MINING PATTERNS FROM GENETIC IMPROVEMENT EXPERIMENTS

GI@ICSE 2019 - MONTREAL - MAY 28TH 2019

Oliver Krauss (presenting), Hanspeter Mössenböck, Michael Affenzeller

ABSTRACT

- GI runs produce and evaluate many individuals
- Mine this information to find:
  - **anti-patterns** - restrict search space
  - **optimization-patterns** - use in grafting operators

ENABLING DATA FOR MINING

- Represent code as Abstract Syntax Tree (AST)
  - should be fine granular
- Log evaluations (performance, successful executions, ...)
- Log relationships between individuals (crossover, mutation...)

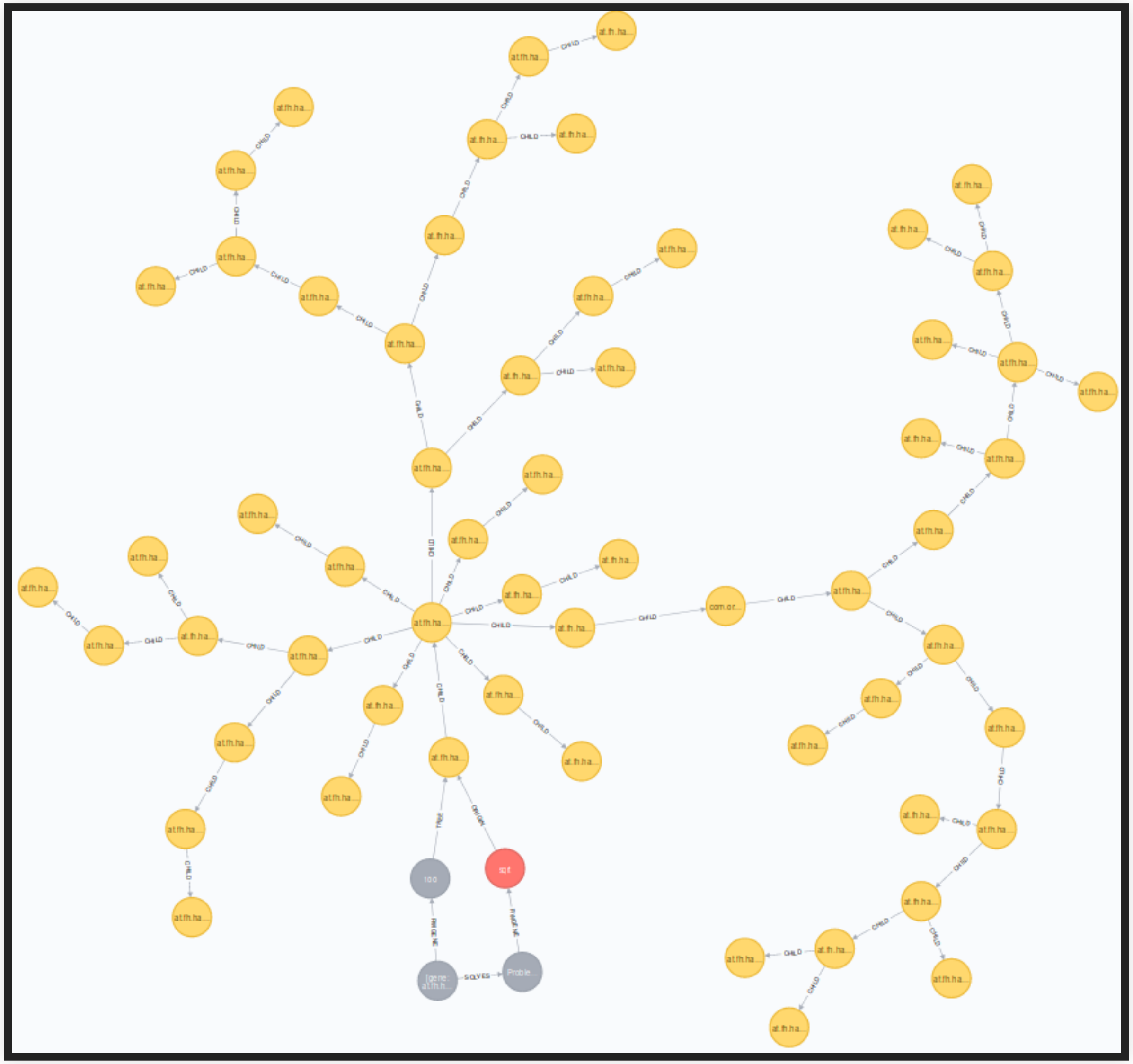


Figure: Newtonian approximation (sqare root) as AST

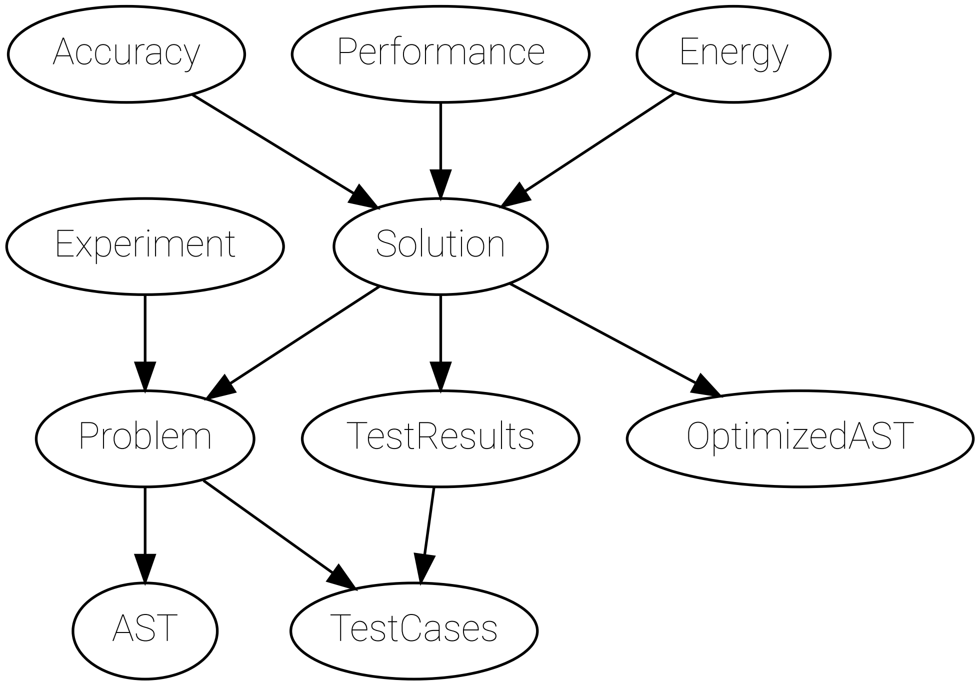


Figure: Data model for pattern mining

FREQUENT PATTERN MINING (FPM)

- Finds largest fequently recurring subgraphs
- Does not handle significance
  - currently manual
  - future statistics or observed/expected frequency
- SOTA -> SLEUTH 2

MINING OPTIMIZATION / ANTI PATTERNS

- Set found patterns into **context**
  - In/output datatypes
  - Problem domain
  - Similar behaviour in fitness (energy, ...)
- FPM in solution space
- FPM in problem space
- See if any found patterns **correlate**

OPTIMIZATION PATTERS

Goal: find source code that can be replaced by something better

- Search solution space with high quality for *optimization-patterns* (ex. performance)
- Search original ASTs of found patterns for *unoptimized-patterns*

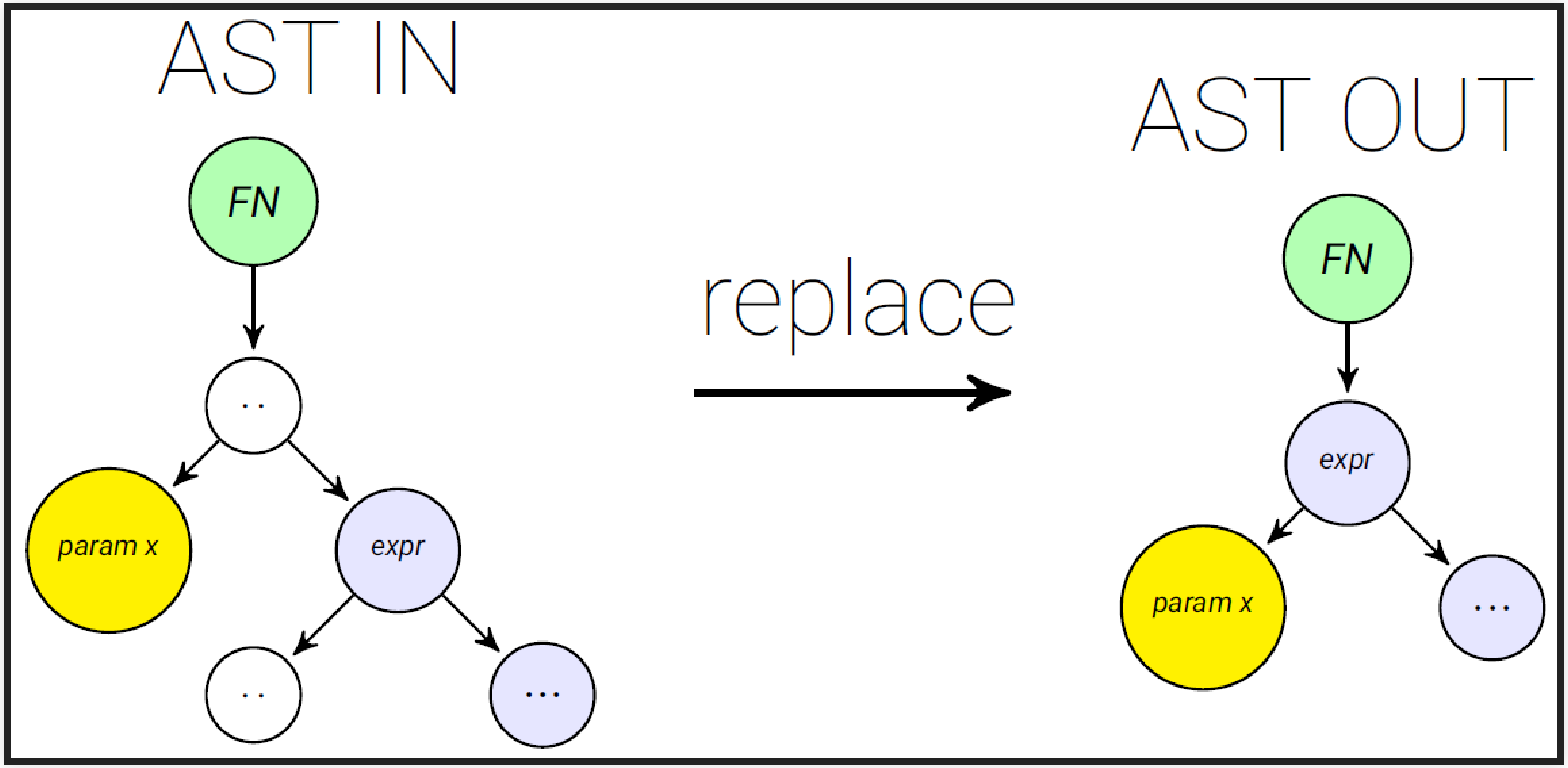


Figure: Optimization pattern (left unoptimized, right optimized)

ANTI PATTERNS

Goal: find patterns that negatively influence solutions

- Can also be used to reduce the search space
- Search solution space with low quality for *anti-patterns*
- Optional: Search original ASTs to find out if anti patterns match *specific domains*

GENERIFYING PATTERNS BY HIERARCHY

- GI is done with **Truffle** 3 and **Graal**
- Operators have a *class hierarchy*

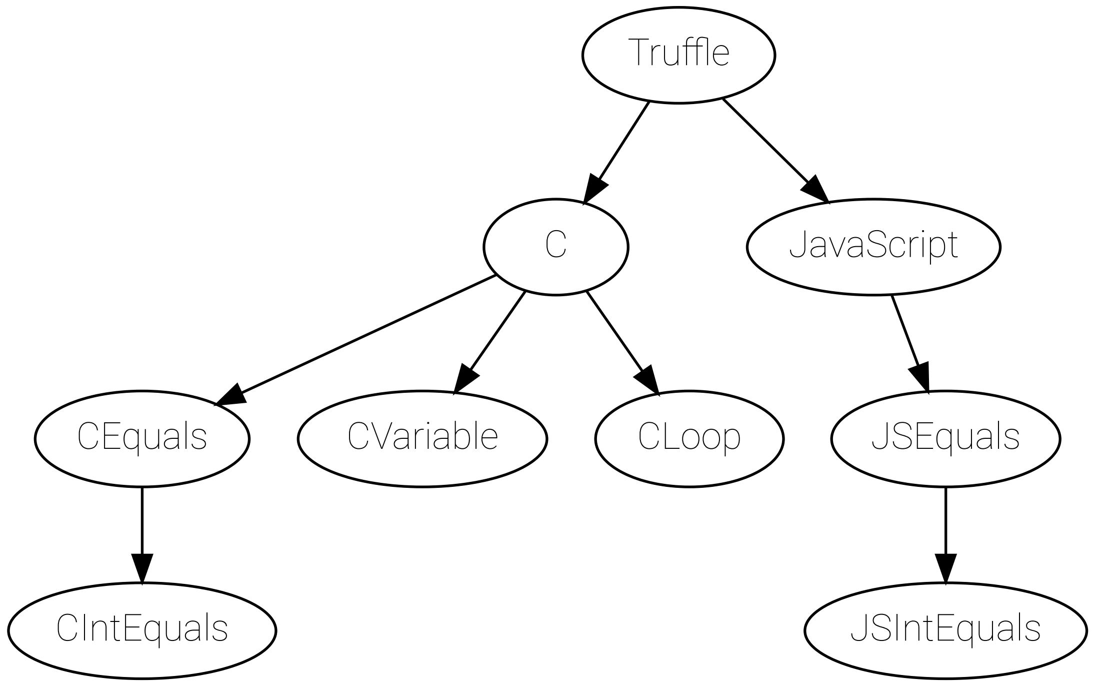
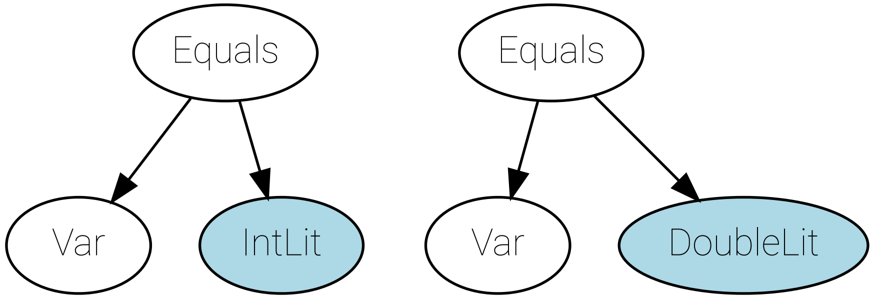


Figure: Truffle node class hierarchy



raise hierarchy of literals:

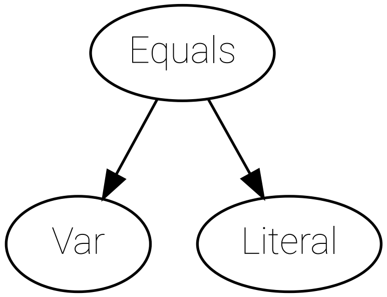


Figure: Finding larger patterns by hierarchy

GENERIFYING PATTERNS BY WILDCARDS (FUTURE WORK)

- Combine smaller patterns with "*bridges*"
- * = 0..*
- . = 1..1
- ? = 0..1

CONCLUSION

- Successfully used in biasing operators
- Still early stage
- Currently restricted to trees, due to SLEUTH

OUTLOOK

- Empirical study for energy consumption patterns
  - planned @GPCE 2019
- Extending wildcard patterns
- Your feedback here

QUESTIONS?

CONTACT

Oliver Krauss

oliver.krauss@fh-hagenberg.at

+43 (0)50804-27195

REFERENCES

1: O. Krauss, H. Mössenböck, M. Affenzeller, 2019 *Mining Patterns from Genetic Improvement Experiments*, 6th International Workshop on Genetic Improvement, May 2019. last accessed 19.05.2019

2: M. J. Zaki, 2005 *Efficiently mining frequent embedded unordered trees*, Fundamenta Informaticae, vol. 66, no. 1-2, pp. 33–52, Mar. 2005, special issue on Advances in Mining Graphs, Trees and Sequences. last accessed 19.05.2019

3: C. Wimmer, T. Würhtinger, 2012 Truffle: A Self-optimizing Runtime System. In Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH '12). ACM, New York, NY, USA, 13–14. last acessed: 19.05.2019