Microsoft Research

Dafny

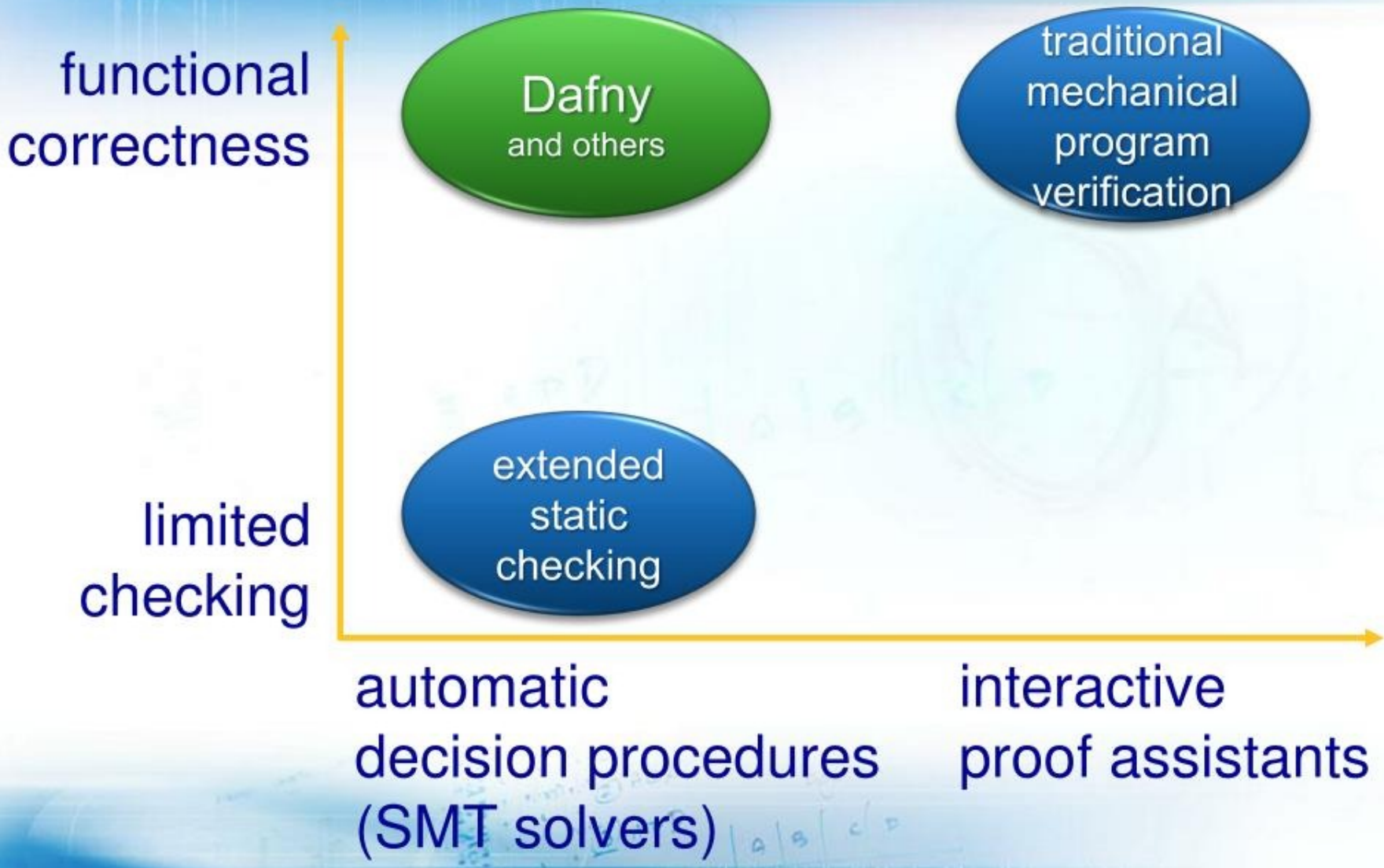
An automatic program verifier for functional correctness

K. Rustan M. Leino
Research in Software Engineering (RiSE)
Microsoft Research, Redmond

LPAR-16
Dakar, Senegal
27 April 2010

Microsoft

Program verification



User interaction

Program

Specifica

Program oriented:
invariants,
assertions, …

Formula

Formula oriented:
theorem-prover
commands, tactics

Dafny

demo

Binary search

Dafny

- Object-based language
  - generic classes, no subclassing
  - object references, dynamic allocation
  - sequential control
- Built-in specifications
  - pre- and postconditions
  - framing
  - loop invariants, inline assertions
  - termination
- Specification support
  - Sets, sequences, algebraic datatypes
  - User-defined functions
  - Ghost variables

Top-level grammar

- Program ::= Type*
- Type ::= Class | Datatype
- Class ::= **class** Name { Member* }
- Member ::= Field | Method | Function
- Datatype ::= **datatype** Name { Constructor* }

- Generic (that is, accepts type parameters)

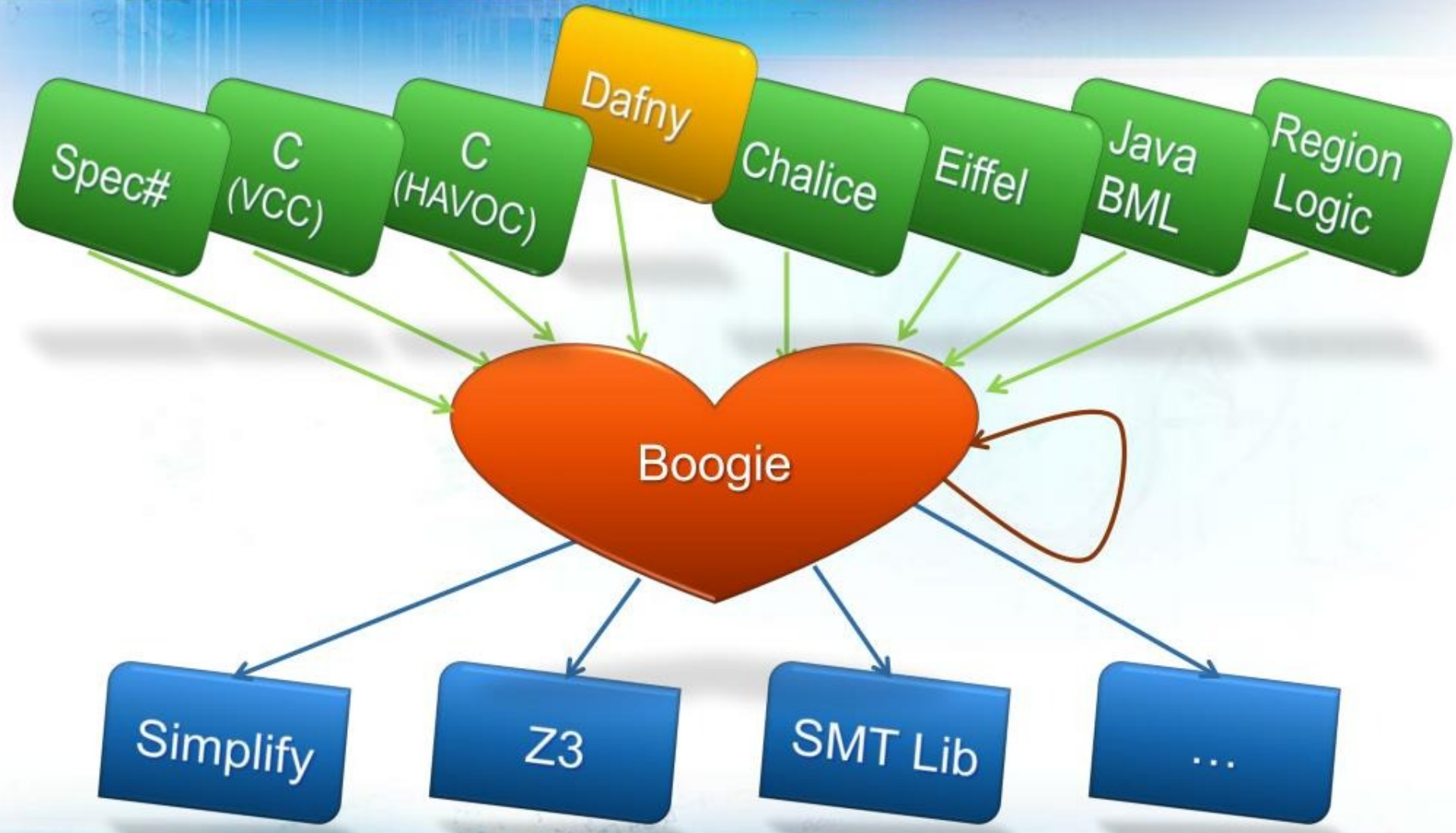
Types

- Booleans
- Mathematical integers
- Finite sets
- Sequences
- Class types
- Algebraic datatypes

Dafny

Calculator

Verification architecture



Dafny, Boogie, VC

From Dafny to verification-condition formulas

Axiomatizing functions

- function F(x: T): U … { Body }
  - (∀x • F(x) = Body)
- datatype Tree { Leaf(int); Split(Tree,Tree); }

  function G(x: Tree): U …

  { match x

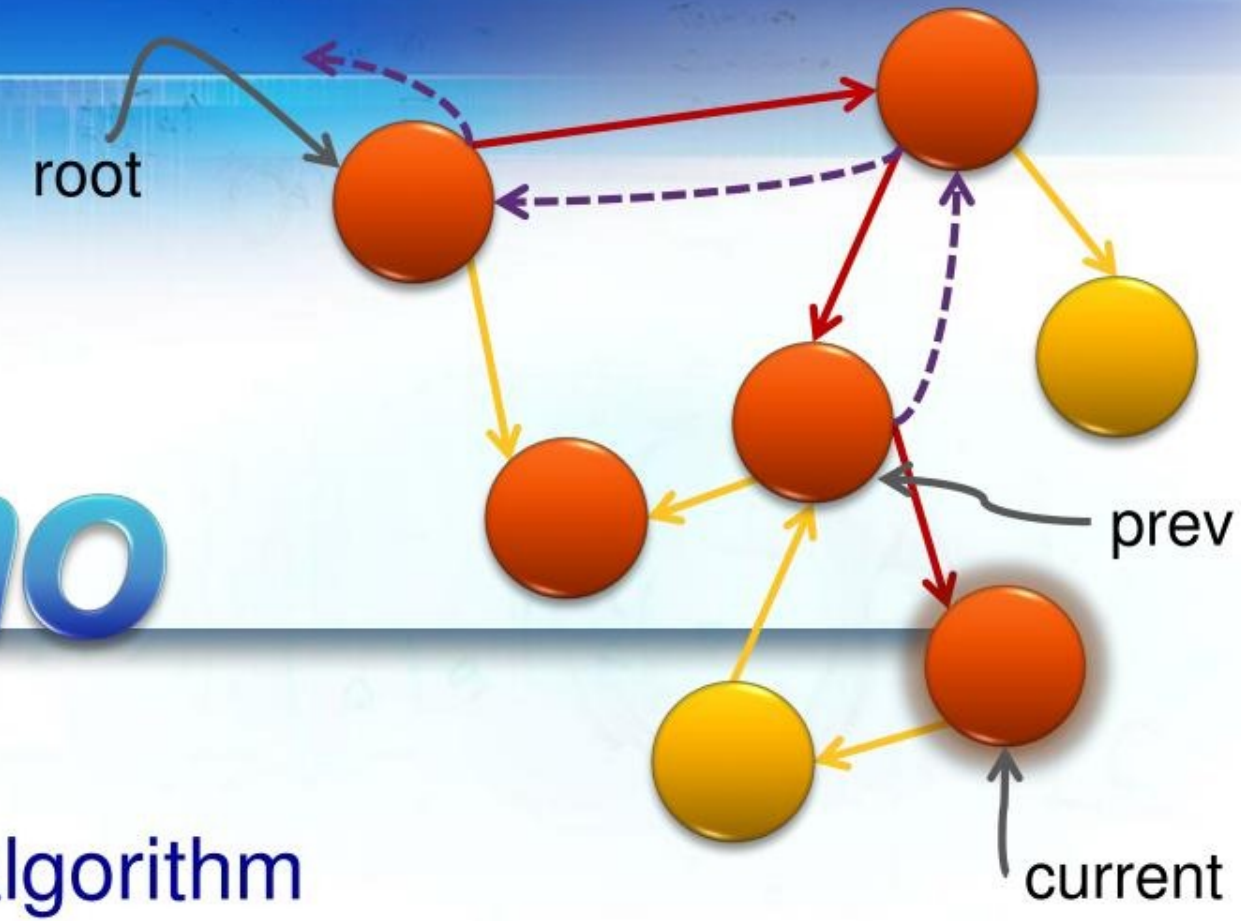
  case Leaf(n) ⇒ n

  case Split(a,b) ⇒ G(a) + G(b) }
  - (∀t • G(t) = if … else G(left(t)) + G(right(t)))
  - (∀n • G(Leaf(n)) = n)
  - (∀a,b • G(Split(a,b)) = G(a) + G(b))

Dafny

demo

Schorr-Waite algorithm



Verifying termination

- Functions
- Loops
- Methods

- **decreases** clause
  - lexicographic tuple
  - components of tuple can be of any types
  - to compare, consider longest commonly typed prefix of the lexicographic tuple

Dafny

Using a program to prover a theorem

Conclusions

- Full functional-correctness verification is becoming more automatic
- Interaction is moving closer to the problem domain
- A well-designed language and verifier, plus a great SMT solver, go a long way

Dafny (and Boogie) open source: boogie.codeplex.com