Caching Results from KLEE's Independent Solver

Zikai Liu

with Steven S. Lumetta

ZJU-UIUC Institute

Zhejiang University / University of Illinois at Urbana-Champaign Institute

Independent Solver Eliminates Irrelevant Constraints

		Selected?
Query:	$i = 20$?	
Constraints:	$i < j$	
	$j < 20$	
	$k > 0$	

Reference: the OSDI 2008 KLEE paper

First Step: Construct IndependentElementSets

		Selected?	IndependentElementSet
Query:	$i = 20$?		{ *i* }
Constraints:	$i < j$		{ *i*, *j* }
	$j < 20$		{ *j* }
	$k > 0$		{ *k* }

Reference: the OSDI 2008 KLEE paper

Start with Variables in the Query Expression

		Selected?	IndependentElementSet	Element Closure
Query:	$i = 20$?		$\{ i \}$	$\{ i \}$
Constraints:	$i < j$		$\{ i, j \}$	
	$j < 20$		$\{ j \}$	
	$k > 0$		$\{ k \}$	

Reference: the OSDI 2008 KLEE paper

Select Constraints with Relevant Variables

		Selected?	IndependentElementSet	Element Closure
Query:	$i = 20$?		$\{ i \}$	$\{ i \}$
Constraints:	$i < j$	✓	$\{ i, j \}$	
	$j < 20$		$\{ j \}$	
	$k > 0$		$\{ k \}$	

Reference: the OSDI 2008 KLEE paper

Update Working Element Closure

		Selected?	IndependentElementSet	Element Closure
Query:	$i = 20$?		{ i }	**{ *i, j* }**
Constraints:	$i < j$	✔	**{ *i, j* }**	
	$j < 20$		{ j }	
	$k > 0$		{ k }	

Reference: the OSDI 2008 KLEE paper

Repeat Until No More Constraints to Add

		Selected?	IndependentElementSet	Element Closure
Query:	$i = 20$?		$\{ i \}$	$\{ i, j \}$
Constraints:	$i < j$	✓	$\{ i, j \}$	
	$j < 20$	✓	$\{ j \}$	
	$k > 0$		$\{ k \}$	

Reference: the OSDI 2008 KLEE paper

Irrelevant Constraints are Eliminated

		Selected?	IndependentElementSet	Element Closure
Query:	$i = 20$?		$\{ i \}$	$\{ i, j \}$
Constraints:	$i < j$	✓	$\{ i, j \}$	
	$j < 20$	✓	$\{ j \}$	
	$k > 0$	✗	$\{ k \}$	

Reference: the OSDI 2008 KLEE paper

Cache IndependentElementSets

With a hashing map from expressions (Exprs) to IndependentElementSets

To avoid redundant constructions

Also avoid copying overhead

```
static IndependentElementSet getIndependentConstraints(
  const Query& query, std::vector< ref<Expr> > &result) {

  IndependentElementSet eltsClosure(query.expr);

  // newly added cache that maps Exprs to IndependentElementSets
  static ExprHashMap<IndependentElementSet> caches;

  // worklist now holds only pointers
  std::vector< std::pair<ref<Expr>,
               const IndependentElementSet *> > worklist;

  // get or construct an IndependentElementSet for each
constraint
  for (const auto &c : query.constraints) {
    auto it2 = caches.find(c);
    if (it2 == caches.end())
      it2 = caches.emplace(c, IndependentElementSet(c)).first;

    worklist.emplace_back(c, &it2->second);
  }


  // KLEE's matching and swapping loop follows…
}
```

Cache IndependentElementSets

With a hashing map from expressions (Exprs) to IndependentElementSets

To avoid redundant constructions

Also avoid copying overhead

```
static IndependentElementSet getIndependentConstraints(
  const Query& query, std::vector< ref<Expr> > &result) {

  IndependentElementSet eltsClosure(query.expr);

  // newly added cache that maps Exprs to IndependentElementSets
  static ExprHashMap<IndependentElementSet> caches;

  // worklist now holds only pointers
  std::vector< std::pair<ref<Expr>,
               const IndependentElementSet *> > worklist;

  // get or construct an IndependentElementSet for each
constraint
  for (const auto &c : query.constraints) {
    auto it2 = caches.find(c);
    if (it2 == caches.end())
      it2 = caches.emplace(c, IndependentElementSet(c)).first;

    worklist.emplace_back(c, &it2->second);
  }


  // KLEE's matching and swapping loop follows…
}
```

Cache IndependentElementSets

With a hashing map from expressions (Exprs) to IndependentElementSets

To avoid redundant constructions

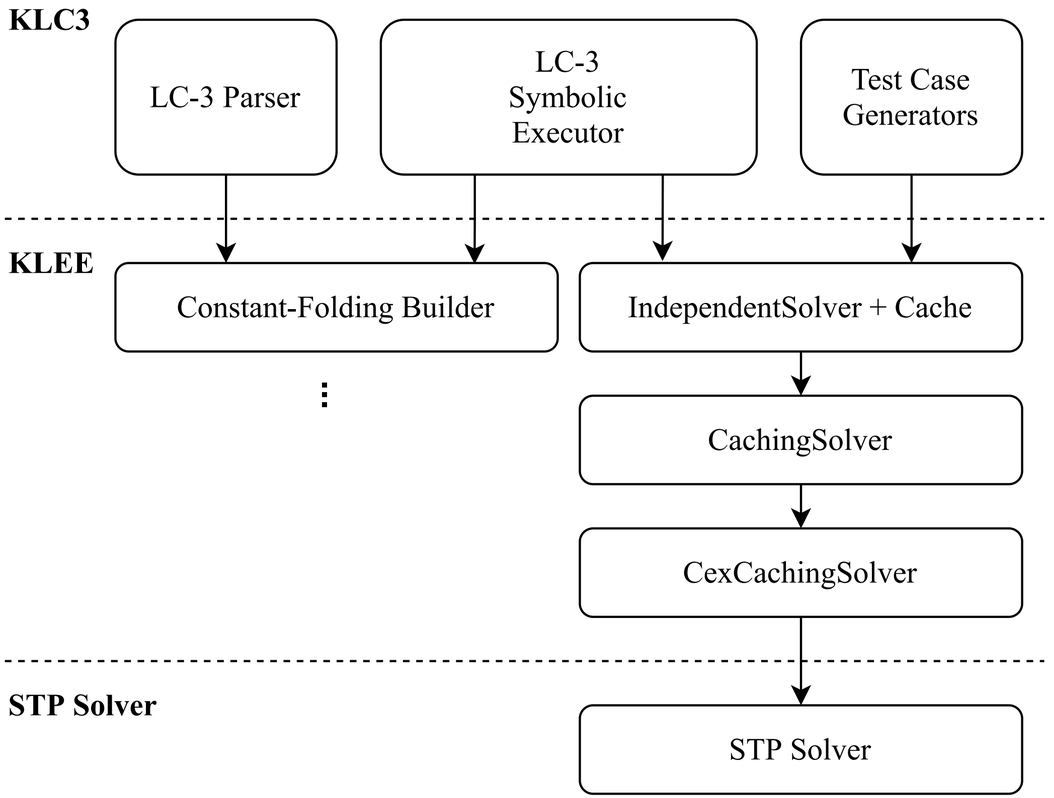
Also avoid copying overhead

```
static IndependentElementSet getIndependentConstraints(
  const Query& query, std::vector< ref<Expr> > &result) {

  IndependentElementSet eltsClosure(query.expr);

  // newly added cache that maps Exprs to IndependentElementSets
  static ExprHashMap<IndependentElementSet> caches;

  // worklist now holds only pointers
  std::vector< std::pair<ref<Expr>,
               const IndependentElementSet *> > worklist;

  // get or construct an IndependentElementSet for each
constraint
  for (const auto &c : query.constraints) {
    auto it2 = caches.find(c);
    if (it2 == caches.end())
      it2 = caches.emplace(c, IndependentElementSet(c)).first;

    worklist.emplace_back(c, &it2->second);
  }


  // KLEE's matching and swapping loop follows…
}
```

Context: Automating Feedback on Student Assembly Programs

Extended KLEE for LC-3: KLC3

- LC-3: an educational assembly
- Use KLEE infrastructure
- Customized symbolic executor
- Talk by Tingkai Liu tomorrow at 13:00 on KLC3 and other tools



The Cache Speeds up KLC3 for 3.47×

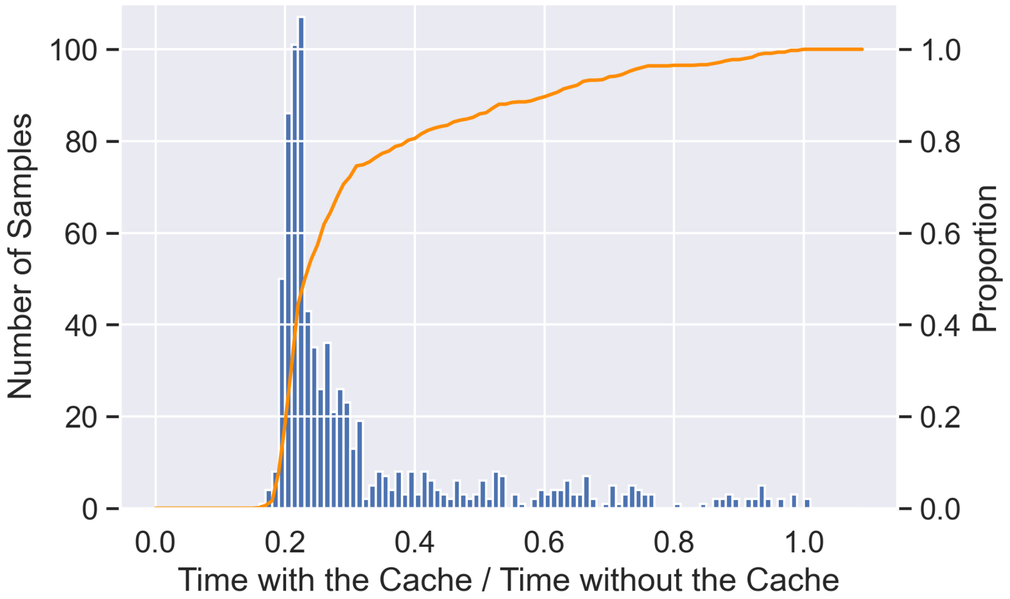
Most complex LC-3 assignment:

- DFS to fill weekly schedules
- 960 student code samples

Cache boosts performance:

- Construct 2.20×10^8 IndependentElementSets on average
- Hit rate > 99.9% for 98.5% samples
- 3.47× speedup

Our goal: feedback to students in 5 minutes



Timeout Rates for 960 Student Submissions

Timeout (Minute)	Cache	No Cache
5	13.54%	65.10%
10	5.31%	22.08%

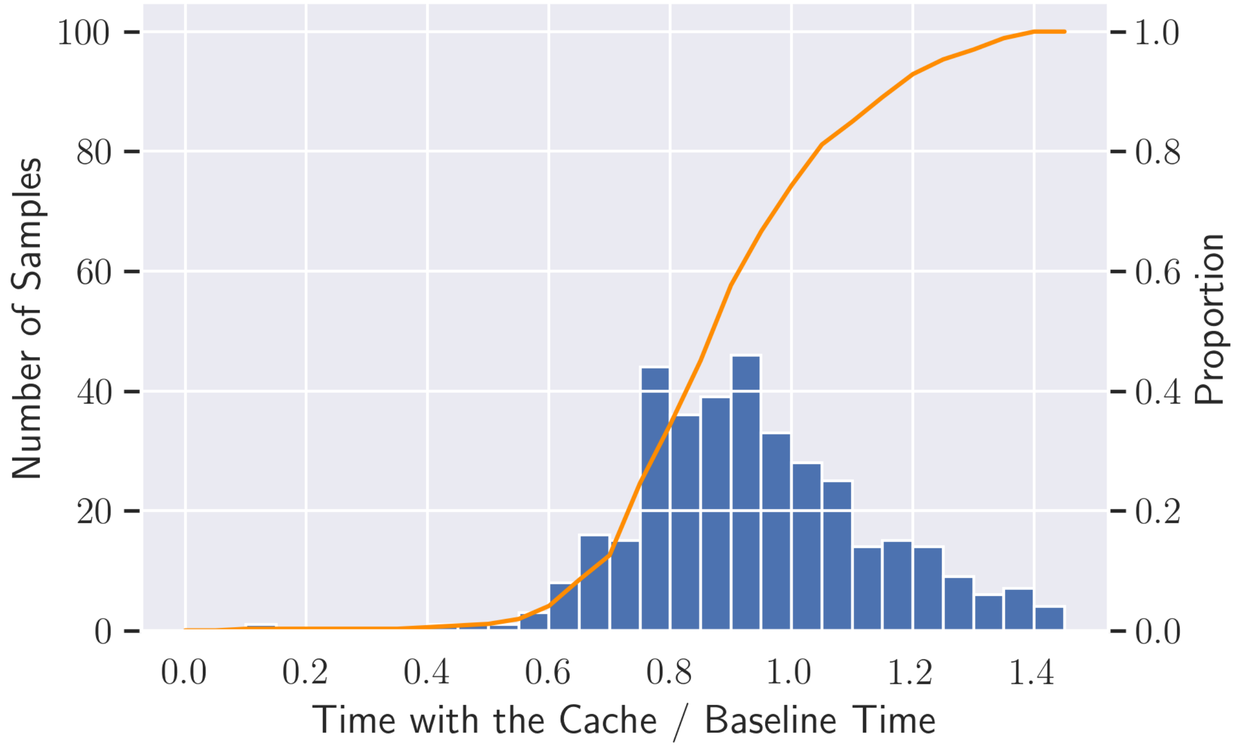
Speedup Less Impressive for C Programs

C programming assignment

- DFS to solve a maze
- 379 student code samples (Clang -O2)

Baseline using array transformations*

- 65% finish faster
- 1.07× average speedup



* D. M. Perry, A. Mattavelli, X. Zhang, and C. Cadar, "Accelerating Array Constraints in Symbolic Execution," ISSTA 2017

Why are the KLEE/LLVM results different?

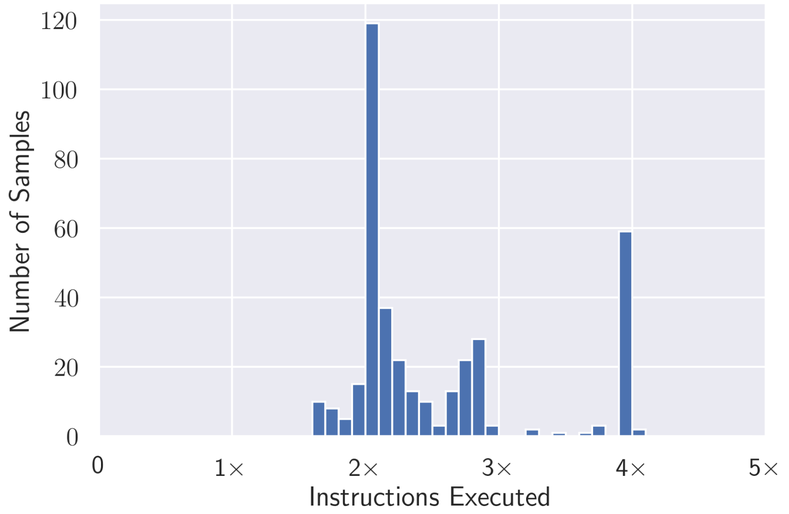
LC-3 vs LLVM IR:
simpler ISA → more instructions

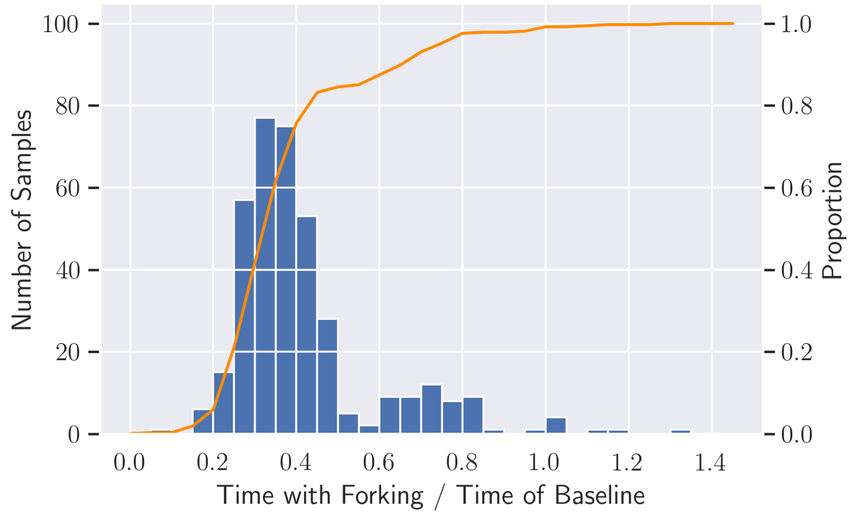
Also, for simplicity,

- KLC3 forks states on any symbolic memory reference
- More states, more instructions
- But simpler queries

Added (bounded) forking into KLEE

- 2.57× speedup





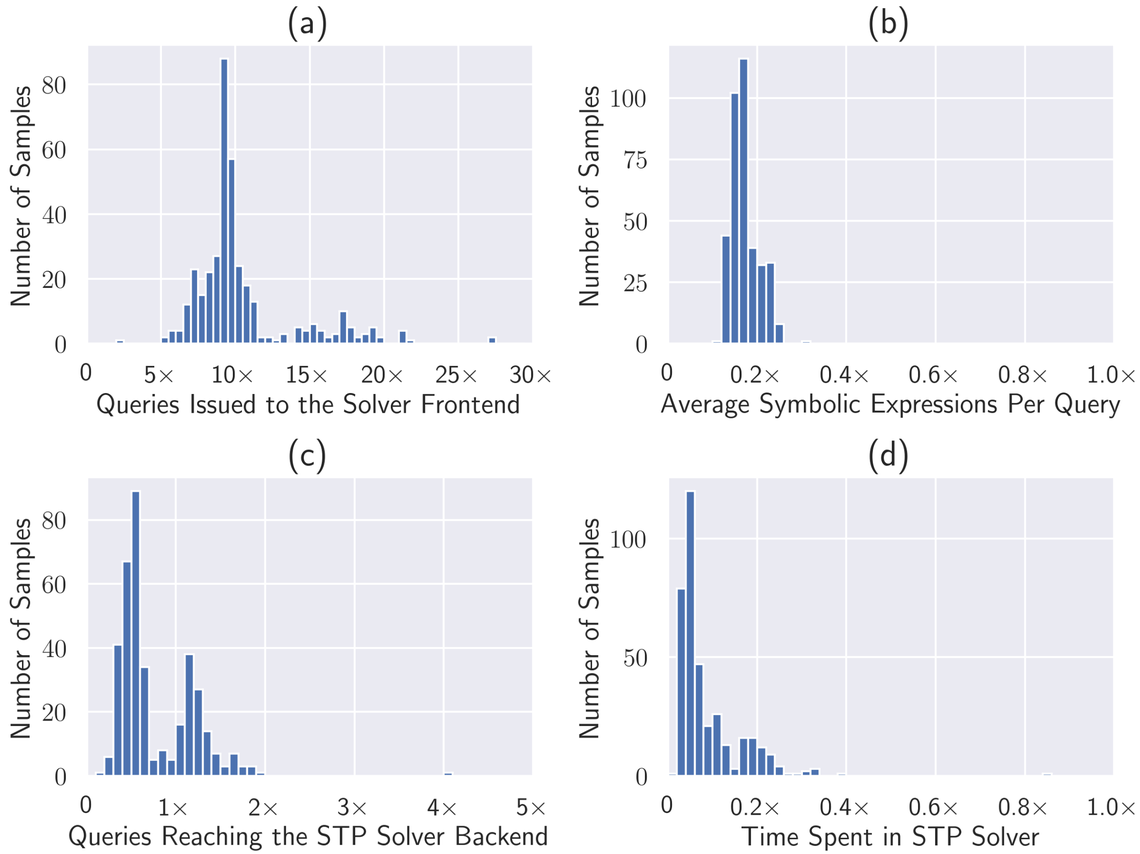
Forks Reduce Query Complexity and Improve Hit Rates

(a) Many more queries

(b) But fewer expression per query → simpler queries

(c) About the same number of queries reach STP solver

(d) Much less time spent in STP solver



Complements the IndependentElementSet cache

	No Cache	Cache
No Forking	1	1.07×
Forking	2.57×	6.59×

Summary: When IndependentElementSet Caching is Effective

Control paths highly dependent on data (ex: DFS)

Many array accesses

Small overhead for external calls