Z3 and Me

Chad Brewbaker
DataCulture LLC

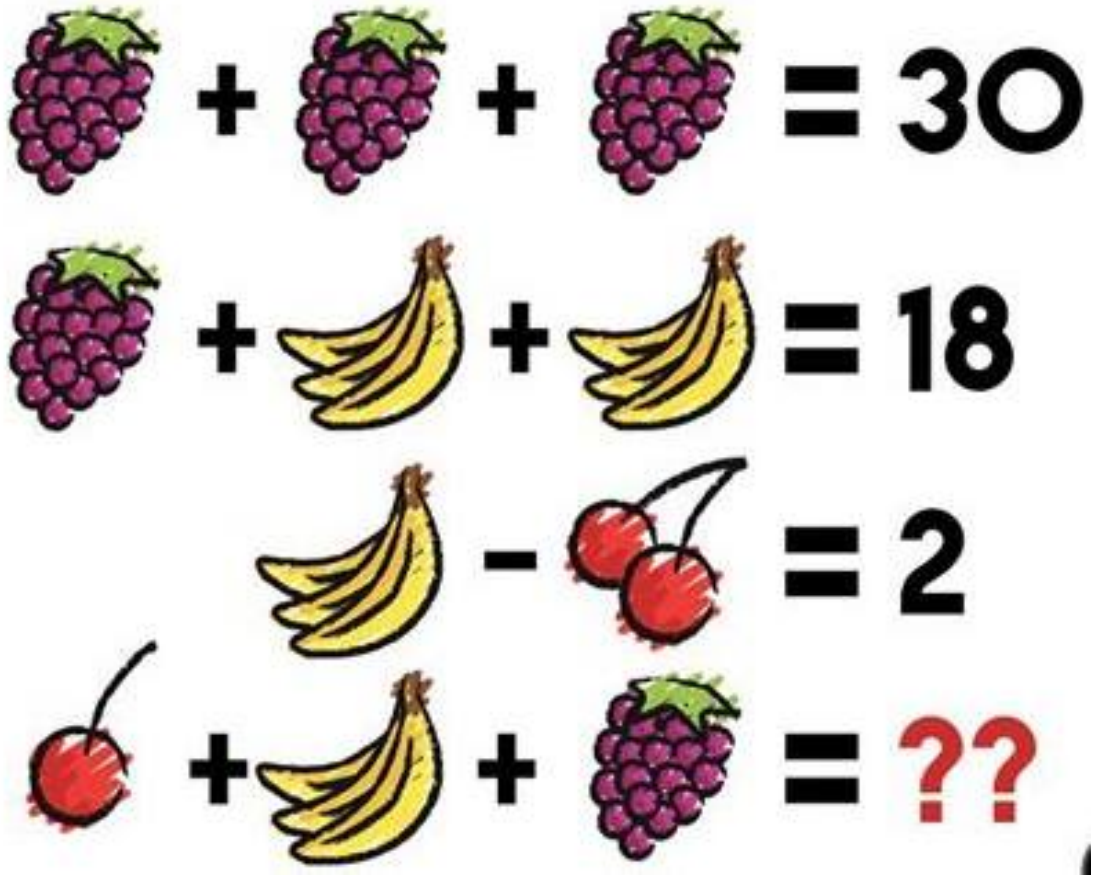
May 23, 2016

Z3 is an open source SMT solver released by Microsoft.
SMT = Satisfiability Modulo Theories

Boolean logic (True, False, And, Or, Not) +
Theories: empty theory, linear arithmetic, nonlinear arithmetic, bitvectors, arrays, datatypes, and quantifiers

Why would I want to use a SMT solver?

Solve logic problems.
Optimize processes.
Prove correctness.


+ + = 30
+ + = 18
- = 2
+ + = ??

```
from z3 import *

s = Solver()
cherry = Int('cherry')
cherries = Int('cherries')
grape = Int('grape')
banana = Int('banana')
s.add( (grape + grape + grape) == 30)
s.add( (grape + banana + banana) == 18)
s.add( (banana - cherries) == 2)
if( s.check() == unsat):
        print 'FAIL'
else:
        print s.model()
```

[cherries = 2, banana = 4, grape = 10]

SAT/SMT Solver Performance over time

1998 1,000 contraints

2001 10,000 constraints

2005 100,000 constraints

2010 1,000,000 constraints

(Data by Vijay Ganesh)

"Control-flow hijacking vulnerabilities represent more than 50% of all vulnerabilities reported." - Oliveira 2016

"Rainbow Books scene", Hackers, United Artists, 1995

Microsoft SAGE Fuzzer
Built on Z3.
Proprietary.
Found $\frac{1}{3}$ bugs in Windows 7-10.

Symbolic + Concrete = Concolic testing

Compile with code coverage.

Find untouched code branches.

Use SMT solver find assignment of variables that allows branch to be entered.

Concolic testing example:

Fizz Buzz Jaberwocky

```
from z3 import *

s = Solver()
num = Int('num')
s.add( num > 300)
s.add( num % 13 == 7)
s.add( num % 7 == 0)
s.add( num % 5 != 0)
s.add( num % 3 != 0)
if( s.check() == unsat):
        print 'FAIL'
else:
        print s.model()
```

[num = 826]

American Fuzzy Lop 

https://fuzzing-project.org/

ImageTragick, Shellshock, PHP, OpenSSL, Mozilla, SQLlite, ...

American Fuzzy Lop demo on stock OSX.

Systems data one liners.
How do we inspect assembler calls?
How do we inspect raw operating system calls?
How do we inspect network calls?

Spying on C builds.
make VERBOSE=1 (Cmake)
make V=1 (GNU autotools)

clang -Os -S -emit-llvm sample.c -o sample.ll

llc -O3 sample.ll -march=x86 -o sample-x86.s

llc -O3 sample.ll -march=arm -o sample-arm.s

Spying on system calls.
strace (Linux)
dtrace (OS X)

Spying on network calls.

sudo tcpdump -i en0

Use functional tools for your platform.

C++ $\rightarrow$ (Haskell, Elixir OTP)

Java VM $\rightarrow$ (Scala, Frege)

Javascript $\rightarrow$ (Elm, Typescript)

Hopper, formally specify multi-party contracts

Don't let contract attorneys hand waive. Prove coorectness.

Don't let contract attorneys waste executive manpower. Automate transactions.

https://github.com/hopper-lang/hopper

1) Formal parsers on all inputs.

2) Isolate pure code from IO code.

3) Make the compiler do more work, not the tester.

4) Runtime monitoring with strace on vendor code.

5) Decouple monoliths.

6) Use Packer to maintain all dependencies. No black boxes.

http://leanprover.github.io

http://z3prover.github.io

http://www.sympy.org/en/index.html

http://saw.galois.com

http://leventerkok.github.io/sbv/

Girard (1987), Linear Logic
Tinelli (2006), The Satisfiability Modulo Theories Library
Ganesh (2007), Decision Procedures for Bit-Vectors, Arrays and Integers
Lezama (2008), Program synthesis by sketching
De Moura (2008), Z3: An efficient SMT solver
Brummayer (2009), Fuzzing and delta-debugging SMT solvers
Jovanovic (2015) Analysis and Design of Symmetric Cryptographic Algorithms