Oliver Krauss MSc

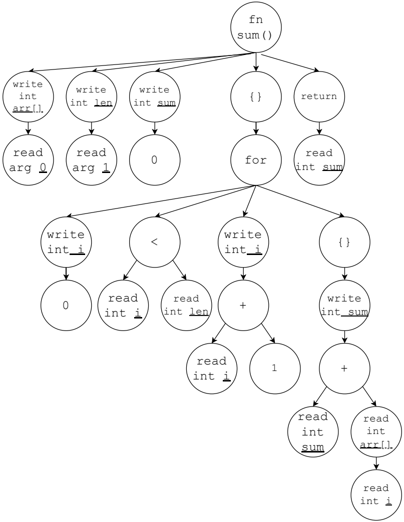
Amaru - A Framework for Combining Genetic Improvement with Pattern Mining (amaru.dev)

Boston / Online 9. July 2022

GI@GECCO 2022

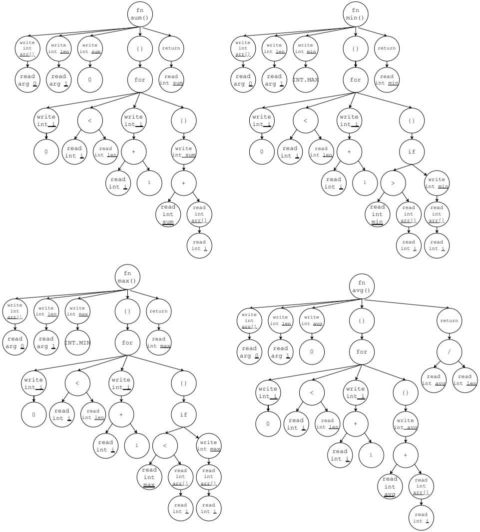
What is a pattern?

Abstract Syntax Tree (AST)



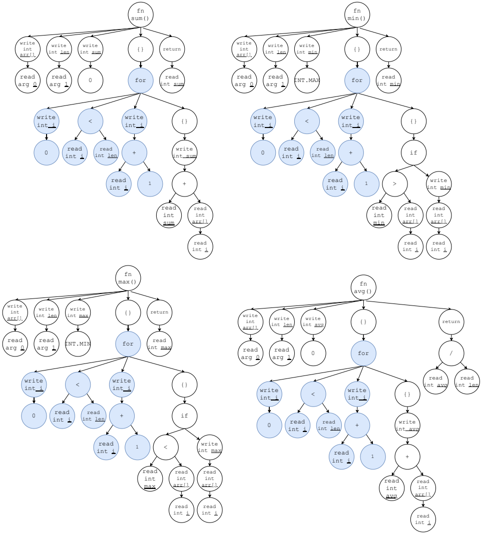
What is a pattern?

Many Abstract Syntax Trees (ASTs)

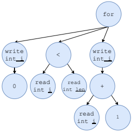


What is a pattern?

Many Abstract Syntax Trees (ASTs)



Frequent **Pattern**



Why Mine Patterns?

JOHANNES KEPLER UNIVERSITÄT LINZ

- Software engineering is challenging

Why Mine Patterns?

- Software engineering is challenging
- More challenging when needing to optimize for **Non-Functional Properties (NFP)**

Why Mine Patterns?

- Software engineering is challenging
- More challenging when needing to optimize for **Non-Functional Properties (NFP)**
- Example: 75% of Software Maintenance costs improve performance or fix bugs [1]

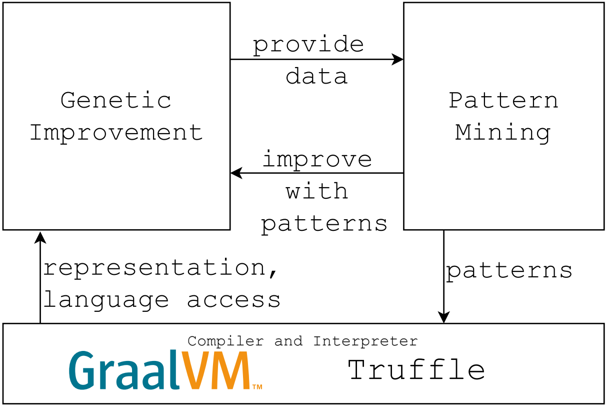
[1] Jussi Koskinen - *Software Maintenance Costs* - 2015

Why Mine Patterns?

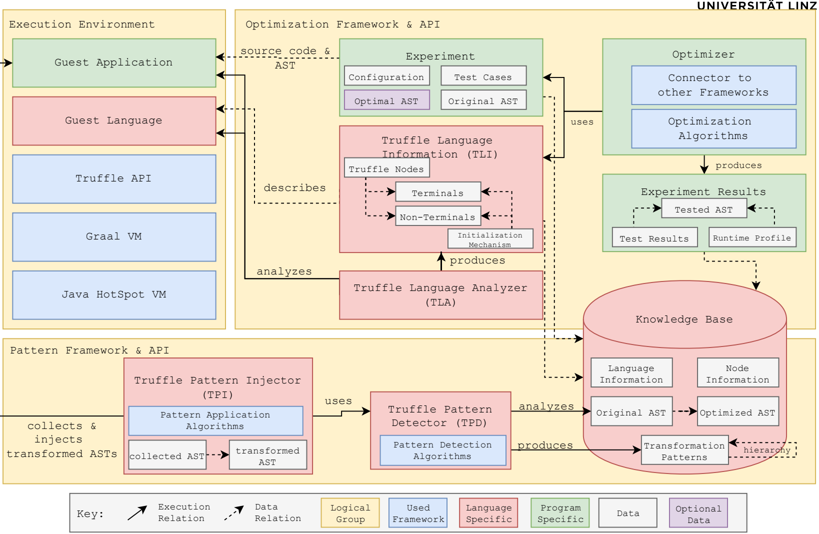
- Software engineering is challenging
- More challenging when needing to optimize for **Non-Functional Properties (NFP)**
- Example: 75% of Software Maintenance costs improve performance or fix bugs [1]
- Goal: Find patterns, validate patterns, use patterns

[1] Jussi Koskinen - *Software Maintenance Costs* - 2015

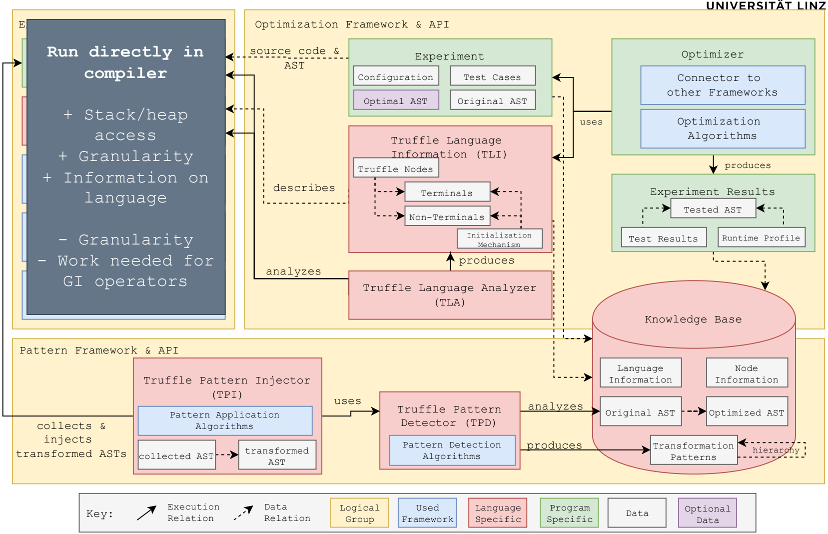
Introduction I



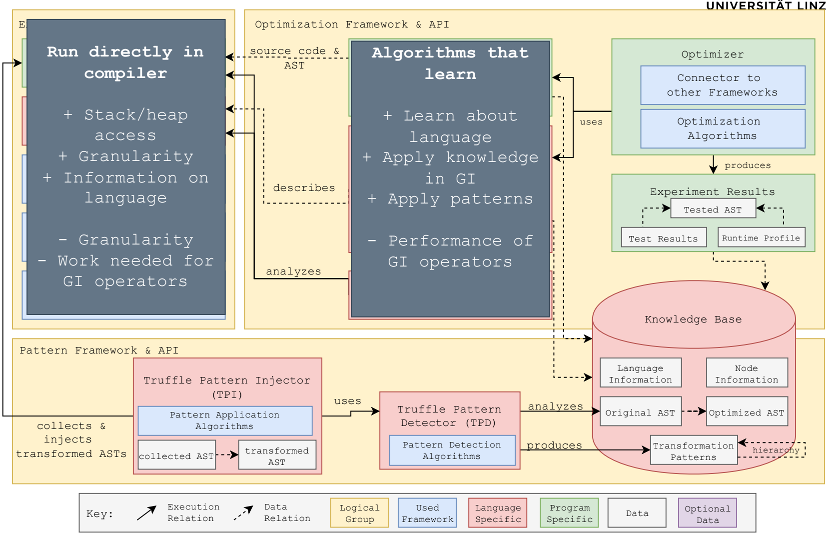
The Amaru Framework



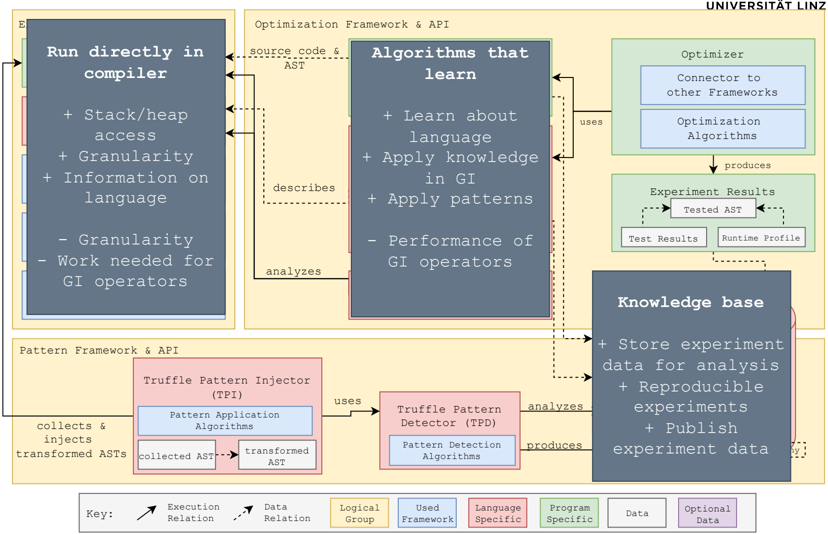
The Amaru Framework



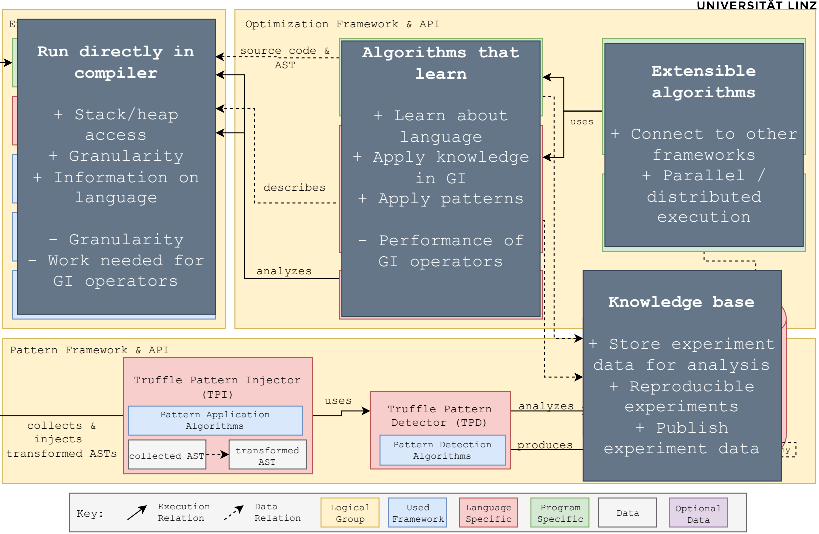
The Amaru Framework



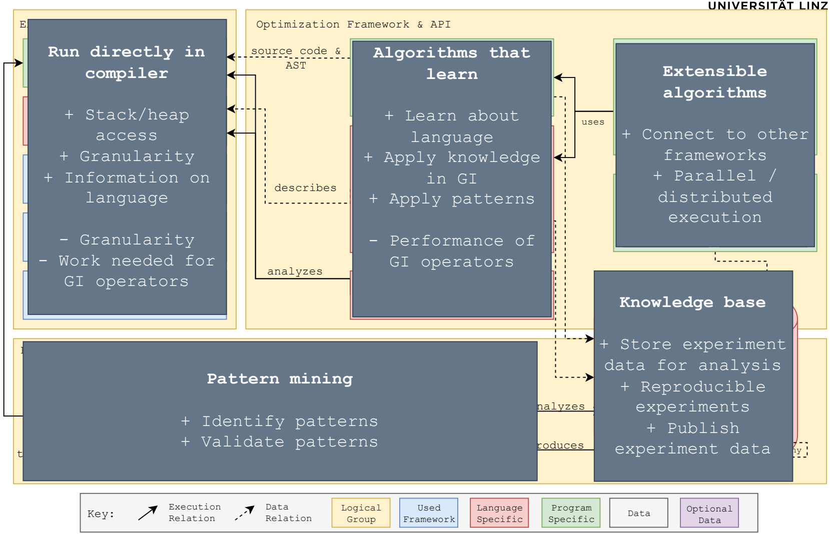
The Amaru Framework



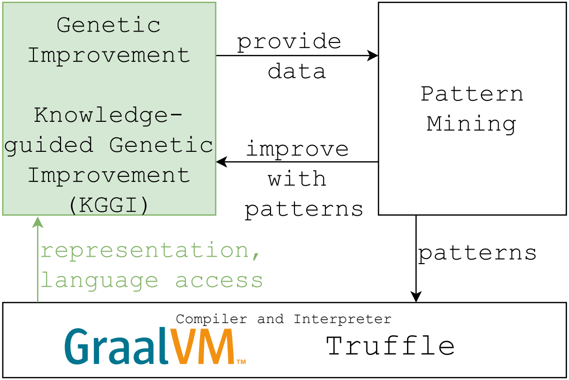
The Amaru Framework



The Amaru Framework

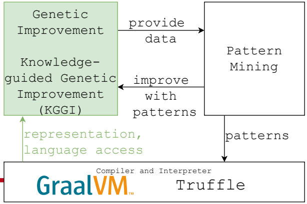


Knowledge-guided Genetic Improvement



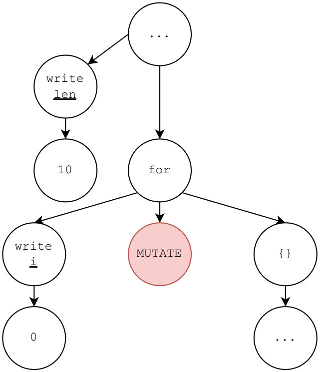
Knowledge-guided Genetic Improvement

- Grammar-guided Genetic Programming
- Tree Genetic Programming
- Enriched with *metadata*
  - loops, branches, NFP, ...
- Operators access *context*
  - stack, heap, functions, ...
- Applies patterns and requirements via *Syntax Graph*

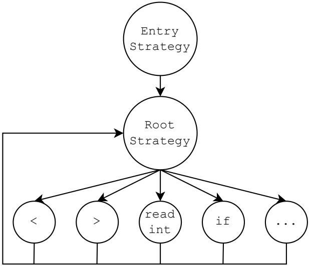


Knowledge-guided Genetic Improvement

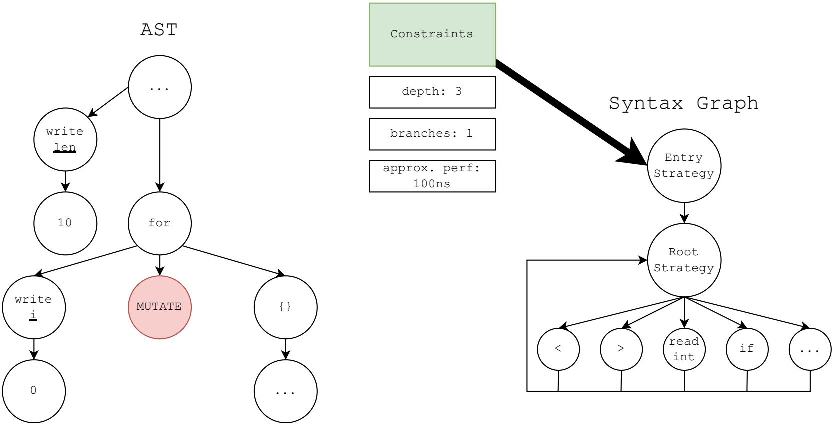
AST



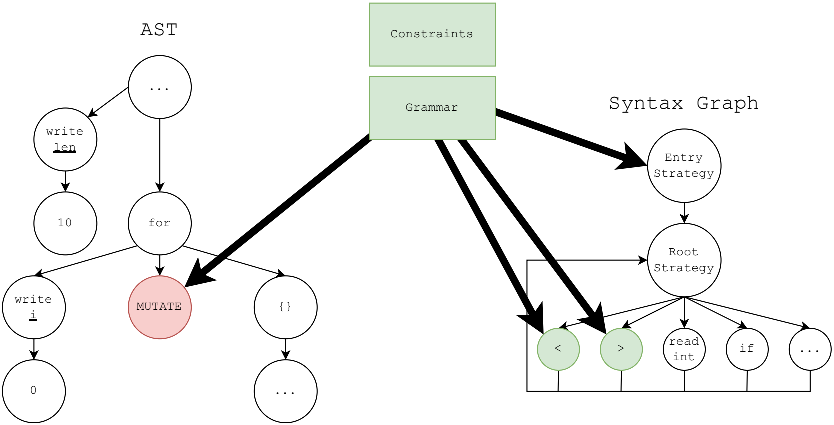
Syntax Graph



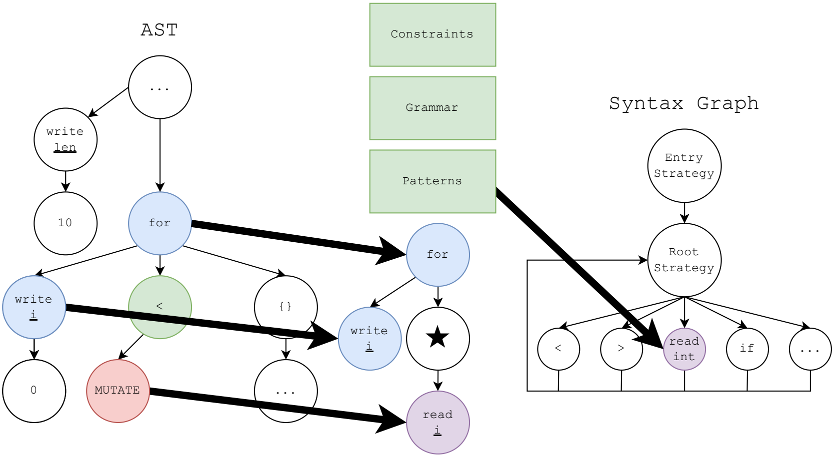
Knowledge-guided Genetic Improvement



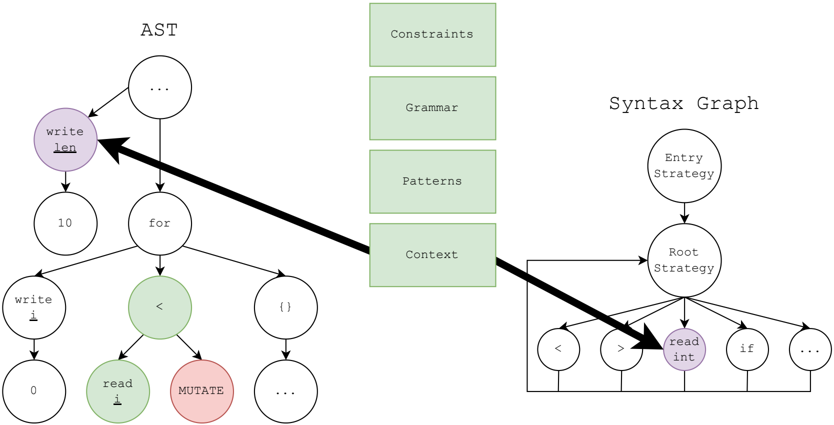
Knowledge-guided Genetic Improvement



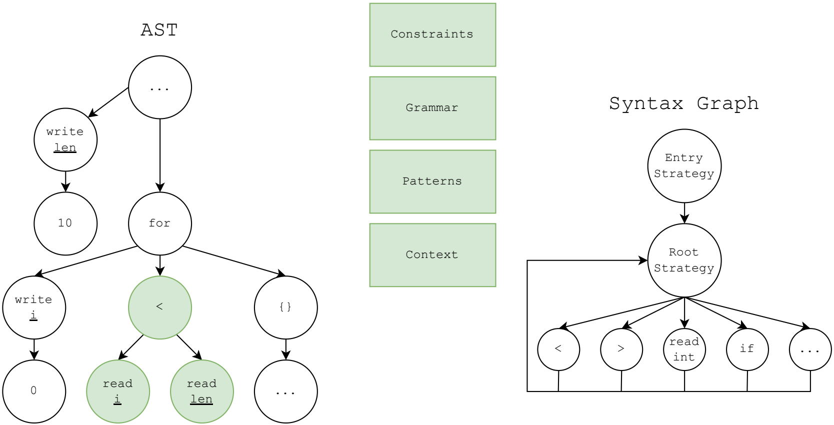
Knowledge-guided Genetic Improvement



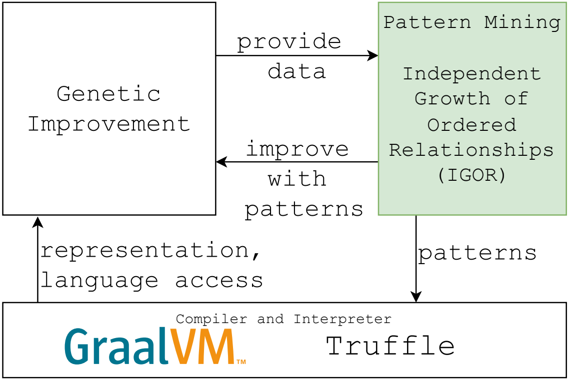
Knowledge-guided Genetic Improvement



Knowledge-guided Genetic Improvement

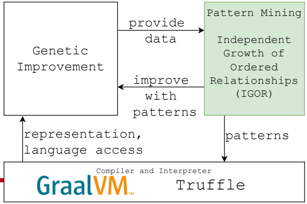


Independent Growth of Ordered Relationships

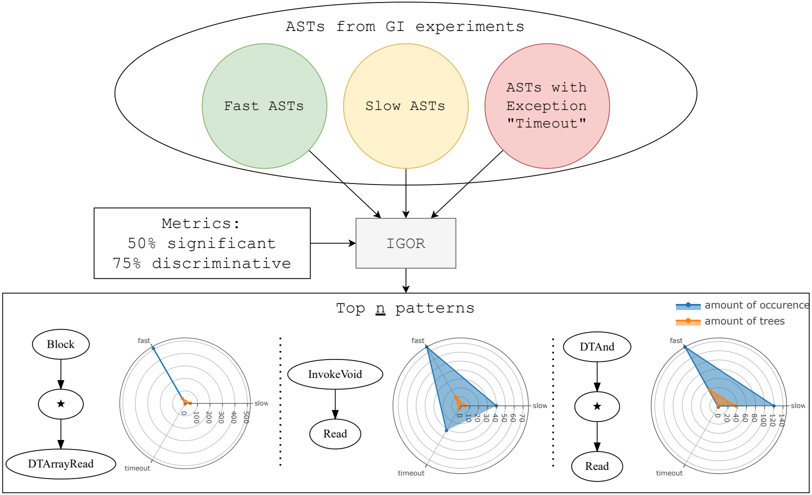


Independent Growth of Ordered Relationships

- Mining of *frequently* recurring substructures
- *Significant* if occuring with a minimum support
- *Discriminative* pattern mining
  - Often used for software fault mining
  - Mining in two groups - *succeeding* and *failing*
  - Discriminative pattern occurs more often in one group

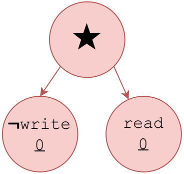


Independent Growth of Ordered Relationships

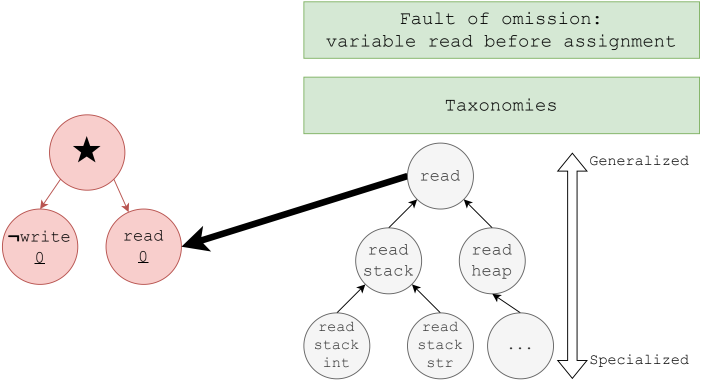


Example: Bug Pattern Uninitialized Variables

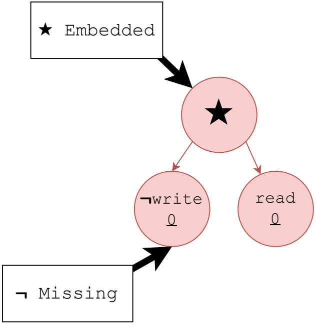
Fault of omission:
variable read before assignment



Example: Bug Pattern Uninitialized Variables



Example: Bug Pattern Uninitialized Variables

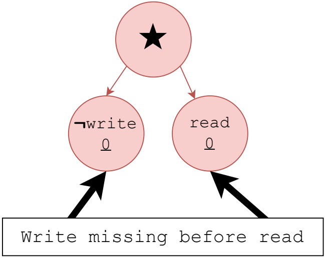


Fault of omission:
variable read before assignment

Taxonomies

Wildcards

Example: Bug Pattern Uninitialized Variables



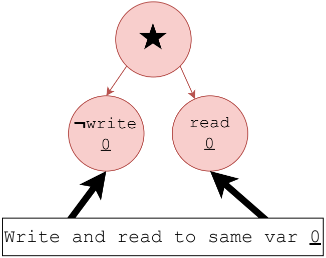
Fault of omission:
variable read before assignment

Taxonomies

Wildcards

Ordered Patterns

Example: Bug Pattern Uninitialized Variables



Fault of omission:
variable read before assignment

Taxonomies

Wildcards

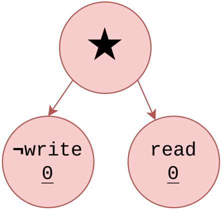
Ordered Patterns

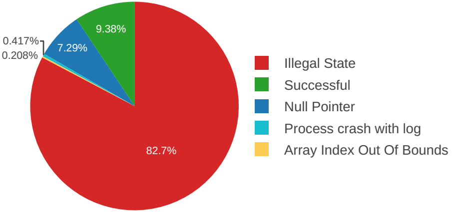
Variables Considered

Validating patterns

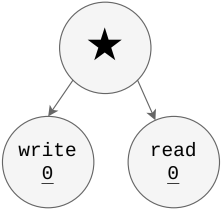
- Using *KGGI Syntax Graph*
- Create Mutants of *n* ASTs
- Validate confidence in hypothesis
  - x% have a speedup due to pattern
  - x% fail due to exception
- Side effects prevent 100% confidence

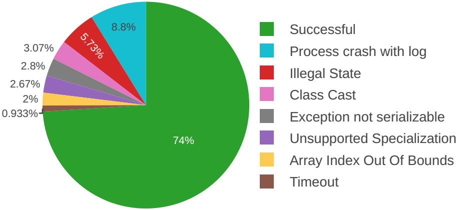
Pattern Validation - Bug





(cause 82.7% confidence)





(fix 94.27% confidence)

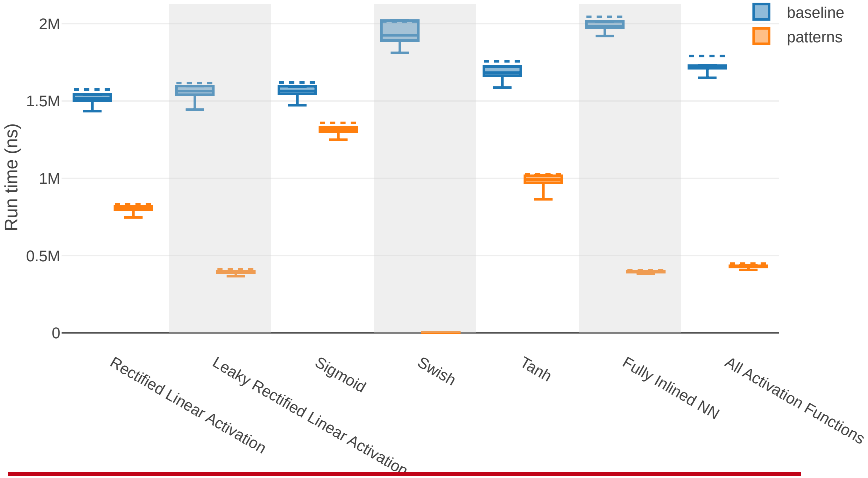
Using Patterns in GI

JOHANNES KEPLER
UNIVERSITÄT LINZ

Figure: Top: population in GI without patterns; bottom: with patterns

Improvement of Run-time Performance

JOHANNES KEPLER
UNIVERSITÄT LINZ



Application of patterns in GI

GI population

- Amount of ASTs overall doubled
- Amount of successful ASTs doubled
- Only 32.7% of ASTs with exceptions (down from 60.3%)

Run-time performance

- 22 / 25 ASTs improved
- Average of 33.5% faster

Summary

Benefits

- GI at the compiler level
- Identify and explain patterns
- Apply patterns in GI
- Improves population quality and diversity

Drawbacks

- Large search spaces
- Mutation and Crossover costly
- Run-time performance measurement costs

Outlook

JOHANNES KEPLER
UNIVERSITÄT LINZ

- Improve Amaru
  - Ease of use
  - Additional algorithms
  - Automation of pattern mining

Outlook

- Improve Amaru
  - Ease of use
  - Additional algorithms
  - Automation of pattern mining
- Additional Connectors
  - More Truffle languages
  - Additional compilers

Outlook

- Improve Amaru
  - Ease of use
  - Additional algorithms
  - Automation of pattern mining
- Additional Connectors
  - More Truffle languages
  - Additional compilers
- Answer your questions

Contact

Code available under the MIT License at `https://amaru.dev`

Oliver Krauss
oliver.krauss@fh-hagenberg.at

+43 (0)50804-27195

Bibliography I

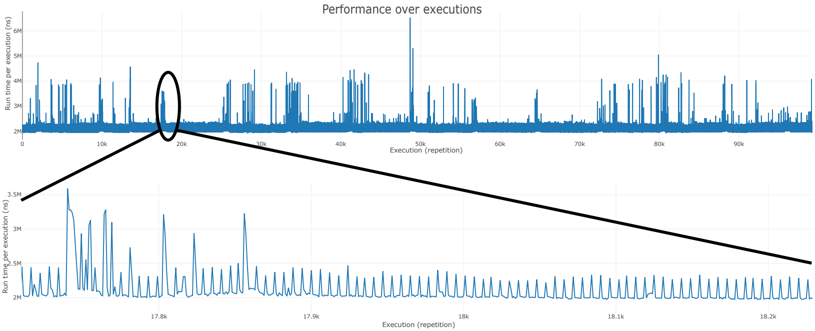
[1] Jussi Koskinen, *Software Maintenance Costs,* [Online; accessed 13. Apr. 2022], Apr. 2022. [Online]. Available: https://wiki.uef.fi/display/tktWiki/Jussi+Koskinen?preview=/38669960/38634345/SMCOSTS.pdf.

Measuring Runtime Performance

- Semantic validity $\rightarrow$ test based $+$ coverage metrics

Measuring Runtime Performance

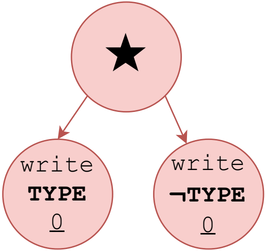
- Semantic validity → test based + coverage metrics
- Accurate measures for NFP → 200,000 runs per AST in own JVM



Measuring Runtime Performance

- Semantic validity → test based + coverage metrics
- Accurate measures for NFP → 200,000 runs per AST in own JVM
- Takes time

Generalizable Optimizations



- Switch of variable type
- Affects example language MiniC
- Possibly generalizable

Generalizable Optimizations

- Other performance (anti)-patterns useful for GI
- Patterns can hint at issues in language
  - Ex. Inlining pattern
  - Inlining identified as performance pattern
  - Graal inlines by itself
  - Root cause was bug