Instruction tables will have to be made up by mathematicians with computing experience and perhaps a certain puzzle-solving ability…

This process of constructing instruction tables should be very fascinating. There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself.

Alan Turing, 1945

Advances and Challenges in Program Synthesis

Armando Solar-Lezama

MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

CSAIL

The promise of automation

The FORTRAN Automatic Coding System

J. W. BACKUS†, R. J. BEEBER†, S. BEST‡, R. GOLDBERG†, L. M. HAIBT†,
H. L. HERRICK†, R. A. NELSON†, D. SAYRE†, P. B. SHERIDAN†,
H. STERN†, I. ZILLER†, R. A. HUGHES§, AND R. NUTT||

Introduction

THE FORTRAN project was begun in the summer of 1954. Its purpose was to reduce by a large factor the task of preparing scientific problems for IBM's next large computer, the 704. If it were possible for the 704 to code problems for itself and produce as good programs as human coders (but without the errors), it was clear that large benefits could be achieved. For it was known that about two-thirds of the cost of solving most scientific and engineering problems on large computers was that of problem preparation. Furthermore, more than 90 per cent of the elapsed time for a problem was usually devoted to planning, writing, system is now complete. It has two components: the FORTRAN language, in which programs are written, and the translator or executive routine for the 704 which effects the translation of FORTRAN language programs into 704 programs. Descriptions of the FORTRAN language and the translator form the principal sections of this paper.

The experience of the FORTRAN group in using the system has confirmed the original expectations concerning reduction of the task of problem preparation and the efficiency of output programs. A brief case history of one job done with a system seldom gives a good measure of its usefulness, particularly when the

The promise of automation

The FORTRAN Automatic Coding System

J. W. BACKUS†, R. J. BEEBER†, S. BEST‡, R. GOLDBERG†, L. M. HAIBT†,
H. L. HERRICK†, R. A. NELSON†, D. SAYRE†, P. B. SHERIDAN†,
H. STERN†, I. ZILLER†, R. A. HUGHES§, AND R. NUTT||

IBM's next large computer, the 704. If it were possible for the 704 to code problems for itself and produce as good programs as human coders (but without the errors), it was clear that large benefits could be achieved.

for a problem was usually devoted to planning, writing,

good measure of its usefulness, particularly when the

Automation Today

Domain Specific Languages

High-level general purpose languages


Some day we won't even need coders any more. We'll be able to just write the specification and the program will write itself
Oh wow, you're right! We'll be able to write a comprehensive and precise spec and bam, we won't need programmers any more!
Exactly!
And do you know the industry term for a project specification that is comprehensive and precise enough to generate a program?
Uh... no....
Code
It's called code
CommitStrip.com

Program Synthesis

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 4, JULY 1979

Synthesis: Dreams ⟹ Programs

ZOHAR MANNA AND RICHARD WALDINGER

e techniques are presented for deriving programs
iven specifications. The specifications express the
ed program without giving any hint of the algo-
l. The basic approach is to transform the specifi-
cording to certain rules, until a satisfactory pro-

Introduction

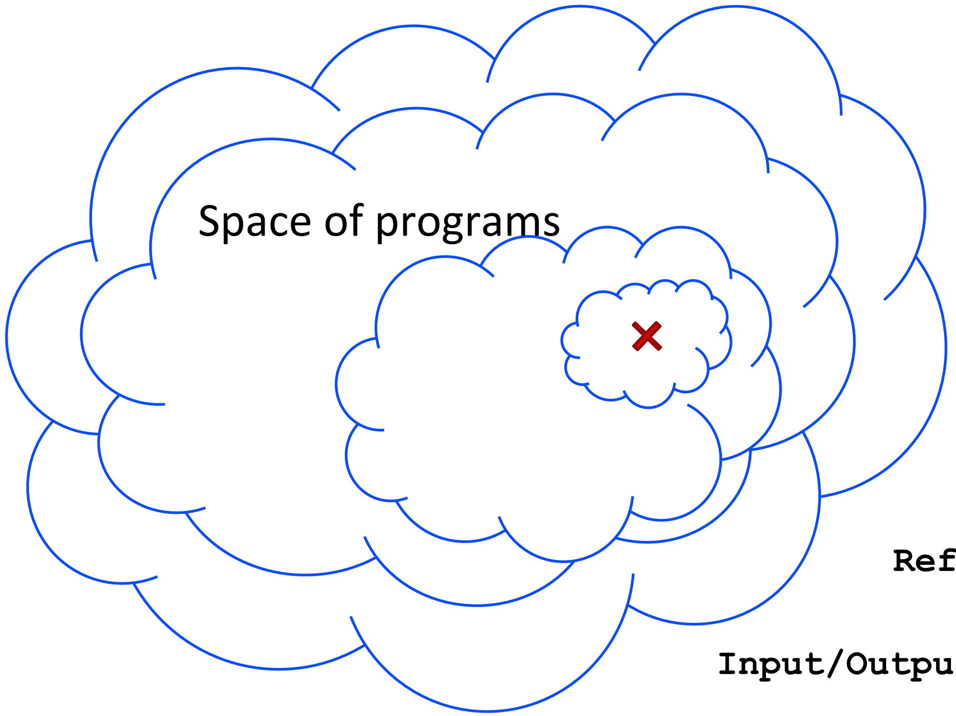
IN RECENT years there has been increasing activity in the field of program verification. The goal of these efforts is to construct computer systems for determining whether a

Zohar Manna

Richard Waldinger

Synthesis: modern view

$$R = \{p_0 \ldots p_i\}$$



$$\varphi(p)$$

Reference Implementation

Input/Output Examples

Test Harnesses

$$\varphi(p) = \forall in. \; \ldots \; p(in) \ldots$$

Example

Sketch

```
bit[W] avg(bit[W] x, bit[W] y)
implements avgSpec{

    return expr@signed({x,y}, 4);
}
```

```
expr ::=   const
         | var
         | expr>>??
         | ~expr
         | expr + expr
         | expr ^ expr
         | expr & expr
```

Spec

```
bit[W] avgSpec(bit[W] x, bit[W] y){

  bit[2*W] xx = extend@signed(x, 2*W);
  bit[2*W] yy = extend@signed(y, 2*W);

  bit[2*W] r = rshift@signed(xx+yy, 1);
  return (r[0::W]);
}
```

And 8 seconds later…

After considering 2^{1296} possibilities

(x & y) + (x ^ y) >> 1

Cool!

Now can you synthesize programs with more than 1 line of code?

Early successes

Concurrent data-structures

Small but high-impact code

- Herlihy calls them the “ball bearings” of concurrent software

Difficult for humans to reason about

Well defined space of possible synchronization and coordination approaches

Paraglide [IBM]

Synthesis of concurrent code

Program Code

```
int take() {
    long b = bottom - 1;
    item_t * q = wsq;
    bottom = b
    long t = top
    …
```

Architecture Description

Paraglide Tool

SAT solvers
Model Checking
Abstraction Directed
Testing
Semantic Search

Minimal Synchronization

```
int take() {
    long b = bottom - 1;
    item_t * q = wsq;
    bottom = b
    long t = top
    …
```

✓

Highly impactful work by Yahav, Vechev and Yorsh at IBM

Domain specific system

Lessons

Focus on high-impact domains

- Leverage domain specific structure

Engineer for interaction with experts

Reverse engineering

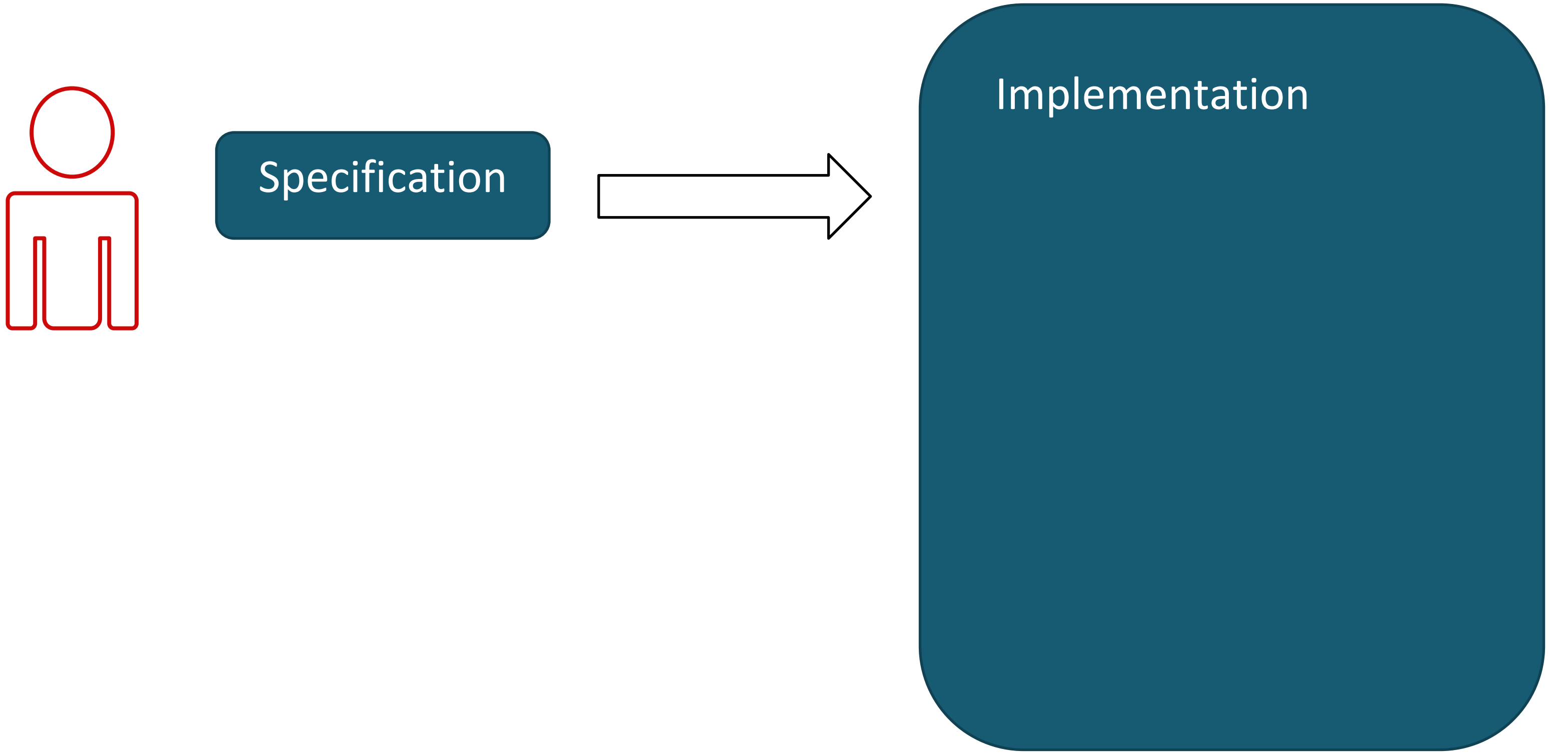
Oracle-guided component-based program synthesis

- ICSE 2010 paper by Jha, Gulwani, Seshia and Tiwari

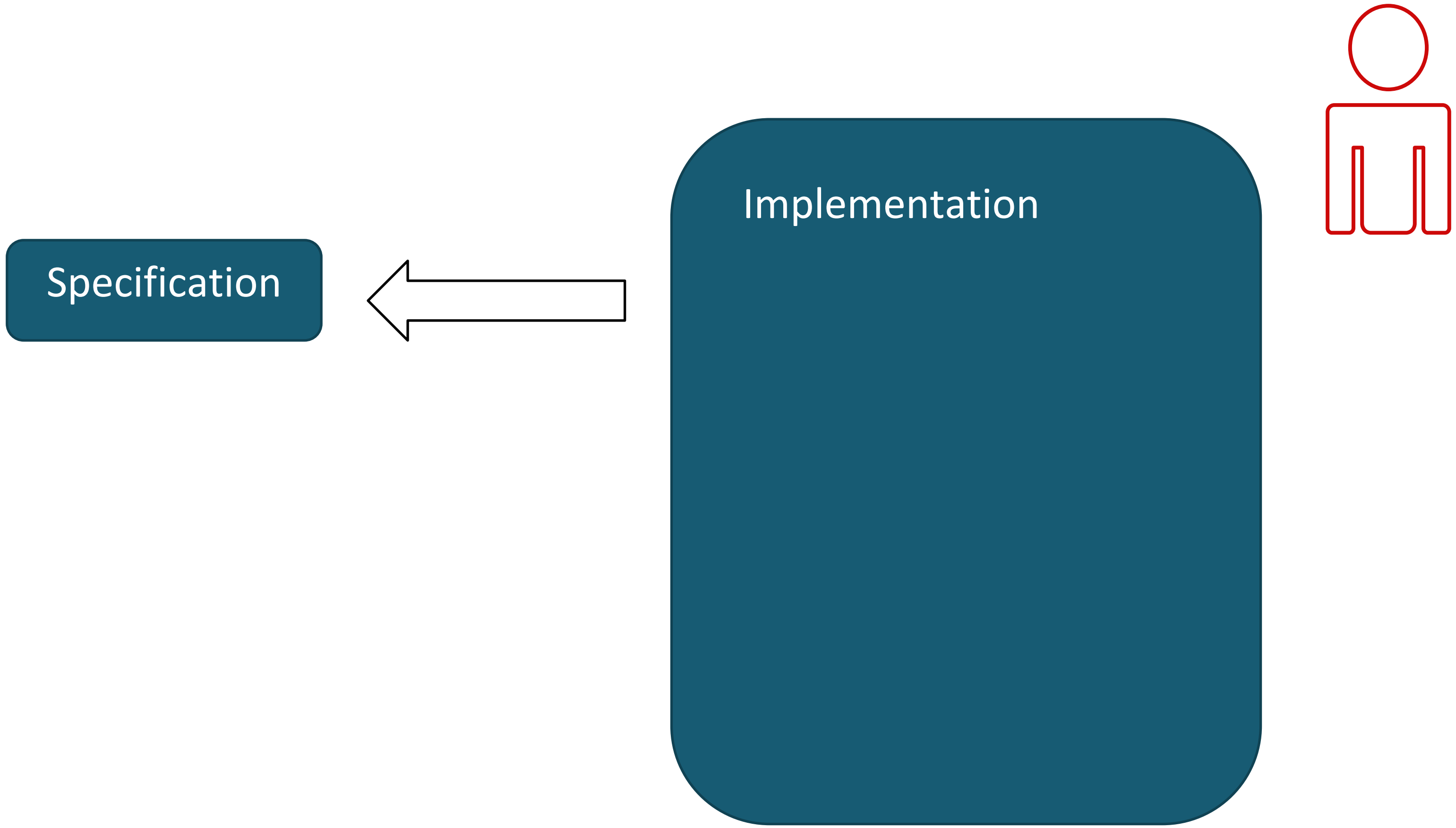
Pioneered a number of new ideas at the algorithmic level

Synthesis for reverse engineering

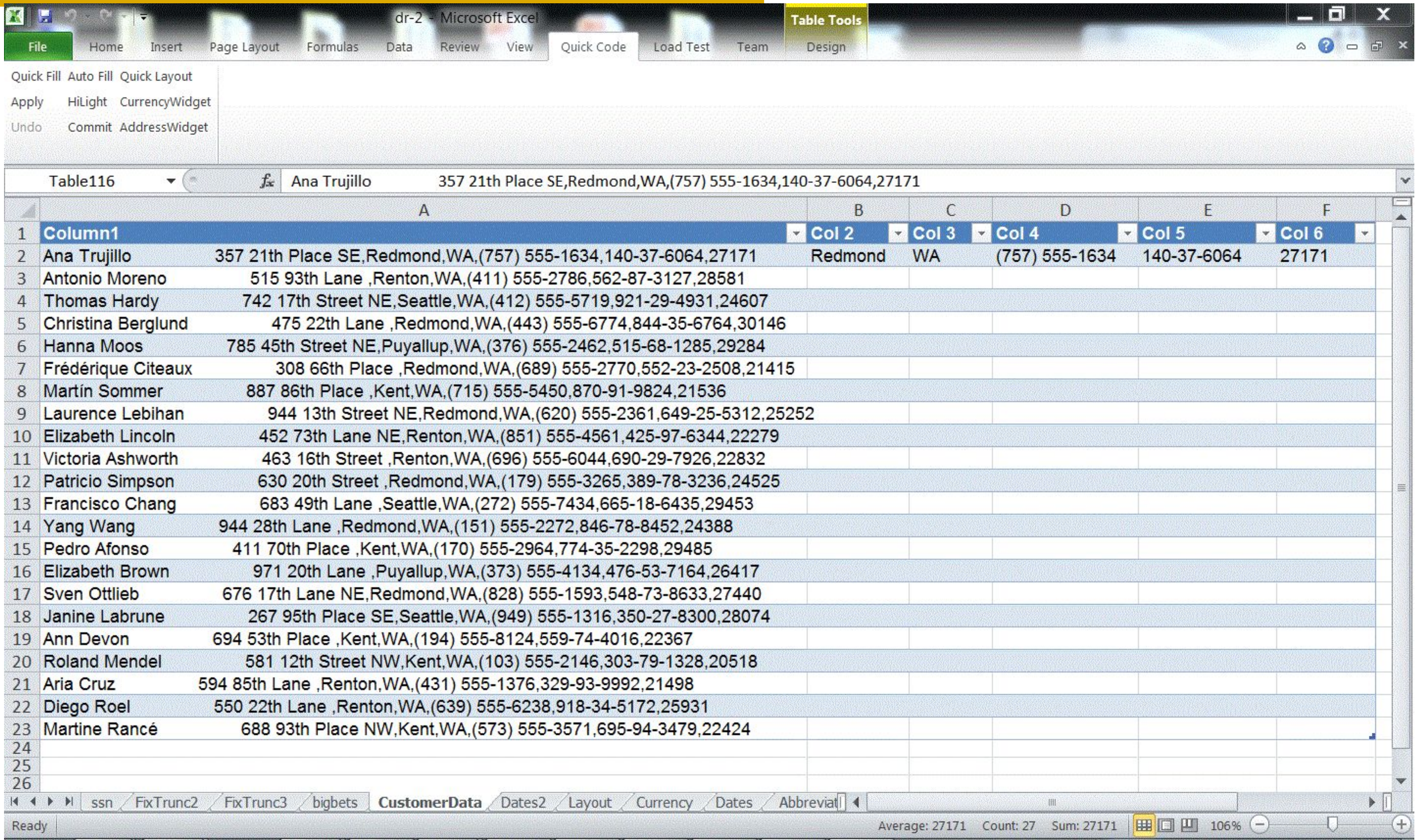
Reverse engineering



Reverse engineering



FlashFill



	Column1	Col 2	Col 3	Col 4	Col 5	Col 6
2	Ana Trujillo 357 21th Place SE,Redmond,WA,(757) 555-1634,140-37-6064,27171	Redmond	WA	(757) 555-1634	140-37-6064	27171
3	Antonio Moreno 515 93th Lane ,Renton,WA,(411) 555-2786,562-87-3127,28581					
4	Thomas Hardy 742 17th Street NE,Seattle,WA,(412) 555-5719,921-29-4931,24607					
5	Christina Berglund 475 22th Lane ,Redmond,WA,(443) 555-6774,844-35-6764,30146					
6	Hanna Moos 785 45th Street NE,Puyallup,WA,(376) 555-2462,515-68-1285,29284					
7	Frédérique Citeaux 308 66th Place ,Redmond,WA,(689) 555-2770,552-23-2508,21415					
8	Martín Sommer 887 86th Place ,Kent,WA,(715) 555-5450,870-91-9824,21536					
9	Laurence Lebihan 944 13th Street NE,Redmond,WA,(620) 555-2361,649-25-5312,25252					
10	Elizabeth Lincoln 452 73th Lane NE,Renton,WA,(851) 555-4561,425-97-6344,22279					
11	Victoria Ashworth 463 16th Street ,Renton,WA,(696) 555-6044,690-29-7926,22832					
12	Patricio Simpson 630 20th Street ,Redmond,WA,(179) 555-3265,389-78-3236,24525					
13	Francisco Chang 683 49th Lane ,Seattle,WA,(272) 555-7434,665-18-6435,29453					
14	Yang Wang 944 28th Lane ,Redmond,WA,(151) 555-2272,846-78-8452,24388					
15	Pedro Afonso 411 70th Place ,Kent,WA,(170) 555-2964,774-35-2298,29485					
16	Elizabeth Brown 971 20th Lane ,Puyallup,WA,(373) 555-4134,476-53-7164,26417					
17	Sven Ottlieb 676 17th Lane NE,Redmond,WA,(828) 555-1593,548-73-8633,27440					
18	Janine Labrune 267 95th Place SE,Seattle,WA,(949) 555-1316,350-27-8300,28074					
19	Ann Devon 694 53th Place ,Kent,WA,(194) 555-8124,559-74-4016,22367					
20	Roland Mendel 581 12th Street NW,Kent,WA,(103) 555-2146,303-79-1328,20518					
21	Aria Cruz 594 85th Lane ,Renton,WA,(431) 555-1376,329-93-9992,21498					
22	Diego Roel 550 22th Lane ,Renton,WA,(639) 555-6238,918-34-5172,25931					
23	Martine Rancé 688 93th Place NW,Kent,WA,(573) 555-3571,695-94-3479,22424					

FlashFill

Program spaces through DSLs

“<<hello>>” → “hello”

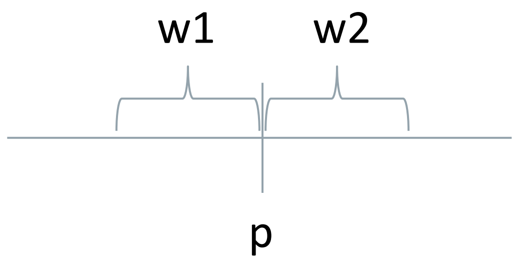
JavaScript:

```
in.substring(in.search(“<<")+2,in.search(“>>"));
```

FlashFill:

```
SubString(in, Pos(“<<“,””), Pos(“”,
“>>”));
```

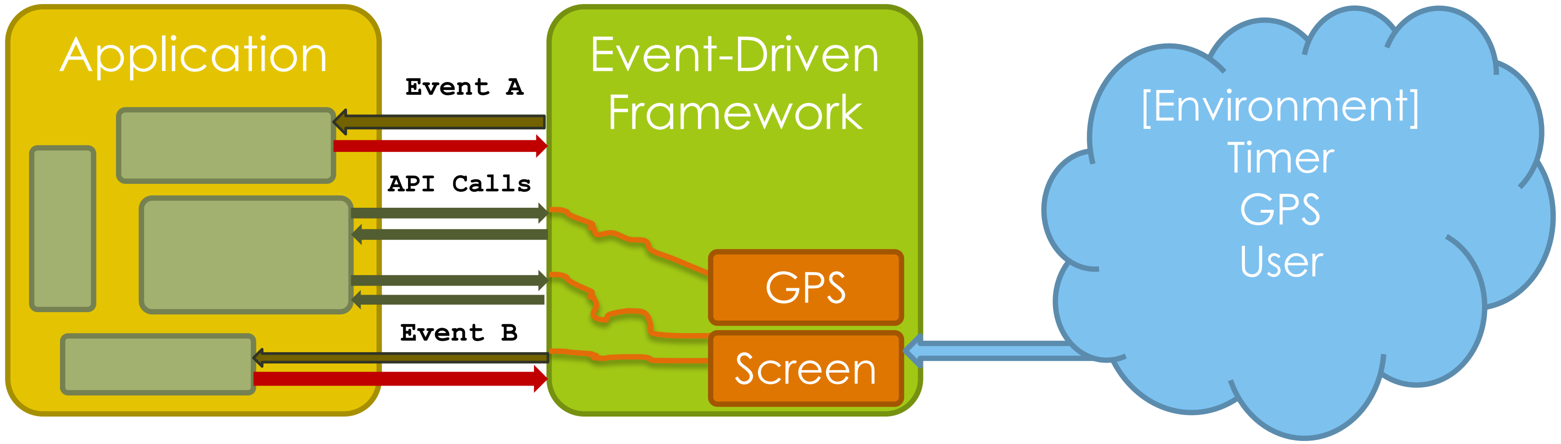
Pos(w1, w2)



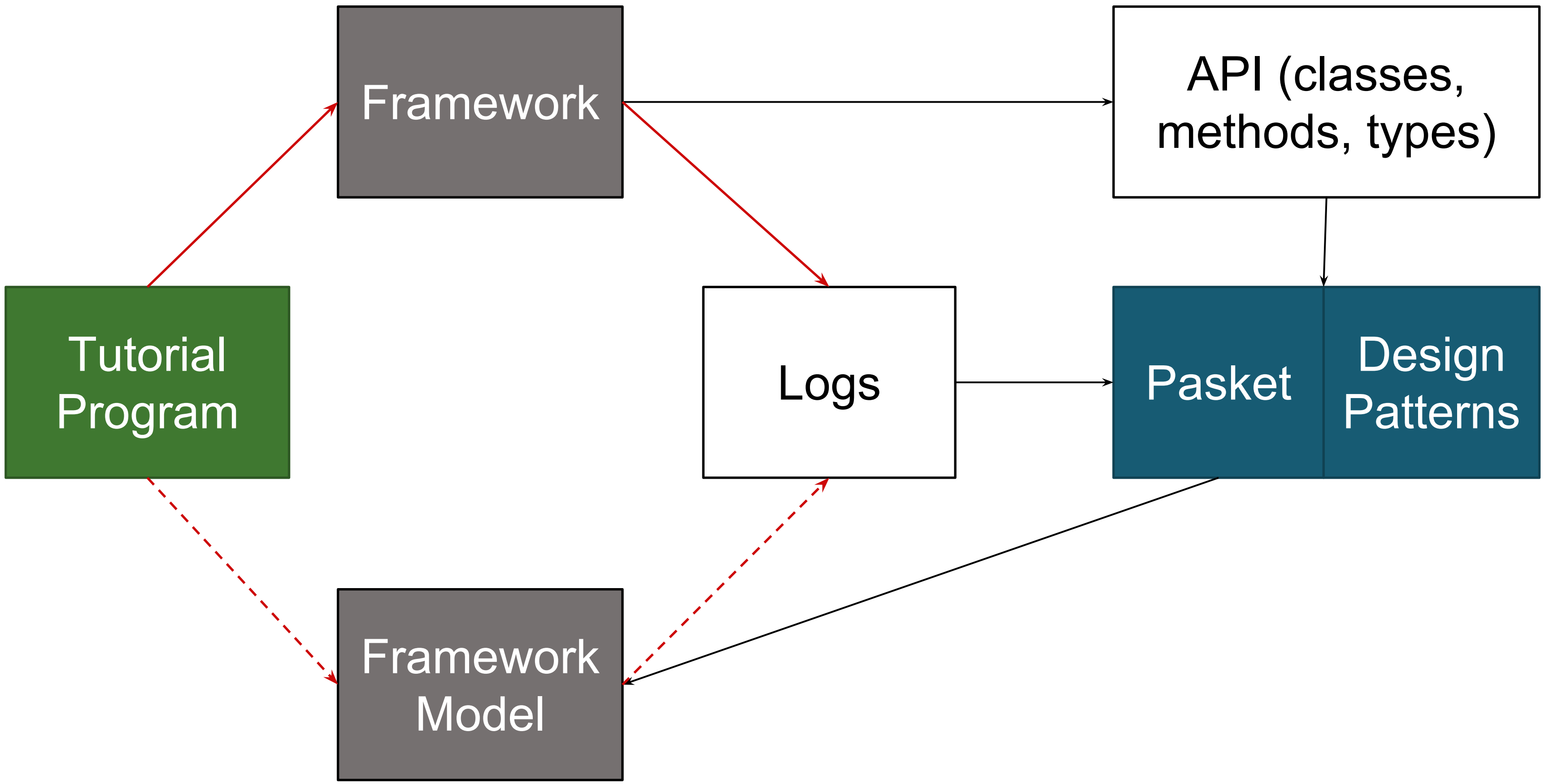
Exciting Directions: Reverse Engineering

Framework Models for Symbolic Execution

Pasket system by J. Jeon, X. Qiu, J. Fetter-Degges, J. S. Foster, and A. Solar-Lezama



Pasket


Framework
API (classes, methods, types)
Tutorial Program
Logs
Pasket
Design Patterns
Framework Model

JPF w(/o) Synthesized Model

```
================== search started: ...
button_demo
--------->property violated
================== error 1
gov.nasa.jpf.vm.NoUncaughtExceptionsProperty
java.lang.NoSuchMethodError:
        javax.swing.JButton.setVerticalTextPosition(I)V
  at ButtonDemo.<init>(ButtonDemo.java:63)
  at ButtonDemo.createAndShowGUI(ButtonDemo.java:131)
  ...
```

(a) With JPF's Swing model.

```
================== search started: ...
button_demo
...
================== results
no errors detected
```

(b) With PASKET's merged model.

JPF along with our synthesized model can run tutorials.

JPF's own hand-written models are insufficient.

- lack of methods: `setVerticalTextPosition`, etc.

An automated process (via Pasket) can avoid simple but nonetheless frustrating problems, like missing methods.

Verified Lifting

Synthesis based reverse engineering can help with optimization

Recent work with by Alvin Cheung and Shoaib Kamil

Optimization then and now

Naïve source code

~~Optimal executable~~

Kind-of-OK executable

Domain specific problem description

SPIRAL

TCE

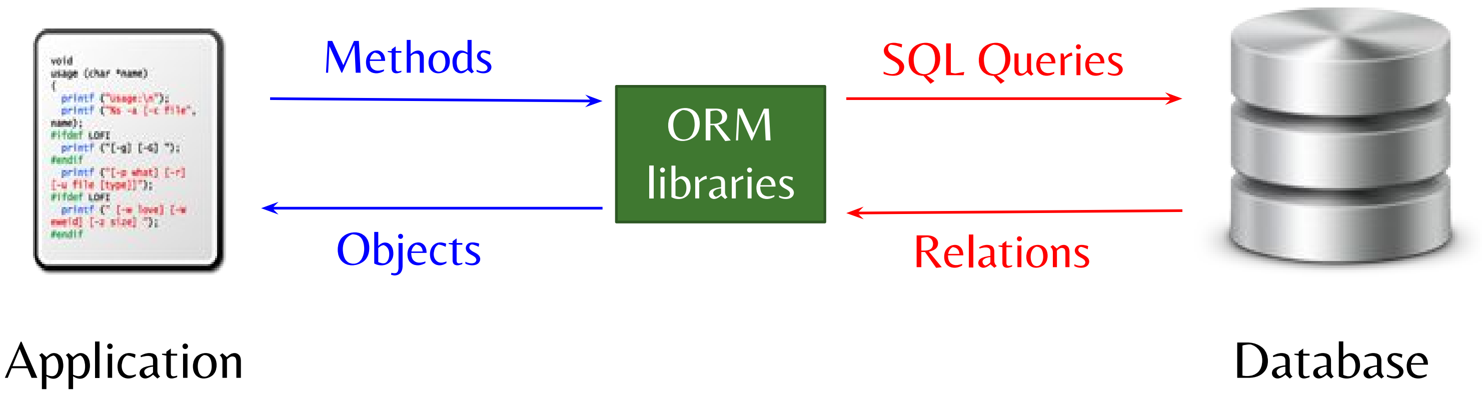
ATLAS

Pochoir

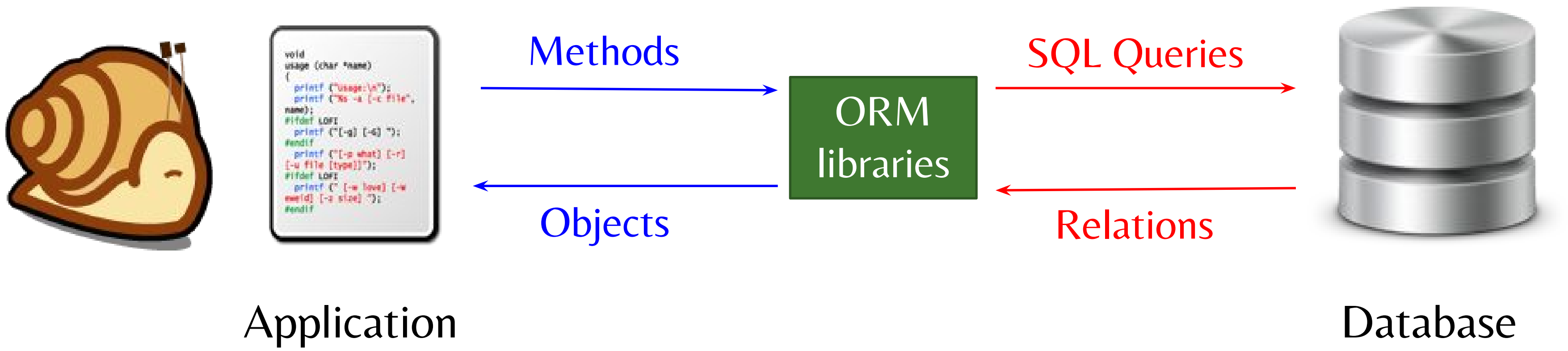
Halide

Close to optimal implementation

Java to SQL


Methods
Objects
ORM
libraries
SQL Queries
Relations
Application
Database

Java to SQL


Methods
Objects
ORM libraries
SQL Queries
Relations
Application
Database

Java to SQL

```
List getUsersWithRoles () {
  List users = User.getAllUsers();
  List roles = Role.getAllRoles();
  List results = new ArrayList();
  for (User u : users) {
   for (Role r : roles) {
        if (u.roleId == r.id)
            results.add(u); }}
  return results; }
```

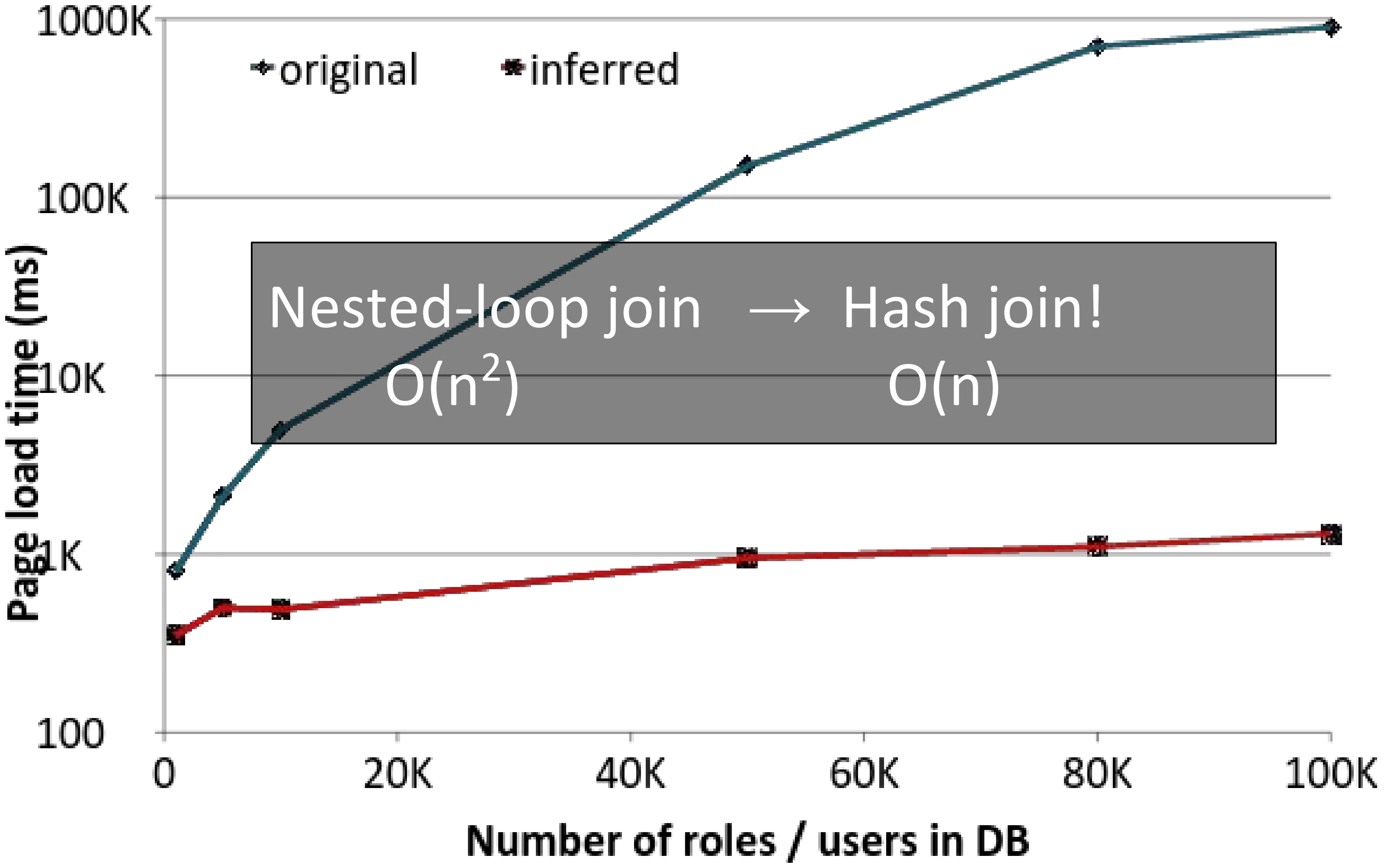
```
SELECT * FROM user
```

```
SELECT * FROM role
```

convert to

```
List getUsersWithRoles () {
 return executeQuery(
  "SELECT   u FROM user u, role r   WHERE    u.roleId == r.id
      ORDER BY u.roleId, r.id"; }
```

Join Query



1000K
100K
10K
1K
100
original
inferred
Page load time (ms)
0
20K
40K
60K
80K
100K
Number of roles / users in DB
Nested-loop join → Hash join!
O(n²)
O(n)

Example: MultiGrid

```
DO i3 = 2, n3 - 1
DO i2 = 2, n2 - 1
DO i1 = 1, n1
r1(i1) = r(i1,i2 - 1,i3) + r(i1,i2 + 1,i3) + r(i1,i2,i3 - 1) + r(i1,i2,i3 + 1)
r2(i1) = r(i1,i2 - 1,i3 - 1) + r(i1,i2 + 1,i3 - 1) + r(i1,i2 - 1,i3 + 1) + r(i1,i2 + 1,i3 + 1)
END DO
DO i1 = 2, n1 - 1
u(i1,i2,i3) = u(i1,i2,i3) + c(0) * r(i1,i2,i3) + c(1) * (r(i1 - 1,i2,i3) + r(i1 + 1,i2,i3) + r1(i1)) + c(2) * (r2(i1) + r1(i1 - 1) + r1(i1 + 1))
END DO
END DO
END DO
```

Example: MultiGrid

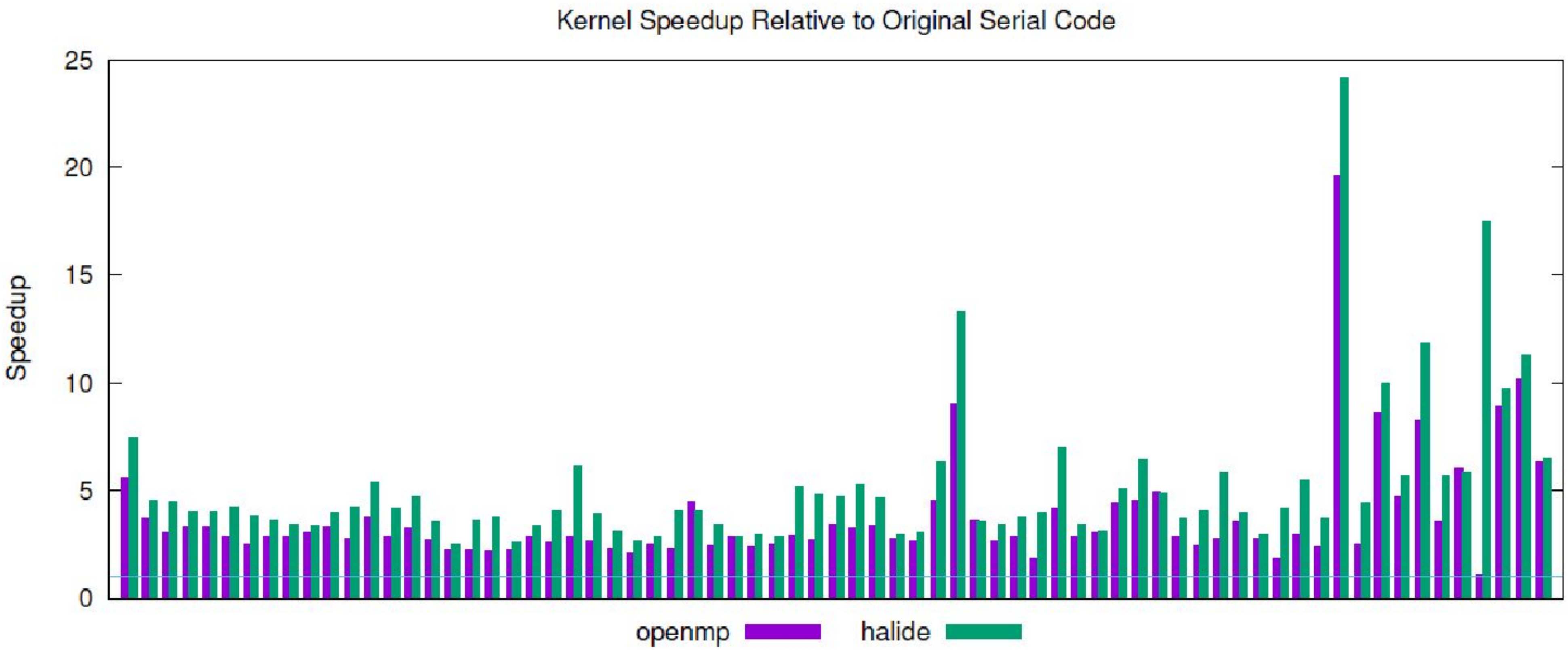
```
/*Range declarations go here */

r1_out(n1) = r(n1,n2-2,n3-1) + r(n1,n2,n3-1) + r(n1,n2-1,n3-2) + r(n1,n2-1,n3)
r2_out(n1) = r(n1,n2-2,n3-2) + r(n1,n2,n3-2) + r(n1,n2-2,n3) + r(n1,n2,n3)

u_out(i1,i2,i3) = u(i1,i2,i3) + c(0) * r(i1,i2,i3)
          + c(1) * (r(i1 - 1,i2,i3) + r(i1 + 1,i2,i3) + r(i1,i2 - 1,i3) + r(i1,i2 + 1,i3) + r(i1,i2,i3 - 1) + r(i1,i2,i3 + 1))
          + c(2) * ((r(i1,i2 - 1,i3 - 1) + r(i1,i2 + 1,i3 - 1) + r(i1,i2 - 1,i3 + 1) + r(i1,i2 + 1,i3 + 1))
                   + (r(i1 - 1,i2 - 1,i3) + r(i1 - 1,i2 + 1,i3) + r(i1 - 1,i2,i3 - 1) + r(i1 - 1,i2,i3 + 1))
                   + (r(i1 + 1,i2 - 1,i3) + r(i1 + 1,i2 + 1,i3) + r(i1 + 1,i2,i3 - 1) + r(i1 + 1,i2,i3 + 1)))

Tuple my_output(r1_out, r2_out, u_out);
```

Speedups



Speedups on 24 cores

Exciting Directions: Synthesis for Synthesis

Can our solvers help us write better solvers?

Solvers are hard to write

Tradeoff between performance and maintainability

No single best approach

- NP complete problems after all

Clean formalizations

Good target for synthesis!

Sketch Simplifier

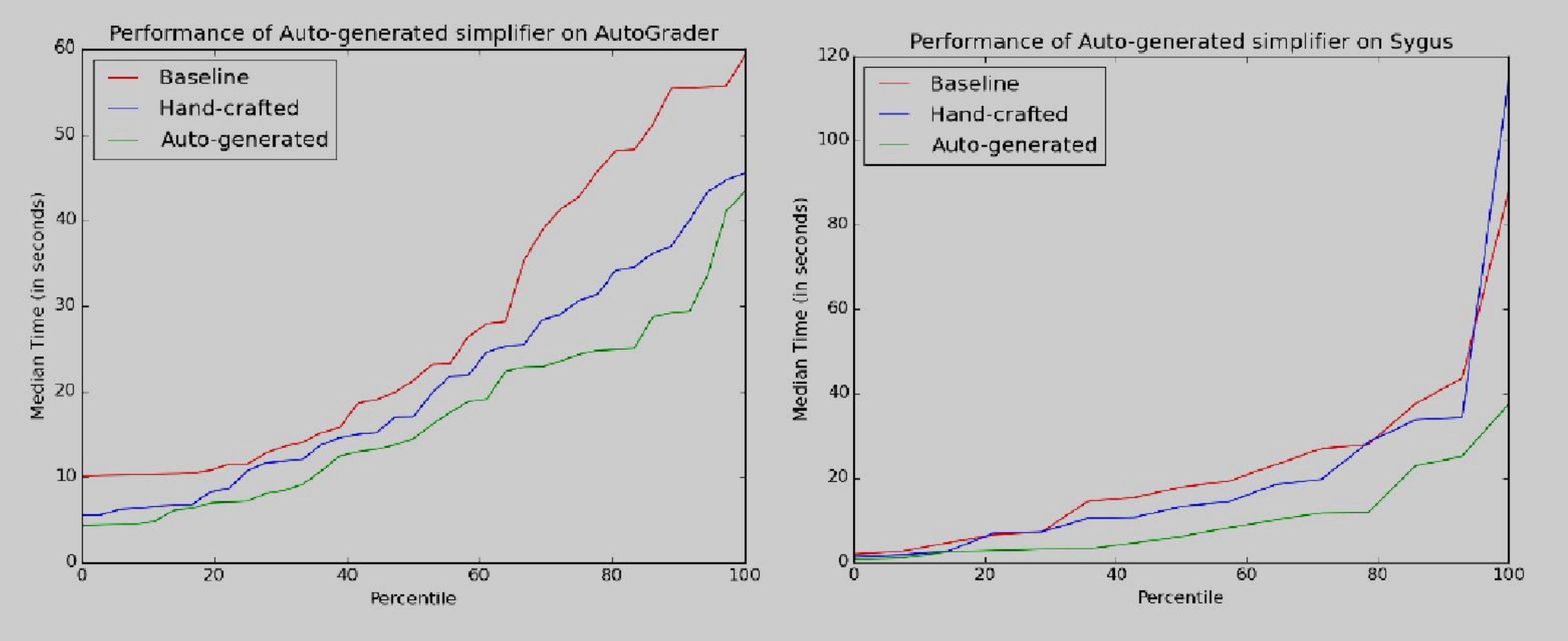
$$a+e < x \quad \& \quad e+b < x \quad \xrightarrow{b<a} \quad a+e < x$$

```
if(nfather->type == LT && nmother->type == LT){
  // (a+e<x) & (b+e<x) ---> a+e<x when b<a
  if(nfather->mother->type == PLUS && nmother->mother->type == PLUS){
    bool_node* nfm = nfather->mother;
    bool_node* nmm = nmother->mother;

    bool_node* nmmConst = nmm->mother;
    bool_node* nmmExp = nmm->father;
    if(isConst(nmmExp)){
      bool_node* tmp = nmmExp;
      nmmExp = nmmConst;
      nmmConst = tmp;
    }
    bool_node* nfmConst = nfm->mother;
    bool_node* nfmExp = nfm->father;
    if(isConst(nfmExp)){
      bool_node* tmp = nfmExp;
      nfmExp = nfmConst;
      nfmConst = tmp;
    }
if(isConst(nfmConst) && isConst(nmmConst) && nfmExp== nmmExp){
      if(val(nfmConst) < val(nmmConst)){
        return nmother;
      }else{
        return nfather;
      }
} } }
```

Performance

Impact on times



AutoGrader: 27.5s,20s,18s average times

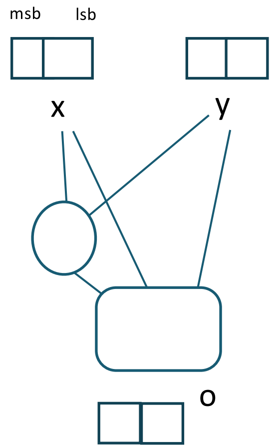
Sygus: 22s,21s,10s average times

Bit-vector encoding

Boolean predicate P → CNF clauses C

$$o = ITE_N((GT_N, x, y), x, y)$$ →

$t1 = true$
$t2 = true$
$for\ i\ from\ N\ to\ 1:$
$\quad t3 = newVar$
$\quad t4 = newVar$
$\quad clause(\{x[i], y[i], \overline{o[i]}\})$
$\quad clause(\{x[i], \overline{t1}, t3\})$
$\quad clause(\{x[i], \overline{t2}, o[i], t4\})$
$\quad clause(\{x[i], \overline{o[i]}, \overline{t3}\})$
$\quad clause(\{\overline{x[i]}, \overline{y[i]}, o[i]\})$
$\quad clause(\{\overline{x[i]}, \overline{t2}, o[i]\})$
$\quad clause(\{\overline{x[i]}, \overline{t2}, t4\})$
$\quad clause(\{y[i], \overline{t2}, t4\})$
$\quad clause(\{y[i], \overline{o[i]}, \overline{t3}\})$
$\quad clause(\{\overline{y[i]}, \overline{t1}, t3\})$
$\quad clause(\{\overline{y[i]}, o[i], \overline{t3}\})$
$\quad clause(\{t1, \overline{t3}\})$
$\quad clause(\{\overline{t1}, o[i], t3\})$
$\quad clause(\{t2, \overline{t4}\})$
$\quad clause(\{t3, \overline{t4}\})$
$\quad t1 = t3$
$\quad t2 = t4$



Solve more problems

Benchmark Family	Solved by CVC4 → Our Solver
Log-slicing (79)	*33 → 62*
ASP (365)	*240 → 288*
Mcm (61)	*40 → 43*
Brummayerbiere2 (33)	*28 → 29*
Float (62)	*59 → 60*
Brummayerbiere3 (40)	*23 → 24*
Bruttomesso (676)	*623 → 623*
TOTAL	***1046 → 1129***

83 more problems in total

Cross domain performance

Solver → / Domain ↓	log-slicing	asp	mcm	brumma2	float	brumma3	brutto
log-slicing	62	58	36	59	32	35	35
asp	227	288	255	227	236	253	240
mcm	39	38	43	40	39	39	41
brumma2	29	28	28	29	29	29	29
float	57	57	59	57	60	60	59
brumma3	22	22	25	22	23	24	23
brutto	607	606	623	609	623	623	623

Best → Worst

Exciting Directions: Quantitative Synthesis

Synthesis meets ML

STOKE

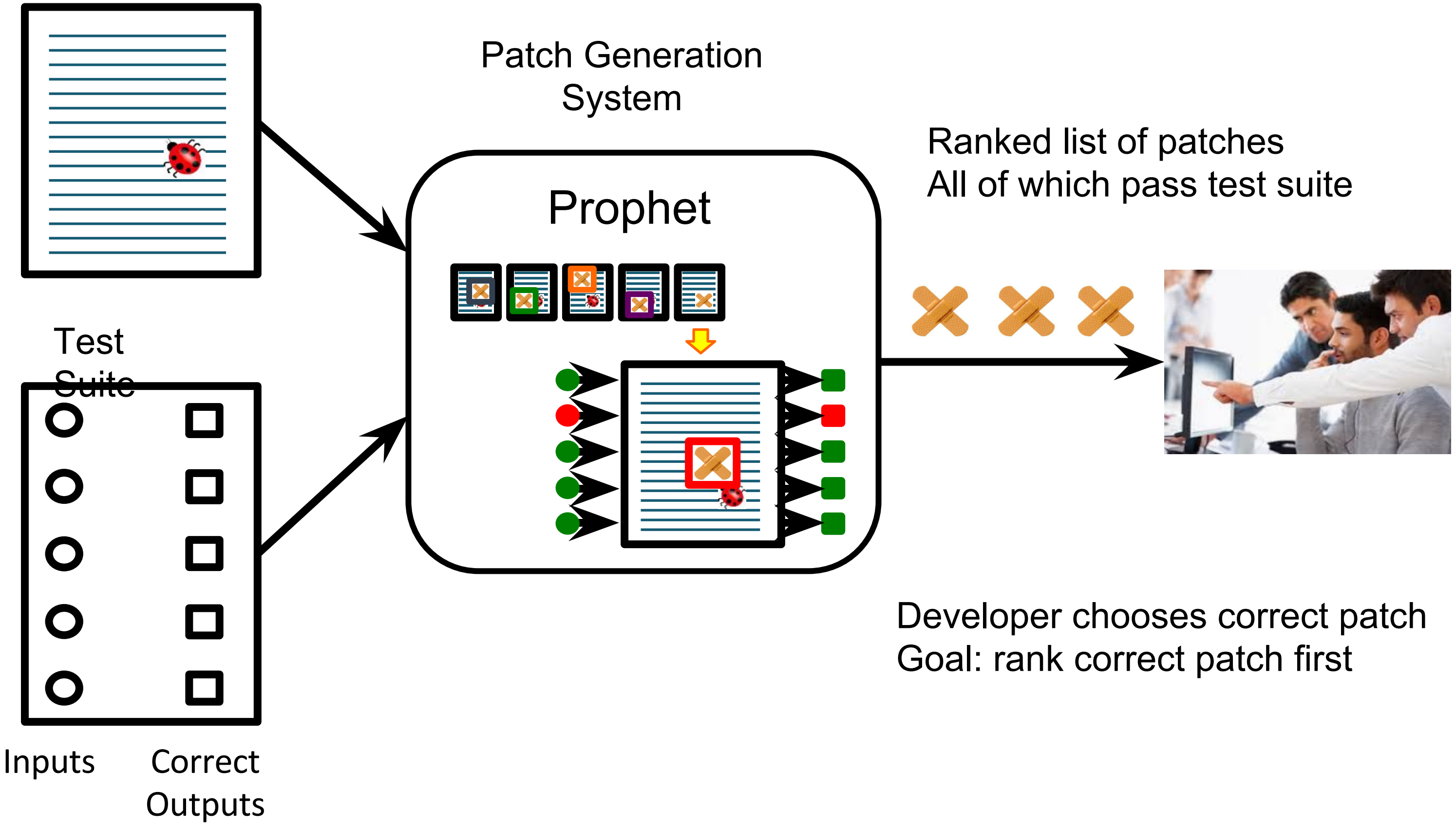
Project by Schkufza, Sharma, Heule, Aiken

Leverages Stochastic Search (MCMC) to incorporate quantitative parameters such as precision and performance

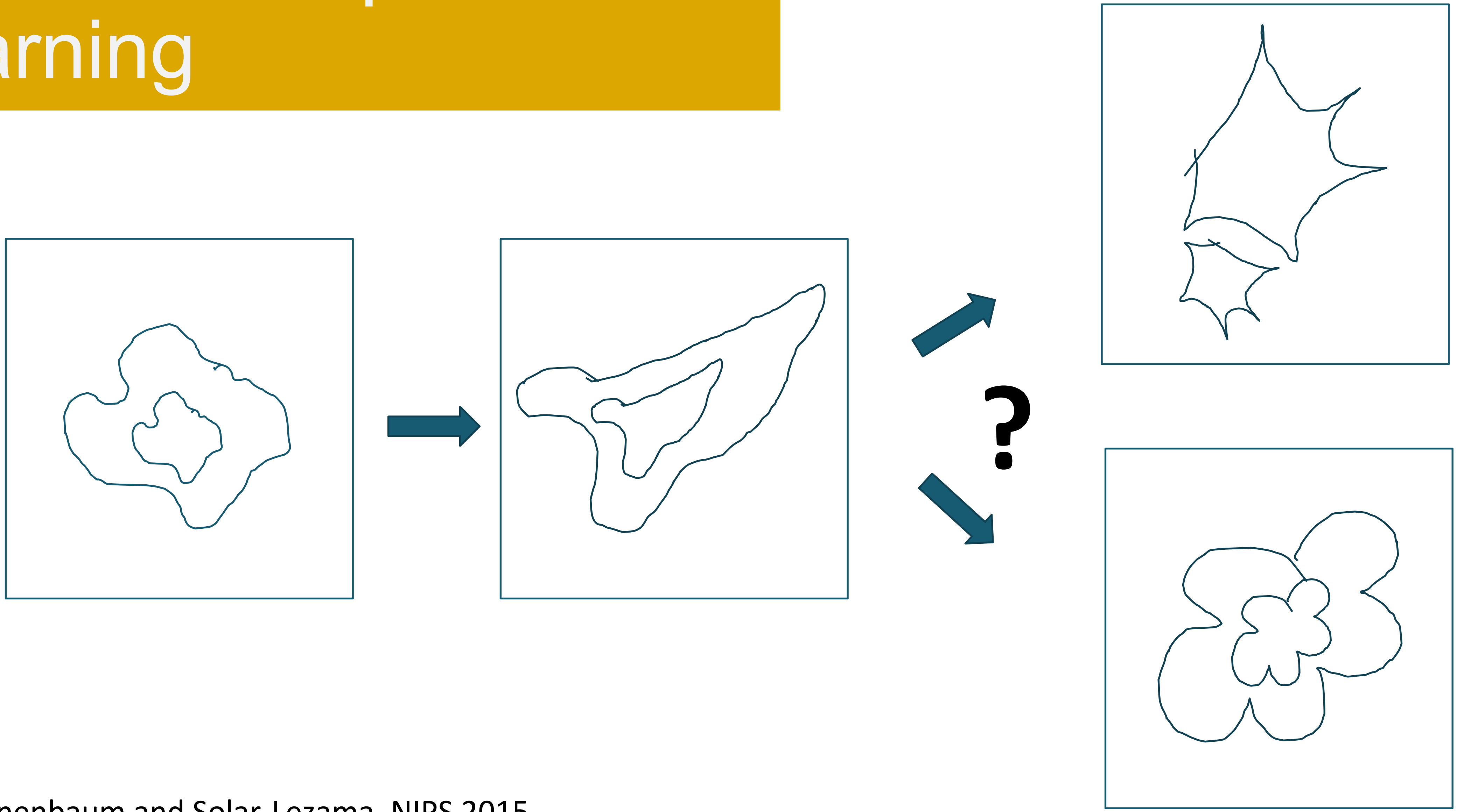
Focus on optimization

Prophet/Genesis

Project by Fan Long, Stelios Sidiroglou and Martin Rinard

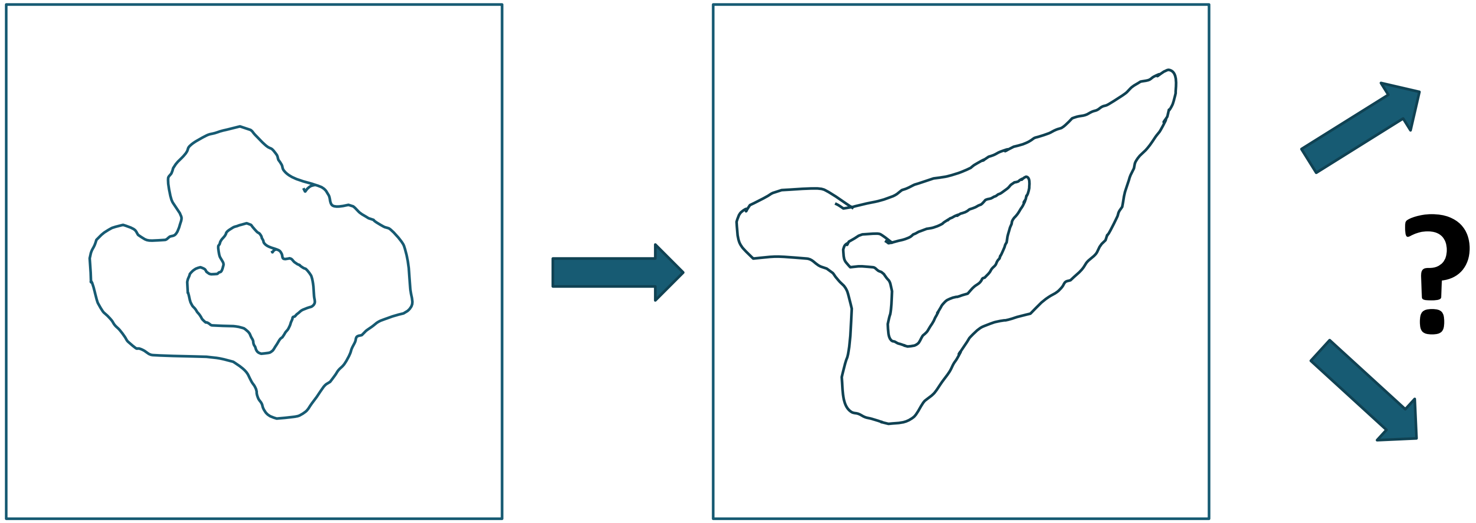


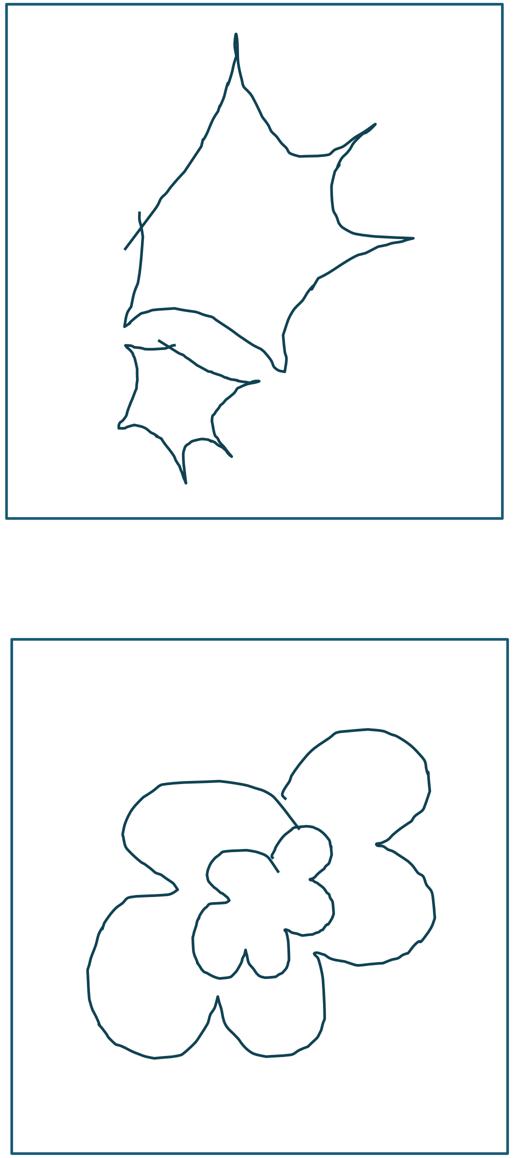
Visual Concept Learning



Ellis, Tenenbaum and Solar-Lezama, NIPS 2015

Visual Concept Learning





```
teleport(position[0], 0)
draw(shape[0], scale=1.0)
draw(shape[0], scale=0.5)
```

Synthesis vs. ML

Quantitative synthesis is at the intersection of synthesis and ML

Synthesis > ML

Big data vs. Small data

- Sometimes generating examples is expensive

I know what I want

- ML is heavily concerned with noise
- By design, it won't give you what you ask for

I know what I want (2)

- Difficult to incorporate hard constraints

ML > Synthesis

Big data vs. Small data

- Sometimes you really do have a lot of data, why waste it?

I know what I want

- Do you really?

You can do this too!

Synthesis Infrastructure

Sketch

- Just released v. 1.7.4
- Mature infrastructure with an expressive frontend language

SyGuS

- Family of solvers supporting emerging SYNT-LIB standard
- Less expressive than sketch, but higher performance
- Strong community support

Prose

- Infrastructure by Sumit Gulwani's team for DSL-based synthesis

Conclusion

The drive for automation continues

Synthesis provides a new set of tools to attack complex problems

We are just beginning to understand how to use this technology to improve productivity