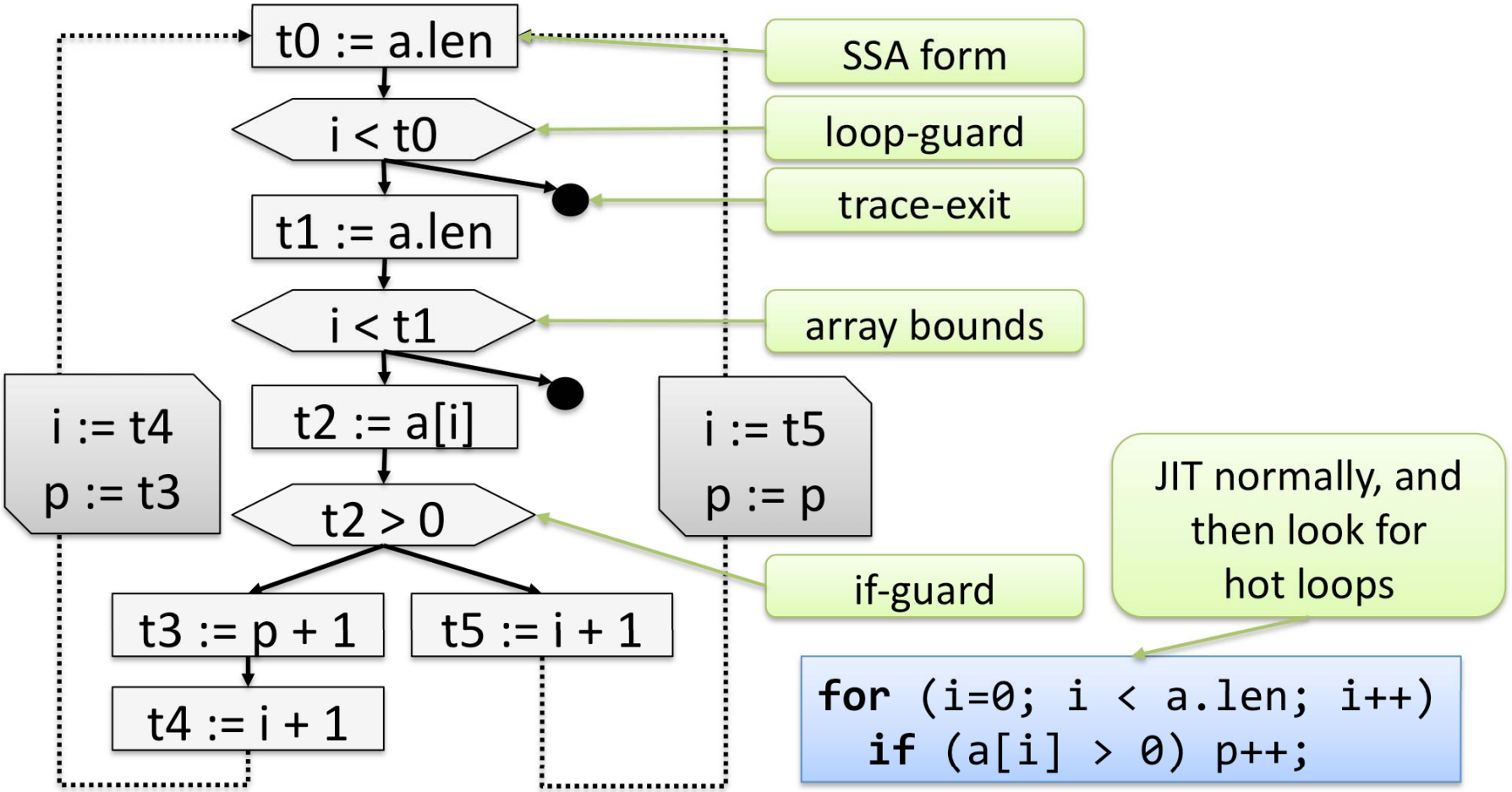
The Unthinkable:
Automated Theorem Provers For (Tracing) Just-in-Time Compilers

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

Microsoft Research Redmond

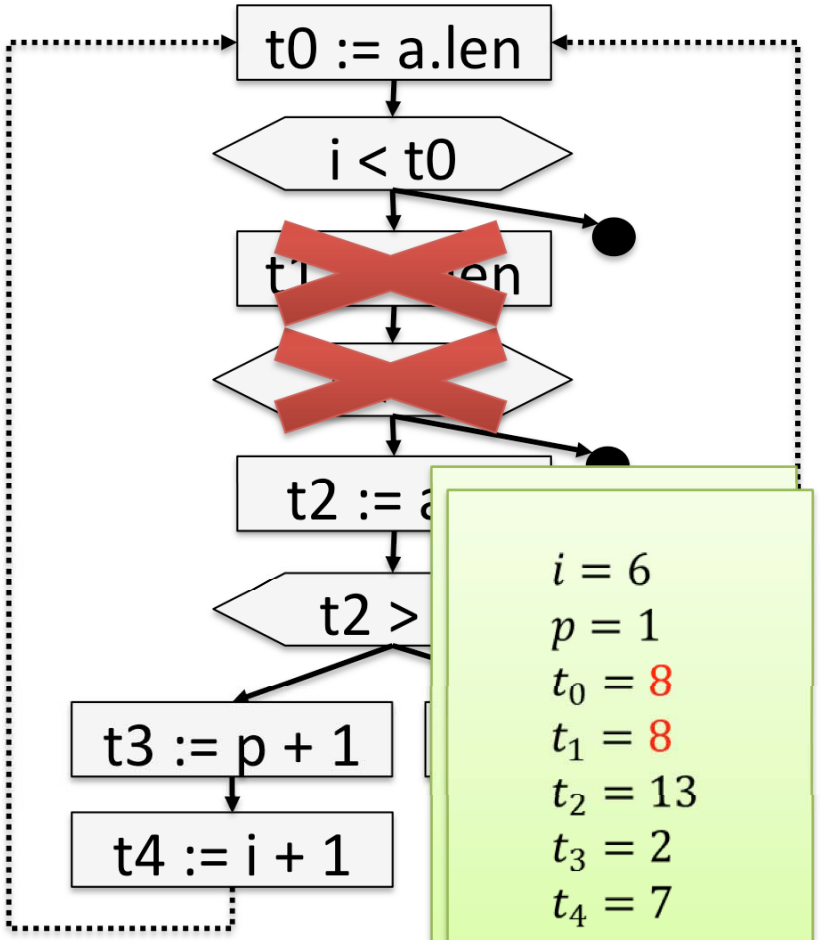
Tracing JIT


t0 := a.len
i < t0
t1 := a.len
i < t1
t2 := a[i]
t2 > 0
t3 := p + 1
t4 := i + 1
t5 := i + 1
i := t4
p := t3
i := t5
p := p
SSA form
loop-guard
trace-exit
array bounds
if-guard
JIT normally, and then look for hot loops
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



$$t_0 = len_0[a]$$

$$bv_<(i, t_0)$$

$$t_1 = len_0[a]$$

$$\neg bv_<(i, t_1)$$

$$t_2 = a[i]$$

$$bv_>(t_2, 0) \qquad \neg bv_>(t_2, 0)$$

$$t_3 = bv_+(p, 1) \qquad t_3 = bv_+(p, 1)$$

$$t_4 = bv_+(i, 1)$$

$$\neg(t_0 = t_4)$$

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?