How to find a good program abstraction automatically?

Hongseok Yang University of Oxford

Joint work with Ravi Mangal, Mayur Naik, Xin Zhang (Georgia Tech), Kihong Heo, Wonchan Lee, Hakjoo Oh, Kwangkeun Yi (SNU), Radu Grigore (Oxford) Mooly Sagiv, Ghila Castelnuovo (Tel-Aviv)

How to find a good program abstraction automatically? + some old story

Hongseok Yang University of Oxford

Joint work with Ravi Mangal, Mayur Naik, Xin Zhang (Georgia Tech), Kihong Heo, Wonchan Lee, Hakjoo Oh, Kwangkeun Yi (SNU), Radu Grigore (Oxford) Mooly Sagiv, Ghila Castelnuovo (Tel-Aviv)

How to find a good program abstraction automatically? + some old story

Hongseok Yang University of Oxford

Joint work with Ravi Mangal, Mayur Naik, Xin Zhang (Georgia Tech), Kihong Heo, Wonchan Lee, Hakjoo Oh, Kwangkeun Yi (SNU), Radu Grigore (Oxford) Mooly Sagiv, Ghila Castelnuovo (Tel-Aviv) A problem has been detected and Windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer, If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x0000001 (0x000000C,0x0000002,0x00000000,0xF86B5A89)

WWW

qv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory Physical memory dump complete. Contact your system administrator or technical support group for further assistance. You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを 数秒間押し続けるか、リセットボタンを押してください。

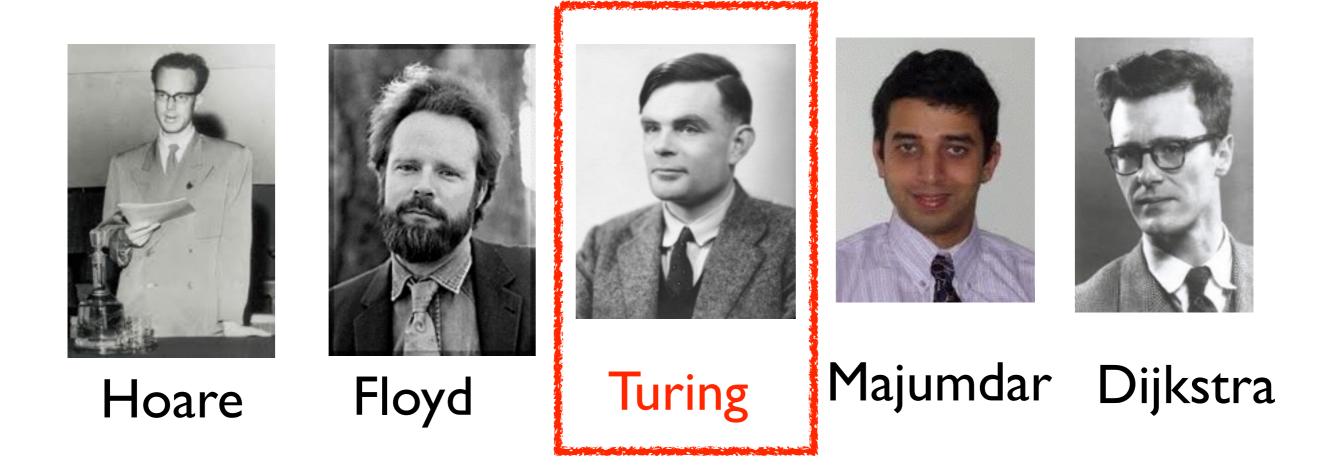
Software verification

- Active research area in computer science.
- Aims at verifying "no blue screen", i.e., programs do not have errors.
- Develops methods for such verification.

[Quiz] Who wrote the earliest paper on software verification?



[Quiz] Who wrote the earliest paper on software verification?



Turing in June 1949

Friday, 24th June, .

Checking a large routine. by Dr. A. Turing.

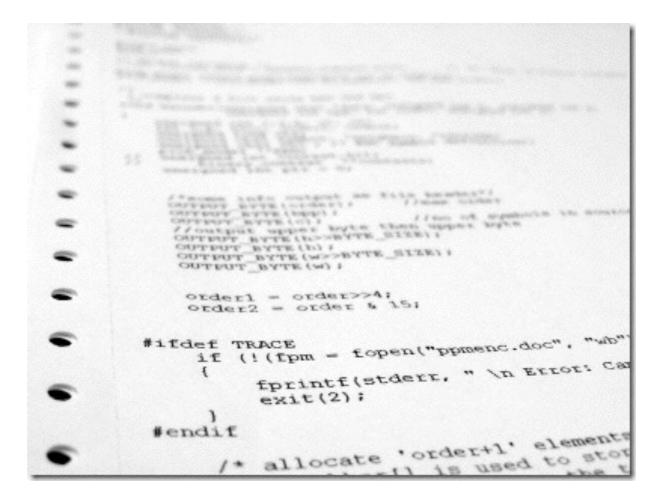
How can one check a routine in the sense of making sure that it is right?

Turing's idea

Use intermediate assertions.

Turing's idea

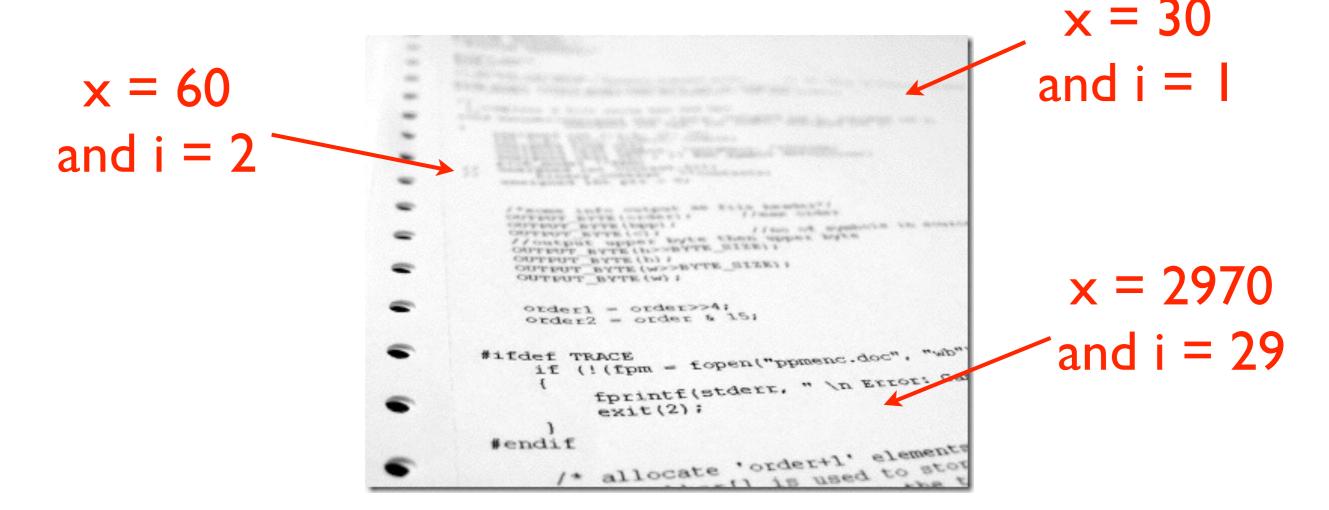
Use intermediate assertions.



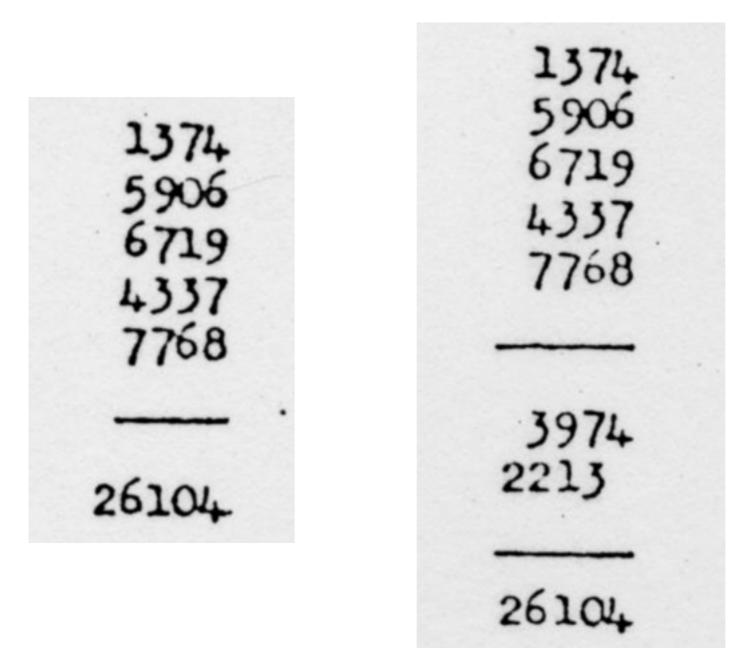
Verify that this program computes "100 * 30".

Turing's idea

Use intermediate assertions.



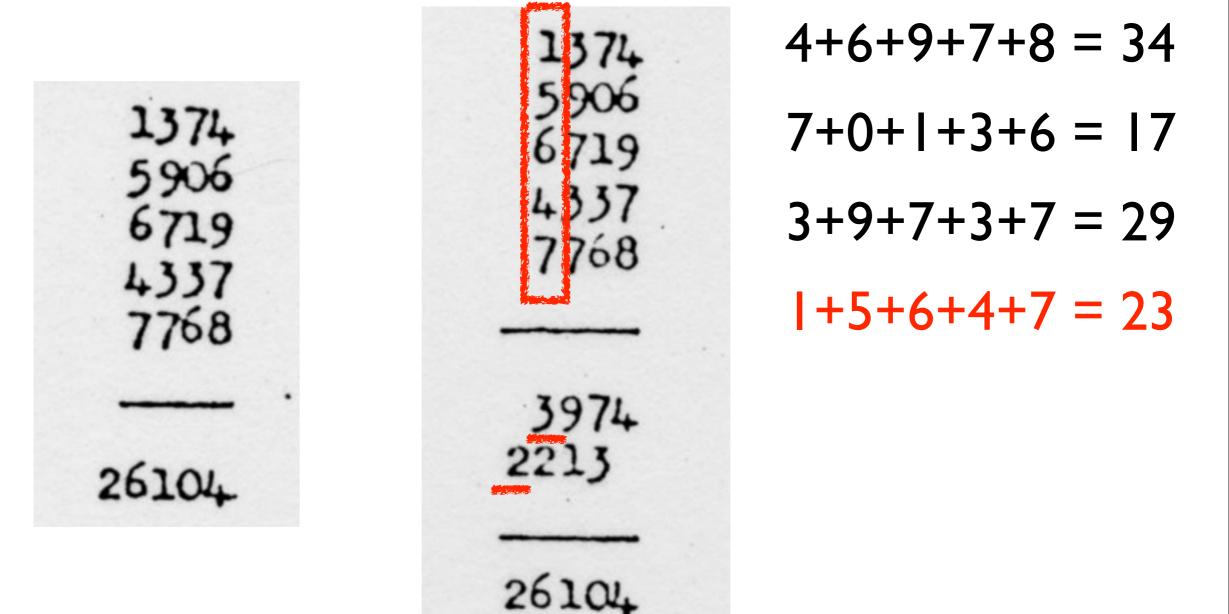
Verify that this program computes "100 * 30".



4+6+9+7+8 = 34

4+6+9+7+8 = 34 7+0+1+3+6 = 17

4+6+9+7+8 = 347+0+1+3+6 = 173+9+7+3+7 = 29



4+6+9+7+8 = 347+0+1+3+6 = 173+9+7+3+7 = 291+5+6+4+7 = 23

Intermediate assertions

- Form the basis of modern verification methods.
- Inferred automatically by commercial tools nowadays.

Microsoft SDV

View Trace Tree Help		_ 8 ×
	Source Code	fal_driver1
sdv_CheckDriverUnload		- 😵 Defect(s)(5)
switch (choice) {	123: DispatchRead (- Spinlock
if (sdv_dispatch_routines)		- 😢 injkesetevent - 😢 lowerdriverretum
sdv stub driver init	124: IN PDEVICE_OBJECT DeviceObject, 125: IN PIRP Irp	- 🙆 irglioapcite
sdv_RunDispatchFunction		- 😢 cancelspinlock
: int x = SdvMakeChoice();	127: (Properties
: UCHAR minor_function = (U	128: /*	
: PIO STACK LOCATION ps = SI	129: This defect is injected for the "SpinLock" ru	
: sdv_end_info = sdv_start :	130: */	
: sdv_end_info = sdv_start_:	131: KSPIN LOCK queueLock;	
sdv_SetStatus	132: KIRQL oldIrgl;	
: pirp->CancelRoutine = NUL	133:	
: ps->MinorPunction = minor	134: UNREFERENCED_PARAMETER(DeviceObject);	
: ps->CompletionRoutine = NI	135: UNREFERENCED_PARAMETER(Irp);	
: sdv_dpc_io_registered = Pi		
: sdv_stub_dispatch_begin	<pre>137: KeAcquireSpinLock(&queueLock, &oldIrql);</pre>	
: switch (x) {	138:	
: ps->MajorFunction = IRP_M	139: return STATUS_SUCCESS;	
: DispatchRead	140: }	
37: adv_KeAcquireSpinLock	141:	
37: SLIC_sdv_KeAcquireSpinL	142: NTSTATUS	
<pre>39: return STATUS_SUCCESS;</pre>	143: DispatchPower (
39: SLIC_DispatchRead_exit	144: IN PDEVICE_OBJECT DeviceObject,	
•	145: IN PIRP Irp	
	146:)	
	147: {	
	148: NTSTATUS status;	
ate:	· · · · · · · · · · · · · · · · · · ·	
	File: .Val_driver1.c Line: 137, Function 'DispatchRead'	lesults

Cr Coverity at 1 32:	v Prevent	Admin User Sign out Prefere	ences Help	About Jump to	CID:
37: Coverity* Integrity Manager	Malle	Dashboard	Projects	Configuration	Administrati
50: 52: Projects >> Covtel Operator I	Phone 1000	(~
53:		n/tmp/linux/linux-2.4.28/net/sctp/ulpevent.c	ts Source	Metrics Trend	as Dashboard
761: Files Components Defect 762: Resource leak 764: In	Calling allocation function "skb_copy_expand(c skb = skb_copy_expand(c)	xpand". [show details]			
767: sctp_ulpevent_make_send_failed() Leak of a system resource such as memory, file handles, or sockets (CWE-404).	462 0), /* toilroom */ (fp);			
775: Defect Impact: The system resource will not be reclaimed and reused, reducing the future availability of the resource. More information	465 goto fail; 466	chunk beader and DATA header. */			
778: Project Impact: Also detected in 2 other projects (show)	468 skb_pull(skb, sizeof(st	truct_sctp_data_chunk)); {/linux-2.4.28/include/linux/skbuff.h			
780 : 782 : In 🛶 Linux 2.4 full		eits pointer parameter skb . char * skb_pull(struct sk_buff *skb, unsigned int le	n)		
956: Events contributing to defect: 958: var_assign (ulpevent.c:460)	847 { 848 if (len > skb->) 848 return				
alloc_fn (ulpevent.c:460) var assign (skbuff.c:719)	850 return _skb_pu				
1 alloc_fn (skbuff.c:719) noescape (skbuff.c:724)	469 len -= sizeof(struct sc	tp_data_chunk);			
e-1 noescape (skbuff.h:913)	470 471 /* Embed the event fiel	ds inside the cloned skb. */			
moescape (skbuff.c:727)		in function "sctp_skb2event". [show details]			
noescape (skbuff.h:788)	472 event = sctp_skb2event(472				
noescape (skbuff.c:733)	473 sctp_ulpevent_init(even 474	<pre>k, HSG_NOTIFICATION);</pre>			
noescape (skbuff.c:412)	Variable "skb" is not freed or pointed-to	in function "skb_push". [show details]			
t sta return_alloc (skbuff.c:734)	475 ssf = (struct sctp_send				
noescape (ulpevent.c:468)	476 skb_push(skb, s 477	sizeof(struct sctp_send_failed));			
noescape (skbuff.h:846)	478 /* Socket Extensions fo	IF SCTP			
noescape (ulpevent.c:472)	473 * 5.3.1.4 SCTP_SEND_FA	ILEO			
noescape (upevent.h:75)	450 * 451 * ss≠_type:				
noescape (ulpevent.c:475)	482 * It should be SCTP_SE	NO_FAILED.			

32: Coveri 37: Coveri 50: Integri 52: Projects >> C 53: Files Compo .761: Resource le .764:	Example 1: scenarios State of the scenarios	Analyzed file: /invalid/path/scenarios.c Driginal source: C/Prples/scenarios/src/scenarios.c ()
37: Coveri 50: Integri 52: Projects >> C 53: Files Compo 761: Resource le	Local settings Preprocessing Mapping to original sources Reports Analysis options	25 30 /* 26 39 * Type cast causing overflow. 27 40 */ 28 * SPEED_SENSOR; 91 # SPEED_SENSOR; shboard
767: sctp_ulpevent Leak of a syste memory, file ha (CWE-404). 769: (CWE-404). 771: Defect Impact will not be reck reducing the fu resource. More 776: Project Impact other projects 780: In ma Linux 2.4 956: Events contribut	 Analysis start (man) ParaBelization A81 Global directives General Domoins Output: Files	<pre>29 29 29 29 29 29 29 29 29 29 29 29 29 2</pre>
var assign (ul		C 2 Column 0 Line 49, Column 0
e 1. var assign (
E 1 alloc_fn (skb		Emors X Alams Net analyzed Coverage Files
-1: noescape (sk		⊕ 2 (2) 5 (5) 0 100% somarios.c
E-1: noescape (sk		Alams Overflow in conversion
moescape (Out-of-Sound array access Possible overflow upon dereference Possible overflow upon dereference
noescape (sk	Errors: 2(2)	Assertion failure
noescape (Alarna: 5(5) Warnings: 1	Definite runtime error during assignment in this context. Analysis stopped for this context. Definite runtime error during assignment in this context, Analysis stopped for this context.
return_alloc	Coverage: 100%	
t sta noescape (up)		Summary Warnings Log Graph Watch Messages

Cov	Project Analysis Editor	Grammatech coo	
		CODESONAR Search code in this analysis for	Search 8 Advanced Search
Coveri Integri Projects >> C	Local settings	me > gnuchess-5.07-eck > gnuchess-5.07-eck analysis 2 > Warning 91.121 ata Race at input c:142 No properties have been set. edit properties not be warning location warning details	Text XML ReML Visible Code: all
Files Compo	Participants opposition	w Events Change View Options	C) thread 3
Resource le In sctp_ulpevent	Panalysis start (no ma Panalysis start (no ma (h) (2) (2) (2)	eed 1 in ome/banjaminy/Sandbaxes/TRUNK_CLEAN/codesonar-tests/regression/hockbanch ushess-5.07-rok.temp/gnuchess-5.07/bro/main.c)	thread 2 input_func (/bome/benjisminy/Sandboxes/TRUNK_CLEAN/codesonar-lests/regression/hookbenc /gnuchess-5.07-rok.templgnuchess-5.07/src/input.c)
Ceak of a syste memory, file ha (CWE-404).	Domains	290 int main (int argo, char *argv[]) A Event 1: Thread 1 starts here. * hide 291 (void *input_func(void *arg_attribute_((unused))) A Event 22: Thread 2 starts here. * hide
will not be recl reducing the fu resource. More	Cospic Files	292 int 1: 293 294 /* 295 * Parse command line arguments conforming with	<pre>121 char prompt[MAXSTR] = ""; 122 123 % while (!(flags & QUIT)) (124 g' if (!(flags & XBOARD)) (</pre>
other projects			125 sprintf(prompt,"%s (%d) : ", 126 EealSide 7 "Black" : "White", 127 (RealGameCnt+1)/2 + 1);
In the Linux 2.4 Events contribu		<pre>450 RealSide = board.side; 451 dbg_printf("Waking up input\n"); 452 dbg_printf("input_status = %d\n", input_status); 453 [-] input_wakeup();</pre>	128) 129 pthread_mutex_lock(&input_mutex); 130 gnuchess_getline(prompt); 131 input_status = INFUT_AVAILABLE;
var_assign (ul alloc_fn (ulpe) var_assign ()		Input_wakeup (/home/benjaminy/Sandbaxes/TRUNK_CLEAN/codesonar-tests/regression /hookbench/gnuchess-5.07-rok.temp/gnuchess-5.07/bro/input.c)	<pre>132 pthread_cond_signal(Ginput_cond); 133 pthread_mutex_unlock(4input_mutex); 134 135 pthread_mutex_lock(4vakeup_mutex);</pre>
alloc_fn (skb noescape (sk noescape (s		<pre>void input_wakeup(void) 151 (152 153 pthread mutex lock(&input mutex);</pre>	136 /* 137 * Posix waits can wake up spuriously 138 * so we must ensure that we keep waiting 139 * until we are woken by something that has 140 * consumed the input
noescape (sk		154 input_status = INFUT_NONE7 Data Race	141 */ 142 while (input status == INFUT AVAILABLE) (
noescape (sk noescape (t return_alloc	Errorsi 2 (2) Alarns: 5 (5) Warnings: 1 Coverage: 190%	This code writes to input_status. • The other thread reads from input_status. See other access. • The following locks are currently held: input_mutes. • None of these locks are held by the other thread when it accesses input_status so a face may occur.	Deta Race This code reads from input_status. • The other thread writes to input_status. See other access. • The following locks are currently held: valueup_mutex. • None of these locks are held by the other thread when it access
noescape (ulp- noescape (sk	Duration: 30s Connected to localhost: 1059	The issue can occur if the highlighted code executes. Show: All events Only primary events	Input_etatus to a tace may occur. The issue can occur if the highlighted code executes.

Abstraction

- Key idea behind automation.
- Keeps only important properties of programs. Forgets all the rest.

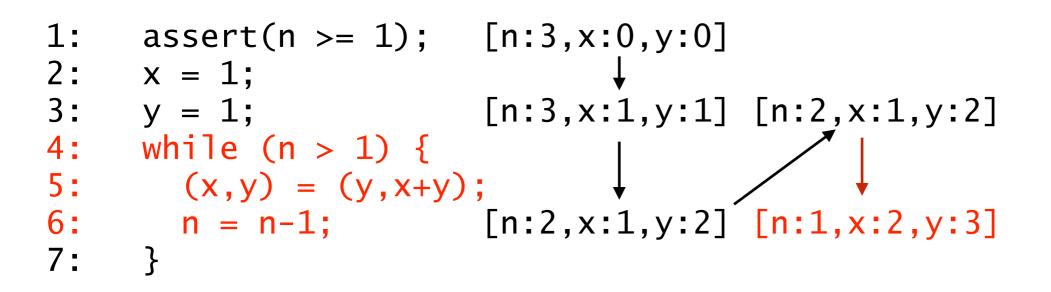
```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
```

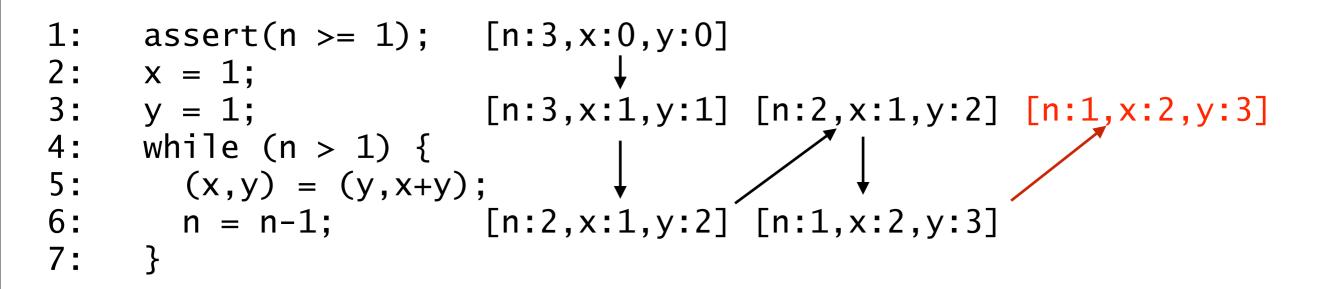
```
1: assert(n >= 1); [n:3,x:0,y:0]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
```

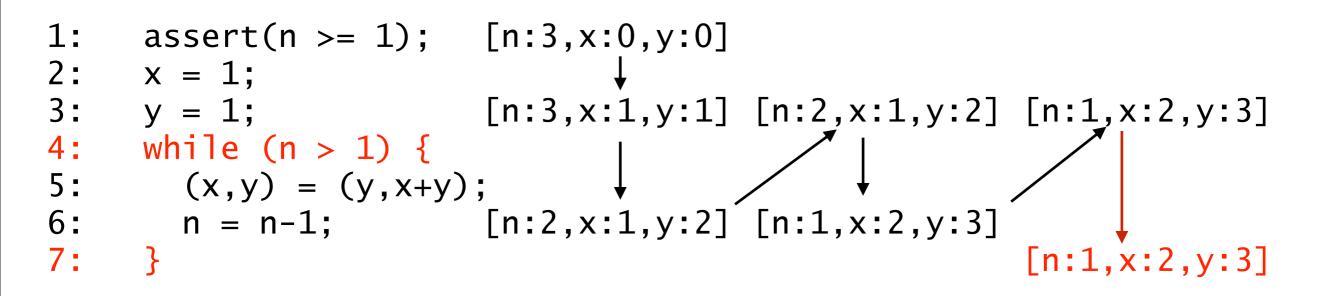
```
1: assert(n >= 1); [n:3,x:0,y:0]
2: x = 1; ↓
3: y = 1; [n:3,x:1,y:1]
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
```

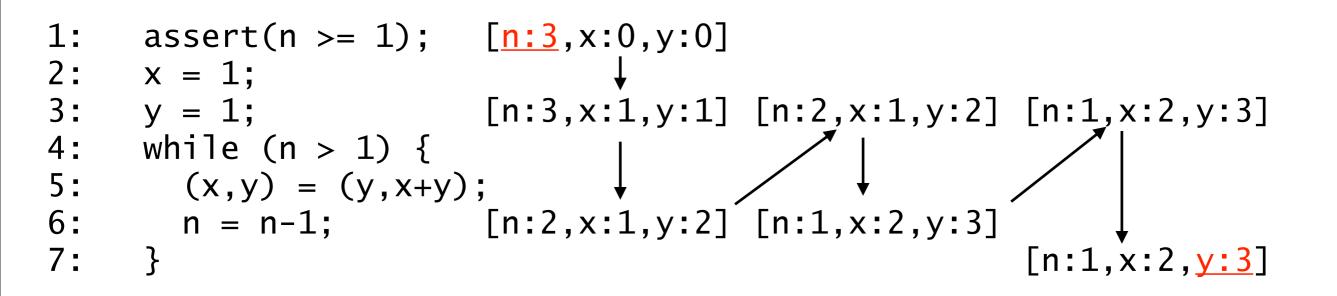
```
1: assert(n >= 1); [n:3,x:0,y:0]
2: x = 1; 
3: y = 1; [n:3,x:1,y:1]
4: while (n > 1) {
5: (x,y) = (y,x+y); 
6: n = n-1; [n:2,x:1,y:2]
7: }
```

```
1: assert(n >= 1); [n:3,x:0,y:0]
2: x = 1; 
3: y = 1; [n:3,x:1,y:1] [n:2,x:1,y:2]
4: while (n > 1) {
5: (x,y) = (y,x+y); 
6: n = n-1; [n:2,x:1,y:2]
7: }
```









```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: assert(y >= 0);
```

 $F_0 = F_1 = I$ $F_n = F_{n-1} + F_{n-2}$

```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: assert(y >= 0);
```

Because it computes fib. number.

Fibonacci number

 $F_0 = F_1 = I$ $F_n = F_{n-1} + F_{n-2}$

```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: assert(y >= 0);
```

Because it computes fib. number. Irrelevant n. No negative numbers nor minus.

Simple sign abstraction

• Abstract values:

 An abstract state is a map from variables to abstract values. E.g. [n:⊤, x:+, y:+].

```
1: assert(n >= 1); [n:+,x:T,y:T]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: assert(y >= 0);
```

```
1: assert(n >= 1); [n:+,x:\top,y:\top]

2: x = 1; 

3: y = 1; [n:+,x:+,y:+]

4: while (n > 1) {

5: (x,y) = (y,x+y);

6: n = n-1;

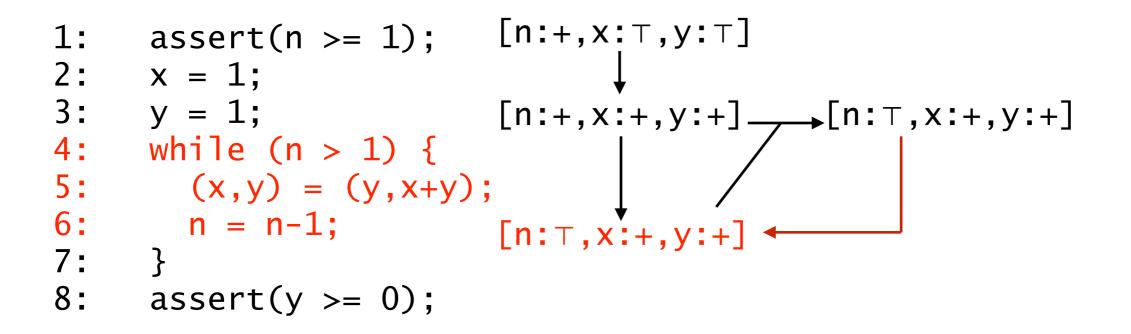
7: }

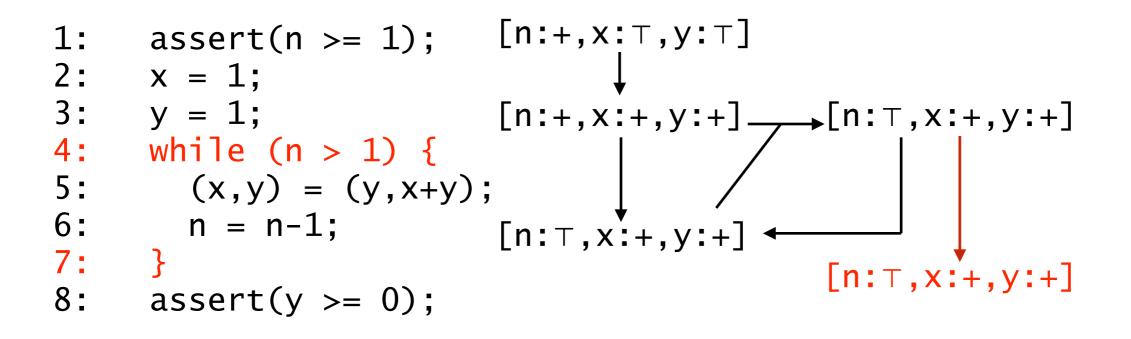
8: assert(y >= 0);
```

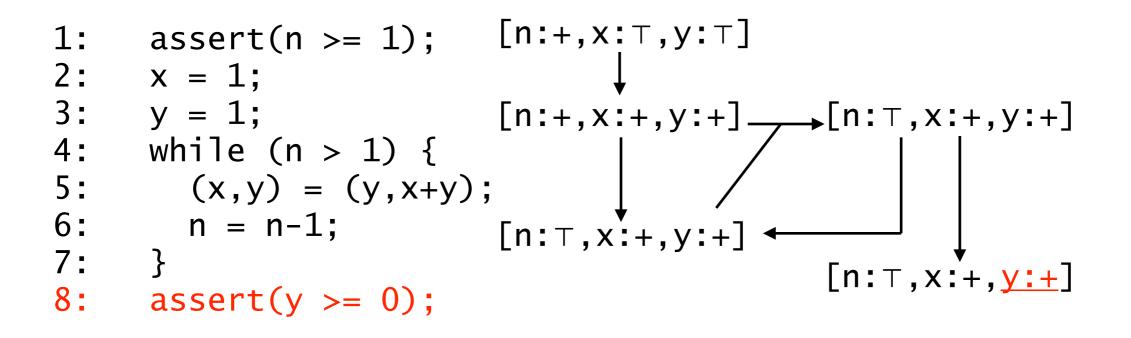
```
assert(n >= 1); [n:+,x:\top,y:\top]
1:
2: x = 1;
3:
  y = 1;
                       [n:+,x:+,y:+]
  while (n > 1) {
4:
    (x,y) = (y,x+y);
5:
    n = n-1;
6:
                     [n: \top, x: +, y: +]
7:
    }
8:
  assert(y \ge 0);
```

1: assert(n >= 1);
$$[n:+,x:\top,y:\top]$$

2: x = 1;
3: y = 1; $[n:+,x:+,y:+] \rightarrow [n:\top,x:+,y:+]$
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1; $[n:\top,x:+,y:+]$
7: }
8: assert(y >= 0);







Finding a good abstraction

```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: assert(y >= 0);
```

• Typically done by hand.

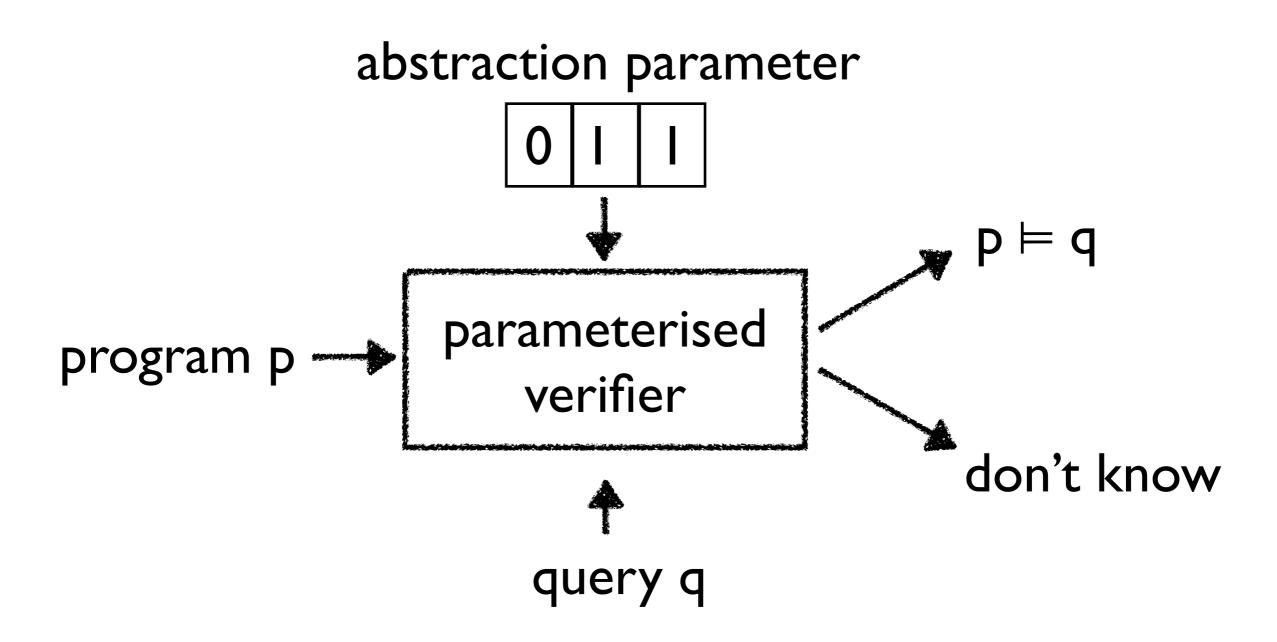
Finding a good abstraction

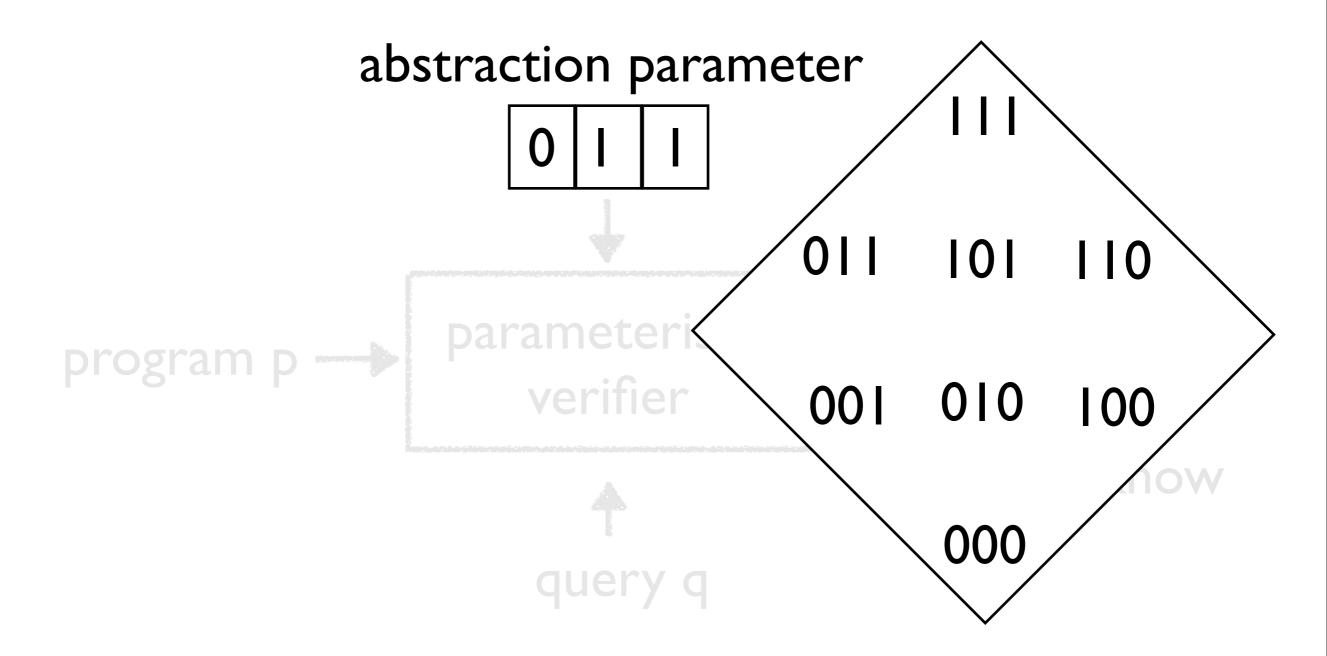
```
assert(n >= 1);
1:
2:
     x = 1;
3:
    y = 1;
   while (n > 1) {
4:
5:
     (x,y) = (y,x+y);
6:
      n = n - 1;
7:
     }
8:
  y = y - 1;
     assert(y >= 0);
9:
```

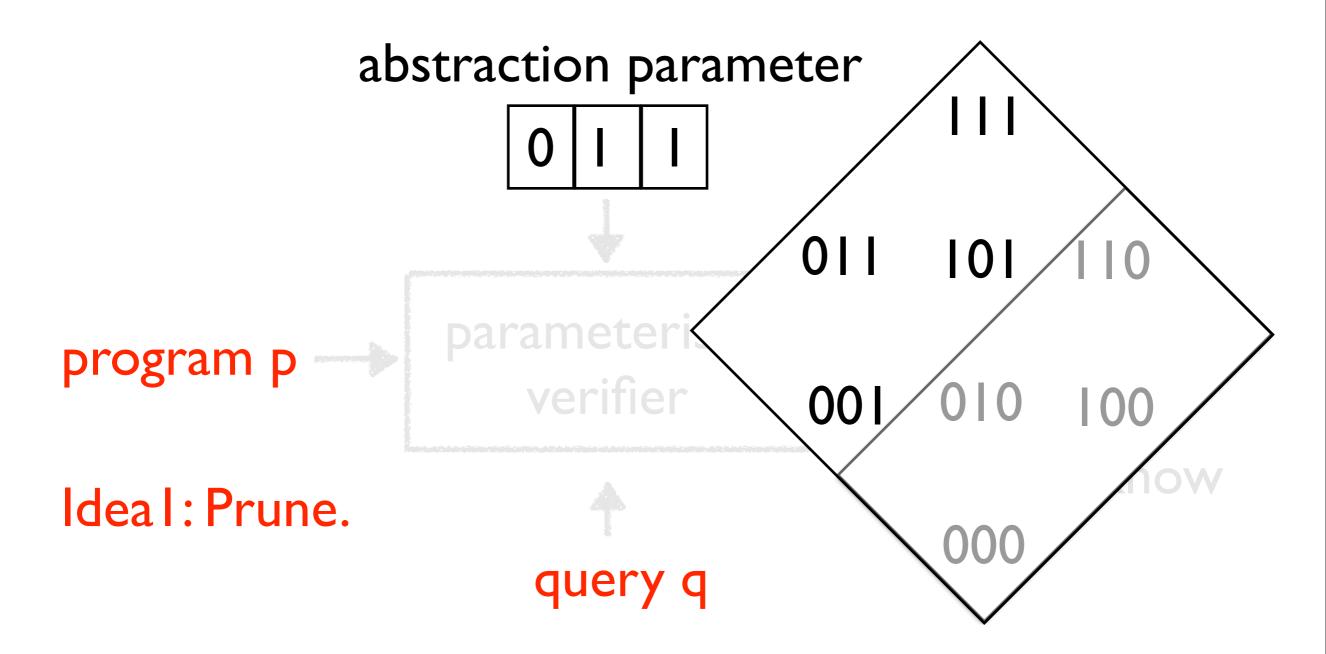
- Typically done by hand.
- Tricky.
- Active research area: how to automate this?

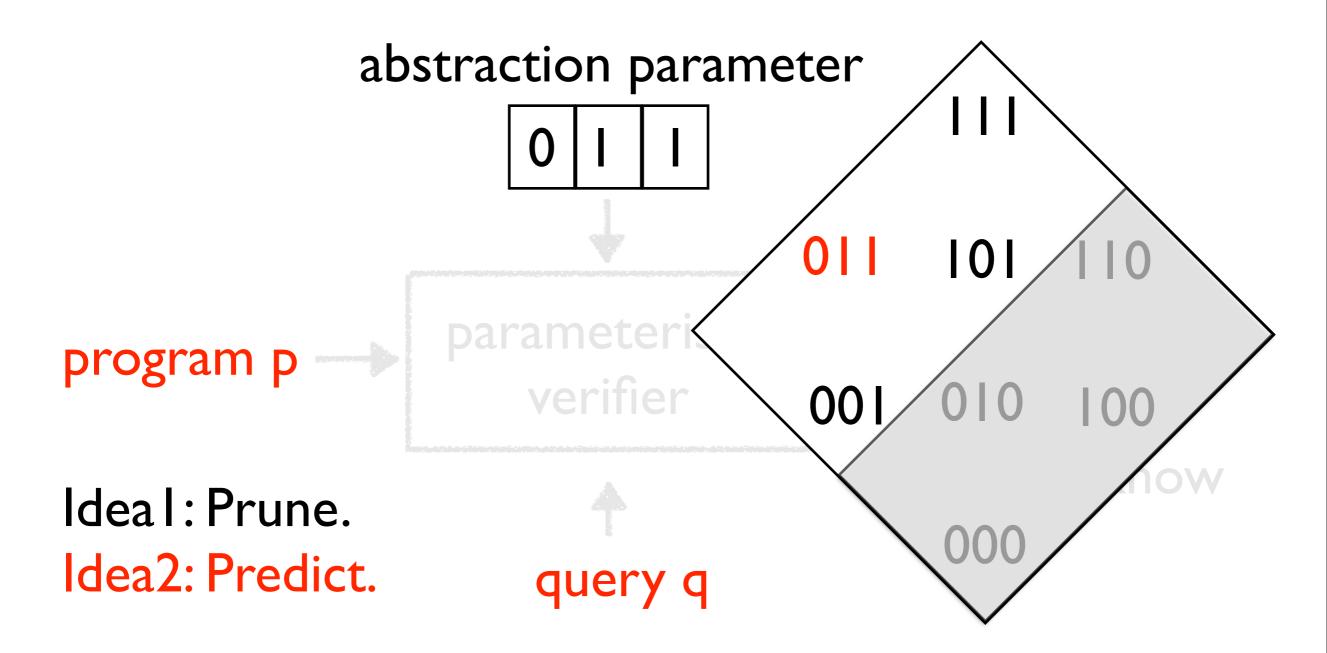
Our approach since 2011

- Formulate abstraction finding as a search problem [POPL'12,PLDI'13,PLDI14a,PLDI14b].
- Choose search space carefully.
- Develop an efficient search algorithm.





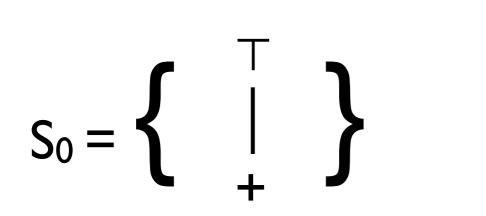




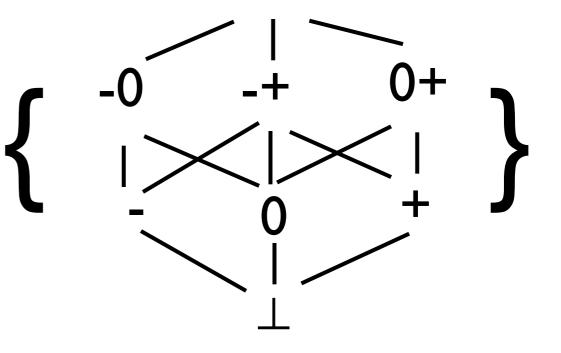
Pruning based on testing results

Two sign abstractions $s_0 = \left\{ \begin{array}{c} T \\ I \\ + \end{array} \right\} \qquad s_1 = \left\{ \begin{array}{c} -0 & -1 \\ I \\ -0 & -+ & 0+ \\ -0 & -- & 0+ \\$

S₁=



Abs = { n, x, y } \rightarrow {0, 1} abs_0 = [n:0,x:0,y:0] abs_1 = [n:1,x:1,y:1] abs_2 = [n:0,x:0,y:1]



$$Abs = \{ n, x, y \} \rightarrow \{0, I\}$$

$$Abs = \{ n, x, y \} \rightarrow \{0, 1\}$$

$$assert(n \ge 1);$$

$$x = 1;$$

$$y = 1;$$
while (n > 1) {
(x,y) = (y, x+y);
n = n-1;
}
y = y-1;
assert(y >= 0);

$$[n:\tau, x:+, y:\tau]$$

1:

2:

3:

4:

5:

6:

7:

8:

9:

X =

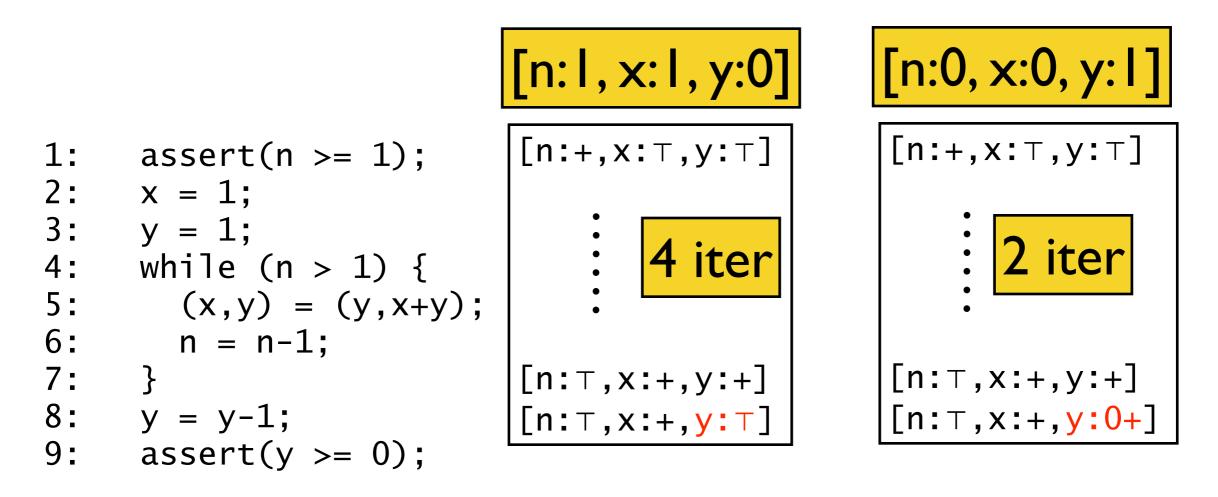
y =

}

y =

n

Abs =
$$\{n, x, y\} \rightarrow \{0, I\}$$



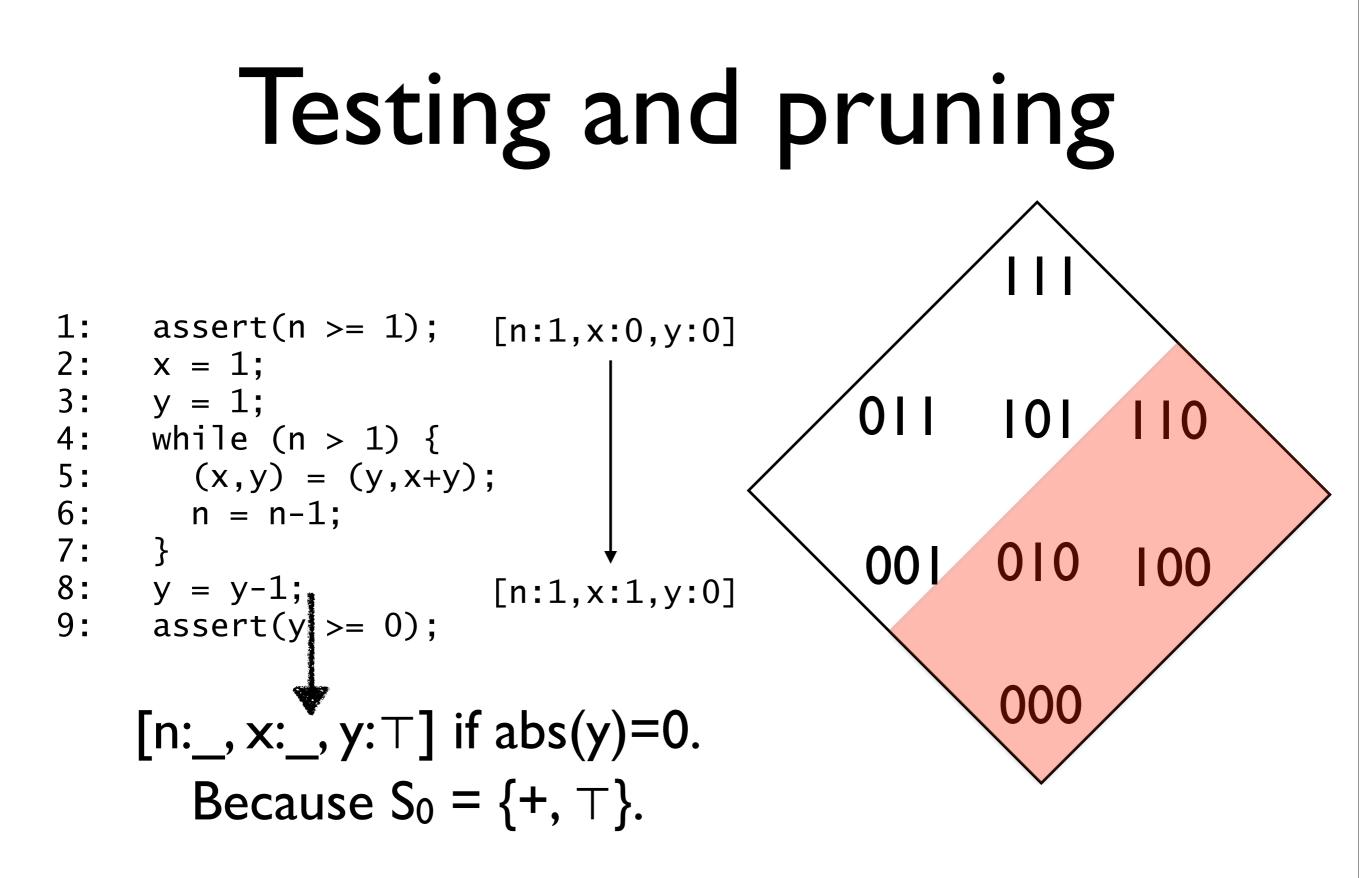
- Test a program.
- If a bug is found, report an error.
- Otherwise, identify bad abstractions and prune the search space.

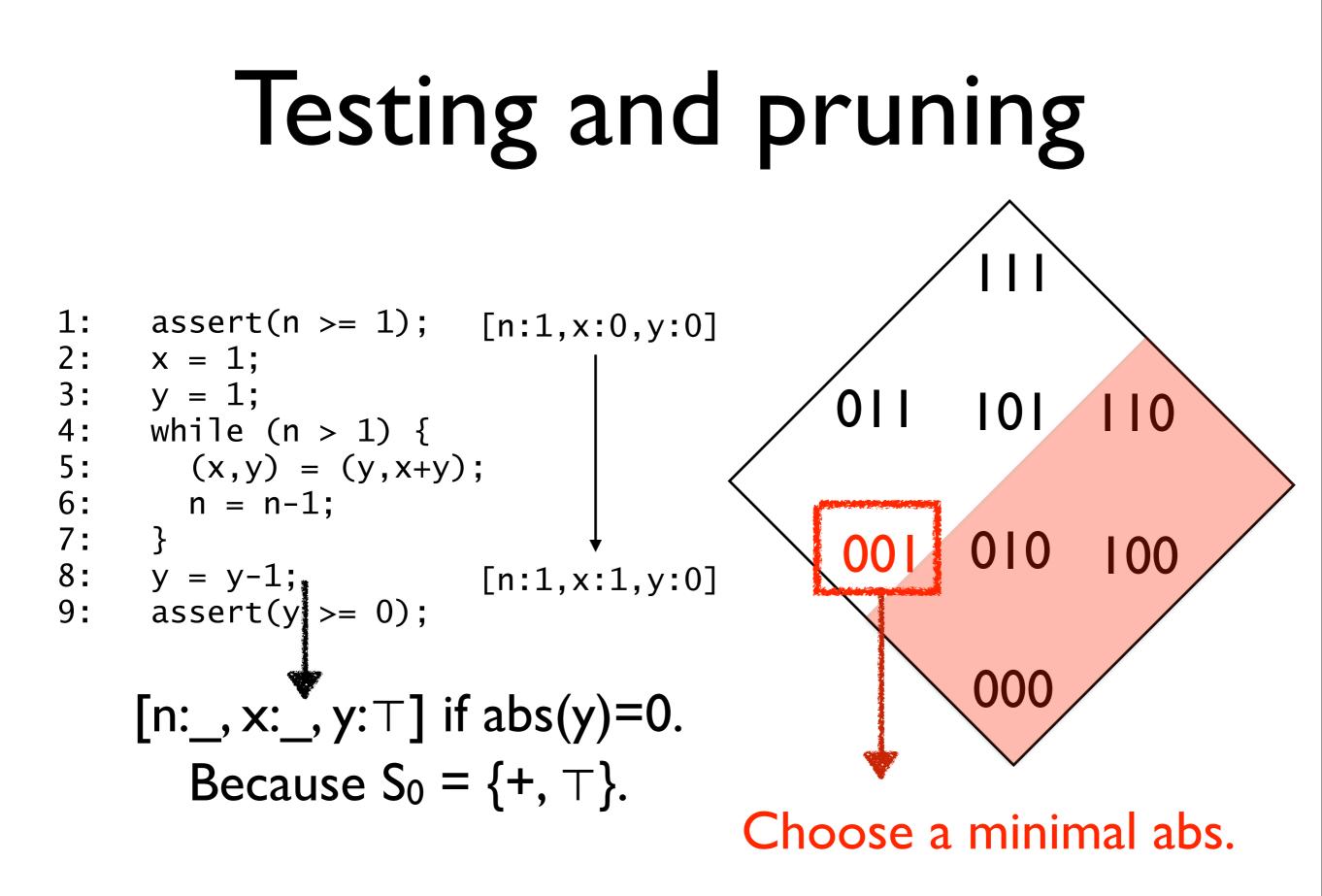
```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y-1;
9: assert(y >= 0);
```

```
1: assert(n >= 1); [n:1,x:0,y:0]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y-1; [n:1,x:1,y:0]
9: assert(y >= 0);
```

```
assert(n >= 1);  [n:1,x:0,y:0]
1:
2: x = 1;
3:
  y = 1;
4:
  while (n > 1) {
5:
  (x,y) = (y,x+y);
6:
  n = n - 1;
  }
7:
8:
  y = y-1;
                    [n:1,x:1,y:0]
9:
  assert(y >= 0);
```

[n:_, x:_, y:⊤] if abs(y)=0. Because S₀ = {+, ⊤}.





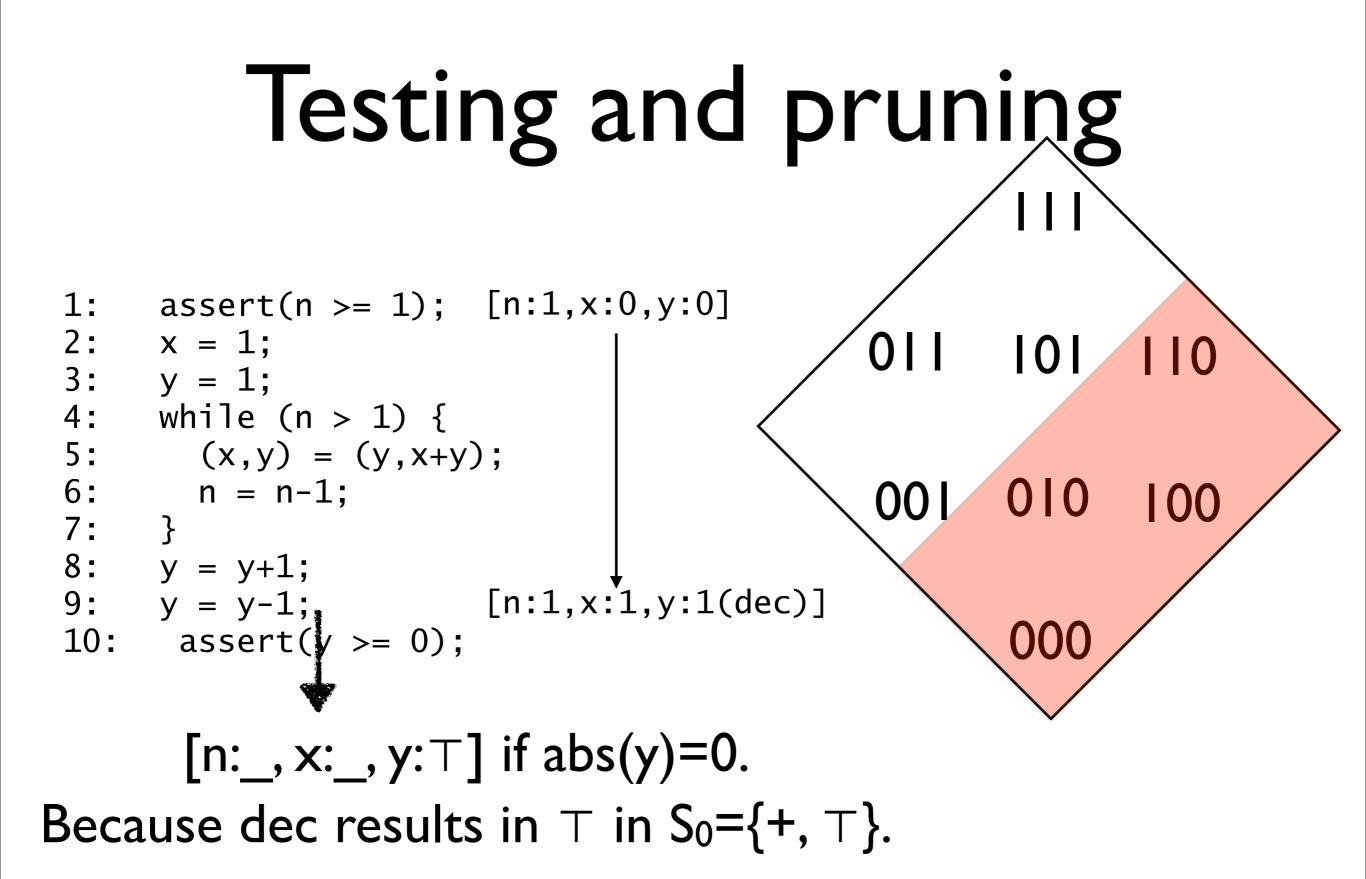
```
1: assert(n >= 1);
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y+1;
9: y = y-1;
10: assert(y >= 0);
```

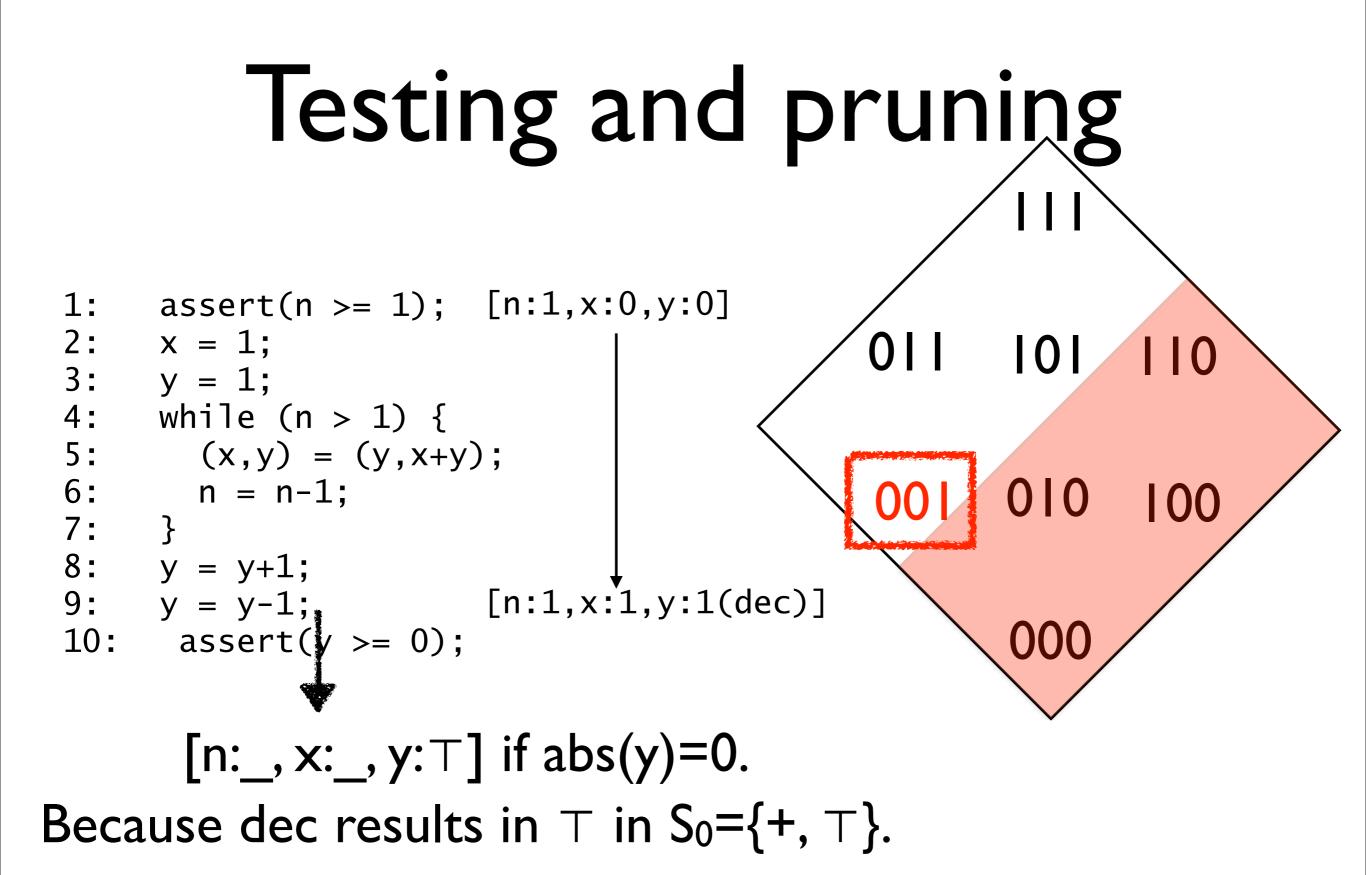
```
1: assert(n >= 1); [n:1,x:0,y:0]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y+1;
9: y = y-1; [n:1,x:1,y:1]
10: assert(y >= 0);
```

```
1: assert(n >= 1); [n:1,x:0,y:0]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y+1;
9: y = y-1; [n:1,x:1,y:1(dec)]
10: assert(y >= 0);
```

```
1: assert(n >= 1); [n:1,x:0,y:0]
2: x = 1;
3: y = 1;
4: while (n > 1) {
5: (x,y) = (y,x+y);
6: n = n-1;
7: }
8: y = y+1;
9: y = y-1; [n:1,x:1,y:1(dec)]
10: assert(y >= 0);
```

[n:_, x:_, y: \top] if abs(y)=0. Because dec results in \top in S₀={+, \top }.





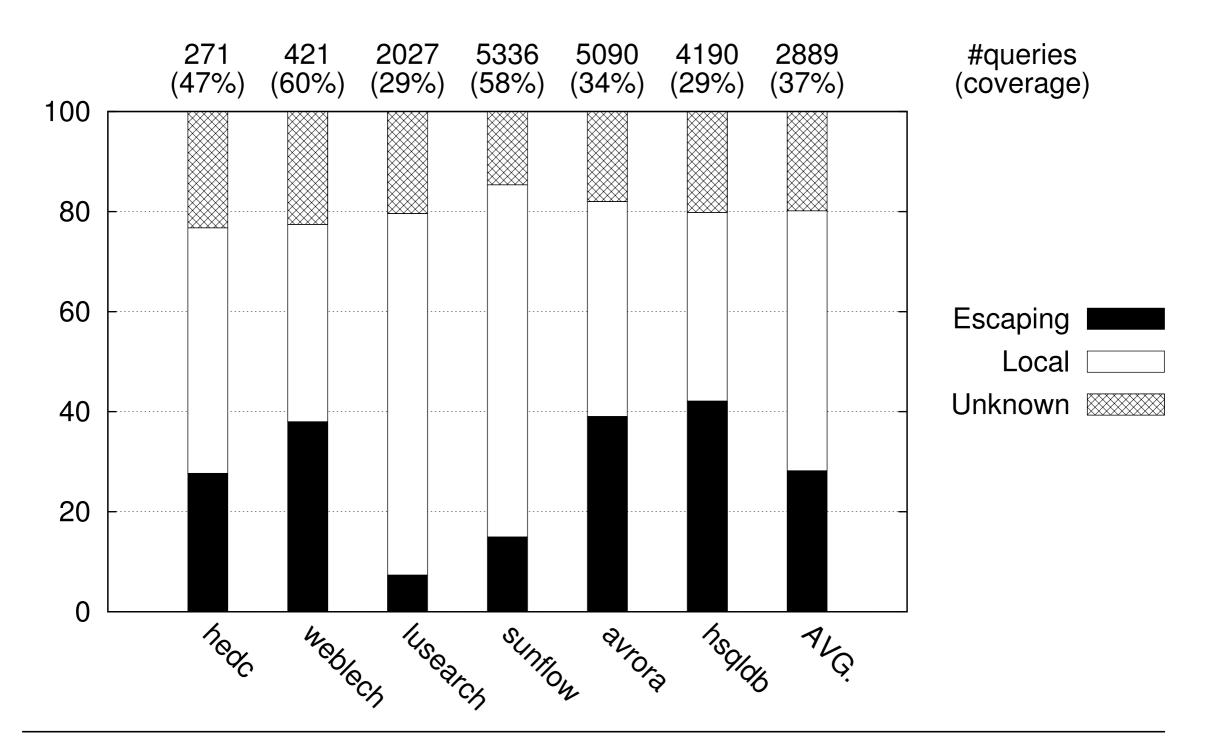


Figure 3. Precision results for our thread-escape analysis. [POPL'12]

Pruning based on refinement

Limitation of testing

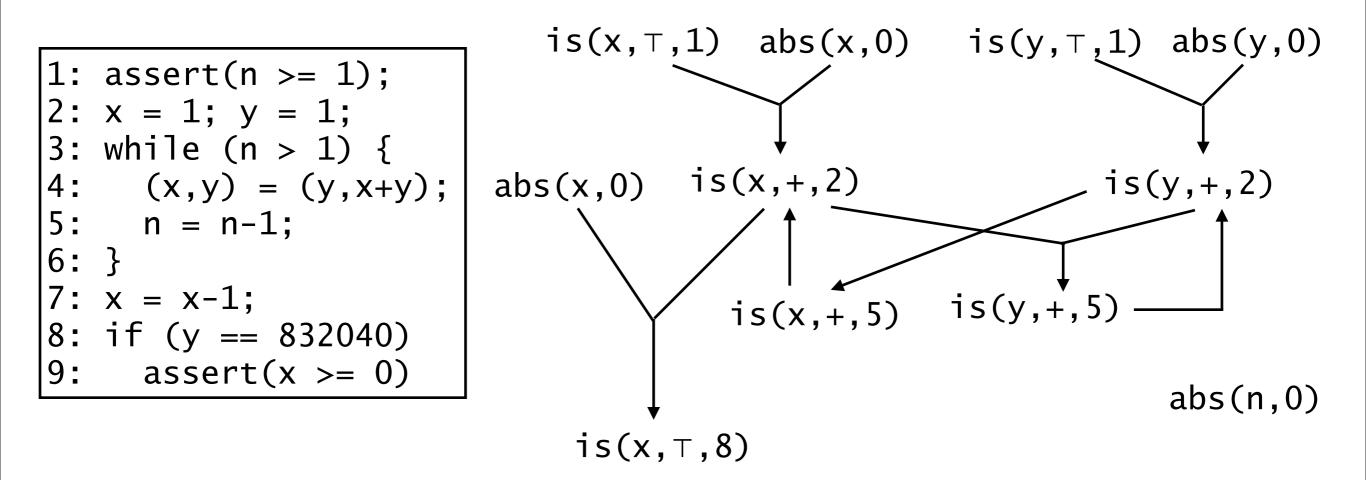
Limitation of testing

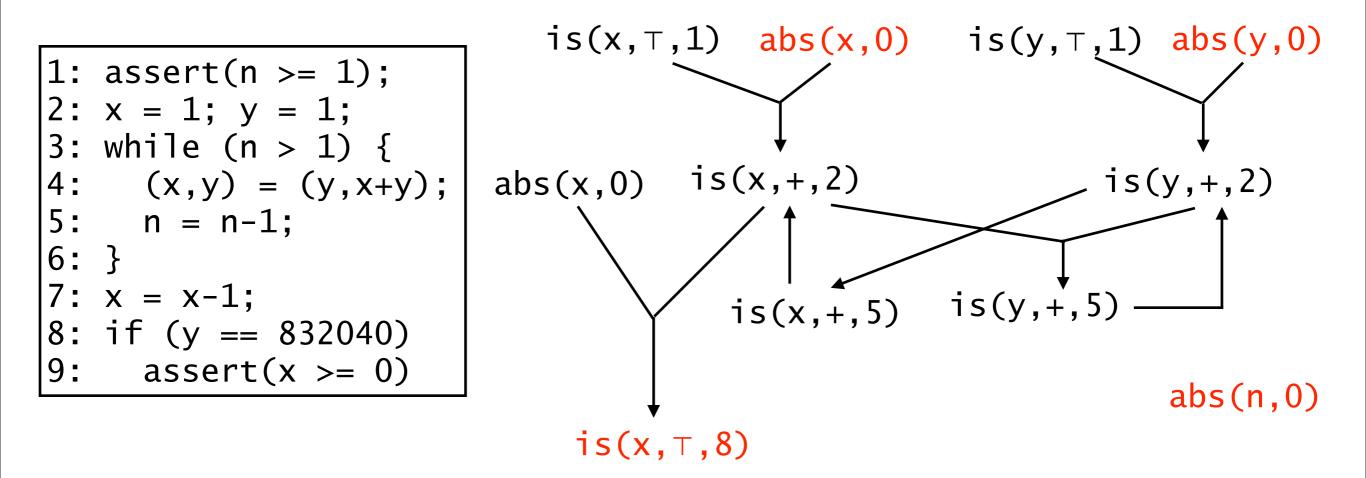
Reaching assert(...) by testing is not easy.

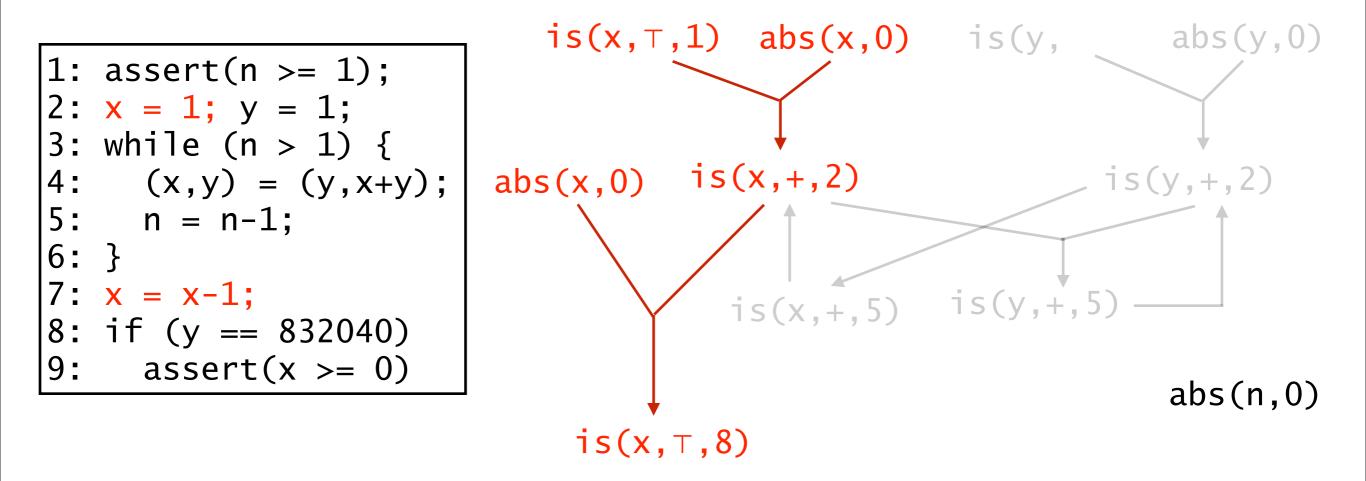
Iterative refinement

- Run a verifier with a cheap abstraction.
- Prune all abstractions that lead to similar verification failures.

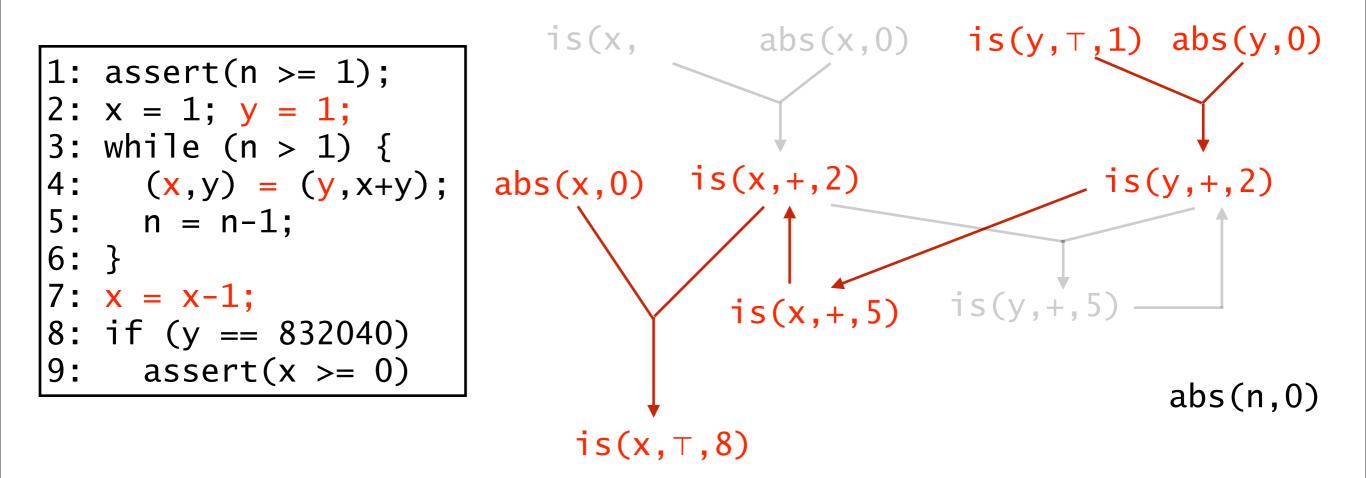
Result with [n:0,x:0,y:0]



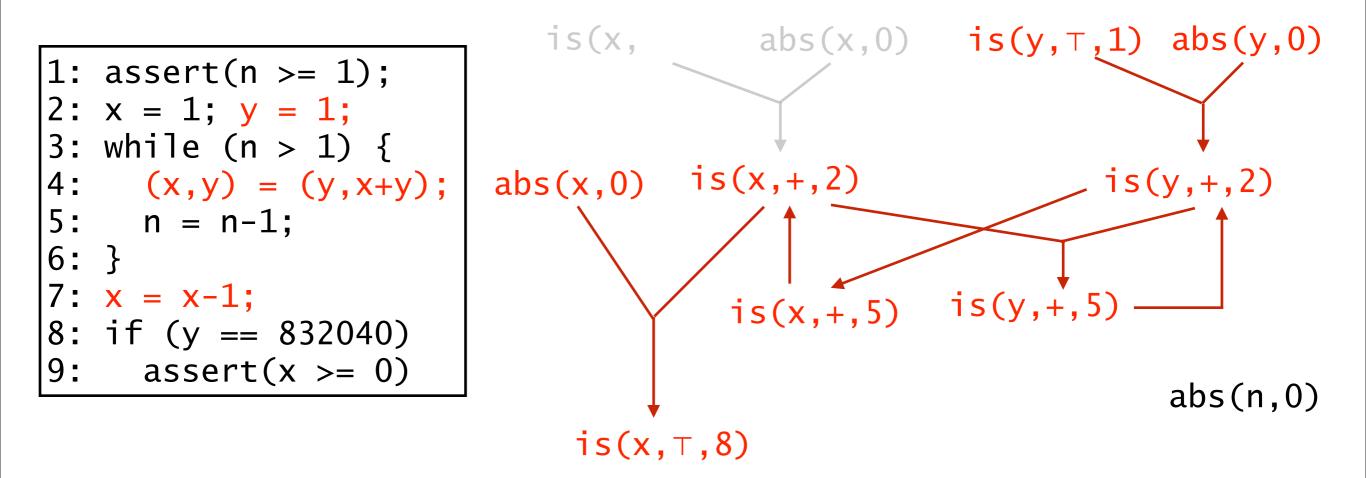




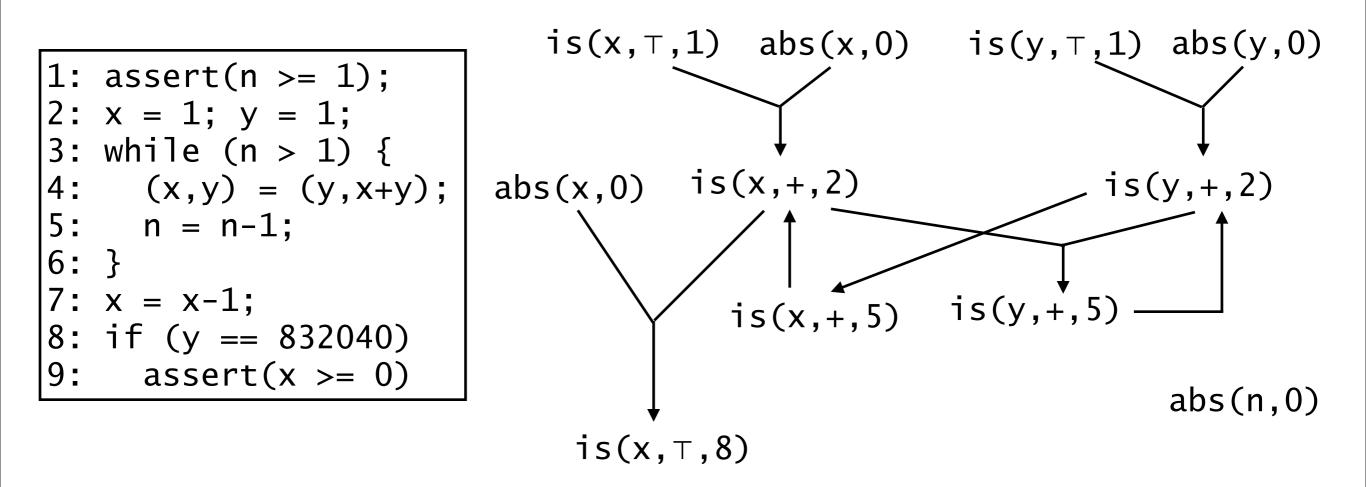
If abs(x,0), we cannot prove the query.



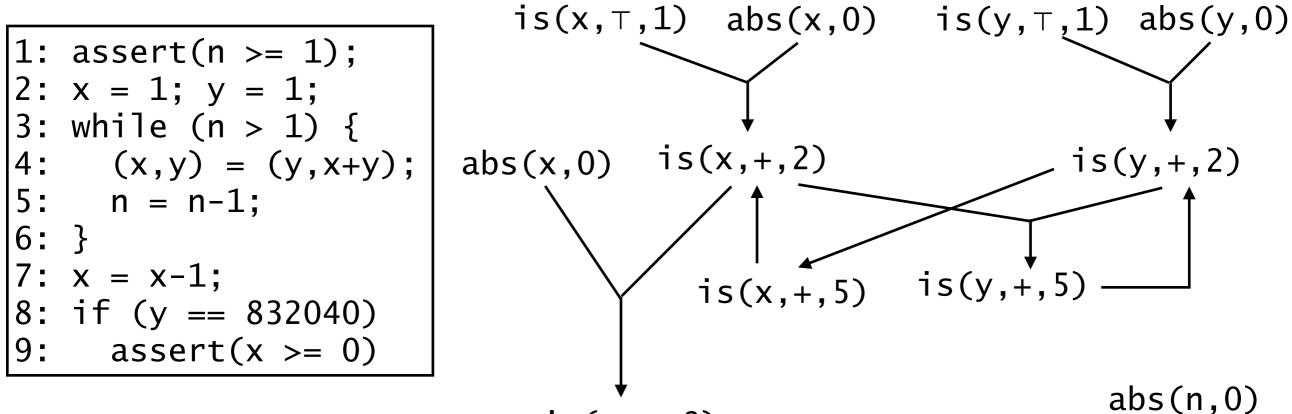
If abs(x,0), we cannot prove the query. If abs(x,0) or abs(y,0), we cannot prove the query.



If abs(x,0), we cannot prove the query. If abs(x,0) or abs(y,0), we cannot prove the query.

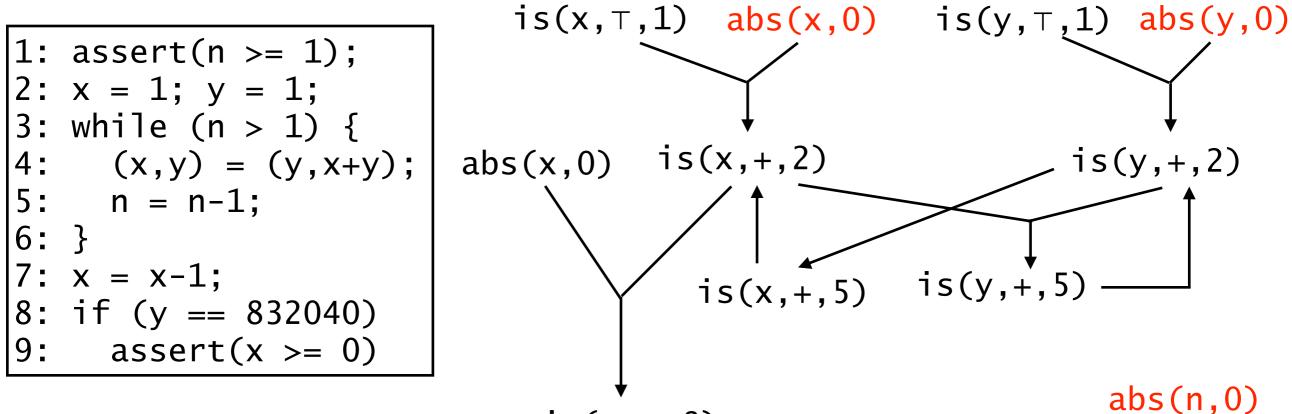


If abs(x,0), we cannot prove the query. If abs(x,0) or abs(y,0), we cannot prove the query.



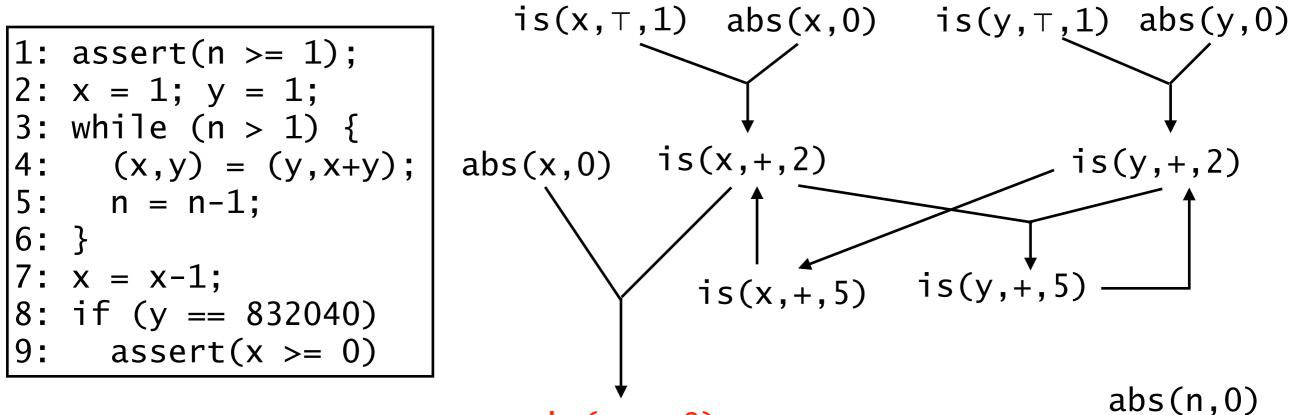
 $is(x, \tau, 8)$

MaxSat Encoding: Hard clauses: Soft clauses: (is(x, T, 1) & abs(x, 0) => is(x, +, 2))abs(n,0)& abs(x,0) & $(abs(x,0) \& is(x,+,2) => is(x, \top, 8))$ & abs(y,0) & ... & (not is(x,T,8))



 $is(x, \tau, 8)$

MaxSat Encoding: Soft clauses: Hard clauses: (is(x, T, 1) & abs(x, 0) => is(x, +, 2))abs(n,0)& abs(x,0) & $(abs(x,0) \& is(x,+,2) => is(x, \top, 8))$ & abs(y,0) & ... & (not is(x,T,8))



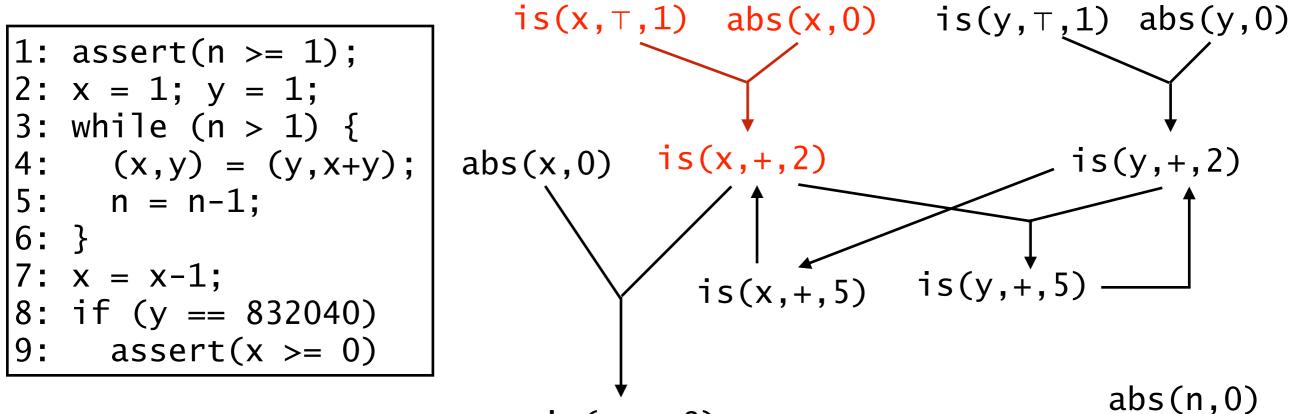
is(x, T, 8)

MaxSat Encoding: Hard clauses: (is(x, T, 1) & abs(x, 0) => is(x, +, 2))& $(abs(x,0) \& is(x,+,2) => is(x, \top, 8))$ & ...

& (not is(x,T,8))

Soft clauses:

abs(n,0)& abs(x,0) & abs(y,0)



 $is(x, \tau, 8)$

MaxSat Encoding: Hard clauses: Soft clauses: (is(x, T, 1) & abs(x, 0) => is(x, +, 2))abs(n,0)& abs(x,0) & $(abs(x,0) \& is(x,+,2) => is(x, \top, 8))$ & abs(y,0) & ... & (not is(x,T,8))

Full story

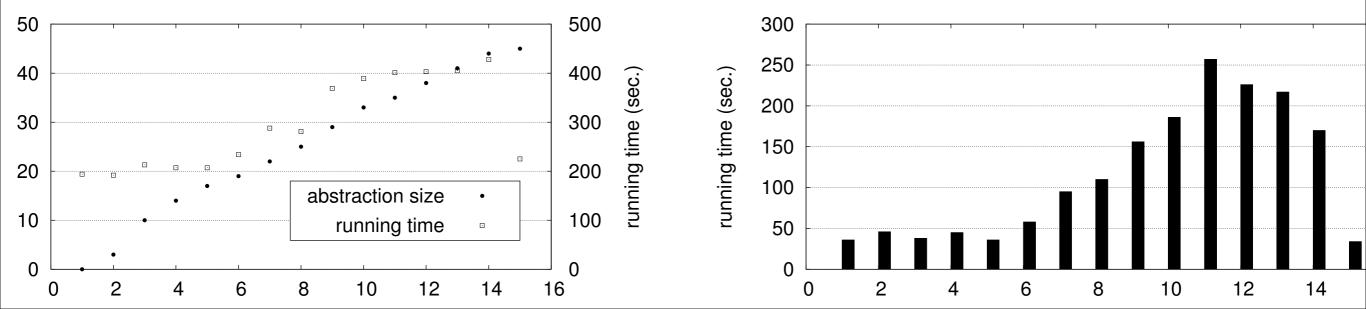
- The process is repeated until we prune the whole search space or prove the query.
- Implemented in the context of program analyses (or verifiers) written in Datalog.
- See PLDI'I4a for details.

			pointer ana	lysis		typestate analysis						
		queries			abstraction size		queries		abstraction size		iterations	
	total	reso	resolved		max.	iterations	total	resolved	final	max.		
	total	CURRENT	BASELINE	final ma		•	iotai	lesolved	mai	шал.	CURRENT	BASELINE
toba-s	7	7	0	17	1,782	10	543	543	62	14,781	15	159
javasrc-p	46	46	0	47	1,845	13	159	159	89	13,653	14	92
weblech	5	5	2	14	3,095	10	13	13	33	25,781	14	16
hedc	47	47	6	73	2,948	18	24	24	14	23,622	7	10
antlr	143	143	5	97	2,917	15	77	77	66	24,815	12	45
luindex	138	138	67	116	4,055	26	248	248	79	33,835	16	72
lusearch	322	322	29	146	3,936	17	45	45	74	33,526	13	52
schroeder-m	51	51	25	45	5,826	15	194	194	71	54,741	9	49

Table 3: Results showing statistics of queries, abstractions, and iterations of our approach (CURRENT) and the baseline approaches (BASELINE).

	running time of the Datalog solver (in seconds)								running time of the MAXSAT solver (in seconds)						
	pointer analysis				typestate analysis			pointer analysis			typestate analysis				
	BASELINE	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.	min.	max.	avg.		
toba-s	11	5	7	6	49	82	68.1	2	7	3.1	1	6	3.1		
javasrc-p	29	7	11	9	76	152	120.8	<1	4	1.6	2	19	6.4		
weblech	2,574	44	54	47.5	121	172	146.6	5	11	6.7	3	8	5.3		
hedc	5,058	21	37	27.9	52	58	54.3	1	23	3.7	1	2	1.7		
antlr	3,723	30	55	39.3	193	325	264.8	11	44	24.1	5	27	13.25		
luindex	913	59	84	76.4	311	512	426.7	8	48	16.3	6	26	14.7		
lusearch	7,040	59	85	72.7	238	437	343.9	7	62	23.9	6	29	15.9		
schroeder-m	23,038	192	428	289.6	1,778	2,681	2,304.6	34	257	114	37	308	138.6		

Table 4: Running time of the Datalog and MAXSAT solvers in each iteration.





	pointer a	nalysis	typestate analysis			
	# variables	# clauses	# variables	# clauses		
toba-s	784k	1,485k	741k	938k		
javasrc-p	470k	877k	1,022k	1,333k		
weblech	1,620k	3,307k	1,374k	1,807k		
hedc	1,245k	2,664k	606k	751k		
antlr	3,621k	6,875k	2,318k	3,009k		
luindex	2,406k	5,643k	2,829k	3,784k		
lusearch	2,103k	5,011k	2,626k	3,524k		
schroeder-m	6,706k	23,680k	16,293k	22,257k		

Table 5: Statistics of MAXSAT formula in the final iteration.

How to find a good program abstraction automatically?

- Formulate it as a search problem.
- Develop a good pruning strategy.
- Predict based on the knowledge of a verifier.