Introduction to Formal Analysis

Mark Greenstreet

CpSc 513 – Term 1, 2015/16

- Formal verification uses algorithms to rigorously prove properties of hardware or software design.
- 20 years ago, we had the `FDIV` bug: `42.0 / 7.0 = 8.0`
  - Formal methods were on the edge of solving HW problems.
  - Now they are standard throughout the industry.
- Today, we've had HeartBleed, and many, many others.
  - Formal methods are on the rise for SW verification.
  - Formal methods are used at Amazon, IBM, Microsoft, Oracle, and many other companies.

Fun with Formal Methods

Here's a puzzle from the most recent *ACM XRDS* magazine.
There are five ships in a port:

1. The Greek ship leaves at six and carries coffee.
2. The ship in the middle has a black chimney.
3. The English ship leaves at nine.

. . .

15. The ship to Hamburg leaves at six.

Which ship is going to Port Said? Which ship carries tea?

We could solve this by thinking, or . . .

. . . we can program!

Using the Z3 SMT solver with its Python API:

```
# define the sets used in the puzzle
Nationality, (Brazilian, ...)  = \
   EnumSort("Nationality", ("Brazilian", ...))
...
# the clues
clues.append(And(departs(6) == nationality(Greek), ...))
clues.append(pier(3) == chimney(black))
clues.append(departs(9) == nationality(English))
...
# invoke the solver and print
s = Solver()
s.add(And(*clues))
s.check()
m = s.model()
for p in range(1,6):
   print "pier=" + str(p) + ...
```

And the answer is…

```
pier=1:  French tea blue Genoa 5
pier=2:  Greek coffee red Hamburg 6
pier=3:  Brazilian cocoa black Manila 8
pier=4:  English rice white Marseille 9
pier=5:  Spanish corn green Port Said 7
```

- The Spanish ship is going to Port Said, and
- The French ship is carrying tea.

Finding bugs is just solving gargantuan puzzles:

Can we find:

- A combination of inputs that enables a security exploit?
- A sequence of memory operations that corrupts the processor's cache and memory?
- A traffic situation that causes a self-driving cars to collide?

Can we *prove* that such failures *never* occur?

Today's formal verification tools can:

- Solve satisfiability problems with tens of thousands of variables.
- Automatically analyse programs that are tens to hundreds of thousands of lines, checking for common errors.
- Check logical properties of hardware designs with millions of flip-flops and gates.

Formal verification combines

- The latest research in algorithms, automata theory, game theory, control theory, and heuristics
- Implementing software to scale to realistic problems
- To have impact on real software, hardware, and cyber-physical designs.

To learn more:

- Come to CpSc 513, T/Th, 11am, DMP 101.
- http://www.cs.ubc.ca/~mrg/cs513/index.html

p.s.

- The ISD faculty (Alan Hu and Mark Greenstreet) have funding and are looking for students.
- You don't need to know about all the areas listed above to get started. If you like theory and math and want to do something with the potential for big impact in industry, then CpSc 513 is for you!