The Role of Alloy in Developing Scientific Software

John Baugh and Tristan Dyer

Civil Engineering and Operations Research
North Carolina State University, Raleigh, NC
{jwb, atdyer}@ncsu.edu

Workshop on the Future of Alloy
Formative Experiences

Formal methods
- Larch Shared Language and Prover
- CCS/Concurrency Workbench, FSP/LTSA, SPIN, Alloy

Scientific software
- ADCIRC: large-scale ocean circulation, USACE, NOAA
- DM2: discrete meso-dynamic, multiphysics, LANL, NCSU
- POLO/FINITE: structural analysis, NASA, NIST
- SYSTRID: CAD/CAM, Airbus, PNNL, Dassault Systèmes

Instruction
- Computing in civil engineering and operations research
- Mathematical programming, e.g., linear and integer programming, building declarative models
What is Scientific Software?

Tools of the trade:

- Fortran
- Numerical analysis
- Matrix computations
- Parallel programming libraries

Tools convey expectations:

they’re what scientific software is about.
Why do we see a role for Alloy?

The essence of scientific software:

- **Structure**
  - Rich state in the form of spatial, geometric, material, topological, and other attributes

- **Behavior**
  - Explicit parallelism in a variety of forms
  - Continuous processes encoded as finite systems

In principle, such characteristics are a match for state-based formalisms like Alloy.

But what about the reals?
Scientific Software

We naturally think of continuous processes:

\textit{e.g., circulation of ocean currents}

But what does the computational apparatus underlying ocean circulation models really look like?

- purely analytic functions \( \times \)
- an amalgam of discrete data structures, algorithms, and \( \ldots \) numerical expressions \( \checkmark \)

Separating concerns:

\[
\begin{align*}
\text{scientific programs} & = \text{numerical expressions} + \text{interstitial machinery}
\end{align*}
\]
Tools Revisited

Fortran

- Instructive to look at the evolution of the language:
  
  *I don’t know what the language of the year 2000 will look like, but I know it will be called Fortran.* – Tony Hoare

Numerical analysis

- Once performed it often applies, unchanged, throughout a broad range of implementation choices and modifications over the life of the program.

Matrix computations

- Often left unassembled or sparse, rarely dense

Parallel programming libraries

- Language constructs and paradigms still being explored
Storm Surge Simulation

ADCIRC: a large scale ocean model used in production

Explore implementation choices and ensure soundness of an extension that improves performance

Storm Surge Simulation

ADCIRC: a large scale ocean model used in production

Explore implementation choices and ensure soundness of an extension that improves performance

- partitioning a finite element mesh: planar triangulations with variable topology and physical attributes
  - rich state and implicit definition of mesh structure

- interaction with a discrete wetting and drying algorithm encoded as empirical rules
  - represent as a series of transition relations

- safety, equivalence checking, predicate abstraction

Modeling Approach

ADCIRC (Fortran) → Ω → Full Domain Model (Alloy)

Extension to ADCIRC → Ω₁ → Subdomain Model (Alloy)

assert SameFinalStates within Ω₁

satisfied (up to bounds)

Alloy Analyzer

counter-example

adjust boundary conditions on Γ
Structural Analysis

Moment distribution: an iterative technique for finding internal member forces in building structures

Check soundness of an abstract implementation

Structural Analysis

Moment distribution: an iterative technique for finding internal member forces in building structures

Check soundness of an abstract implementation

- method is similar to asynchronous, chaotic relaxation algorithms, where portions of a building structure converge numerically at differing rates
- as with elliptic PDEs, the nondeterminism available here can be exploited in different ways depending on problem characteristics and hardware features
- refinement checking, predicate abstraction

Current Work

A design-centered approach to differential and integral equations found in practice, for which there are no closed form solutions

PDEs

discretization $\downarrow$ FEM, FD, FV

Finite System of Equations

invariants $\downarrow$ structural properties

Abstract Implementation

*Lightweight* in another sense: can draw useful conclusions about scientific software without simultaneously reproducing the sometimes deep, semantic proofs of numerical analysis.
Visualization

Backend visualization by reading XML instances from Alloy
  ▸ Atom editor, HTML, Javascript, CSS, D3
  ▸ consistent layout when stepping through states
Domain-specific viewers (in progress)
  ▸ triangulations: planar embedding, annotations

```plaintext
sig Mesh {
  triangles: some Triangle,
  adj: Triangle → Triangle
}
sig Vertex {}
sig Triangle {
  edges: Vertex → Vertex
}
```

![Diagram of mesh with vertices and edges labeled v0, v1, v2, v3, v4, t0, t1, t2.]
Atom + D3

Only the edge with the pending carryover will become non-pending; all others must remain in the same state:

\[
\text{all } x, y: \text{Vertex} \mid x \rightarrow y \text{ in edge} \Rightarrow \text{moment}[x, y].s' = (x = u \text{ and } y = v \Rightarrow \text{False else } n}
\]

Run showHardyCross for 3 but exactly 3 Vertex

An initial state in which there are no moments to carry over:

\[
\text{pred initial_state_no_moment } (s: \text{State}) \{ \text{stutter}[s, s'] \text{ or (some x: Vertex } \mid \text{release}[x, s, s'] \text{) or (some x, y: Vertex } \mid \text{carryover}[x, y, s']\}
\]

A valid state transition:

\[
\text{pred step_state } (s, s': \text{State}) \{ \text{stutter}[s, s'] \text{ or (some x: Vertex } \mid \text{release}[x, s, s'] \text{) or (some x, y: Vertex } \mid \text{carryover}[x, y, s']\}
\]

`pred showHardyCross`
Instruction

Declarative languages are expressive, but it is not always clear how they can be used

- no special constructs for parallelism, message-passing, synchronization or other mechanisms that give some insight into what one is “supposed” to do with it
- there are few affordances (contrast with FSP/LTSA)

Template approach [Schrage]

- used in an undergraduate systems engineering course
- linear, integer, and nonlinear programming models from the field of systems science and operations research

Reimagining freshman programming for engineers

- just a skills course?
Final Thoughts

Working in a domain where quality, reproducibility, and productivity are growing concerns

- retractions of papers in scientific journals
- not an obvious target for formal methods, but scientists and engineers know about and value modeling

Promoting adoption

- documentation: object models and state machines
- user interface: environment and visualization

thank you