Dafny 2025 State of the Union Address

K. Rustan M. Leino

Amazon Web Services

Keynote, Dafny Workshop 2025
Denver, CO, USA
19 January 2025

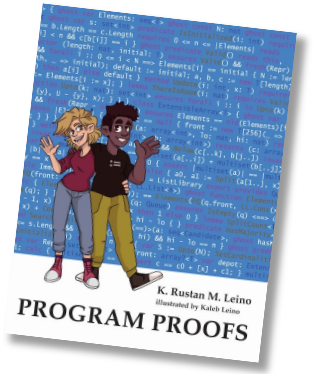
New applications

- AWS Authentication Engine
  Most impactful formal verification project to date?

  Talk at AWS re:Inforce 2024:
  https://youtu.be/oshxAJGrwMU?si=2HRf2XvNR-8RMIXZ

- AWS Encryption SDK
  Authored in Dafny, now released for 5 language platforms
  https://aws.amazon.com/about-aws/whats-new/2025/01/aws-encryption-sdk-go-available/
- AI applications (see today's schedule)
- What are some of yours?

Teaching



- Program Proofs
  K. Rustan M. Leino (MIT Press, 2023)
- Specification and Verification of Distributed Protocols
  Manos Kapritsos and Jon Howell
  https://glados-michigan.github.io/verification-class/2022/
- Training material used at AWS
  John Tristan
  https://dafny.org/blog/2023/12/15/teaching-program-verification-in-dafny-at-amazon/
  https://dafny.org/teaching-material/
- 6.S057 Verified Software Engineering (MIT)
  Adam Chlipala and K. Rustan M. Leino, Spring 2025

- What is some of your class material?

Community

- New: Zulip channel dafny.zulipchat.com/
- Coming soon: Redesigned web site dafny.org
- We want more actively to encourage open-source contributions
- **WANTED** a Dafny online playground and a format for interactive Jupyter-like tutorials

Tool set

- Doc comments
- JUnit/XUnit-like test automation
- Test generation
- **WANTED** QuickCheck for Dafny
- Auditor
- Interface Description Language: Smithy-Dafny
- Boogie extractor
- Standard library
  - Soon: Streaming library
  - Some parts need redesign **WANTED**

Invoking Dafny

- Project files
  No more `include` directives!

`dfyconfig.toml`

```
includes = ["*.dfy"]

[options]
warn-shadowing = true
```

- Verb-based Command-Line Interface
  - `dafny verify MyFile.dfy`
  - `dafny run dfyconfig.toml`

Types

- Basic types
  - `bool`
  - `char`
  - `int`
  - `real`
  - `bv0`, `bv1`, `bv2`, …
  - `ORDINAL`
- Immutable structures
  - `datatype`
  - `codatatype`
  - tuples (built-in special cases of datatypes)
- Built-in (immutable) collections
  - `set`, `iset`
  - `seq`
  - `multiset`
  - `map`, `imap`
- (Monomorphic) arrow types
  - `~>`
- References to mutable storage
  - `class`
  - `array`, `array2`, `array3`, …

Updated type system

- Reimplementation of type inference/checking
- To use it now:
  - `--type-system-refresh`
  - `--general-traits=datatype`
  - `--general-newtypes`
- Will become default in a future release

Subset types

(cf. refinement types, predicate subtypes)

- `type MyType = x: BaseType | Constraint(x)`
- Built-ins
  - `nat`
  - `-->`
  - `->`
- Constraint cannot depend on mutable state
- Cannot declare members
- Type system allows assignments between `MyType` and `BaseType` *in both directions*
  - Verifier enforces constraints
- New: types of bound variables always inferred as base types
  - `forall x :: P(x)`

Traits

(cf. interfaces, type classes)

```
trait MyTrait {
  // members
}
```

- Can have data members
- New: Can be implemented by any type that supports members (i.e., not subset types)
- Extending the built-in trait `object` makes a type a *reference type*
- A `class` automatically extends `object`

Newtypes

(cf. derived types)

- `newtype N = BaseType`
- New: Allows type parameters
- New: `BaseType` can be any non-trait, non-reference type
  - except datatypes **WANTED**
- A newtype is not allowed to extend traits
  - Needs support for fat pointers in the compilers **WANTED**
- Syntax allows a constraint, but this is better viewed as declaring both a subset type and a newtype

Type tests and conversions

- New: Expanded support for
  - type tests (`is`) and
  - type conversions (`as`)

Type parameters

- Variance
- Cardinality preservation
- Test-type (`is`) restriction to preserve parametric polymorphism
  - Coincidentally, this helps multi-target compilation
  - But is it otherwise of importance? **HELP**
- New: Type bounds (bounded polymorphism)
  - `method MyMethod<T extends MyTrait>(x: T)`
  - Bounds are traits or subset types thereof
  - Supports multiple bounds
  - Requires some casts when using as one of the bounds
    - Embellish type inference to not require these casts **WANTED**
  - Allow any type as a bound **WANTED**

Proof organization and stabilization

- `assert PropertyIWantToProve by { // proof goes here }`
- New: `by` clauses on all statements
- New: `opaque` code blocks
- New: `hide` statements (≈ dual of `reveal`)

- New: `{:only}`
  - Temporarily focuses proof effort on the selected component

Internal changes to improve prover performance and stability

- Clean up encoding of functions: CanCall
- Direct verifier in how to use automatic induction hypotheses
- Coming soon:
  - Improved encoding of fuel
  - Change in automatic unfolding for literal arguments to functions (probably)
- Still an issue: encoding of allocatedness

```
class C {
  var data: int
  …
}
```

```
method M(d: C) {
  var c := new C();
  assert c != d;
  c.data := 10;
}
```

- Bugs found and fixed in Z3
  - Dafny will update its Z3 version in Dafny 5.0 (or sooner)

Other new language features

- @-attributes
  - `@SomeAttribute(params)` alternative to `{:someAttribute params}`
  - More control and checking
- Parameter default values

```
method M(a: int,
         b: int := 25,
         c: int := d + 1,
         d: int := 100)
```

-

```
…
M(2, 4, d := 90)     // calls M(2, 4, 91, 90)
```

- `decreases to` expressions
- `forall` statements in expressions
- Sometimes, `{:axiom}` is required to confirm missing information

Rustan's feature wish list

- Regions, linearity, uniqueness
  - Abstraction, simpler specifications
  - Better verifier performance
  - Compiler optimizations
  - Concurrency
- Systems programming features (nominally unsafe memory)
- Named quantifiers
  - A new way to present quantifiers to users
- Letting subset constraint be declared as ghost or compiled
  - Compiled by default, `ghost` indicates ghost
- Stop requiring types to be auto-init by default
- `seq`/`array` constructors with pseudo-lambda expressions
- ^ and + for indexing and ranges
  - `s[^1]` means `s[|s| - 1]`
  - `s[n..+2]` means `s[n..n+2]`
- Redesign semantics of function values
- Post-declaration implementation of traits
- Change type characteristics to use traits **WANTED**
- Type `invariant` (only for immutable state) **WANTED**
- Better IDE support for writing proofs **WANTED**

SMT solving

- SMT research on quantifiers and matching patterns (triggers)
- Some of my assumptions were wrong
  - Triggers are not used in clause learning
- A minimal SMT solver with
  - arithmetic (integer, real, bitvectors)
  - uninterpreted function symbols
  - quantifiers
- Triggers with guards
  - Cf. Egg
  - Cf. work by Jhala, Vazou, et al.

Open design questions

- Termination for dynamic dispatch
- Termination for applications of function values
- Scope of well-formedness checks
- Future of multi-target compilation
- Native compiler for Dafny
- Verified compiler (see Dafny/CakeML talk later today)

Conclusions

This is an exciting time for software program proofs