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 continous processes:

e.g., circulation of ocean currents

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

- purely analytic functions ✗
- an amalgam of discrete data structures, algorithms, and . . . numerical expressions ✓

Separating concerns:

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

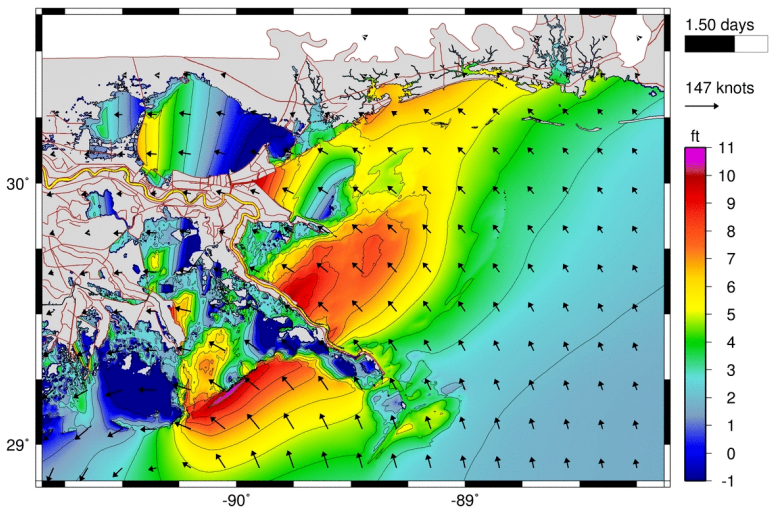
Parallel programming libraries

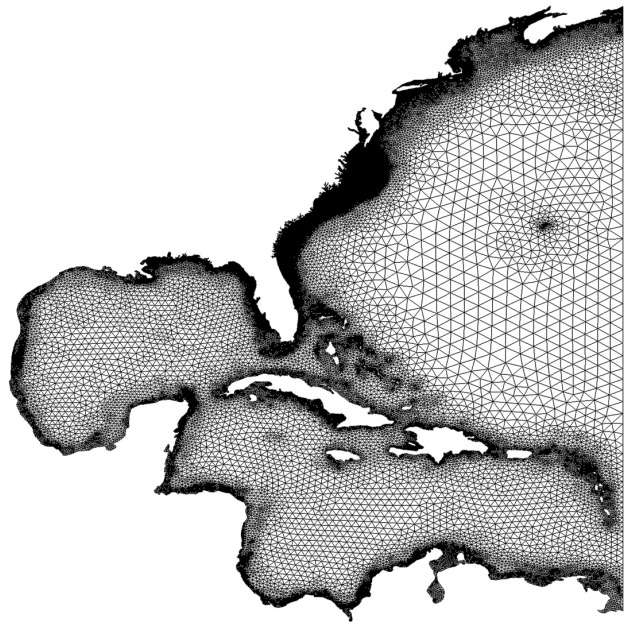
- Language constructs and paradigms still being explored

Storm Surge Simulation

ADCIRC: a large scale ocean model used in production

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





Formal methods and finite element analysis of hurricane storm surge: A case study in software verification. Baugh and Altuntas. *Science of Computer Programming*, 158:100–121, 2018.

Storm Surge Simulation

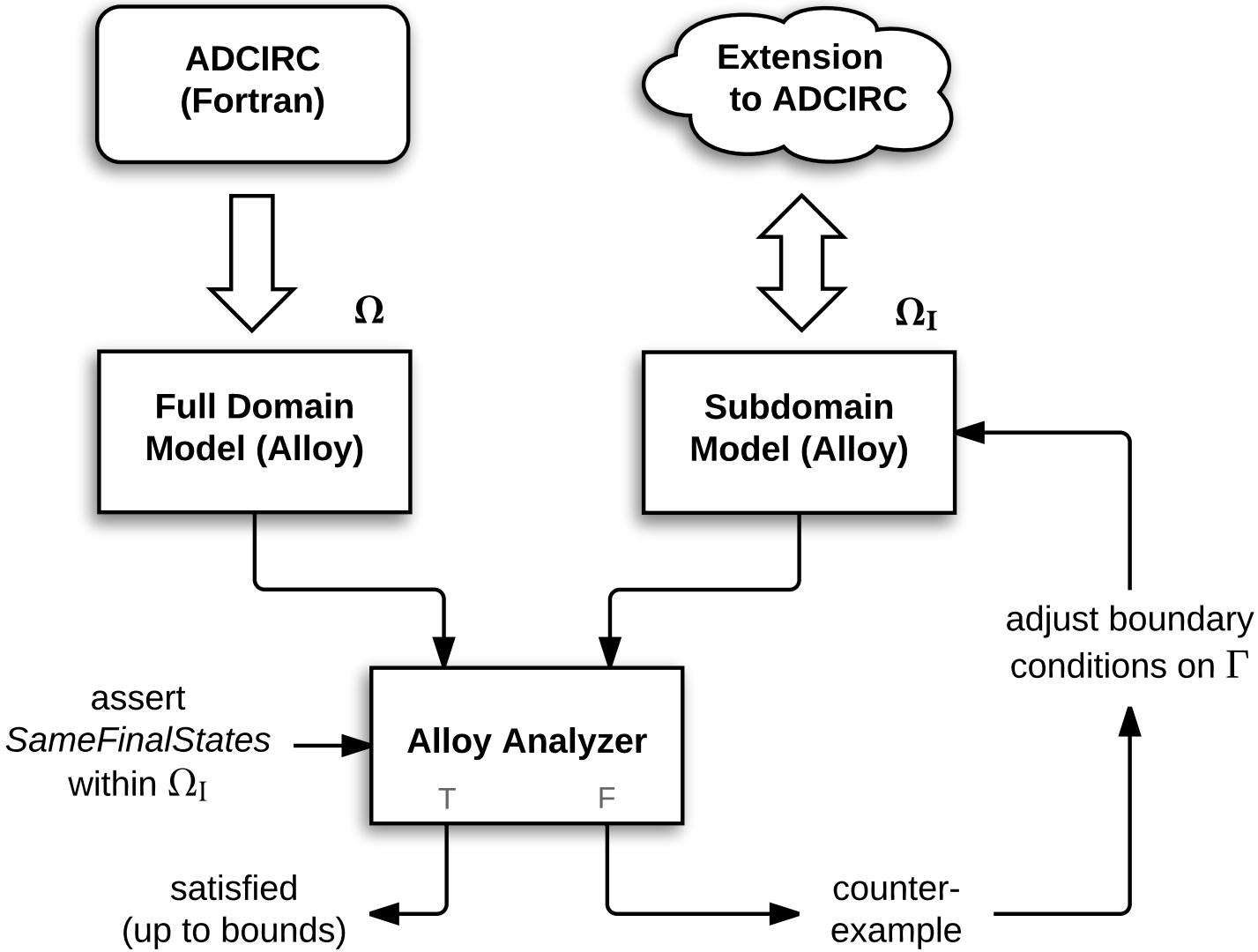
ADCIRC: a large scale ocean model used in production

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

Formal methods and finite element analysis of hurricane storm surge: A case study in software verification. Baugh and Altuntas. *Science of Computer Programming*, 158:100–121, 2018.

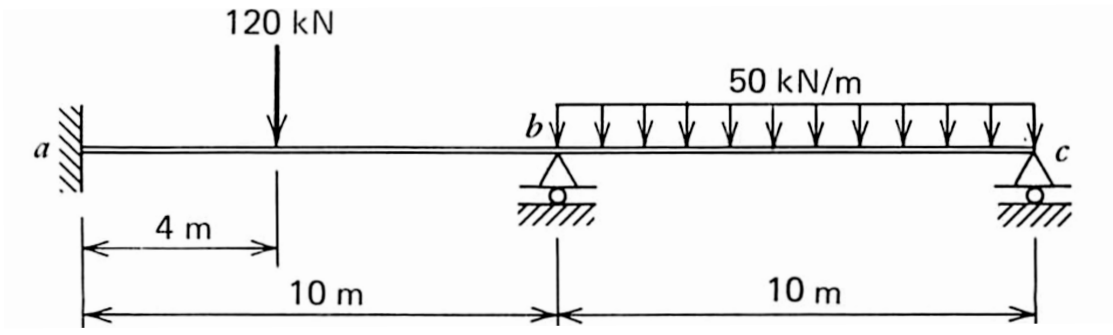
Modeling Approach

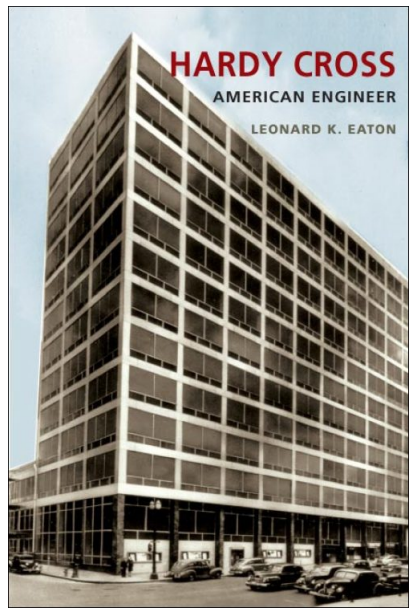


Structural Analysis

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

Check soundness of an abstract implementation





A general characterization of the Hardy Cross method as sequential and multiprocess algorithms. Baugh and Liu. *Structures*, 6:170–181, 2016.

Structural Analysis

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

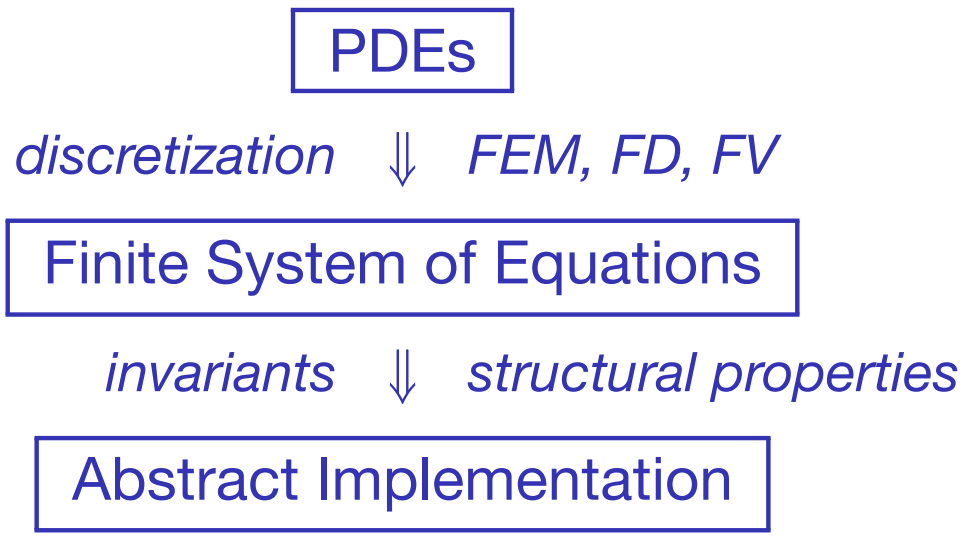
Check soundness of an abstract implementation

- 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

A general characterization of the Hardy Cross method as sequential and multiprocess algorithms. Baugh and Liu. *Structures*, 6:170–181, 2016.

Current Work

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



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

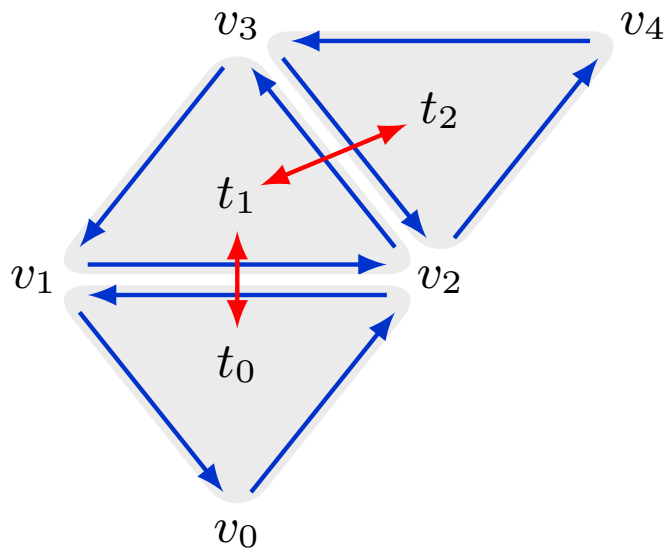
Domain-specific viewers (in progress)

- triangulations: planar embedding, annotations

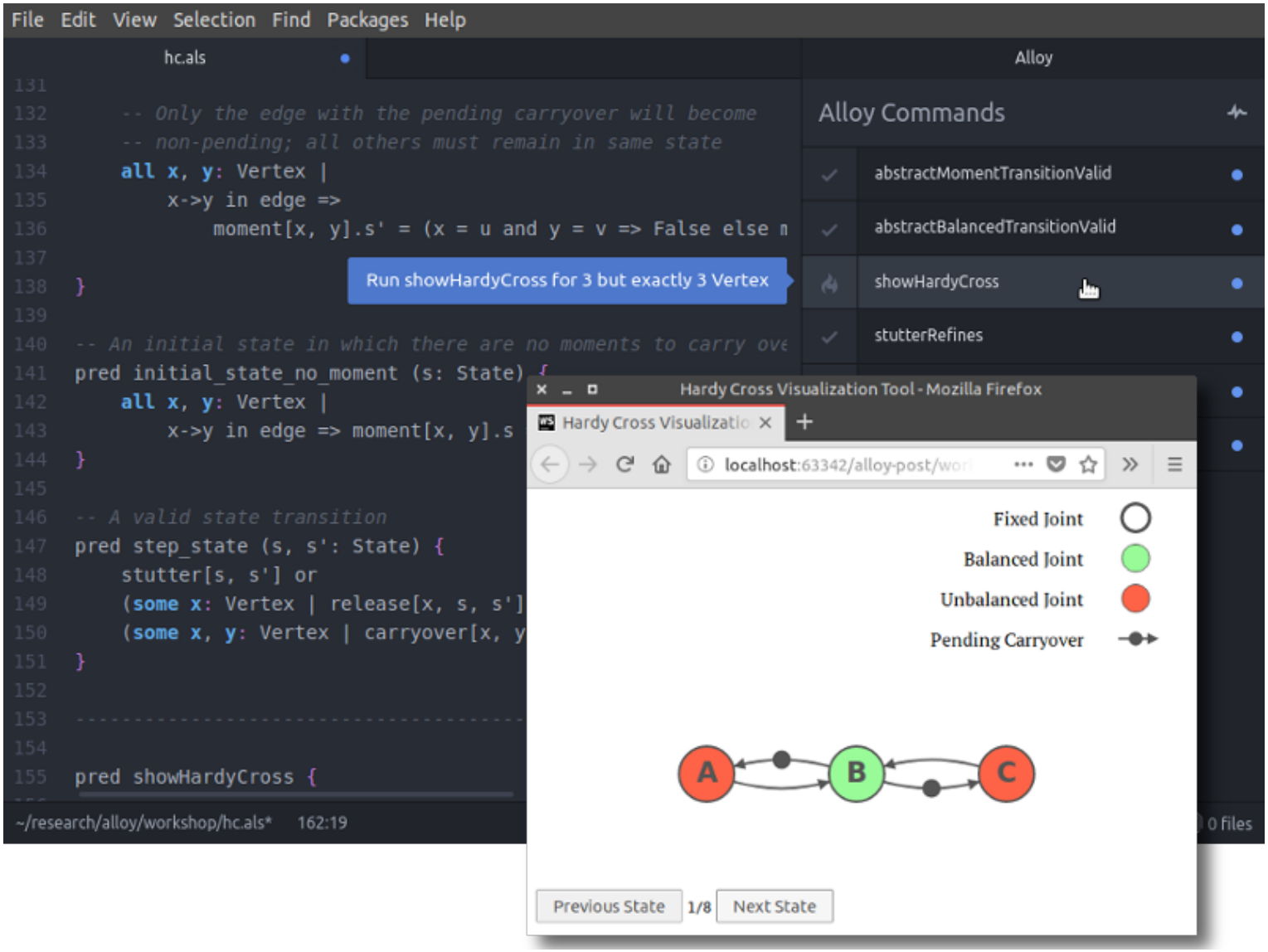
```
sig Mesh {
    triangles: some Triangle,
    adj: Triangle → Triangle
}

sig Vertex {}

sig Triangle {
    edges: Vertex → Vertex
}
```



Atom + D3


File Edit View Selection Find Packages Help
hc.als
Alloy
Alloy Commands
abstractMomentTransitionValid
abstractBalancedTransitionValid
showHardyCross
stutterRefines
Run showHardyCross for 3 but exactly 3 Vertex
-- Only the edge with the pending carryover will become
-- non-pending; all others must remain in same state
all x, y: Vertex |
x->y in edge =>
moment[x, y].s' = (x = u and y = v => False else m
}
-- An initial state in which there are no moments to carry ove
pred initial_state_no_moment (s: State) {
all x, y: Vertex |
x->y in edge => moment[x, y].s
}
-- A valid state transition
pred step_state (s, s': State) {
stutter[s, s'] or
(some x: Vertex | release[x, s, s']
(some x, y: Vertex | carryover[x, y
}
pred showHardyCross {
~/research/alloy/workshop/hc.als* 162:19
0 files
Hardy Cross Visualization Tool - Mozilla Firefox
Hardy Cross Visualizatio
localhost:63342/alloy-post/wor
Fixed Joint
Balanced Joint
Unbalanced Joint
Pending Carryover
A
B
C
Previous State
1/8
Next State

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

State-based formal methods in scientific computation. Baugh and Dyer. To appear in *Abstract State Machines, Alloy, B, TLA, VDM, and Z: 6th International Conference, ABZ 2018*.

thank you