Executable Specifications: Foundations → MS Tools

Yuri Gurevich
Microsoft Research

Agenda

- A short version of the talk
- A piece of theory
- Executable specifications
- Our languages
- Our tools

The Short Version

Genesis

- This project started about 20 years ago when this logician moved to computer science.
- What is CS about? While it is about many things, the central role is played by algorithms (in a broad sense of the term). Operating systems, programming languages, compilers, etc. are all algorithms.

Key Question

- What's an algorithm?
- The Turing machine is not adequate anymore. What is, if anything?

The Proposed Solution

A long analysis led to abstract state machines and the ASM thesis:
For every algorithm there is a behaviorally identical ASM.

Experimental Confirmation

- By the thesis, ASMs fit to model and specify algorithms.
- Many applications are found, in academia and industry.
- In the process, experimental confirmation of the thesis accumulates.

The group on Foundations of Software Engineering at MSR

- Wolfram Schulte
- Margus Veanes
- Colin Campbell
- Lev Nachmanson
- Mike Barnett
- Wolfgang Grieskamp
- Nikolai Tillman

Behavioral theory of computation

- Increasing parts of the ASM thesis are proven from first principles.
- In the process, axiomatic definitions of sequential, parallel, etc. algorithms emerge.

A piece of theory

Sequential Time Postulate

Any algorithm determines

- the set of states,
- the subset of initial states,
- the transition function.

Def. Two seq-time algorithms are *behaviorally equivalent* if they have the same states, initial states and the transition function.

What are states of an algorithm?

- What are states of, say, a C program?
- Transparent (or explicit, or honest) states

Abstract State Postulate

- The states are logic structures.
- ...

Seq algorithms

- Seq-time algorithms with bounded-work steps.
- How to measure work?

Definition

A sequential algorithm is any entity that satisfies the three postulates:

- sequential time,
- abstract state,
- bounded-exploration.

Euclid's algorithm

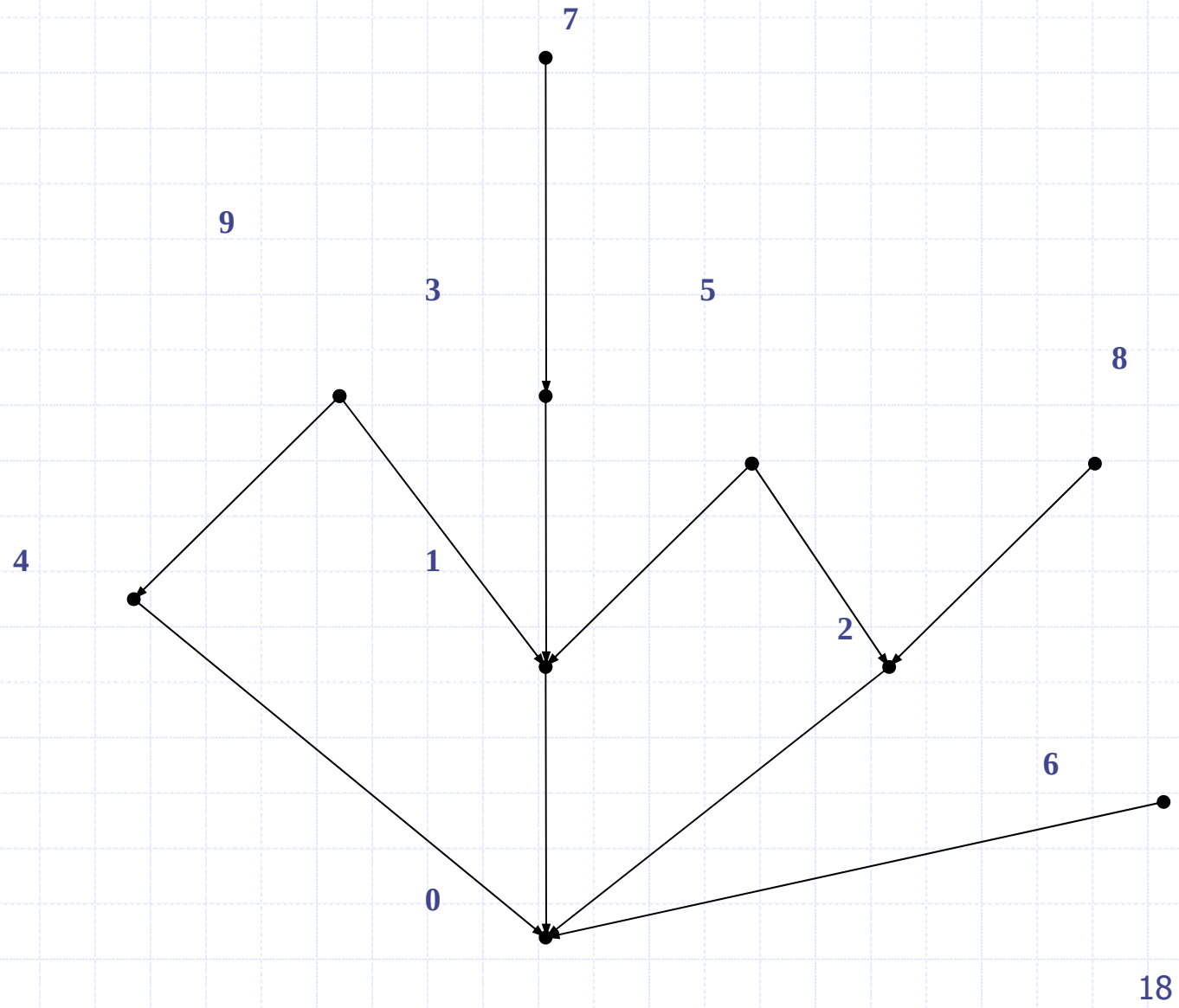
```
if b = 0 then d := a
else
   [do in-parallel]
        a := b
        b := a mod b
```

Seq Characterization Theorem

- *For every sequential algorithm A, there exists a behaviorally equivalent sequential ASM .*
- Ref. #141 at the speaker's website

Parallel algorithms

Example
Slicing
a dag



Slicing a Dag in AsmL

```
forall v in V
 if forall u in V holds
   (u,v) in E implies u in X
 then add v to X
```

Par Characterization Thm

- Analysis
- Theorem:
  For every parallel algorithm A, there is a behaviorally equivalent parallel ASM .
- Ref. #157

Intra-step interaction

- New object creation, choice, remote procedure calls, messages
- The characterization theorems

Ref: #166, 170, 171, and forthcoming
by Andreas Blass, YG, Dean Rosenzweig and Benjamin Rossman

Distributed algorithms

Distributed ASMs were defined long ago, and most ASM applications, at least at Microsoft, are distributed, but the axiomatization problem is still open.

Executable specifications

In-place one-swap-a-time sorting

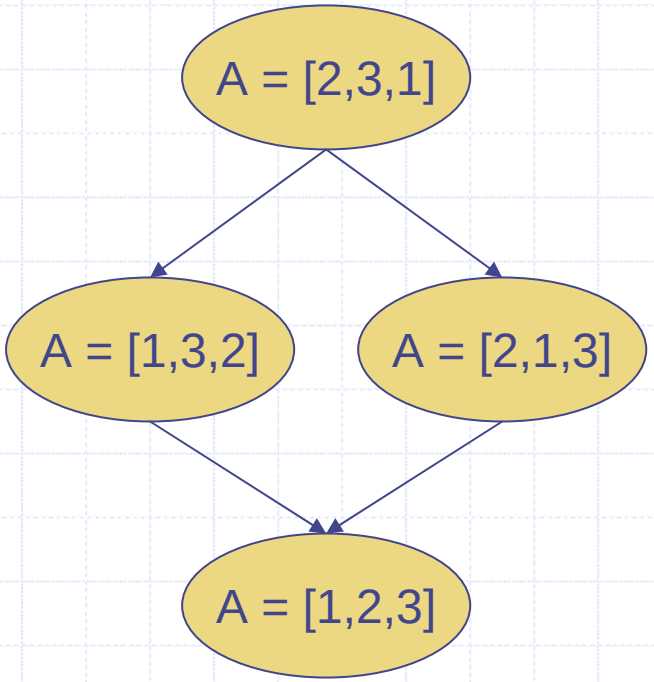
```
var A as Seq of Integer = [3,1,2]

Swap()
  choose i,j in Indices(A)
          where i<j and A(i)>A(j)
     A(i) := A(j)
     A(j) := A(i)

Sort()
  step until fixpoint
     Swap()
```

Nondeterminsm

Parallelism



Our Languages

AsmL and C#

http://research.microsoft.com/fse/asml

- Math e.g. set comprehension
  $\{x^2 \mid x \in \{1,..,10\}$ **where** $x = 0$ **mod** $2\}$
- Transactions, nondeterminism
- OO, interoperability via .NET
- Literate programming via Word, automated programming via XML

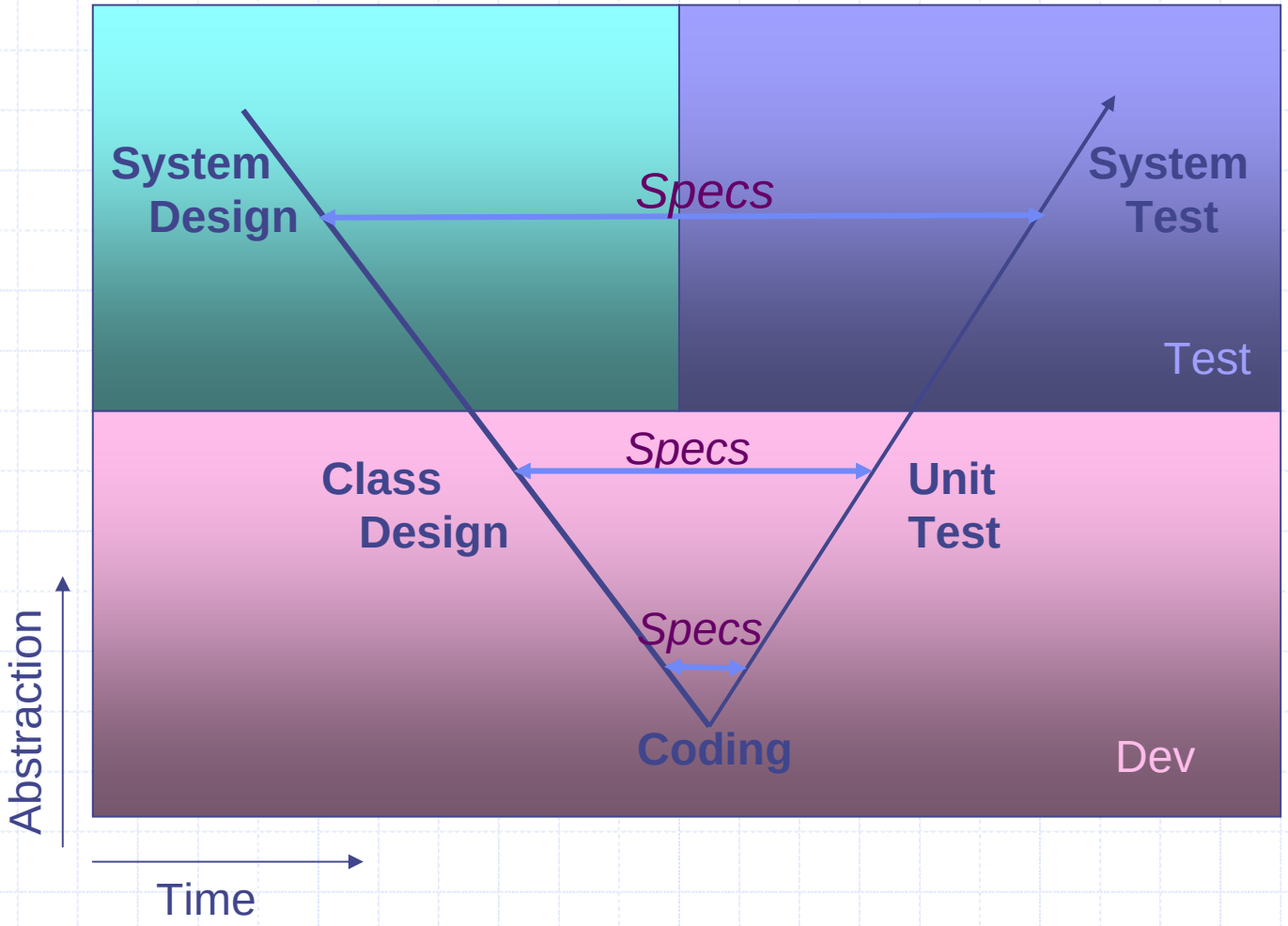
Topological Sorting

```
step while ToSet(S) ne V
 let X = V - ToSet(S)
  if X <> {} then
   S := S + [(any v | v in X
     where not(exists u in X
      where (u,v) in E))]
```

Our Tools

Software development process

Current Projects



Spec Explorer

Parameterized Unit Tests

Spec#

Validating the spec

- Human comprehension
- Playing scenarios
- Deriving an FSM and then testing (including model checking) it

Enforcing the spec

- In the deterministic case, generate a test suite with results, and run the suit on various implementations.
- Execute the model and implementation in lock step.
- Play to test