Self-Adaptive Static Analysis

Mayur Naik

Georgia Tech

Joint work with:

Xin Zhang, Ravi Mangal

Georgia Tech

Hongseok Yang, Radu Grigore

Oxford University

Mooly Sagiv

Tel-Aviv University

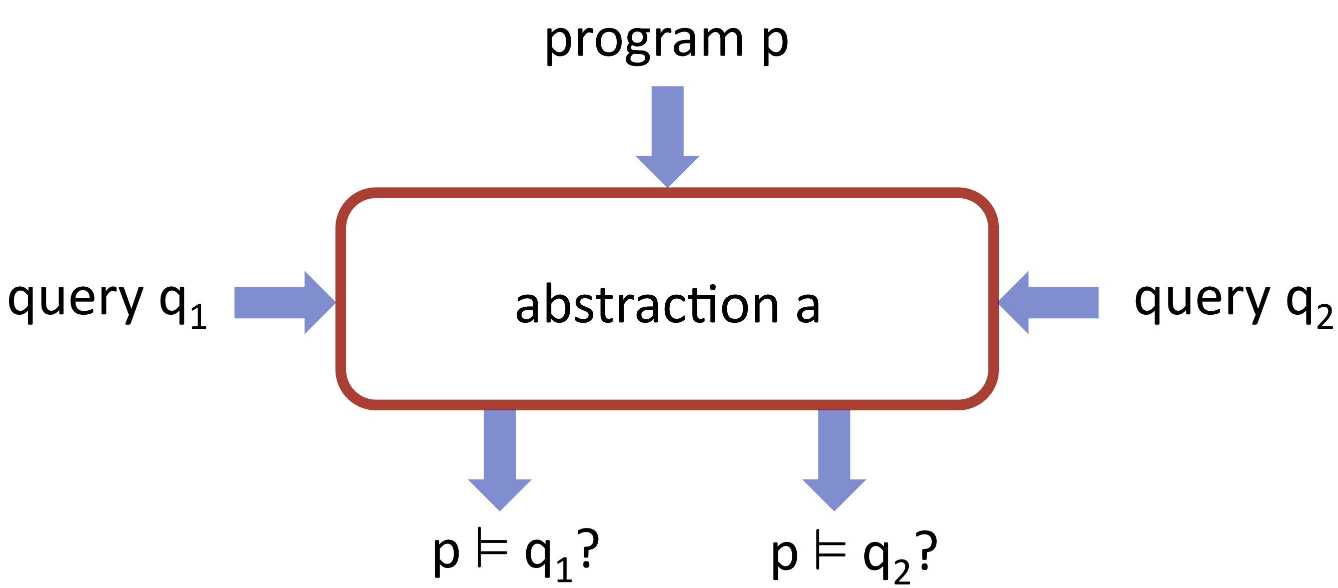
Percy Liang

Stanford University

Static Analysis: 70's to 90's

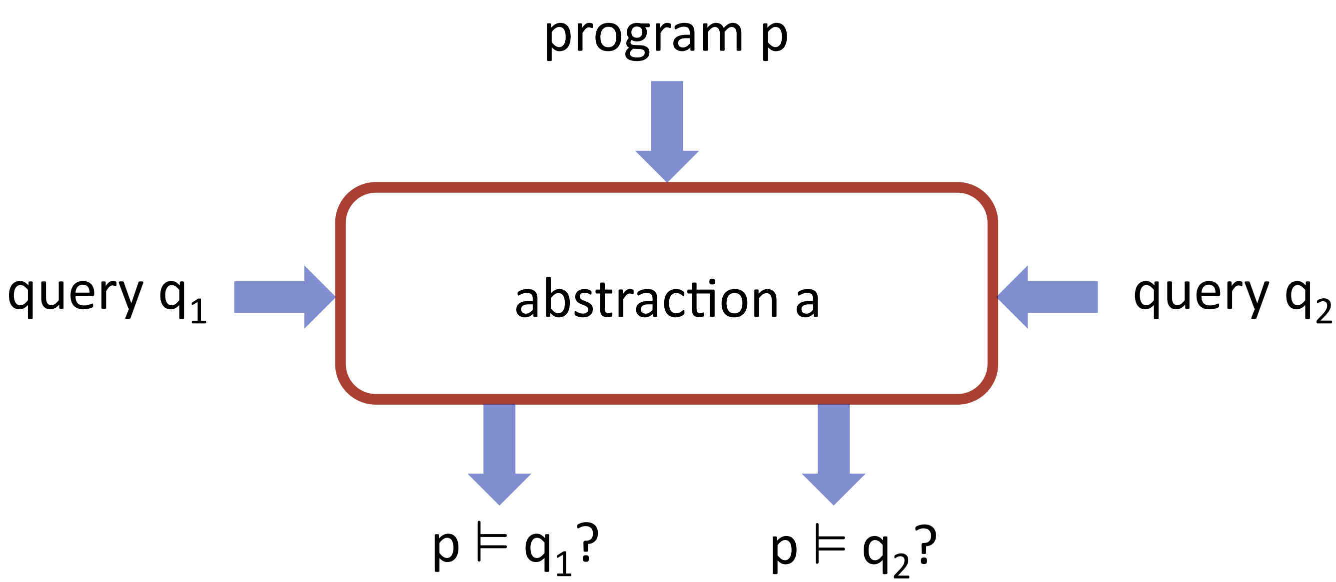
- client-oblivious

"Because clients have different precision and scalability needs, future work should identify the client they are addressing …"
M. Hind, *Pointer Analysis: Haven't We Solved This Problem Yet?*, 2001



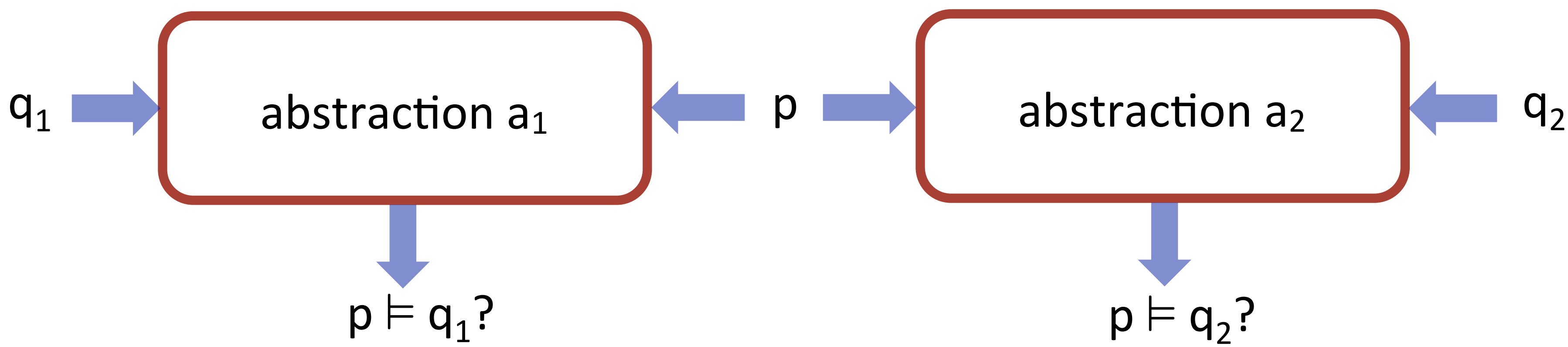
Static Analysis: 00's to Present

- client-driven



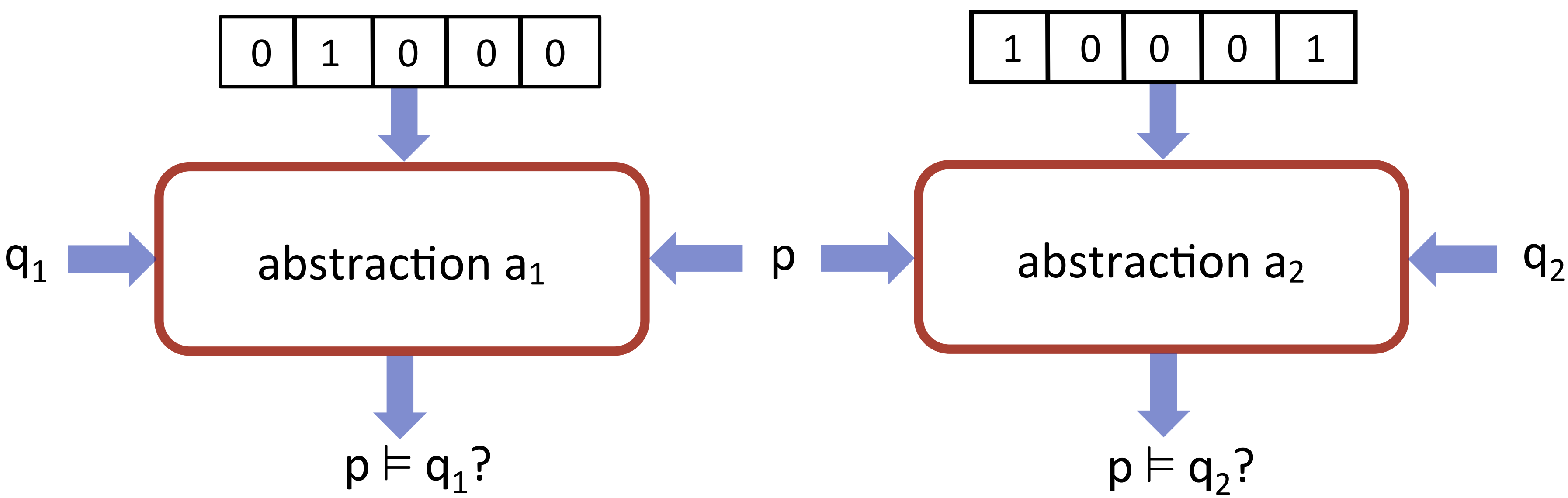
Static Analysis: 00's to Present

- client-driven
  - modern pointer analyses
  - software model checkers



Our Static Analysis Setting

- client-driven + parametric
  - new search algorithms: testing, machine learning, …
  - new analysis questions: optimality, impossibility, …



Example 1: Type-State Analysis

Must-Alias Set to Track:

```
x = new File;
y = x;
if (*) z = x;
x.open();
y.close();
if (*)
   assert1(x, closed);
else
   assert2(x, opened);
```

{ x }	{ x, y }
<*closed*, {x}>	<*closed*, {x}>
<*closed*, {x}>	<*closed*, {x,y}>
<*closed*, {x}>	<*closed*, {x,y}>
<*opened*, {x}>	<*opened*, {x,y}>
<*closed* ∨ *opened*, {x}>	<*closed*, {x,y}>
✗	✓
✗	✗

Example 2: Thread-Escape Analysis

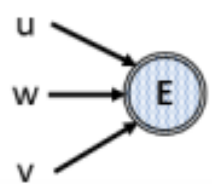
Heap abstraction:

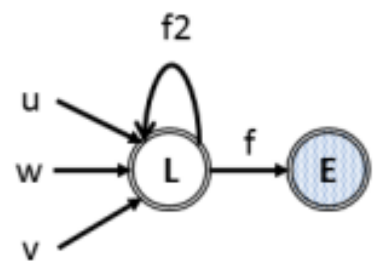
h1	h2	h3	h4
L	L	L	L

h1	h2	h3	h4
L	L	E	L

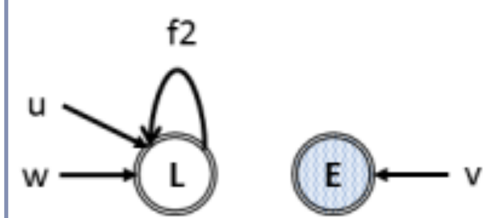
h1	h2	h3	h4
L	E	E	L

```
while (*) {
    u = new h1;
    v = new h2;
    g = new h3;
    v.f = g;
    w = new h4;
    u.f2 = w;
    assert(local(w));
    u.spawn();
}
```





but not optimal



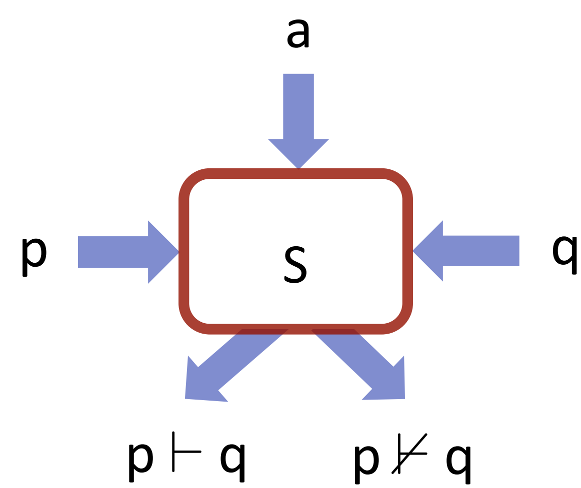
and optimal!

Example 3: Approximation Safety Analysis

Approximated Operations:	**{ 1, 2, 3, 4 }**	**{ 2, 3, 4 }**	**{ 2, 4 }**
1: v1 = input();	{ {v1} }	{ Ø }	{ Ø }
2: v2 = input();	{ {v1,v2} }	{ {v2} }	{ {v2} }
restrict(v1);	{ T }	{ {v2} }	{ {v2} }
while (v1 > 0) {	{ T }	{ {v2}, T }	{ {v2} }
3: v1 = f(v1);	{ T }	{ {v1,v2}, T }	{ {v2} }
4: v2 = g(v2);	{ T }	{ {v1,v2}, T }	{ {v2} }
restrict(v1);	{ T }	{ T }	{ {v2} }
}	{ T }	{ {v2}, T }	{ {v2} }
relax(v2);	{ T }	{ Ø, T }	{ Ø }
restrict(v2);	{ T }	{ Ø, T }	{ Ø }
output(v2);	{ T }	{ Ø, T }	{ Ø }
	✗	✗	✓

Problem Statement

- An efficient algorithm with:



INPUTS:

- program p and query q
- abstractions $A = \{ a_1, \ldots, a_n \}$
- boolean function $S(p, q, a)$

OUTPUT:

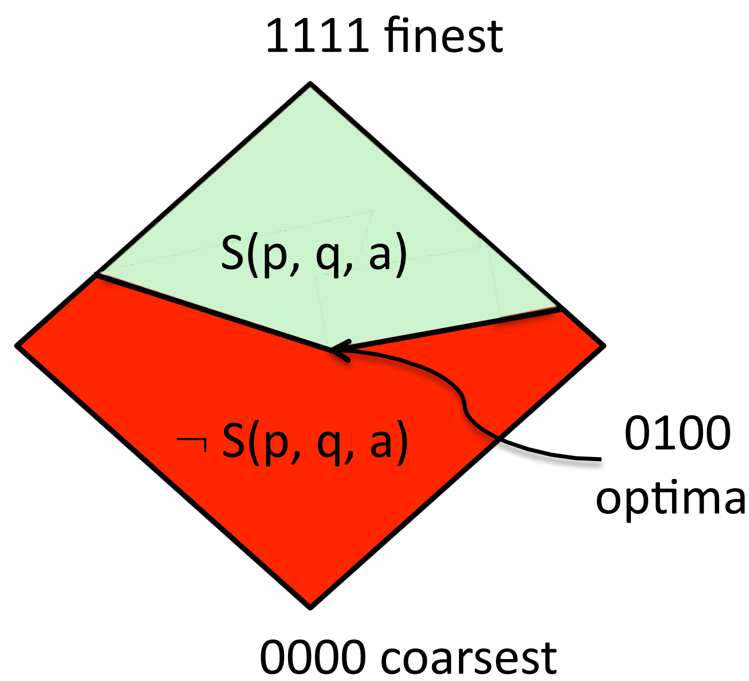
- Impossibility: $\nexists\, a \in A: S(p, q, a) = \text{true}$
- or Proof: some $a \in A: S(p, q, a) = \text{true}$ **AND**

$\forall a' \in A: (a' \leq a \wedge S(p, q, a') = \text{true}) \Rightarrow a' = a$

Optimal Abstraction

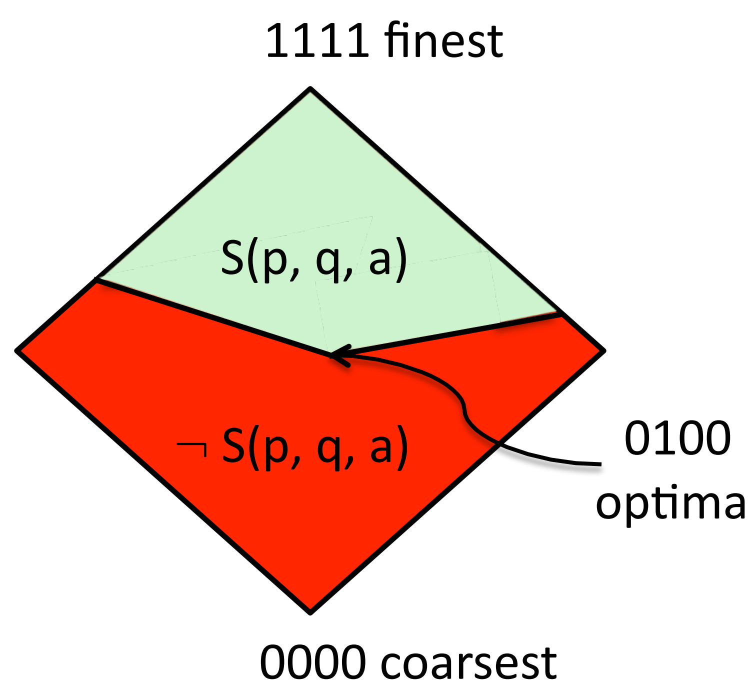
Why Optimal Abstraction?

- Yields smallest/largest solution
- Provides empirical lower bounds
- Efficient to compute
- Better for user consumption
  - analysis imprecision facts
  - assumptions about missing program parts
- Suitable for machine learning



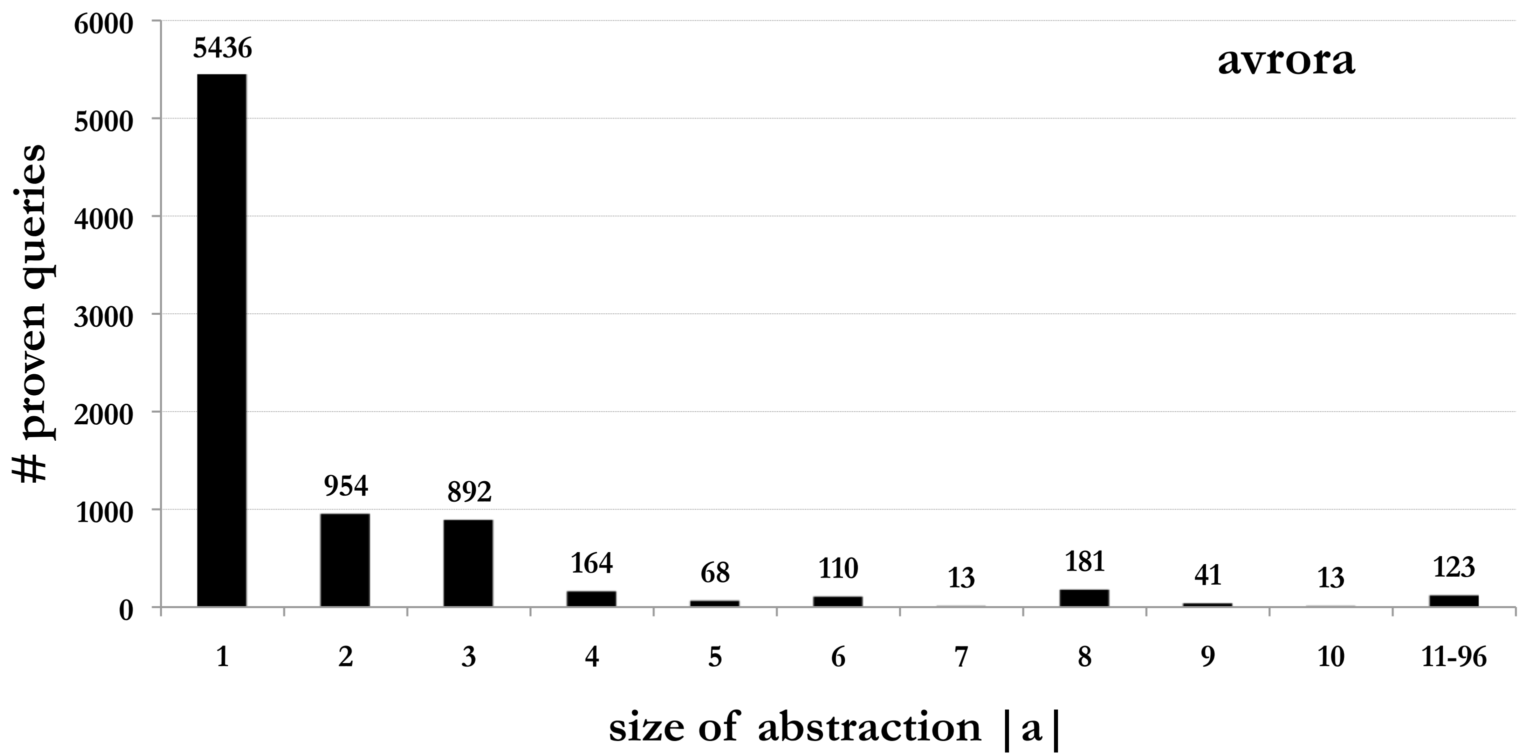
Why is this Hard in Practice?

- $|A|$ exponential in size of **p**, or even infinite
- S(p, q, a) = false for most **p, q, a**
- Different **a** is optimal for different **p, q**

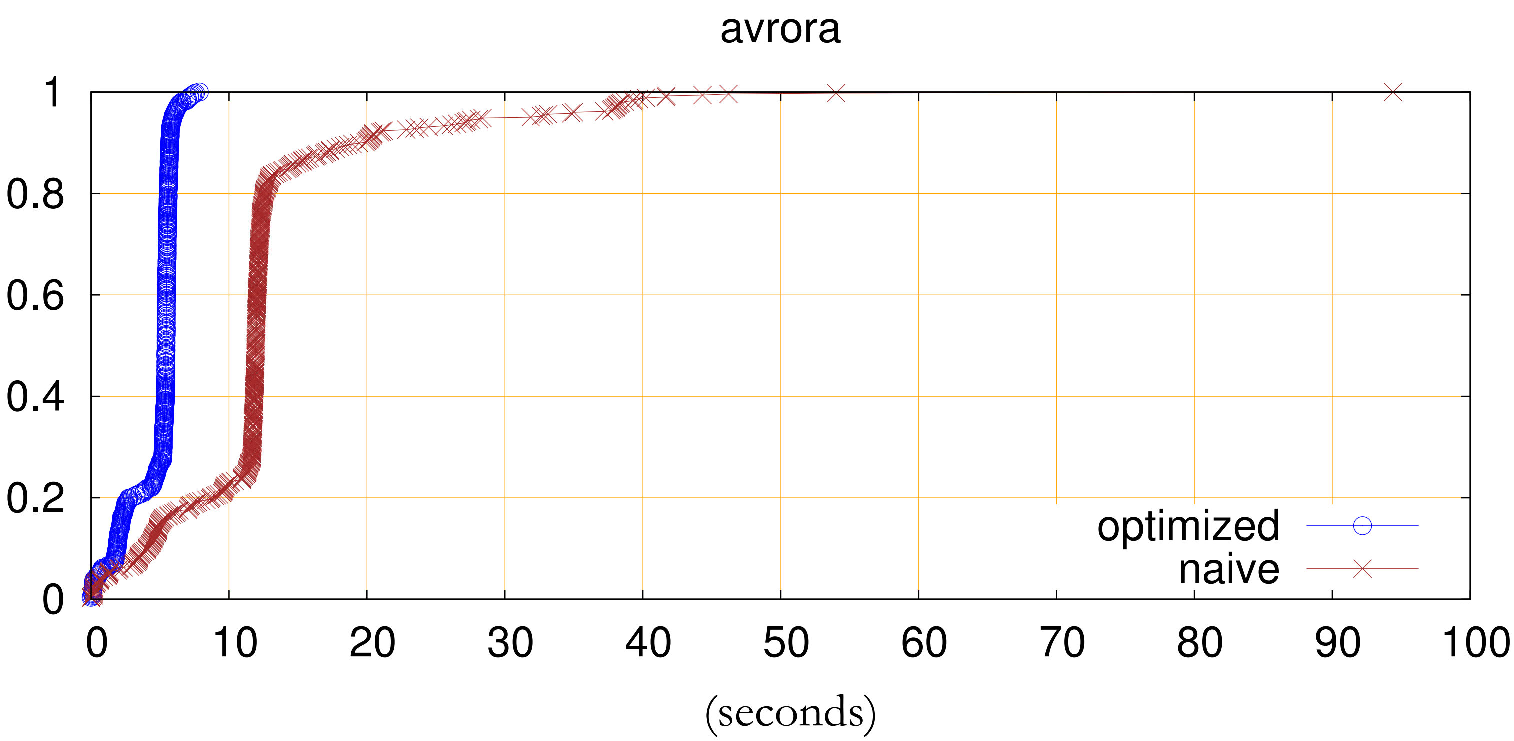


Example: Size of Optimal Abstractions

|A| = 2^10k |Q| = 14k |Q_proven| = 55% of |Q|



Example: Runtime under Optimal Abstractions



Summary of Our Results

- Machine Learning [POPL'11]
  - "Learning Minimal Abstractions"
- Dynamic Analysis [POPL'12]
  - "Abstractions from Tests"
- Static Refinement [PLDI'13]
  - "Finding Optimum Abstractions in Parametric Dataflow Analysis"
- Constraint Solving [PLDI'14]
  - "On Abstraction Refinement for Program Analyses in Datalog"

All implementations available in Chord, an extensible program analysis framework for Java bytecode (jchord.googlecode.com).

Datalog for Program Analysis

JChord

Soot

Doop

LLVM COMPILER INFRASTRUCTURE

What is Datalog?

What is Datalog?

Input relations: edge(i, j).
Output relations: path(i, j).
Rules: (1) path(i, i).
(2) path(i, k) :- path(i, j), edge(j, k).

Least fixpoint computation:
Input: edge(0, 1), edge(1, 2).
path(0, 0).
path(1, 1).
path(2, 2).
path(0, 1) :- path(0, 0), edge(0, 1).
path(0, 2) :- path(0, 1), edge(1, 2).

Why Datalog?

If there exists a path from **a** to **b**, and there is an edge from **b** to **c**, then there exists a path from **a** to **c**:

path(a, c) :- path(a, b), edge(b, c).

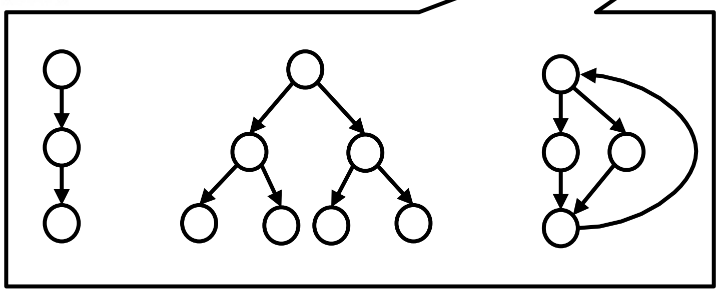
Paddle

Z3

bddbddb

OVER ONE QUATTUORDECILLION RELATIONS SERVED

LogicBlox



Why Datalog?

Paddle

Z3

bddbddb

OVER ONE QUATTUORDECILLION RELATIONS SERVED

LogicBlox

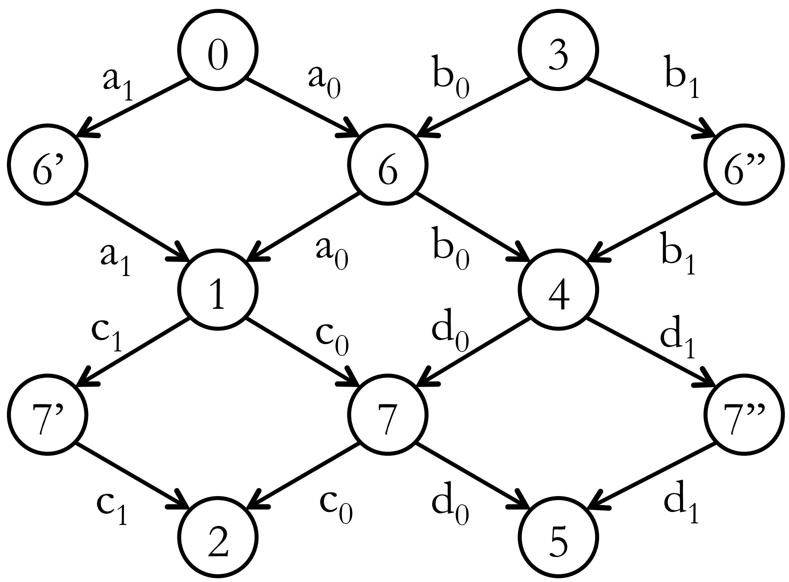
k-object-sensitivity,
k = 2, ~100KLOC

Limitation

k-object-sensitivity,
k = 2, ~100KLOC

k-object-sensitivity,
k = 10, ~500KLOC

Pointer Analysis as Graph Reachability



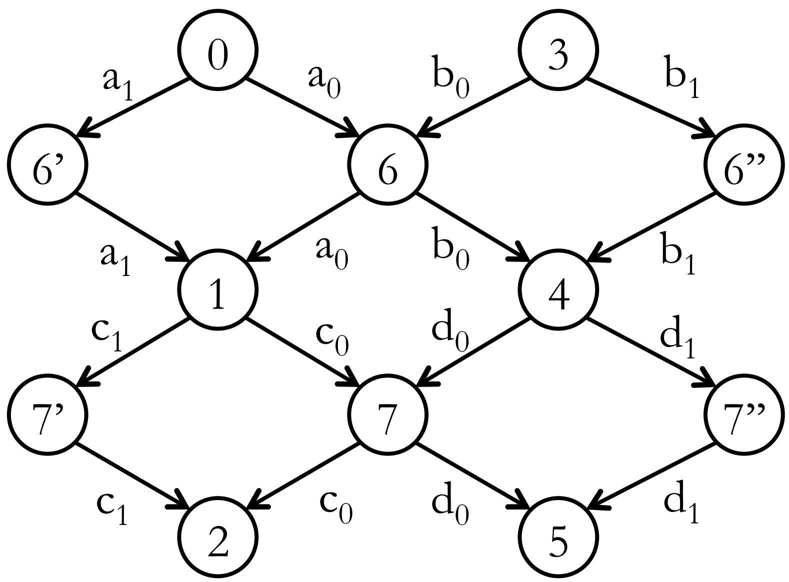

```
f(){
    v1 = new ...;
    v2 = id1(v1);
    v3 = id2(v2);
q2:assert(v3!= v1);
}

id1(v){return v;}
```

```
g(){
    v4 = new ...;
    v5 = id1(v4);
    v6 = id2(v5);
q1:assert(v6!= v1);
}

id2(v){return v;}
```

Graph Reachability in Datalog



Query Tuple	Original Query
q$_1$: path(0, 5)	`assert(v6!= v1)`
q$_2$: path(0, 2)	`assert(v3!= v1)`

Input relations:

edge(i, j, n), abs(n)

Output relations:

path(i, j)

Rules:

path(i, i).

path(i, j) :- path(i, k), edge(k, j, n), abs(n).

Input tuples:

edge(0, 6, a_0), edge(0, 6', a_1), edge(3, 6, b_0),

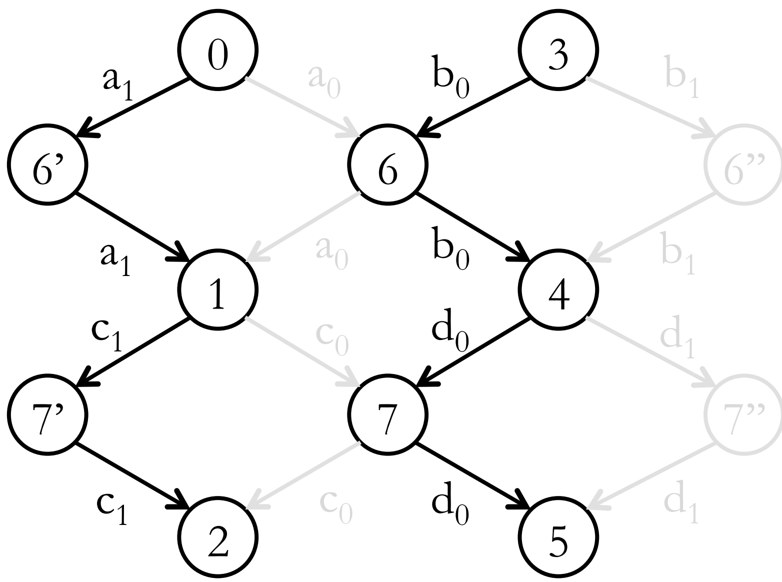
...

abs(a_0)$\oplus$abs(a_1), abs(b_0)$\oplus$abs(b_1),

abs(c_0)$\oplus$abs(c_1), abs(d_0)$\oplus$abs(d_1).

16 possible abstractions in total

Desired Result



Query	Answer
q_1: path(0, 5)	✔ $a_1 b_0 c_1 d_0$
q_2: path(0, 2)	✘ Impossibility

Input relations:

edge(i, j, n), abs(n)

Output relations:

path(i, j)

Rules:

path(i, i).

path(i, j) :- path(i, k), edge(k, j, n), abs(n).

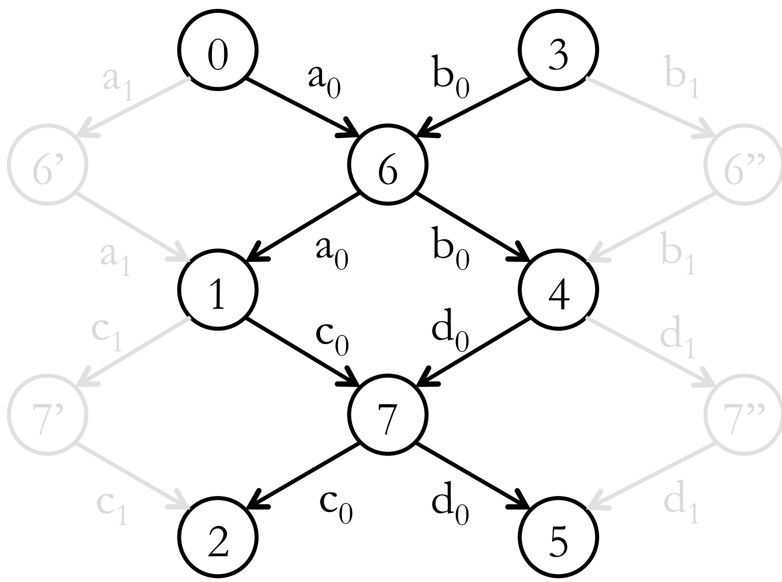
Input tuples:

edge(0, 6, a_0), edge(0, 6', a_1), edge(3, 6, b_0),

...

$\text{abs}(a_0) \oplus \text{abs}(a_1)$, $\text{abs}(b_0) \oplus \text{abs}(b_1)$,
$\text{abs}(c_0) \oplus \text{abs}(c_1)$, $\text{abs}(d_0) \oplus \text{abs}(d_1)$.

Iteration 1

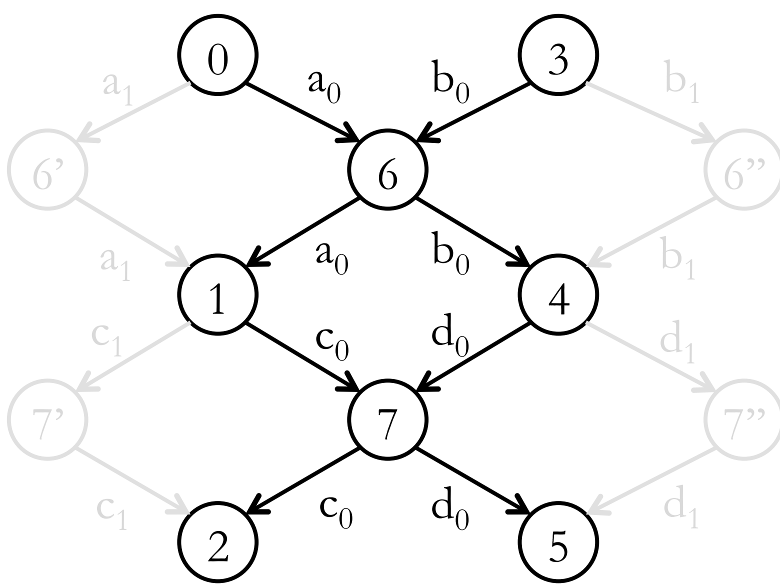


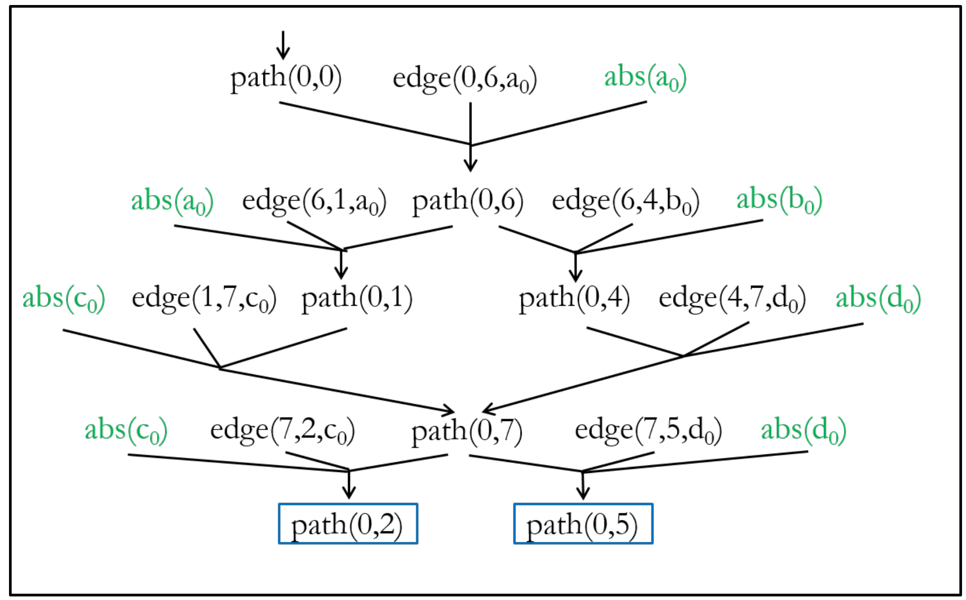
path(0, 0).
path(0, 6) :- path(0, 0), edge(0, 6, a_0), abs(a_0).
path(0, 1) :- path(0, 6), edge(6, 1, a_0), abs(a_0).
path(0, 7) :- path(0, 1), edge(1, 7, c_0), abs(c_0).
$\boxed{\text{path(0, 2)}}$:- path(0, 7), edge(7, 2, c_0), abs(c_0).
path(0, 4) :- path(0, 6), edge(6, 4, b_0), abs(b_0).
path(0, 7) :- path(0, 4), edge(4, 7, d_0), abs(d_0).
$\boxed{\text{path(0, 5)}}$:- path(0, 7), edge(7, 5, d_0), abs(d_0).
…

Query	Eliminated Abstractions
q_1: path(0, 5)	
q_2: path(0, 2)	

abs(a_0)$\oplus$abs(a_1), abs(b_0)$\oplus$abs(b_1),
abs(c_0)$\oplus$abs(c_1), abs(d_0)$\oplus$abs(d_1).

Iteration 1 - Derivation Graph

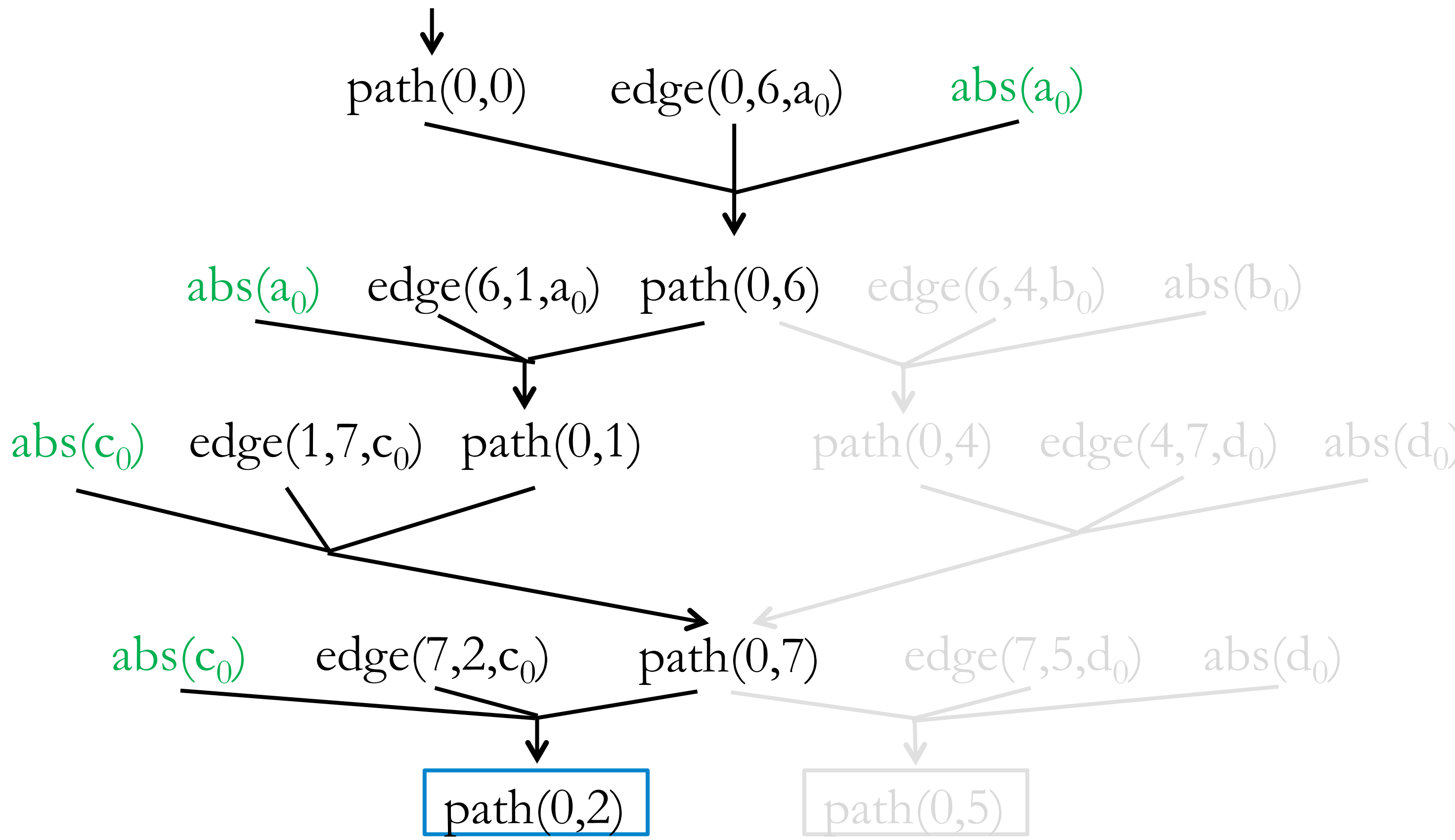




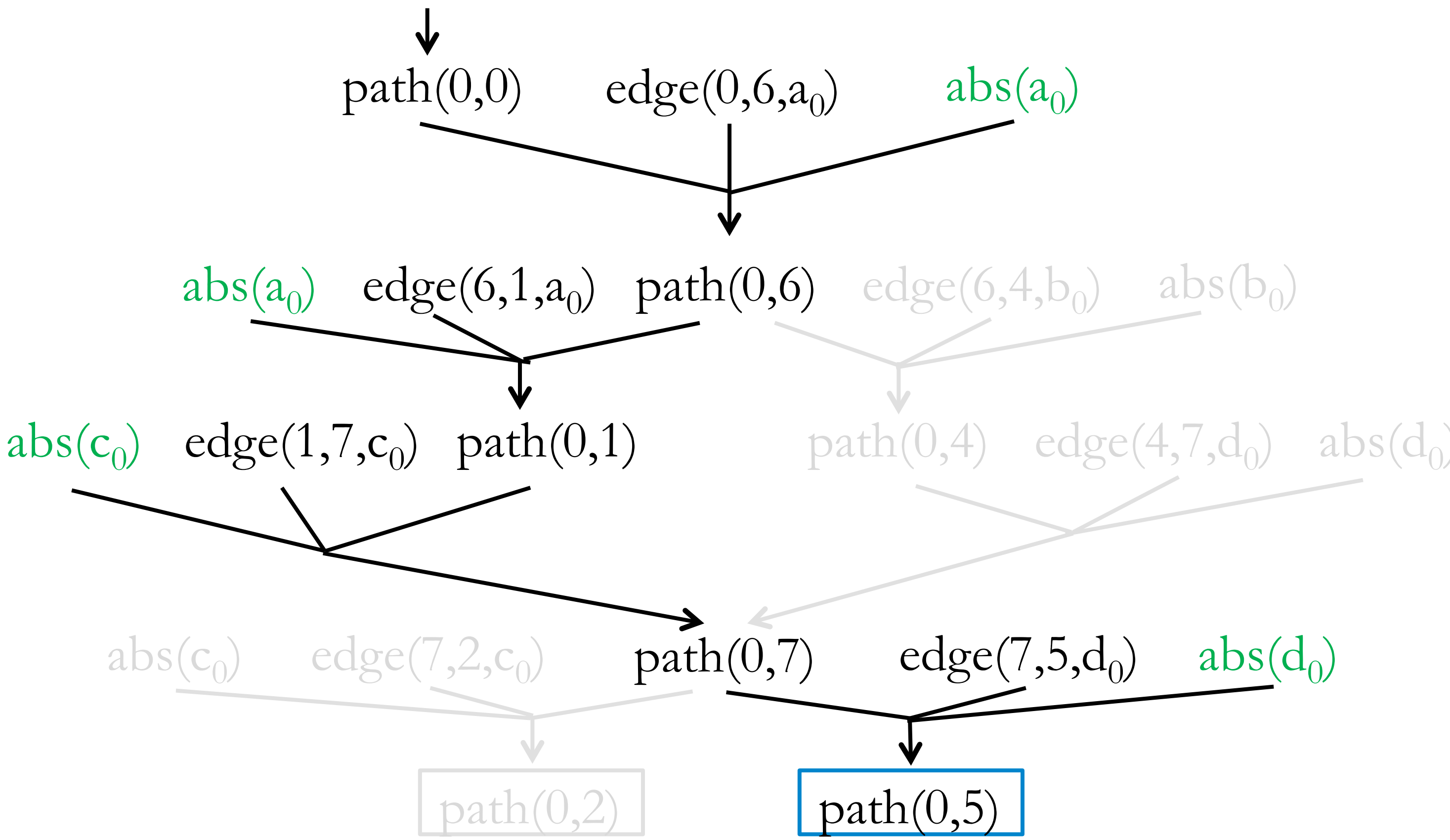
Query	Eliminated Abstractions
q_1: path(0, 5)	
q_2: path(0, 2)	

$abs(a_0) \oplus abs(a_1)$, $abs(b_0) \oplus abs(b_1)$,
$abs(c_0) \oplus abs(c_1)$, $abs(d_0) \oplus abs(d_1)$.

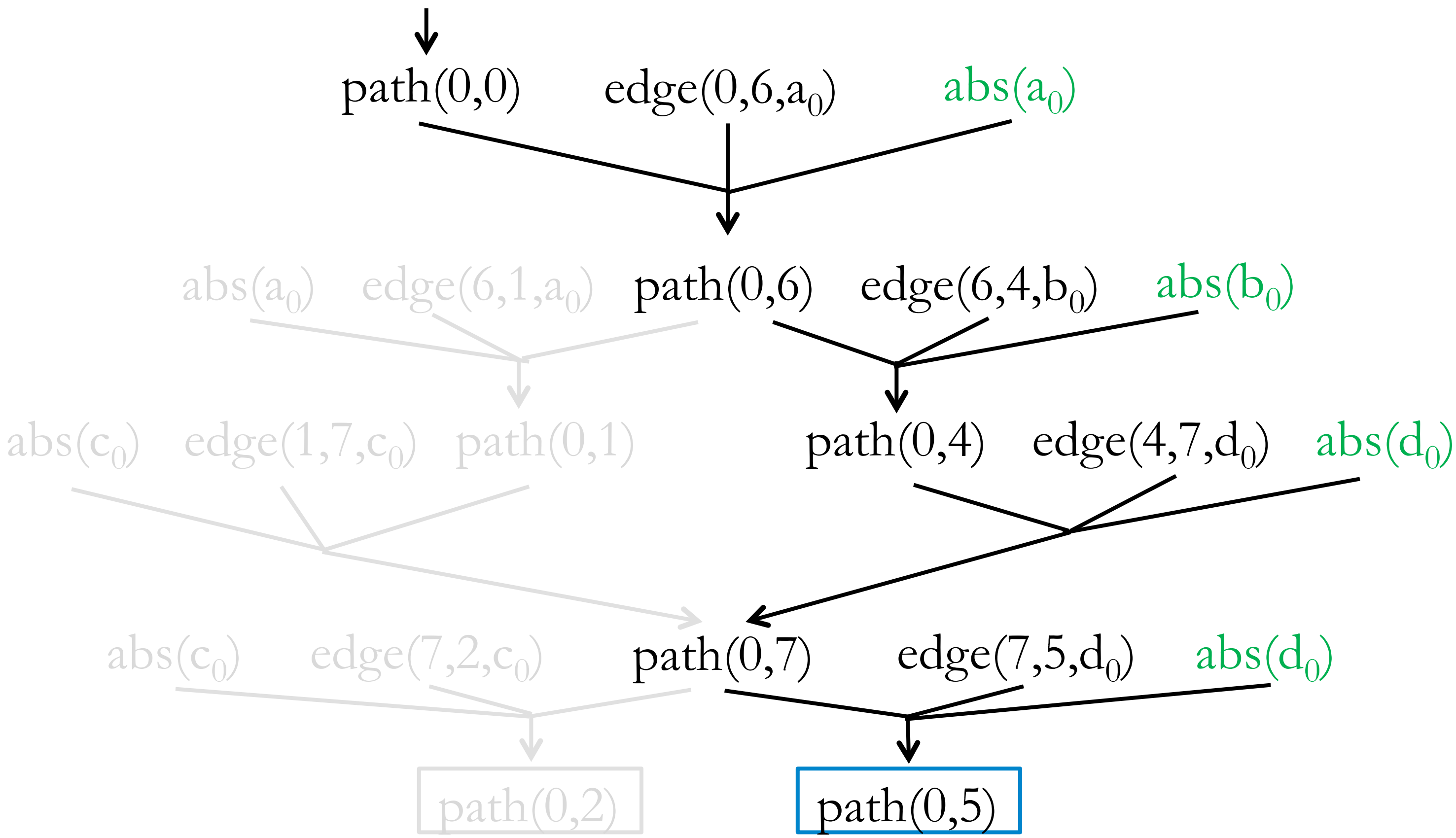
Iteration 1 - Derivation Graph



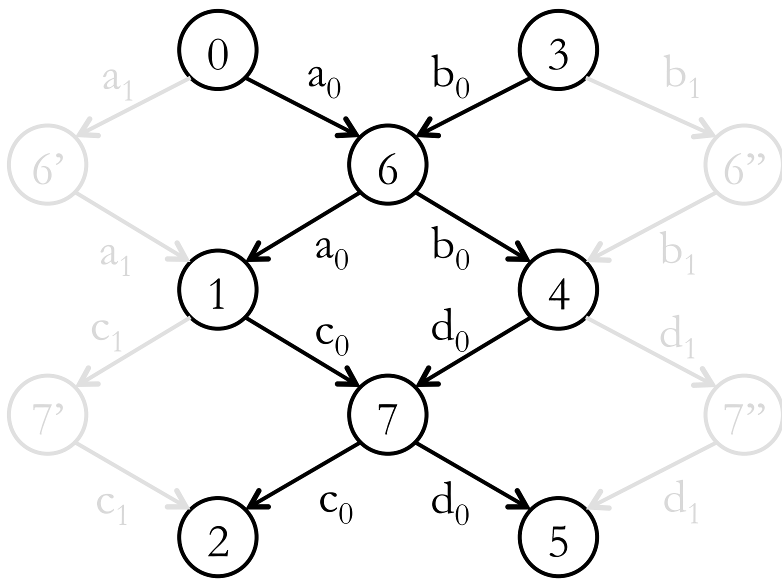
Iteration 1 - Derivation Graph

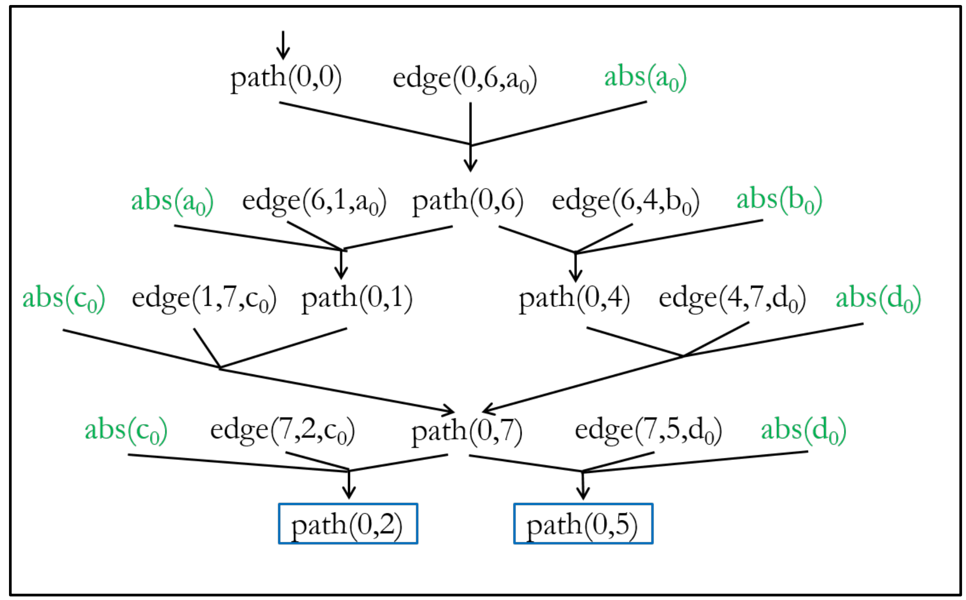


Iteration 1 - Derivation Graph



Iteration 1 - Derivation Graph

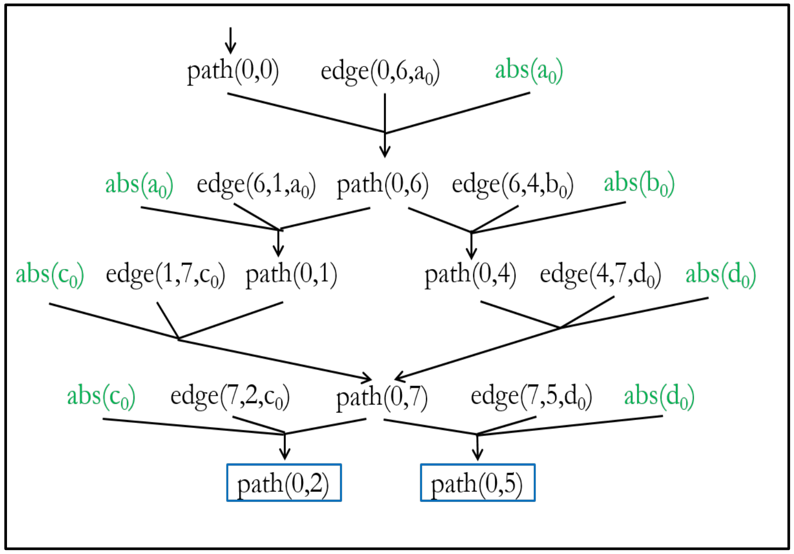




Query	Eliminated Abstractions
q_1: path(0, 5)	$a_0c_0d_0$, $a_0b_0d_0$ **(4/16)**
q_2: path(0, 2)	a_0c_0 **(4/16)**

abs(a_0)$\oplus$abs(a_1), abs(b_0)$\oplus$abs(b_1),
abs(c_0)$\oplus$abs(c_1), abs(d_0)$\oplus$abs(d_1).

Encoded as MAXSAT

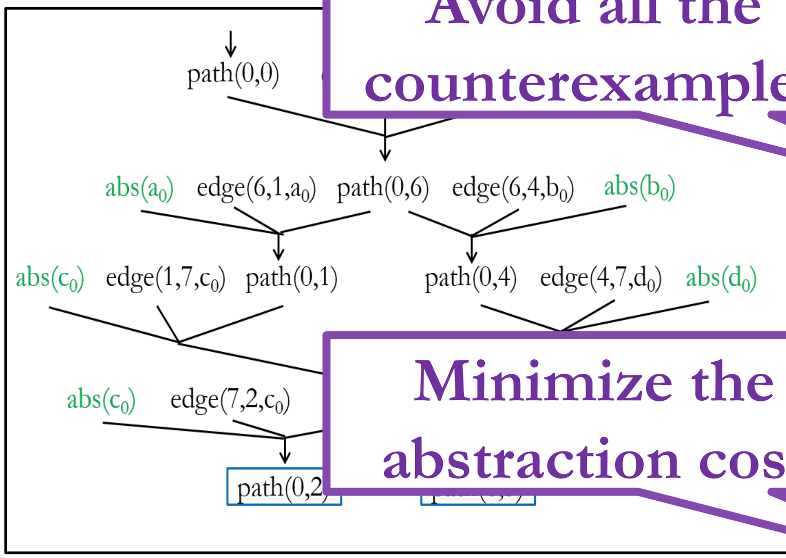


Hard Constraints **Soft Constraints**

$$\textbf{MAXSAT}(\boxed{\boldsymbol{\psi_0}}, \overbrace{(\boldsymbol{\psi_1}, \boldsymbol{w_1}), \ldots, (\boldsymbol{\psi_n}, \boldsymbol{w_n})}):$$

Find	solution S that
Maximize	$\{\psi_i \mid S \vDash \psi_i, 1 \leq i \leq n\}$
Subject to	$S \vDash \psi_0$

Encoded as MAXSAT



Avoid all the counterexamples

Minimize the abstraction cost

abs(a_0)⊕abs(a_1), abs(b_0)⊕abs(b_1), abs(c_0)⊕abs(c_1), abs(d_0)⊕abs(d_1).

Hard constraints:

$$
\begin{aligned}
&path(0,0)\ \wedge\\
&(path(0,6) \vee \neg path(0,0) \vee \neg abs(a_0))\ \wedge\\
&(path(0,1) \vee \neg path(0,6) \vee \neg abs(a_0))\ \wedge\\
&(path(0,7) \vee \neg path(0,1) \vee \neg abs(c_0))\ \wedge\\
&(path(0,4) \vee \neg path(0,6) \vee \neg abs(b_0))\ \wedge\\
&\ldots
\end{aligned}
$$

Soft constraints:

$$
\begin{aligned}
&(abs(a_0)\ \textbf{weight 1})\ \wedge\\
&(abs(b_0)\ \textbf{weight 1})\ \wedge\\
&(abs(c_0)\ \textbf{weight 1})\ \wedge\\
&(abs(d_0)\ \textbf{weight 1})\ \wedge\\
&(\neg path(0,2)\ \textbf{weight 5})\ \wedge\\
&(\neg path(0,5)\ \textbf{weight 5})
\end{aligned}
$$

Encoded as MAXSAT

Solution:

$path(0,0) = \text{true}, \quad path(0,6) = \text{false},$
$path(0,1) = \text{false}, \quad path(0,4) = \text{false},$
$path(0,7) = \text{false}, \quad path(0,2) = \text{false},$
$path(0,5) = \text{false},$

$\boxed{\begin{array}{l} abs(a_0) = \text{false}, abs(b_0) = \text{true}, \\ abs(c_0) = \text{true}, abs(d_0) = \text{true}. \end{array}}$

$\boxed{a_1 b_0 c_0 d_0}$

Query	Eliminated Abstractions
q_1: path(0, 5)	$a_0c_0d_0$, $a_0b_0d_0$ **(4/16)**
q_2: path(0, 2)	a_0c_0 **(4/16)**

Hard constraints:

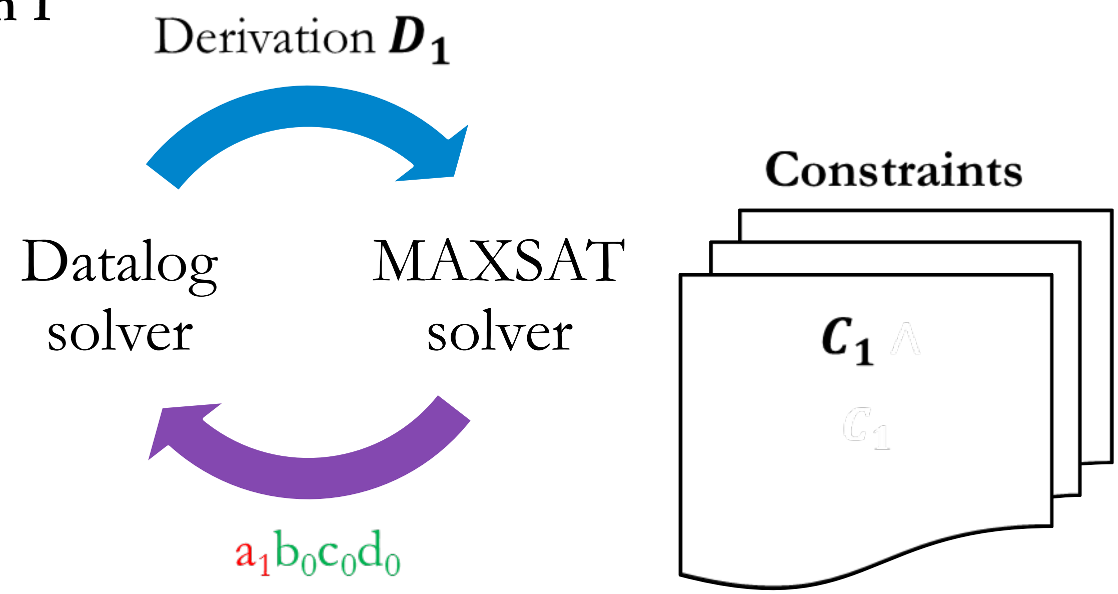
$$\begin{array}{r} path(0,0) \land \\ (path(0,6) \lor \neg path(0,0) \lor \neg abs(a_0)) \land \\ (path(0,1) \lor \neg path(0,6) \lor \neg abs(a_0)) \land \\ (path(0,7) \lor \neg path(0,1) \lor \neg abs(c_0)) \land \\ (path(0,4) \lor \neg path(0,6) \lor \neg abs(b_0)) \land \\ \ldots \end{array}$$

Soft constraints:

$$\begin{array}{r} (abs(a_0)\ \textbf{weight 1}) \land \\ (abs(b_0)\ \textbf{weight 1}) \land \\ (abs(c_0)\ \textbf{weight 1}) \land \\ (abs(d_0)\ \textbf{weight 1}) \land \\ (\neg path(0,2)\ \textbf{weight 5}) \land \\ (\neg path(0,5)\ \textbf{weight 5}) \end{array}$$

Iteration 2 and Beyond

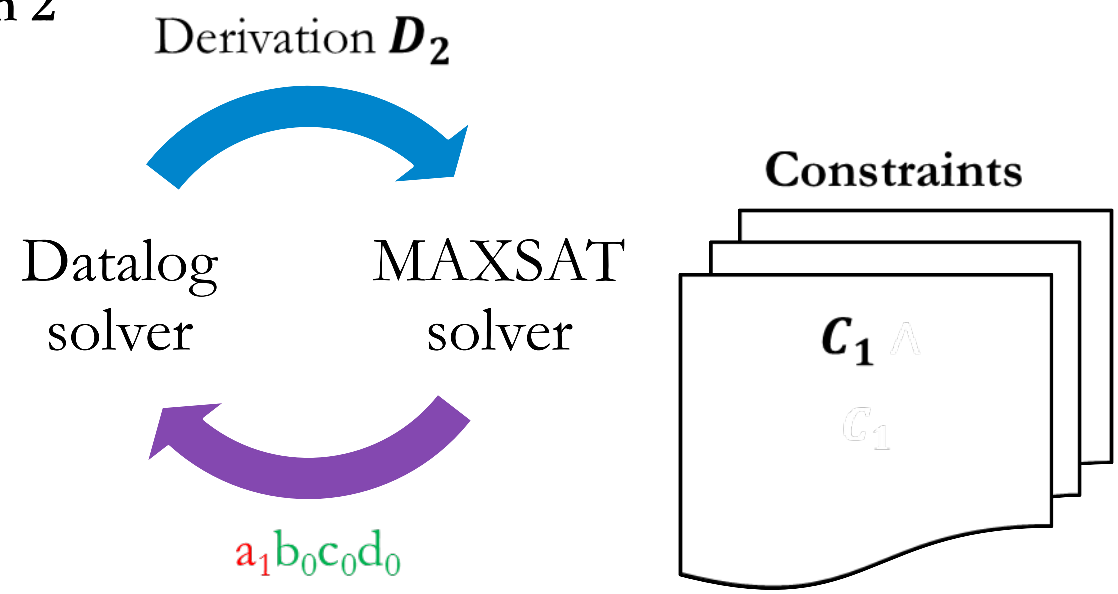
Iteration 1



Query	Answer	Eliminated Abstractions
q_1: path(0, 5)		$a_0c_0d_0$, $a_0b_0d_0$ **(4/16)**
q_2: path(0, 2)		a_0c_0 **(4/16)**

Iteration 2 and Beyond

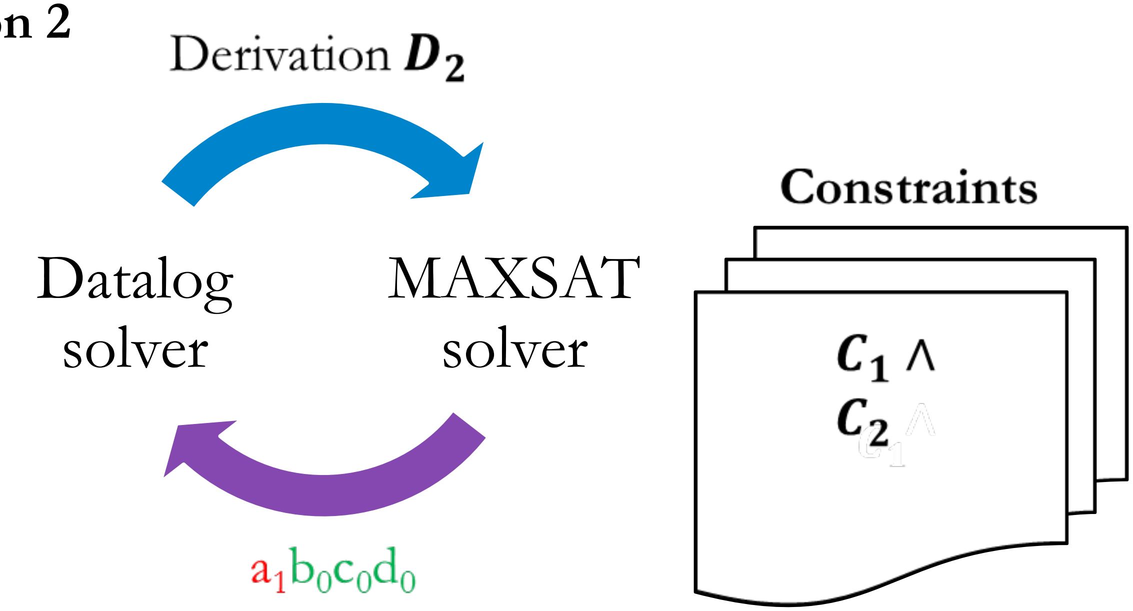
Iteration 2



Query	Answer	Eliminated Abstractions	
q_1: path(0, 5)		$a_0c_0d_0$, $a_0b_0d_0$	**(4/16)**
q_2: path(0, 2)		a_0c_0	**(4/16)**

Iteration 2 and Beyond

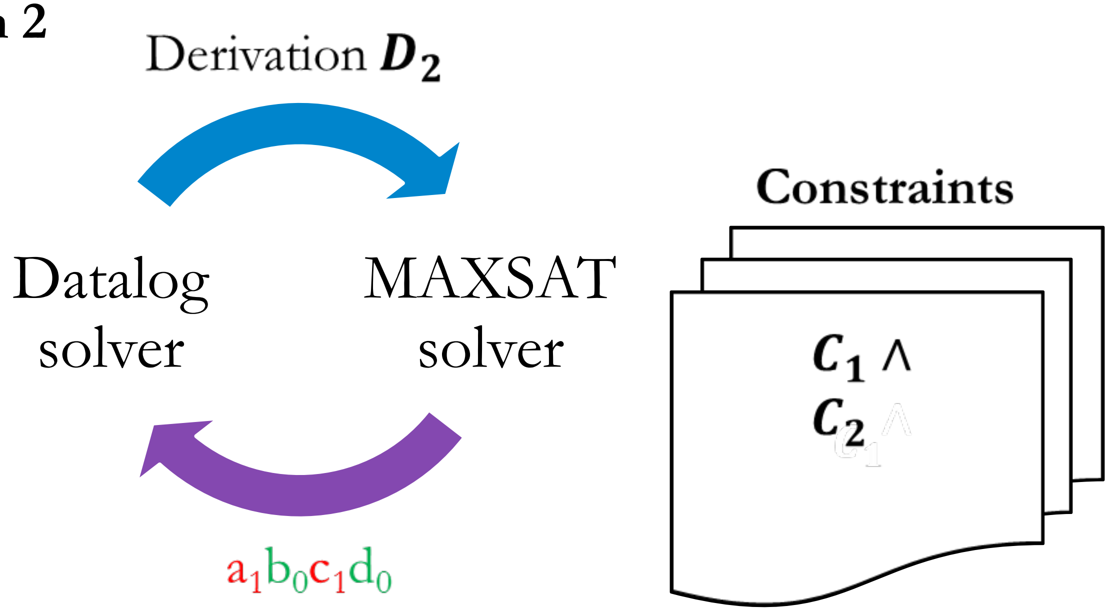
Iteration 2



Query	Answer	Eliminated Abstractions
q_1: path(0, 5)		$a_0c_0d_0$, $a_0b_0d_0$ **(4/16)**
q_2: path(0, 2)		a_0c_0 **(4/16)**

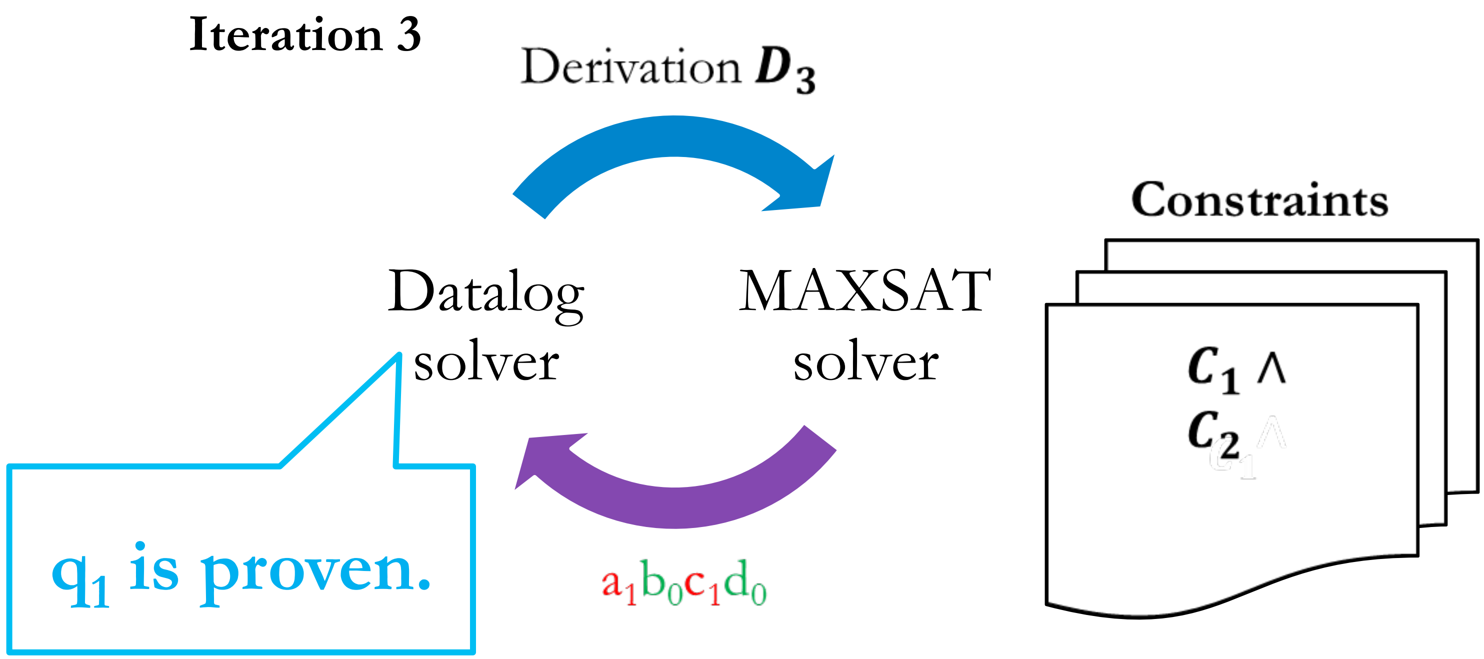
Iteration 2 and Beyond

Iteration 2



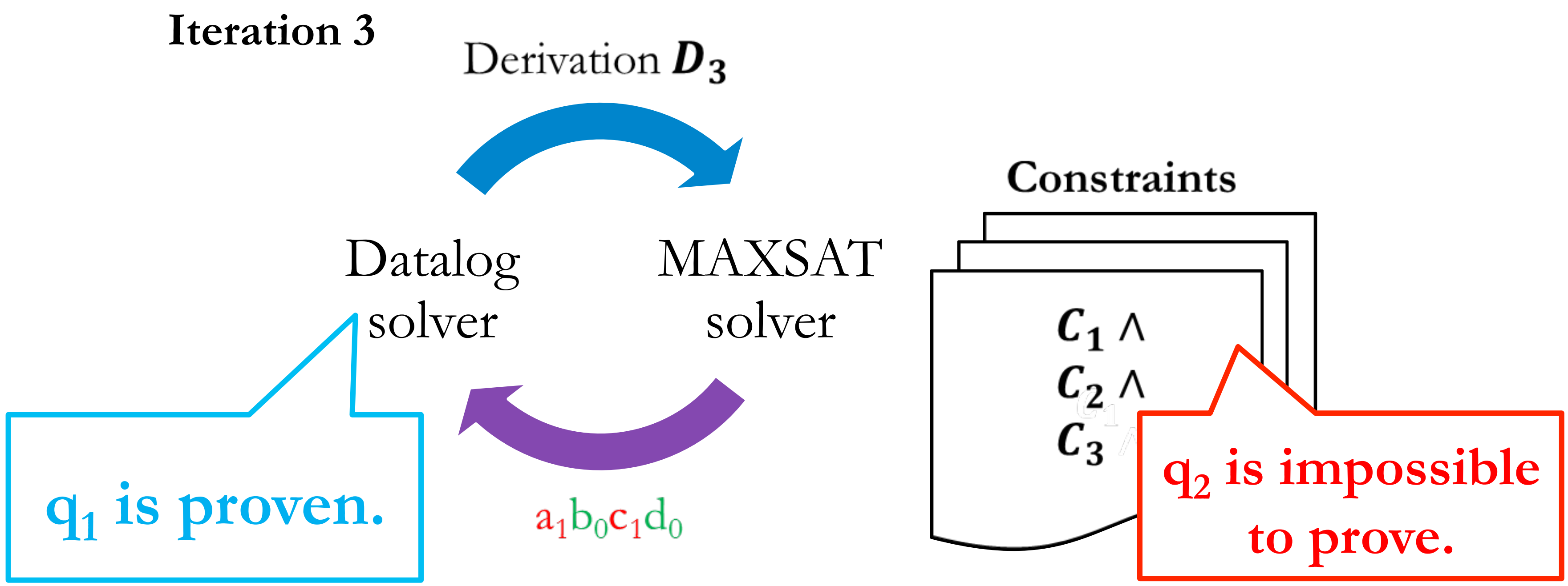
Query	Answer	Eliminated Abstractions
q_1: path(0, 5)		$a_0c_0d_0$, $a_0b_0d_0$, $a_1c_0d_0$ **(6/16)**
q_2: path(0, 2)		a_0c_0, a_1c_0 **(8/16)**

Iteration 2 and Beyond



Query	Answer	Eliminated Abstractions
q_1: path(0, 5)	✔ $a_1b_0c_1d_0$	$a_0c_0d_0$, $a_0b_0d_0$, $a_1c_0d_0$ **(6/16)**
q_2: path(0, 2)		a_0c_0, a_1c_0 **(8/16)**

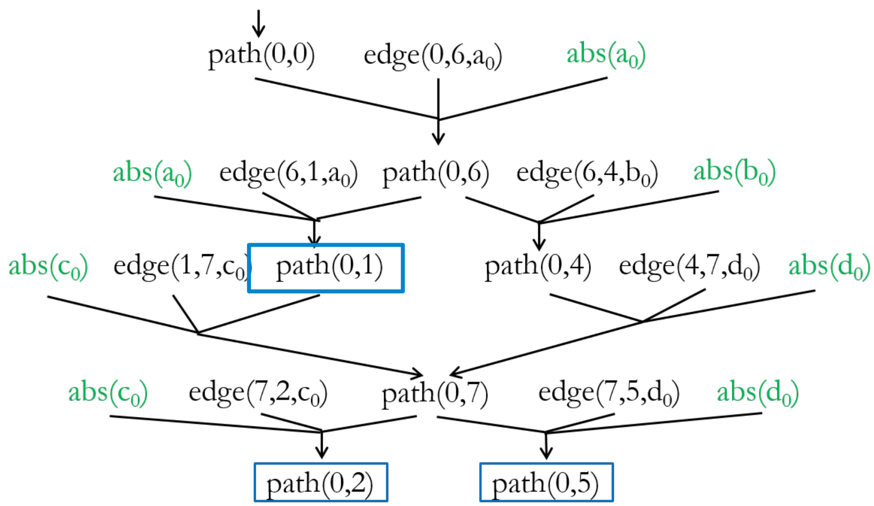
Iteration 2 and Beyond



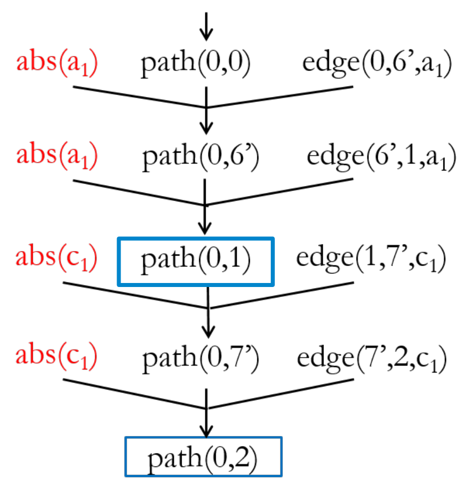
Query	Answer	Eliminated Abstractions
q_1: path(0, 5)	✓ $a_1b_0c_1d_0$	$a_0c_0d_0$, $a_0b_0d_0$, $a_1c_0d_0$ **(6/16)**
q_2: path(0, 2)	✗ Impossibility	a_0c_0, a_1c_0, a_1c_1, a_0c_1 **(16/16)**

Mixing Counterexamples

Iteration 1



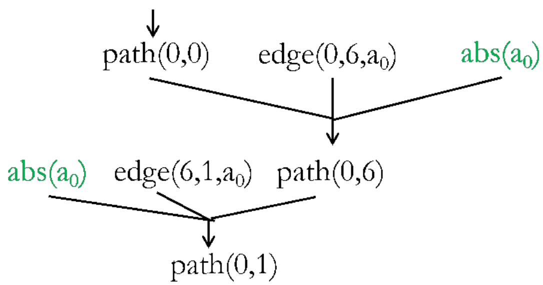
Iteration 3

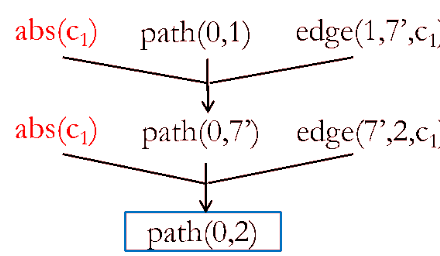


Eliminated Abstractions: $a_0 * c_0 *$ $\quad$ $a_1 * c_1 *$

Mixing Counterexamples

Iteration 1 | Mixed! | Iteration 3





Eliminated Abstractions: $a_0 * c_0 *$ | $a_0 * c_1 *$ | $a_1 * c_1 *$

Experimental Setup

- Implemented in JChord using off-the-shelf solvers:
  - Datalog: bddbddb
  - MAXSAT: MiFuMaX
- Applied to two analyses that are challenging to scale:
  - k-object-sensitivity pointer analysis:
    - flow-insensitive, weak updates, cloning-based
  - typestate analysis:
    - flow-sensitive, strong updates, summary-based
- Evaluated on 8 Java programs from DaCapo and Ashes.

Benchmark Characteristics

	classes	methods	bytecode(KB)	KLOC
toba-s	1K	6K	423	258
javasrc-p	1K	6.5K	434	265
weblech	1.2K	8K	504	326
hedc	1K	7K	442	283
antlr	1.1K	7.7K	532	303
luindex	1.3K	7.9K	508	295
lusearch	1.2K	8K	511	314
schroeder-m	1.9k	12K	708	460

Results: Pointer Analysis

	queries			size		iterations
	total	resolved				
		current	baseline	final	max	
toba-s	7			170	18K	10
javasrc-p	46			470	18K	13
weblech	5	5	2	140	31K	10
hedc	47	47	6	730	29K	18
antlr	143	143	5	970	29K	15
luindex	138	138	67	1K	40K	26
lusearch	322	322	29	1K	39K	17
schroeder-m	51	51	25	450	58K	15

4-object-sensitivity
< 50%

< 3% of max

Performance of Datalog: Pointer Analysis

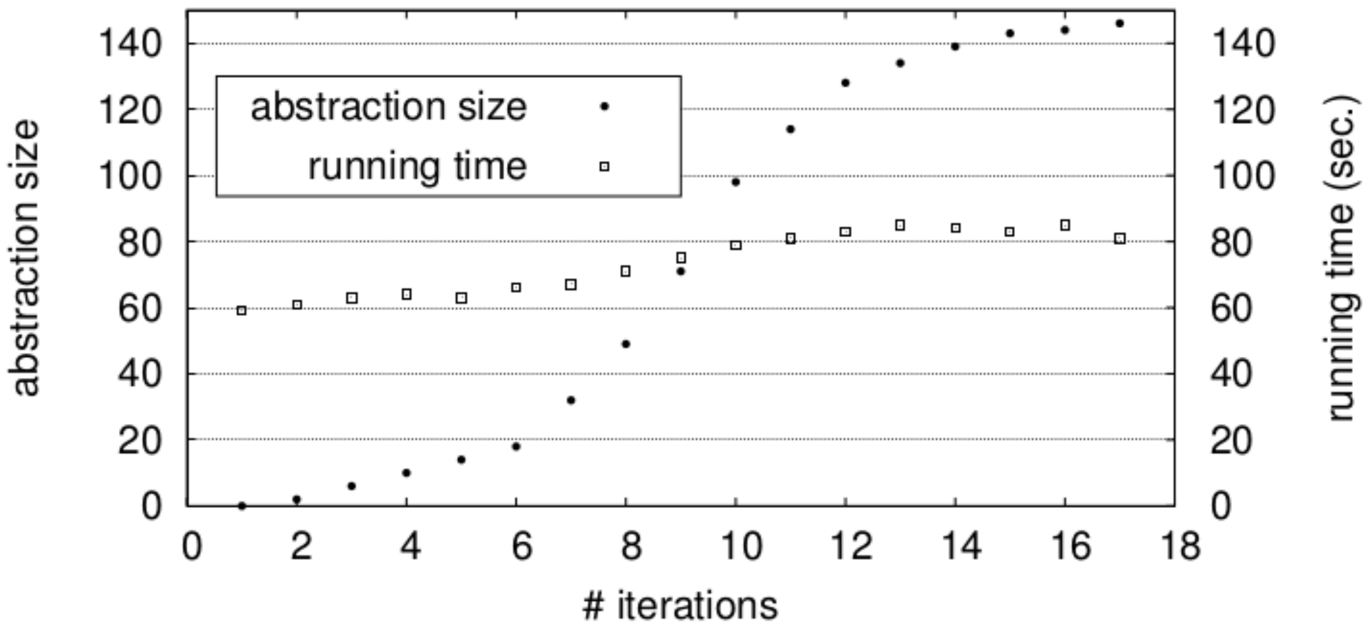
k = 4, 3h28m

Baseline k = 3, 590s

k = 2, 214s

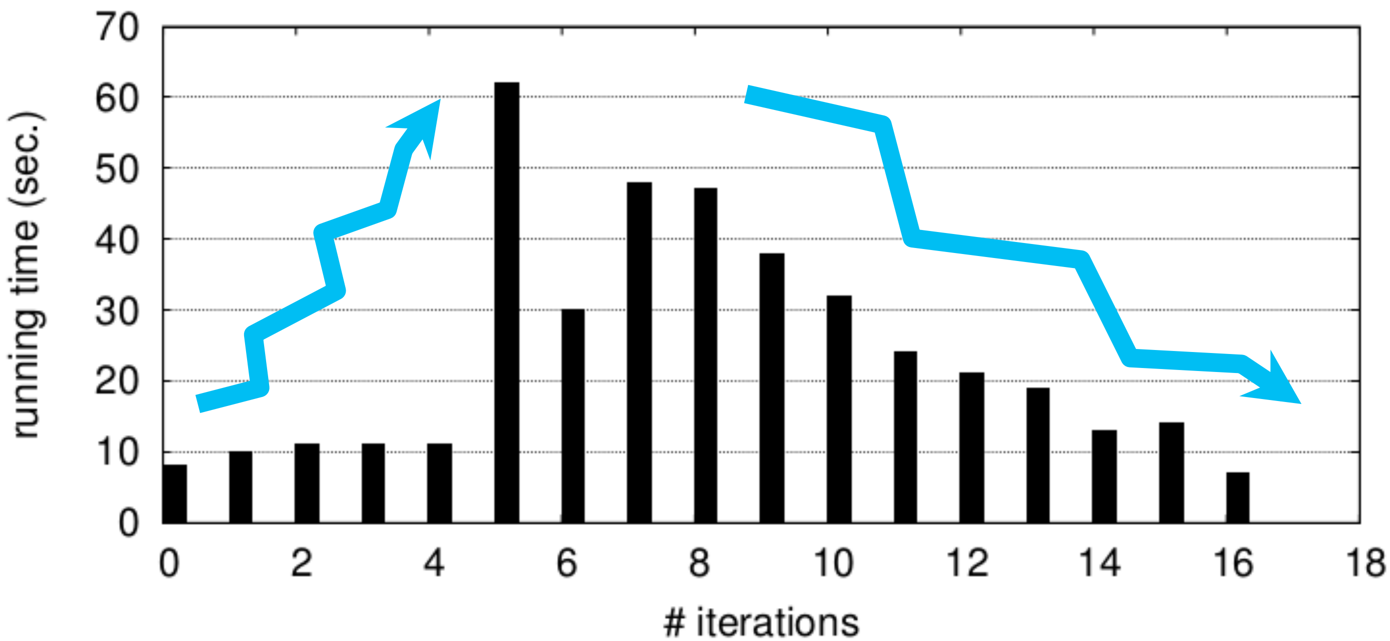
lusearch

k = 1, 153s



Performance of MAXSAT: Pointer Analysis

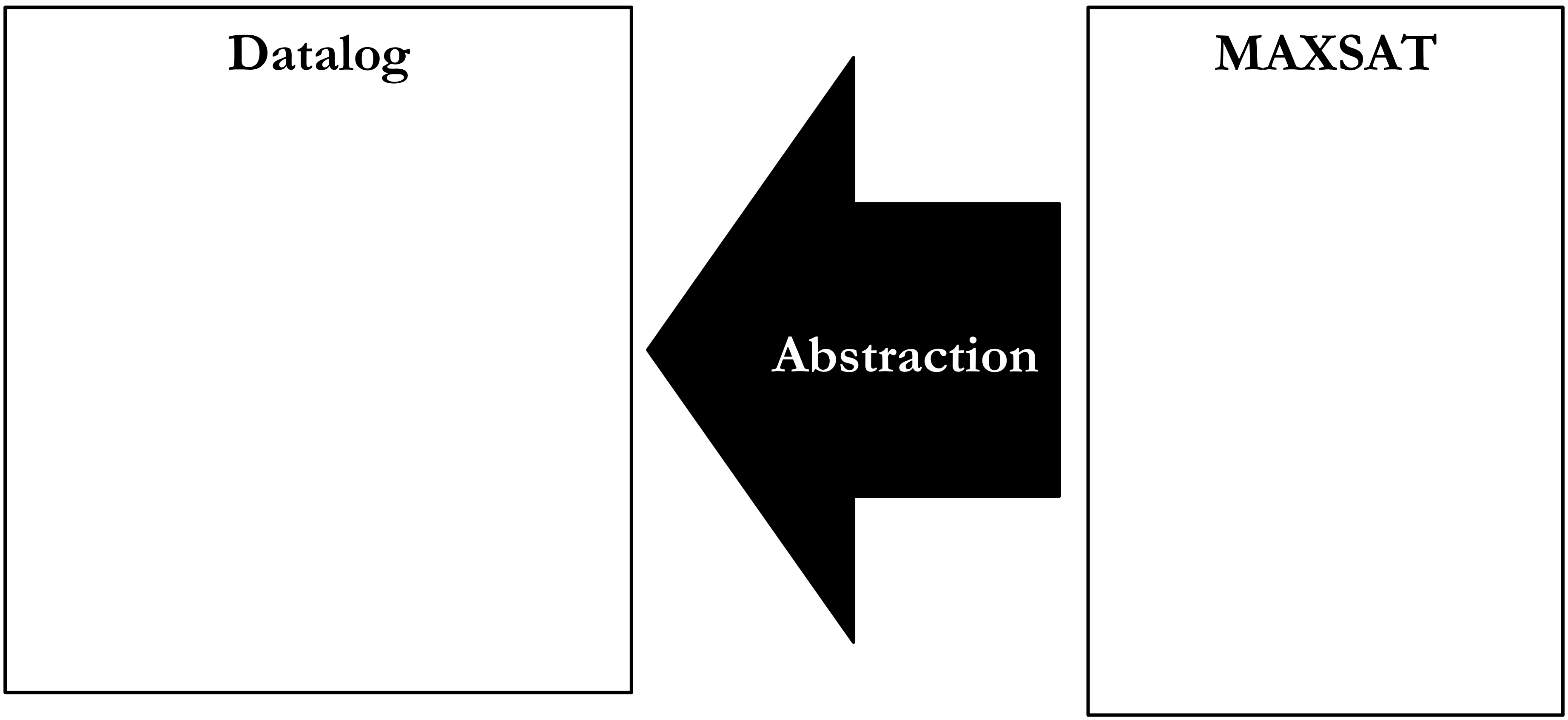
lusearch



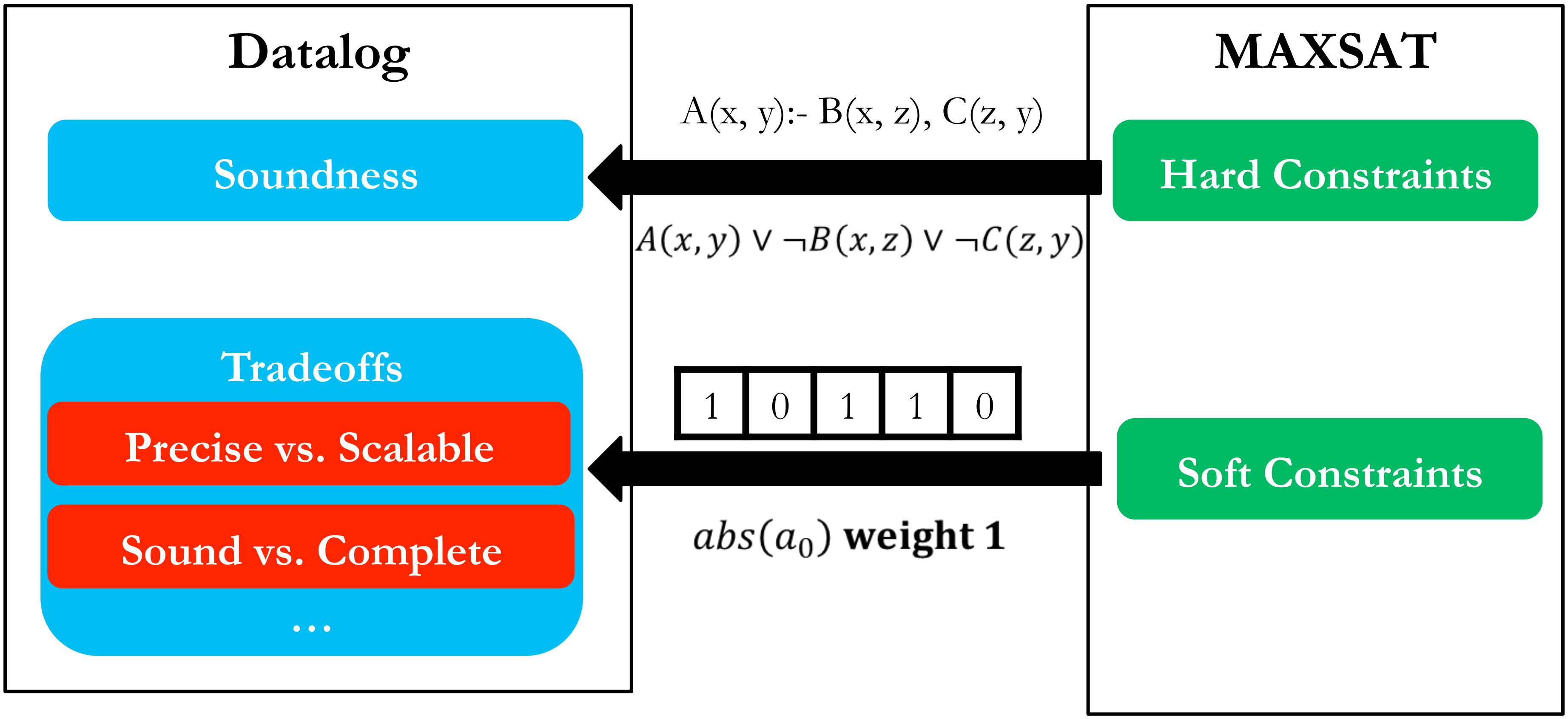
Statistics of MAXSAT Formulae

	pointer analysis	
	variables	**clauses**
toba-s	0.7M	1.5M
javasrc-p	0.5M	0.9M
weblech	1.6M	3.3M
hedc	1.2M	2.7M
antlr	3.6M	6.9M
luindex	2.4M	5.6M
lusearch	2.1M	5M
schroeder-m	6.7M	23.7M

Conclusion

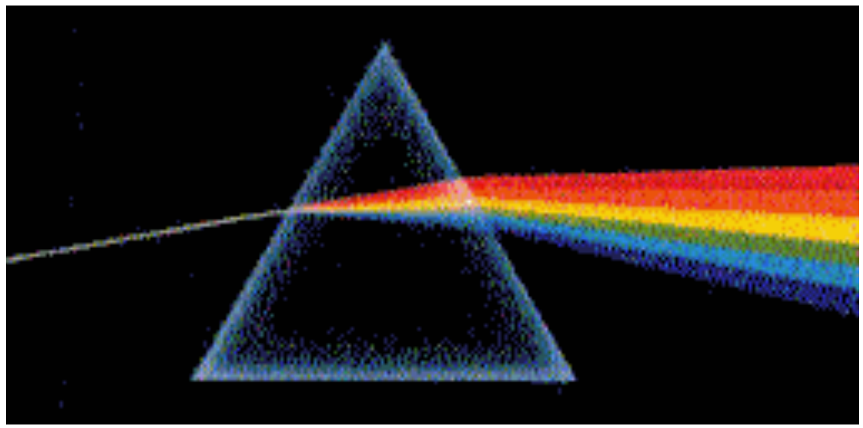


Conclusion



Datalog
Soundness
Tradeoffs
Precise vs. Scalable
Sound vs. Complete
…
A(x, y):- B(x, z), C(z, y)
$A(x,y) \vee \neg B(x,z) \vee \neg C(z,y)$
1 0 1 1 0
$abs(a_0)$ weight 1
MAXSAT
Hard Constraints
Soft Constraints

Thank You!



http://pag.gatech.edu/prism