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
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 FOR-
TRAN 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 con-
cerning 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
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. Further, more than 30 percent of the elapsed time for a problem was usually devoted to planning, writing, and checking the programs produced, a 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?

Code

Uh... no...

It’s called code.
Synthesis: Dreams $\rightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

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

$\phi(p) = \forall in. \ldots p(in) \ldots$
Example

Sketch

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

Spec

```plaintext
    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

$$(x \& y) + (x ^ {\land} y) \gg 1$$

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

1 int take() {
2   long b = bottom - 1;
3   item_t * q = wsq;
4   bottom = b
5   long t = top
6   ...
}

Minimal Synchronization

1 int take() {
2   long b = bottom - 1;
3   item_t * q = wsq;
4   bottom = b
5   long t = top
6   ...
}

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
<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Phone Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ana Trujillo</td>
<td>357 21th Place SE, Redmond, WA</td>
<td>555-1634, 140-37-6064, 27171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antonio Moreno</td>
<td>515 93th Lane, Renton, WA</td>
<td>555-2766, 662-87-3127, 28581</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thomas Hardy</td>
<td>742 17th Street NE, Seattle, WA</td>
<td>555-6719, 921-29-4931, 24607</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Christina Berglund</td>
<td>475 22th Lane, Redmond, WA</td>
<td>555-6774, 844-35-6764, 30146</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hanna Moos</td>
<td>785 45th Street NE, Puyallup, WA</td>
<td>555-2462, 615-68-1285, 29284</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frederick Cieux</td>
<td>308 66th Place, Redmond, WA</td>
<td>699-2700, 552-23-2508, 21416</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Martin Sommer</td>
<td>887 86th Place, Kent, WA</td>
<td>555-6450, 870-91-9824, 21536</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laurence Leibihan</td>
<td>944 13th Street NE, Redmond, WA</td>
<td>655-2331, 649-25-5312, 25252</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elizabeth Lincoln</td>
<td>452 73th Lane, Renton, WA</td>
<td>655-4661, 426-97-6344, 22279</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Victoria Ashworth</td>
<td>463 16th Street, Renton, WA</td>
<td>655-6044, 690-29-7252, 22832</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patricio Simpson</td>
<td>630 20th Street, Redmond, WA</td>
<td>555-3268, 389-73-3236, 24525</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Francisco Chang</td>
<td>683 49th Lane, Seattle, WA</td>
<td>555-7434, 685-18-6435, 29453</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yang Wang</td>
<td>944 28th Lane, Redmond, WA</td>
<td>555-2272, 846-78-8452, 24888</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pedro Afonso</td>
<td>411 70th Place, Kent, WA</td>
<td>555-3964, 799-25-29488, 21448</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elizabeth Brown</td>
<td>971 20th Lane, Puyallup, WA</td>
<td>555-4134, 476-53-7164, 2017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sven Ottoleb</td>
<td>676 17th Lane NE, Redmond, WA</td>
<td>655-1593, 584-73-8633, 27440</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Janine Labrune</td>
<td>267 95th Place SE, Seattle, WA</td>
<td>949-555-1316, 350-27-8300, 20074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ann Devon</td>
<td>694 53th Place, Kent, WA</td>
<td>655-8124, 359-74-4016, 22367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roland Mendel</td>
<td>561 12th Street NW, Kent, WA</td>
<td>655-2146, 303-79-1328, 20518</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arla Cruz</td>
<td>504 85th Lane, Renton, WA</td>
<td>555-1376, 325-99-9992, 21496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diego Roel</td>
<td>550 22th Lane, Renton, WA</td>
<td>655-6238, 918-34-5172, 25931</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Martine Rancé</td>
<td>688 93th Place NW, Kent, WA</td>
<td>573-555-3571, 695-94-3479, 22424</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Program spaces through DSLs

“<<hello>>” → “hello”

JavaScript:
in.substring(in.search("<<")+2,in.search(">>"));

FlashFill:
Substring(in, Pos("<<",""), Pos(""", ">>"));
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

Tutorial Program

Framework Model

Logs

API (classes, methods, types)

Pasket

Design Patterns
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

Close to optimal implementation

ATLAS
Pochoir
Halide

Spiral

GCC
Fortran
C++
Java to SQL
Java to SQL

Application → ORM libraries

Methods → SQL Queries

Objects ← Relations

ORM libraries

Database
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;
}

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

Nested-loop join $\rightarrow$ Hash join!

$O(n^2)$ $\rightarrow$ $O(n)$
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 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

\[ a + e < x \quad \& \quad e + b < x \quad \rightarrow \quad b < a \quad \rightarrow \quad a + e < x \]

```cpp
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 $P$

\[ o = \text{ITE}_N((GT_N, x, y), x, y) \]

CNF clauses $C$

\[
\begin{align*}
t1 &= \text{true} \\
t2 &= \text{true} \\
\text{for } i \text{ from } N \text{ to } 1 : \\
t3 &= \text{newVar} \\
t4 &= \text{newVar} \\
\text{clause}(\{x[i], y[i], \overline{o[i]}\}) \\
\text{clause}(\{x[i], t1, t3\}) \\
\text{clause}(\{x[i], t2, o[i], t4\}) \\
\text{clause}(\{x[i], o[i], t3\}) \\
\text{clause}(\{x[i], y[i], o[i]\}) \\
\text{clause}(\{x[i], t2, o[i]\}) \\
\text{clause}(\{x[i], t2, t4\}) \\
\text{clause}(\{y[i], t2, t4\}) \\
\text{clause}(\{y[i], o[i], t3\}) \\
\text{clause}(\{y[i], t1, t3\}) \\
\text{clause}(\{y[i], o[i], t3\}) \\
\text{clause}(\{t1, t3\}) \\
\text{clause}(\{t1, o[i], t3\}) \\
\text{clause}(\{t2, t4\}) \\
\text{clause}(\{t3, t4\}) \\
t1 &= t3 \\
t2 &= t4
\end{align*}
\]
Solve more problems

<table>
<thead>
<tr>
<th>Benchmark Family</th>
<th>Solved by CVC4</th>
<th>Our Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log-slicing (79)</td>
<td>33 → 62</td>
<td></td>
</tr>
<tr>
<td>ASP (365)</td>
<td>240 → 288</td>
<td></td>
</tr>
<tr>
<td>Mcm (61)</td>
<td>40 → 43</td>
<td></td>
</tr>
<tr>
<td>Brummayerbiere2 (33)</td>
<td>28 → 29</td>
<td></td>
</tr>
<tr>
<td>Float (62)</td>
<td>59 → 60</td>
<td></td>
</tr>
<tr>
<td>Brummayerbiere3 (40)</td>
<td>23 → 24</td>
<td></td>
</tr>
<tr>
<td>Bruttomesso (676)</td>
<td>623 → 623</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>1046 → 1129</strong></td>
<td></td>
</tr>
</tbody>
</table>

83 more problems in total
## Cross domain performance

<table>
<thead>
<tr>
<th>Solver Domain</th>
<th>log-slicing</th>
<th>asp</th>
<th>mcm</th>
<th>brumma2</th>
<th>float</th>
<th>brumma3</th>
<th>brutto</th>
</tr>
</thead>
<tbody>
<tr>
<td>log-slicing</td>
<td>62</td>
<td>58</td>
<td>36</td>
<td>59</td>
<td>32</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>asp</td>
<td>227</td>
<td>288</td>
<td>255</td>
<td>227</td>
<td>236</td>
<td>253</td>
<td>240</td>
</tr>
<tr>
<td>mcm</td>
<td>39</td>
<td>38</td>
<td>43</td>
<td>40</td>
<td>39</td>
<td>39</td>
<td>41</td>
</tr>
<tr>
<td>brumma2</td>
<td>29</td>
<td>28</td>
<td>28</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>float</td>
<td>57</td>
<td>57</td>
<td>59</td>
<td>57</td>
<td>60</td>
<td>60</td>
<td>59</td>
</tr>
<tr>
<td>brumma3</td>
<td>22</td>
<td>22</td>
<td>25</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>brutto</td>
<td>607</td>
<td>606</td>
<td>623</td>
<td>609</td>
<td>623</td>
<td>623</td>
<td>623</td>
</tr>
</tbody>
</table>
Exciting Directions: Quantitative Synthesis

Synthesis meets ML
Project by Schkufza, Sharma, Heule, Aiken

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

Focus on optimization
Project by Fan Long, Stelios Sidiroglou and Martin Rinard
Visual Concept Learning

Ellis, Tenenbaum and Solar-Lezama, NIPS 2015
teleport(position[0], 0)
draw(shape[0], scale=1.0)
draw(shape[0], scale=0.5)
Quantitative synthesis is at the intersection of synthesis and 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
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
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