A Little History

The idea is an old one

- Robert S. Boyer, Bernard Elspas, and Karl N. Levitt. **SELECT–a formal system for testing and debugging programs by symbolic execution**. In ICRS, pages 234–245, **1975**.
- James C. King. **Symbolic execution and program testing**. CACM, 19(7):385–394, **1976**. **(most cited)**
- Leon J. Osterweil and Lloyd D. Fosdick. **Program testing techniques using simulated execution**. In ANSS, pages 171–177, **1976**.
- William E. Howden. **Symbolic testing and the DISSECT symbolic evaluation system**. IEEE Transactions on Software Engineering, 3(4):266–278, **1977**.

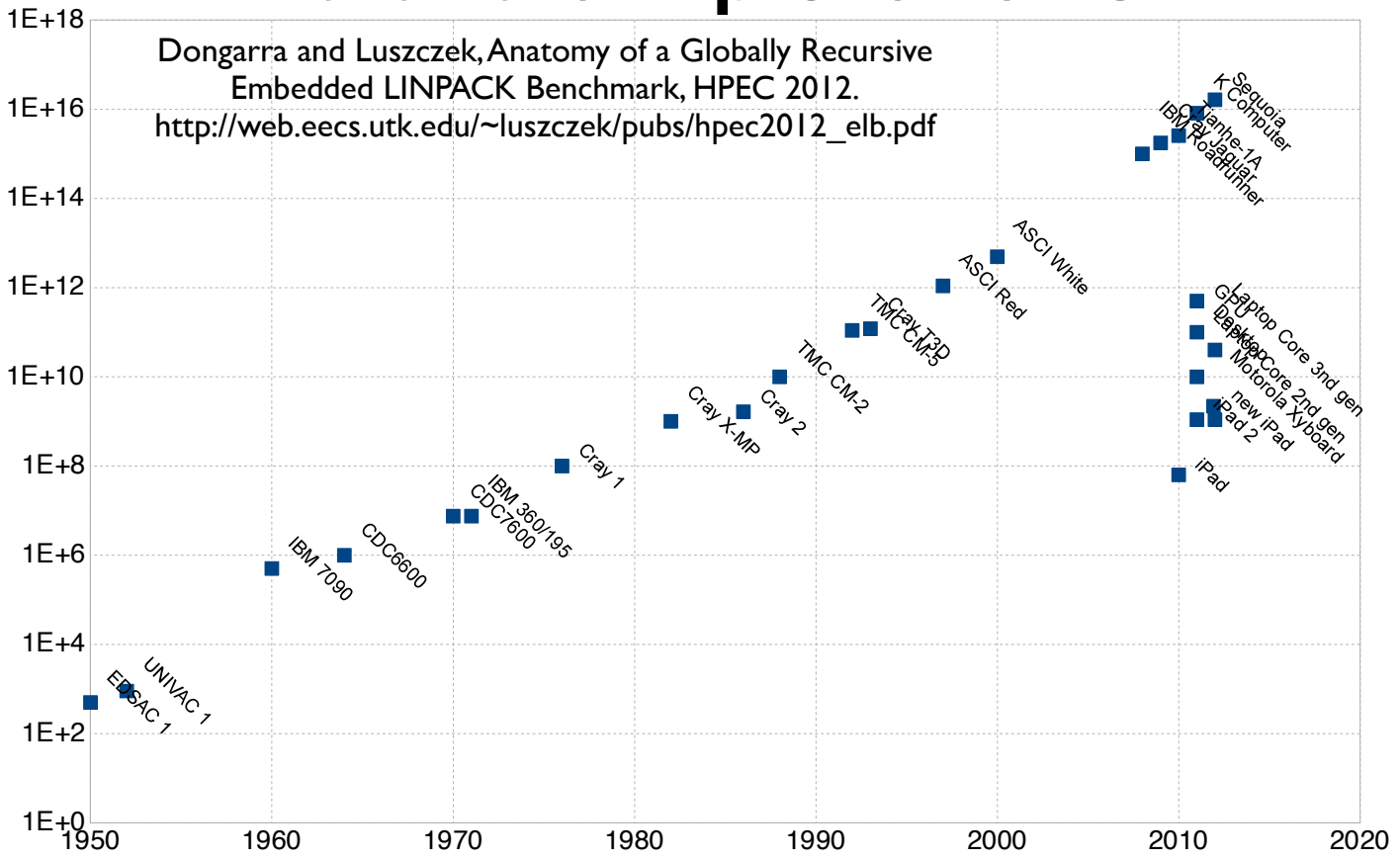
Why didn’t it take off?

- **Symbolic execution can be compute-intensive**
  - Lots of possible program paths
  - Need to query solver a lot to decide which paths are feasible, which assertions could be false
  - Program state has many bits
- **Computers were slow** (not much processing power) **and small** (not much memory)
  - Recent Apple iPads are as fast as Cray-2’s from the 80’s

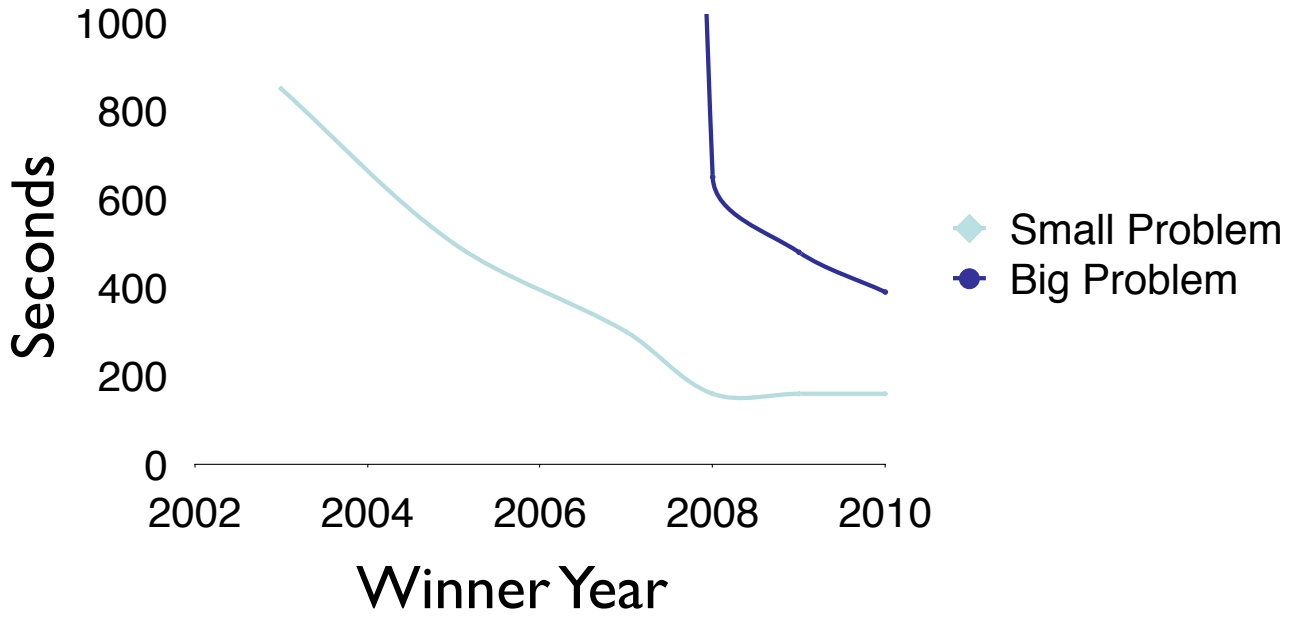
Today

- **Computers** are much **faster**, **bigger**
- **Better algorithms** too: powerful **SMT/SAT solvers**
  - SMT = *Satisfiability Modulo Theories* = SAT++
- Can solve very large instances, very quickly
  - Lets us check assertions, prune infeasible paths

Hardware improvements



SAT algorithm improvements



Results of SAT competition winners (2002-2010)
on SAT'09 problem set, on 2011 hardware

Rediscovery

- 2005-2006 reinterest in symbolic execution
- Area of success: (security) **bug finding**
  - Heuristic search through space of possible executions
  - Find really interesting bugs