Modeling program semantics in an automated program verifier

K. Rustan M. Leino

Principal Researcher
Research in Software Engineering (RiSE), Microsoft Research, Redmond

Visiting Professor
Department of Computing, Imperial College London

PNW PLSE 2016, Seattle, WA, 15 March 2016

Dafny

Programming language designed for *reasoning*

Language features drawn from:

- Imperative programming
  - *if*, *while*, :=, *class*, …
- Functional programming
  - *function*, *datatype*, *codatatype*, …
- Proof authoring
  - *lemma*. *calc*, *refines*, *inductive predicate*, …

Program verifier

Integrated development environment (IDE)

Uses of Dafny

In projects

- ExpressOS [ASPLOS 2013]
- CloudMake algorithms [FM 2014]
- Ironclad Apps [OSDI 2014]
- IronFleet [SOSP 2015]

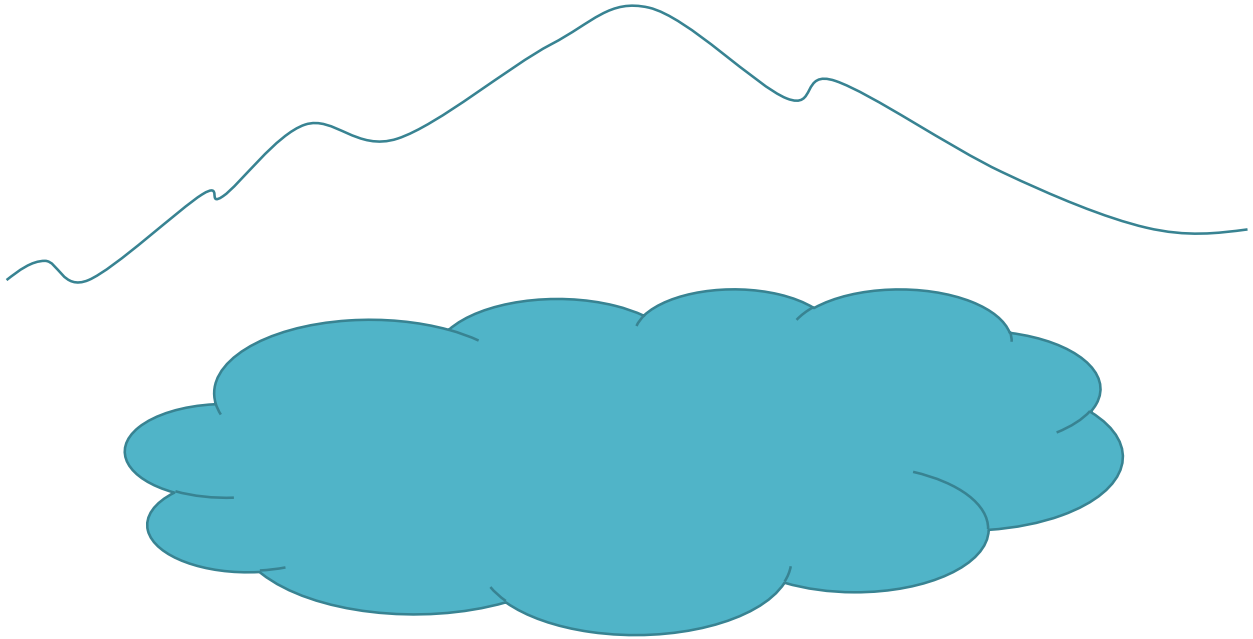
Program verification competitions

- VerifyThis
- VS Comp

In teaching

- At over 30 universities (including UW ☺)

Everest expedition:
Everest Verified End-to-end Secure Transport



Verified,
Performant,
Deployed
HTTPS

What’s in it for PL peeps?

“Can I prove the soundness of my language semantics?”

Semantics of INC

$\text{Cmd} ::= \text{Inc} \mid \text{Cmd}; \text{Cmd} \mid \text{Repeat}(\text{Cmd})$

$$\frac{t = s+1}{(\text{Inc}, s) \to t}$$

$$\frac{(c0, s) \to s' \quad (c1, s') \to t}{(c0; c1,\ s) \to t}$$

$$\frac{t = s}{(\text{Repeat}(body), s) \to t}$$

$$\frac{(body, s) \to s' \quad (\text{Repeat}(body), s') \to t}{(\text{Repeat}(body), s) \to t}$$

Semantics of INC

$$\text{Cmd} ::= \text{Inc} \mid \text{Cmd}\,;\text{Cmd} \mid \text{Repeat}(\text{Cmd})$$

$$\frac{t = s{+}1}{(\text{Inc}, s) \to t}$$

$$\frac{\exists\, s'.\ (c0, s) \to s' \quad (c1, s') \to t}{(c0\,;c1,\ s) \to t}$$

$$\frac{t = s}{(\text{Repeat}(body), s) \to t} \qquad \frac{\exists\, s'.\ (body, s) \to s' \quad (\text{Repeat}(body), s') \to t}{(\text{Repeat}(body), s) \to t}$$

Demo

INC

```
datatype cmd = Inc | Seq(cmd, cmd) | Repeat(cmd)
type state = int

inductive predicate BigStep(c: cmd, s: state, t: state)
{
  match c
  case Inc =>
    t == s + 1
  case Seq(c0, c1) =>
    exists s' :: BigStep(c0, s, s') && BigStep(c1, s', t)
  case Repeat(body) =>
    s == t ||
    exists s' :: BigStep(body, s, s') && BigStep(c, s', t)
}

inductive lemma Monotonic(c: cmd, s: state, t: state)
  requires BigStep(c, s, t)
  ensures s <= t
{
  match c
  case Inc =>
    // trivial
  case Seq(c0, c1) =>
    var s' :| BigStep(c0, s, s') && BigStep(c1, s', t);
    Monotonic(c0, s, s');
    Monotonic(c1, s', t);
  case Repeat(body) =>
    if s == t {
      // trivial
    } else {
      var s' :| BigStep(body, s, s') && BigStep(c, s', t);
      Monotonic(body, s, s');
      Monotonic(c, s', t);
    }
}
```

Conclusions

Dafny is a programming language and program verifier

Dafny now supports inductive/co-inductive predicates

Encoded in a first-order induction-unaware fixpoint-unaware IVL/SMT*

Do your language semantics in Dafny!

Also, see Nada Amin's talk "How to write your next POPL paper using Dafny"

Try Dafny online: `rise4fun.com/dafny`

Sources: `dafny.codeplex.com`

`research.microsoft.com/verificationcorner`

*) ask me how