The Unthinkable:
Automated Theorem Provers
For (Tracing) Just-in-Time Compilers

Nikolai Tillmann, Michal Moskal,
Wolfram Schulte, Herman Venter,
Manuel Fahndrich

Microsoft Research Redmond
Tracing JIT

```plaintext
for (i=0; i < a.len; i++)
    if (a[i] > 0) p++;
```
The Idea

Use Satisfiability Modulo Theories (SMT) solver to understand what the program does.
Trace tree to SMT push/pop

\[ t_0 = len_0[a] \]
\[ bv_<(i, t_0) \]
\[ t_1 = len_0[a] \]
\[ \neg bv_<(i, t_1) \]
\[ t_2 = a[i] \]
\[ bv_>(t_2, 0) \]
\[ \neg bv_>(t_2, 0) \]
\[ t_3 = bv_+(p, 1) \]
\[ t_3 = bv_+(p, 1) \]
\[ t_4 = bv_+(i, 1) \]
\[ \neg (t_0 = t_4) \]

---

\[ i = 6 \]
\[ p = 1 \]
\[ t_0 = 8 \]
\[ t_1 = 8 \]
\[ t_2 = 13 \]
\[ t_3 = 2 \]
\[ t_4 = 7 \]
Optimizations

• forward guard elimination
• common subexpression elimination
  – modulo theories
  – with alias analysis and redundant load elimination
  – all equality analyses combined!
• redundant store elimination
Status: SPUR with Z3

• complete (for bitvectors)
  – strongest possible optimization of this kind
• guard removal implemented in SPUR with Z3
  – easy to implement
    • especially no need to worry about bitvector corner-cases
  – couple times slower than SPUR “Classic”
• use Z3 for runtime validation of SPUR?