Adaptive Recursive Query Optimization

Anna Herlihy, Guillaume Martres, Anastasia Ailamaki, Martin Odersky

Adaptive Metaprogramming in Datalog

Datalog is a logic-based programming language that has been used in Java program analysis, TensorFlow, Rust compilation, Ethereum VM, AWS network analysis, and other performance-critical applications.

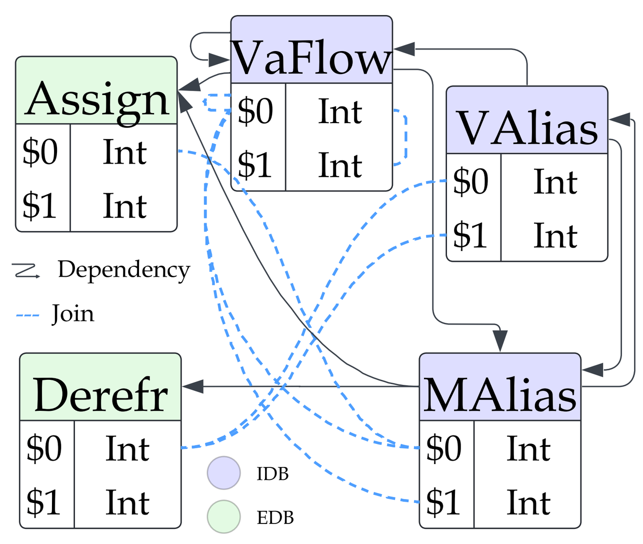
Adaptive Metaprogramming uses Multi-Stage Programming to continuously reoptimize Datalog via **phases of compile and runtime code generation**, so the optimizer can adapt to new information as it becomes available.

Running Example: Graspan's Context-Sensitive Pointer Analysis (CSPA) on Apache httpd

Advanced program analysis requires solving constraint systems with complex, recursive interdependencies.

CSPA Datalog Program

```
val program = new Program(Execution(Storage()))
val VFlow, VAlias, MAlias,
    Assign, Deref = program.relation()
val v0, v1, v2, v3 = program.variable()
VaFlow(v1, v2) :- MAlias(v3, v2), Assign(v1, v3)
VaFlow(v1, v2) :- VaFlow(v3, v2), VaFlow(v1, v3)
MAlias(v1, v0) :- VAlias(v2, v3), Derefr(v3, v0),
                  Derefr(v2, v1)
VAlias(v1, v2) :- VaFlow(v3, v2), VaFlow(v3, v1)
VAlias(v1, v2) :- VaFlow(v0, v2), VaFlow(v3, v1),
                  MAlias(v3, v0)
VaFlow(v2, v1) :- Assign(v2, v1)
VaFlow(v1, v1) :- Assign(v1, v2)
VaFlow(v1, v1) :- Assign(v2, v1)
MAlias(v1, v1) :- Assign(v2, v1)
MAlias(v1, v1) :- Assign(v1, v2)
```



CSPA Database Schema

$$VAlias_{\delta+1} = \bigcup \begin{array}{l} \pi_{\$1,\$3}(\sigma_{\$0=\$2}(VaFlow_{\delta} \times VaFlow_{\star})) \\ \pi_{\$1,\$3}(\sigma_{\$0=\$2}(VaFlow_{\star} \times VaFlow_{\delta})) \end{array} \bigcup \begin{array}{l} \pi_{\$3,\$1}(\sigma_{\$0=\$5,\$2=\$4}(VaFlow_{\delta} \times VaFlow_{\star} \times MAlias_{\star})) \\ \pi_{\$3,\$1}(\sigma_{\$0=\$5,\$2=\$4}(VaFlow_{\star} \times VaFlow_{\delta} \times MAlias_{\star})) \\ \pi_{\$3,\$1}(\sigma_{\$0=\$5,\$2=\$4}(VaFlow_{\star} \times VaFlow_{\star} \times MAlias_{\delta})) \end{array}$$

Generated VAlias Subquery

The wrong execution plan can significantly impact performance: CSPA generates subqueries that have a factorial possible left-deep join orderings, and selecting a suboptimal join order in just a single subquery in only the first iteration leads to extra materialization of 6534GB.

The optimal query plan changes during execution so traditional relational database query optimization techniques cannot scale to iterative query execution.

Carac: An Adaptive Just-in-time Datalog Compiler

The Carac compiler uses Adaptive Metaprogramming to **partially evaluate the input Datalog program and continuously regenerate specialized and parallelized imperative programs.**

Carac compiles Datalog to an IR using a Futamura Projection and the bottom-up Semi-Naive algorithm. The IR is lowered to various targets, including:

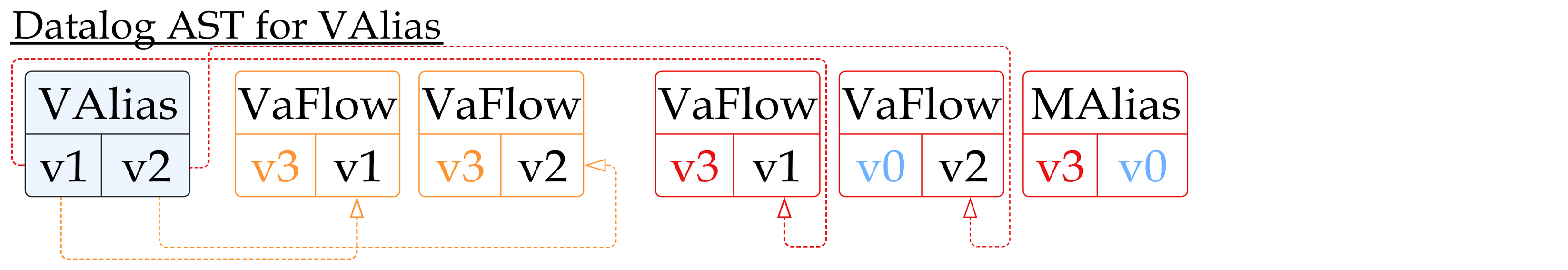
(1) Scala 3 Quotes and Splices

(2) JVM bytecode

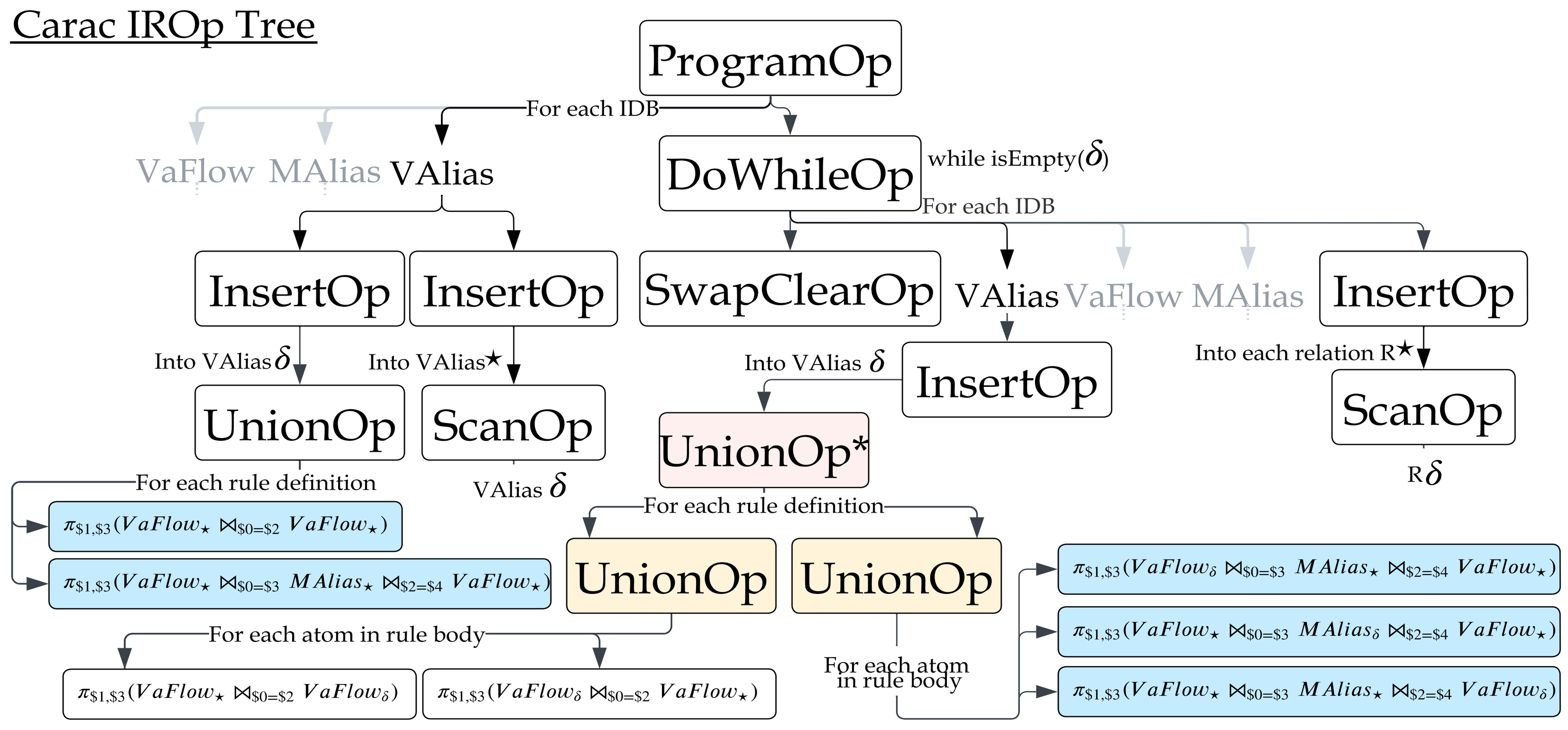
(3) Higher-order lambda functions

Where possible, JIT compilation is combined with **macros to push expensive optimizations offline**.

Datalog AST for VAlias

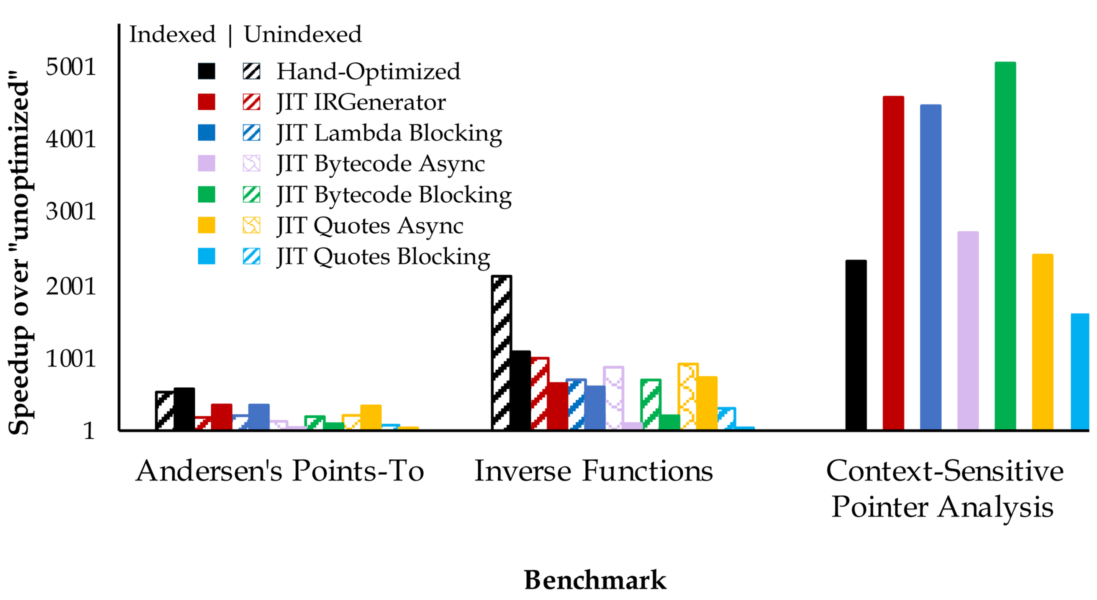


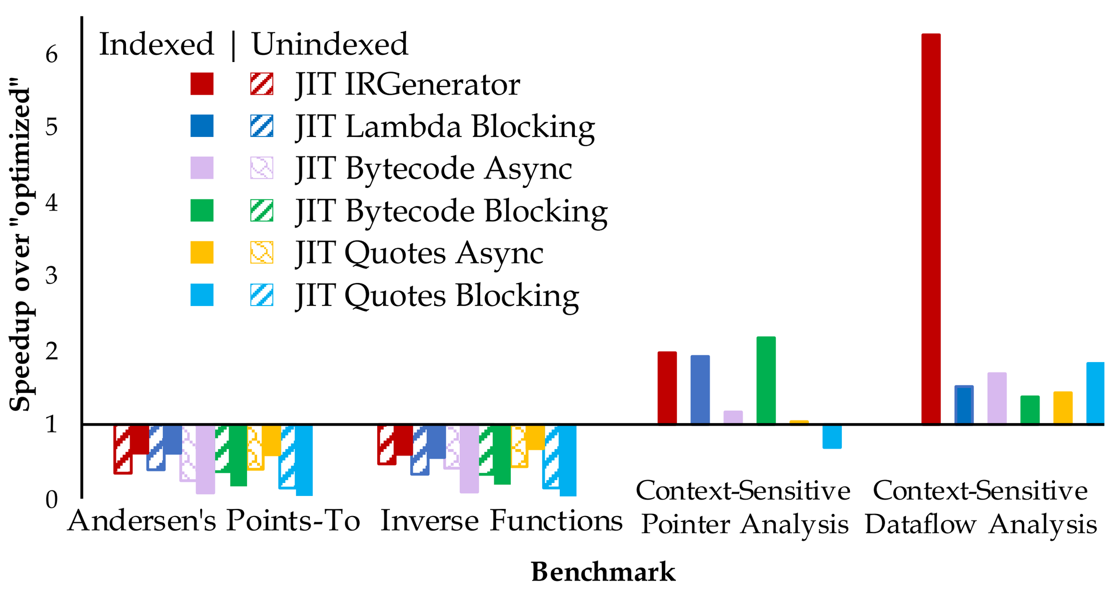
Carac IROp Tree

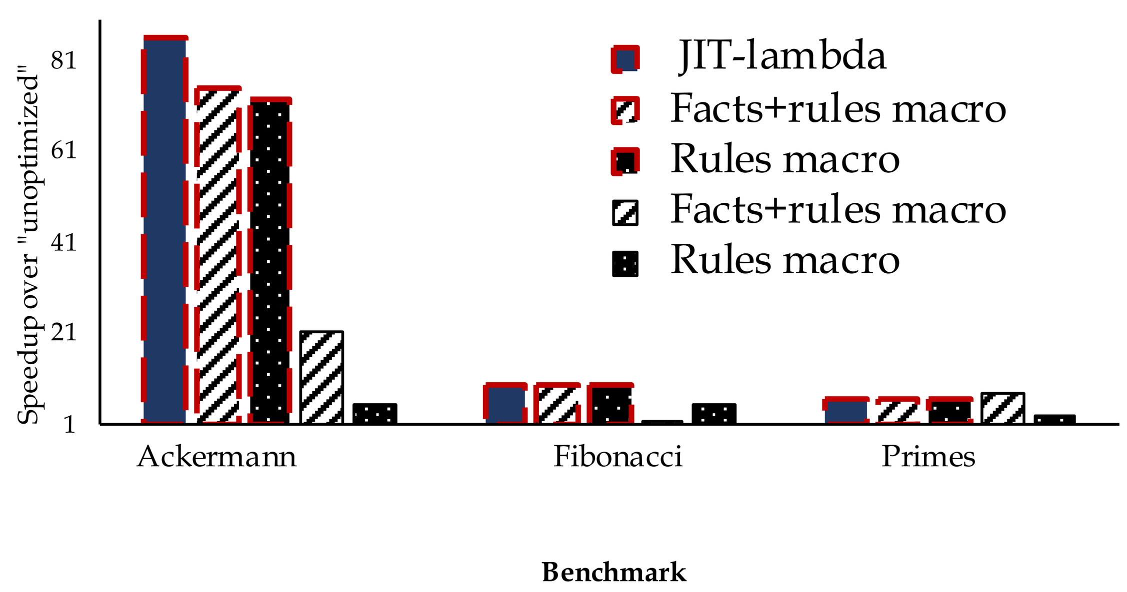


Staged Unoptimized Queries | Staged Hand-Optimized Queries | Ahead-of-time + Staged Queries

Dual-socket Intel Xeon E5-2650L v3 12 physical cores per socket and 396GB RAM.







Up to 5000x speedup over unoptimized queries and 6x over hand-optimized queries using *adaptive metaprogramming*.

Fondo nazionale svizzero
"Capabilities for Typing Resources and Effects" (TMAG-2_209506/1)