ELTE

Thematic Excellence Program

Industry and Digitalisation

Application Domain Specific Highly Reliable IT Solutions

# Static source code analysis and manipulation of Erlang programs

Melinda Tóth and István Bozó

Department of Programming Languages and Compilers,
Eötvös Loránd University (ELTE)
{toth_m, bozo_i}@inf.elte.hu

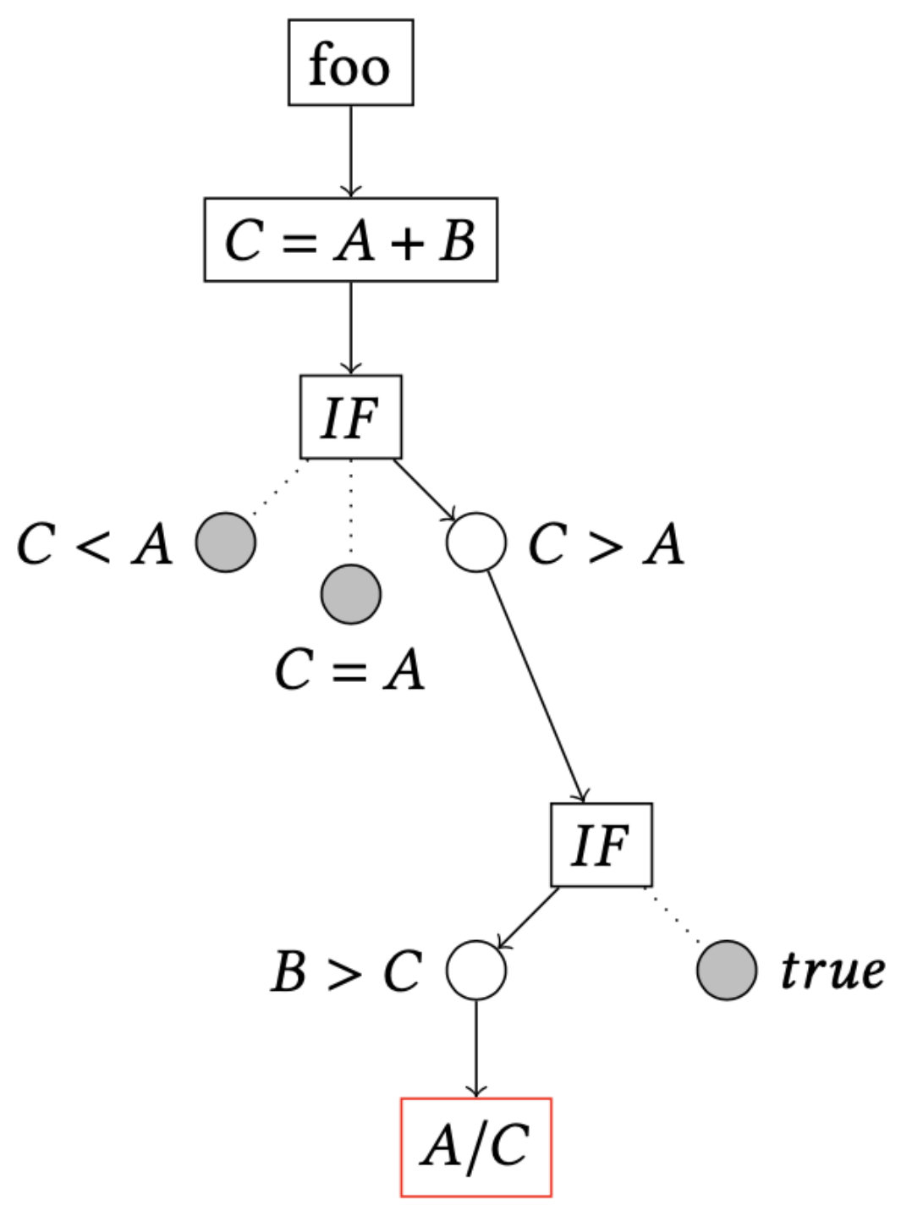
## RefactorErl

- Compile-time analyses designed for Erlang:
  - Lifetime, scope, visibility, reference analyses for semantic entities (functions, records, variables, etc.)
  - Side-effects, hidden dependencies
  - Data-flow, control-flow, dynamic function call
- Main features are
  - Understanding legacy code
  - Refactoring/Application restructuring
  - Code checking: complexity, quality, style, vulnerability, custom properties

## Code Checking with RefactorErl

Secure coding

- Find non-intentional software vulnerabilities in Erlang
  - Interoperability mechanism related vulnerabilities
  - Concurrent programming related issues
  - Distributed programming related issues
  - Injection
  - Memory overload related attacks
- How it works?
  - Determines the function call locations which are associated with unsecure operations.
  - Selects the functions parameters that can be associated with potential vulnerabilities.
  - Runs data-flow analysis on the sensitive parameters.
  - Flags parameters with unknown source.
  - Filters out functions provided by the users for input validation.
- Next improvements:
  - Optimizations and selection heuristics
  - False positive result reductions
  - New checkers

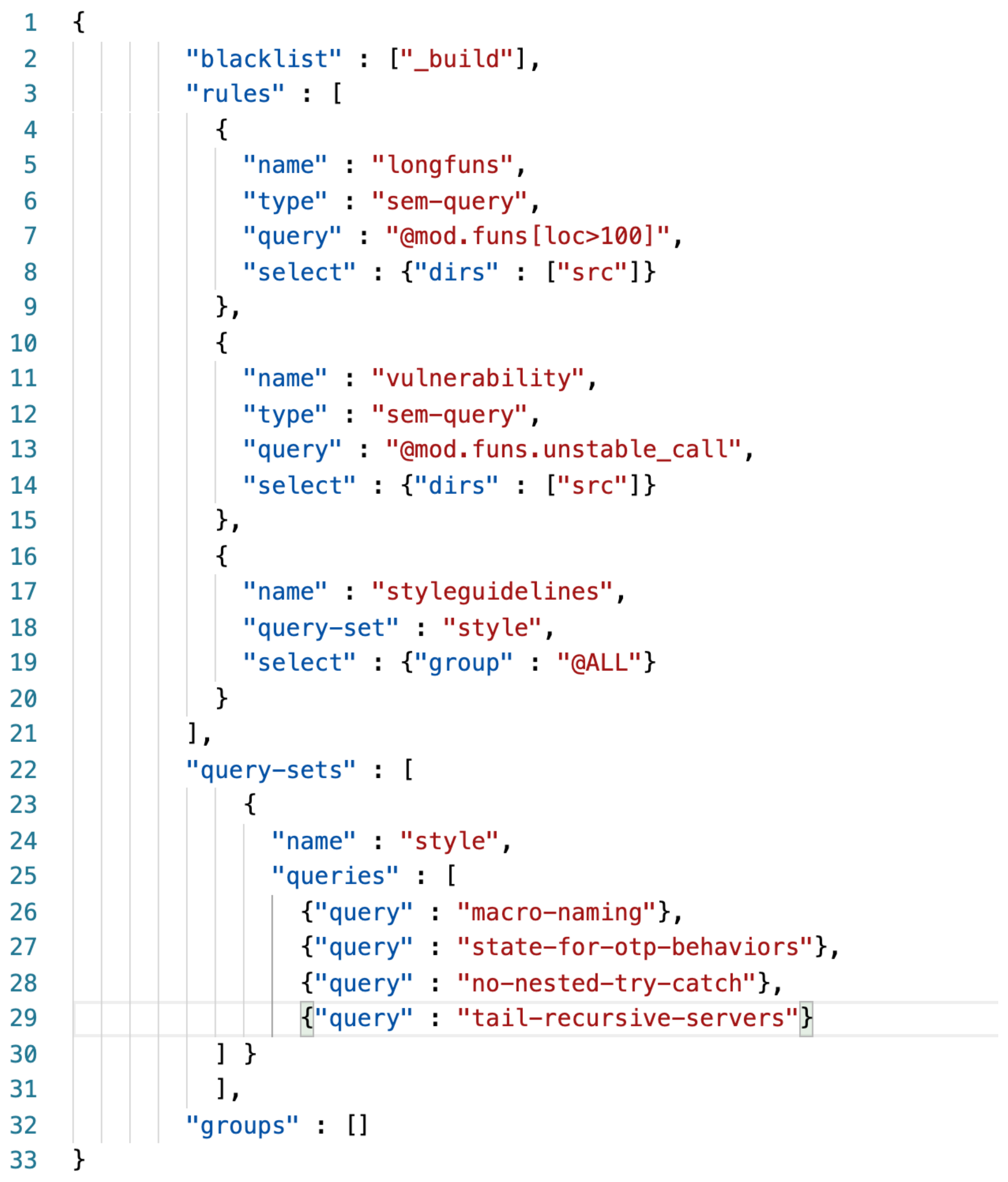


Code checking

- Through the Semantic Query Language
  - Helps also in debugging, grokking, learning
  - Built-in + custom
  - Works with the units of the language
  - `mods.funs.unstable_calls`
  - `mods.funs.unsecure_compile_operations`
- New DRC client
  - Focuses on automatic code checking
  - Easy to integrate and use
  - Input config
  - Connects to a running RefactorErl server
  - Custom output

Finding sources of runtime errors

- Control-flow based static execution paths selection
- Combined with direct symbolic execution
- Using the Z3 SMT solver on the generated constraints
- The runtime error compiled to a constraint
- Suggests execution path that leads to a runtime error



```
{
        "blacklist" : ["_build"],
        "rules" : [
          {
            "name" : "longfuns",
            "type" : "sem-query",
            "query" : "@mod.funs[loc>100]",
            "select" : {"dirs" : ["src"]}
          },
          {
            "name" : "vulnerability",
            "type" : "sem-query",
            "query" : "@mod.funs.unstable_call",
            "select" : {"dirs" : ["src"]}
          },
          {
            "name" : "styleguidelines",
            "query-set" : "style",
            "select" : {"group" : "@ALL"}
          }
        ],
        "query-sets" : [
            {
              "name" : "style",
              "queries" : [
                {"query" : "macro-naming"},
                {"query" : "state-for-otp-behaviors"},
                {"query" : "no-nested-try-catch"},
                {"query" : "tail-recursive-servers"}
          ] }
          ],
        "groups" : []
}
```

## Topics

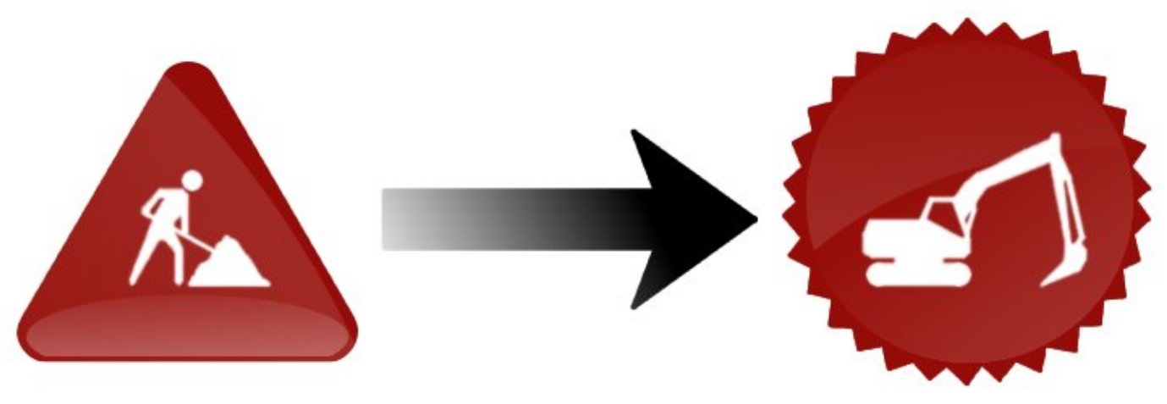
- Semantic queries
- Software complexity metrics
- Bad smell detection
- Duplicated code detection and elimination
- Clustering - software restructuring
- Dependency visualisation
- Secure programming
- Communication/process analyses
- Decompilation
- Pattern candidate discovery and parallelisation
- Program slicing for test case selection
- OTP behaviour analyses
- Distributed software analysis and manipulation
- Improving the “functional style” of the code
- Merging static and dynamic analyses
- Green computing

## Why to use it?

Key benefits for industrial partners

- Shorten learning term of a newcomer
- Shorten bug report solution time
- Make the possibility of a better team work
- Support software delivery product line
- Increase code quality through reducing faults
- Shorten time-consuming daily jobs, such as the source code checking
- Supports secure coding

Effective software maintenance



Application Domain Specific Highly Reliable IT Solutions project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

National Research, Development and Innovation Office
Hungary

PROGRAM FINANCED FROM THE NRDI FUND