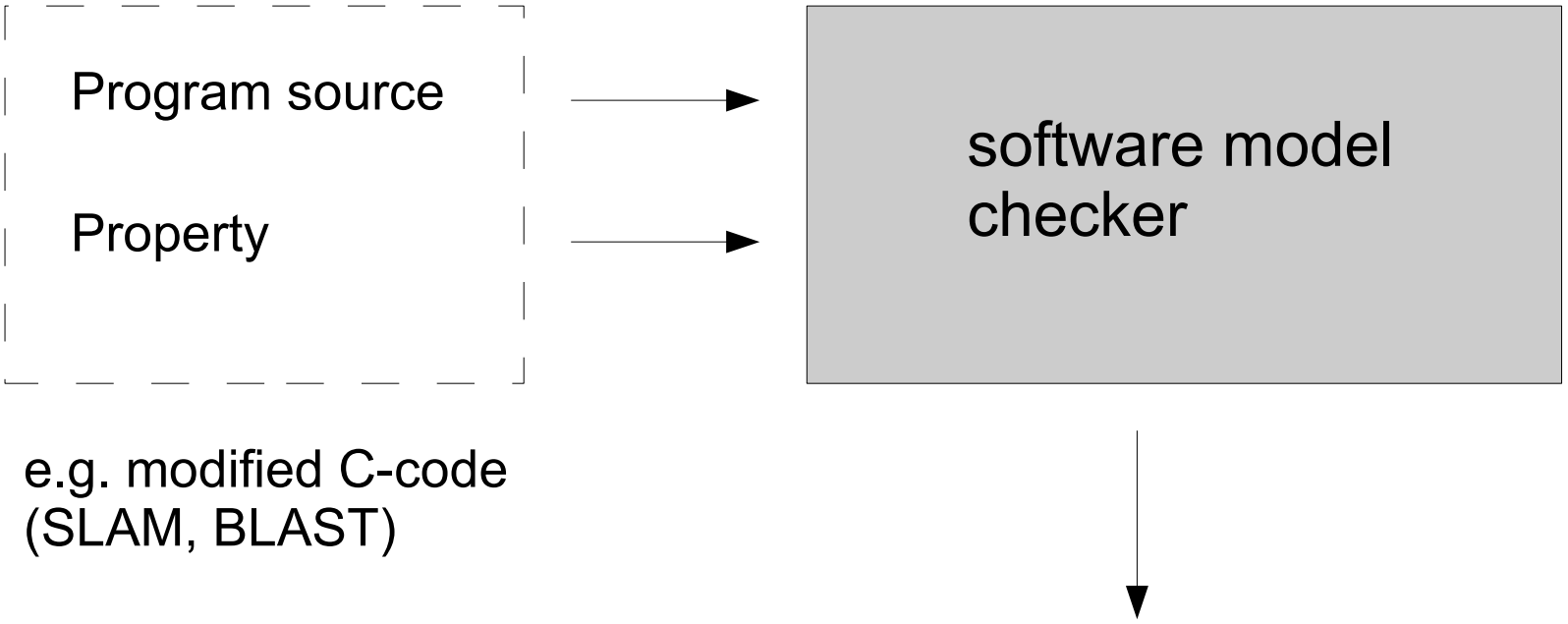
The Software Model Checker BLAST: Applications to Software Engineering

Dirk Beyer, Thomas A. Henzinger,
Ranjit Jhala, Rupak Majumdar

Int. Journal on Software Tools for
Technology Transfer, 2007

Stefan Buchholz
March 17, 2009

Model checking



e.g. modified C-code
(SLAM, BLAST)

- Valid according to the specification
- Error path

Model checking

Goal

Find a path from an initial state to an error state (reachability)

Problem

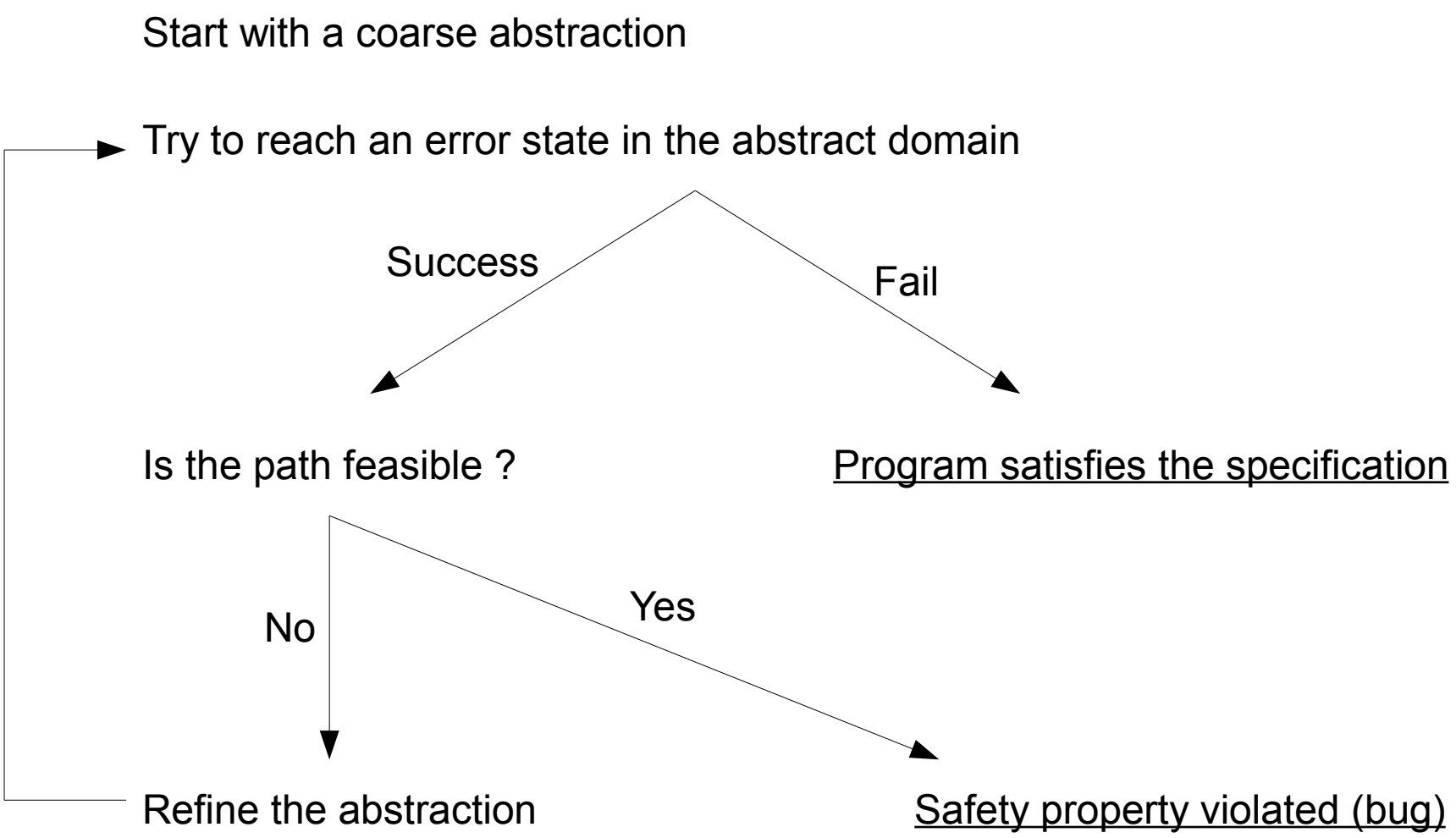
State explosion ! Infeasible to check every state.

BLAST's solution

Dynamic (lazy) predicate abstraction

CEGAR

Counterexample-guided abstraction refinement



Optimizations

Lazy predicate abstraction

- don't refine error-free regions

Interpolation-based predicate discovery

- add predicates locally when refining

Applications

Reduce runtime memory-checks

- Test their reachability and remove unnecessary ones

Automatic test generation

- For any counterexample found try to generate a test vector

BLASTing Linux Code

Demonstration

Open source !

http://mtc.epfl.ch/software-tools/blast/

Limitations

False alarms (can't show infeasability of a path)

- Mathematical integers (no overflows modeled)
- Decision functions only implement linear arithmetic
- Uninterpreted functions (e.g. bit-level manipulations)
- Simple pointer arithmetic and alias-handling

Weak predicate language

- No logical quantifiers
- No precise reasoning about recursive functions possible

Conclusion

+ Good performance and scalability

- Limitations make it impractical for a lot of real world applications (pointer arithmetic !)

- Manually modifying the source code is always required

BACKUP

Data structures

Control flow automaton (CFA)

- Internal representation of a program
- Nodes are control points, edges are operations

Abstract reachability tree (ART)

- Represents a portion of the reachable state space
- A path in the ART corresponds to a program execution
- Each node stores a link to a CFA-location, the current call stack and the state of variables

Automatic test generation

Target Predicate Coverage

Can a location be reached and the variables satisfy a predicate p ? e.g. p = true for location coverage

Coverage of a location L

Test if L is reachable (dead code detection)

When a feasible path is found generate a variable assignment that guides it to the location

Embarrassingly parallel