Computer Game *Civilization*

In the simulation, each country (leader) is described by quantitative attributes that govern its "AI" behavior.

Initially Gandhi's **aggression** := 1

When a country adopts democracy, its **aggression** decreases by 2.
**Computer Nostalgia (1980ies)**

- *empirical* correctness
- *hardware* data type *byte/word* \( \neq \mathbb{Z} \)
- Inconvenient/unelegant \( \rightarrow \) bugs
- *absolute* performance
- *low-level* prog.language

1 MHz CPU
64kB RAM

Coding, **not** Computer **Science**

---

**Contemporary Computer Science**

- *empirical* correctness
- *hardware* data type *byte/word* \( \neq \mathbb{Z} \)
- inconvenient/unelegant \( \rightarrow \) bugs
- *absolute* performance
- *low-level* prog.language

1. **Rigorous specification**
2. Algorithm design&analysis *(always correct & efficient)*
3. **Proof of optimality** *(polynom.-time *reduction*)
4. Design/build on axiomatic Abstract Data Types
5. in obj.-oriented high-level prog.language *(semantics)*
6. **Formal verification**
7. Implementation/coding

---

**LOGIC**
Abstract Data Type = Structure

- empirical correctness
- hardware data type byte/word
- convenient, elegant \(\rightarrow\) less bugs
- absolute performance
- low-level prog.language

Object-oriented high-level programming language:

- \(\mathbb{Z}\) (integer in Python3; bigint in Java; GMP)
- stack of \(X\)
- queue of \(X\)
- file of \(X\)
- array\([X]\) of \(Y\)
- (labeled) graph (of \(X,Y\))

- modular software design
- “Information hiding”
- ideal data types

Contemporary Numerics

- empirical correctness
- hardware data type float/double ≠ \(\mathbb{R}\)
- inconvenient/unelegant \(\rightarrow\) bugs
- absolute performance

“nag_opt_one_var_deriv” normally computes a sequence of \(x\) values which tend in the limit to a minimum of \(F(x)\)

7. Implementation/coding
Medieval Mathematics

- empirical correctness
- hardware data type float/double ≠ \mathbb{R}
- inconvenient/unelegant & inconsistent
- absolute performance

\[ Q, \text{ as opposed to } \mathbb{R}, \text{ violates:} \]
- intermed. value theorem
- compactness of [0,1]
- fixed point theorems,
- logical completeness,
- and even distributive law

Don't test for equality!
How about inequality "<"?

\[ x=0 \iff \neg(x<0) \land \neg(x>0) \]

IEEE754:
\[ ±\infty, ±0, \text{NaNs, denormals} \]

Example Prog. Codes

\[ x_n \to x_{n+1} := r \cdot x_n \cdot (1-x_n) \text{ error } \leq 10\% \]

float: \( n \leq 30 \), double: \( n \leq 85 \),
long double: \( n \leq 200 \)
rational: \( n \leq 24 \)

MATLAB function
\[
y = \text{jmmuller}(m) \\
x = \text{vpa}(11/2); \\
y = \text{vpa}(61/11); \\
\text{while } m > 1; \\
\quad t = \text{vpa}((111 - (1130-3000/x)/y)); \\
x = y; y = t; m = m-1; \\
\text{end; end} \\
\rightarrow 100 \\
x_{m+1} := 111 - (1130-3000/x_{m-1})/x_m \to 6 \]

\[ x_0 := 11/2, \quad x_1 := 61/11 \]
Models of Real Computation

Imperative,
+,-,×,÷,<  exact
Computing by streams of approximations

Intermission
Floating point numbers (IEEE 754 from 1985)
are a hardware-based, legacy data type — like byte:
to be hidden behind object-oriented ideal data types!

Project: Develop continuous abstract data types.
• Fix some continuous structure (example: real numbers).
• Investigate computability of its operations/predicates.
  (Turing machines, Computable Analysis)
• Modify their semantics to make them computable and s.t.
  computing over this structure becomes Turing-complete.

Logic of Computing with Continuous Data:
Foundations of Numerical Software Engineering
Computing Real Numbers

Fact: For \( r \in \mathbb{R} \) call \( r \in \mathbb{R} \) computable if the following are equivalent:

a) \( r \) has a recursive binary expansion

b) There exists a Turing machine printing a sequence \((a_n)_{n \in \mathbb{Z}}\) with \(|r-\alpha/2^n| \leq 2^{-n}\).

c) There exists a Turing machine printing sequences \((a_n),(b_n)_{n \in \mathbb{Z}}\) with \(|r - a_n/b_n| \leq 1/n\).

Ernst Specker (1949): (c)\textsuperscript{Halting problem} \iff (d)

d) There exists a machine printing \((q_n)_{n \in \mathbb{N}}\) with \( q_n \rightarrow r \).

\[ \mathbb{R} \text{ as Computable Abstract Data Type} \]

- Fix some continuous structure (example: real numbers).
- Investigate computability of its operations/predicates.
- Modify their semantics to make them computable and s.t. computing over this structure becomes Turing-complete.

Reminder: \(+, -, \times, \div\) are computable

Fact: There is a computable sequence \((r_j)_{j \in \mathbb{N}}\subset [0,1]\) such that \( \{ j : r_j \neq 0 \} = H \), the Halting problem.

Non-extensional "Parallel-OR" [Escardó/Hofmann/Streicher'04]

- \(+, -, \times, \div, 0, 1\): exact operations provided it exists
- Partial semantics of tests: "\( x \lor y \)" = \( \uparrow \) if \( x = y \).
- choose \((x_1 \lor y_1, \ldots, x_d \lor y_d)\) = any \( j \) s.t. \( x_j > y_j \).
Example 0: Fuzzy/soft test [Chee K. Yap, ...]
choose( y+2^{-n} > x , x+2^{-n} > y )

REAL Trisection(INTEGER p; REAL p->REAL f);
REAL x:=0; REAL y:=1;
WHILE ( choose(2^p>y−x , y−x>2^p−1) == 2)
  IF (choose(0>f((2x+y)/3),f((x+2y)/3)>0)==1)
    THEN x:=(2x+y)/3 ELSE y:=(x+2y)/3 ENDIF

(Z,0,1,+,>) and (R,0,1,+,* , >) with \( \nu : \mathbb{Z} \ni p \rightarrow 2^p \in \mathbb{R} \)

Arguments exact, value approximate to 2^p.
• Partial semantics of tests: "x>y" = ↑ if \( x=y \).
• choose(\( x_1 > y_1 , ... x_d > y_d \)) = any \( j \) s.t. \( x_j > y_j \)

Logic of Computing over \( \mathbb{R} \)

Theorem: a) Computing over the below two-sorted structure is Turing-complete for real functions!
Reconcile Blum-Shub-Smale model & Computable Analysis
b) Said two-sorted structure has decidable First-Order Theory!
c) Generalizing (a) to real functionals.

Also "model complete"

Wrong for \( \nu(n)=1/n \)

(Z,0,1,+,>) and (\( \mathbb{R},0,1,+,* , > \)) with \( \nu : \mathbb{Z} \ni p \rightarrow 2^p \in \mathbb{R} \)

Arguments exact, value approximate to 2^p.
• Partial semantics of tests: "x>y" = ↑ if \( x=y \).
• choose(\( x_1 > y_1 , ... x_d > y_d \)) = any \( j \) s.t. \( x_j > y_j \)
Computing over Continuous Structures

**Theorem:** a) Computing over the below two-sorted structure is Turing-**complete** for real functions!
Reconcile Blum-Shub-Smale model & Computable Analysis

b) Said two-sorted structure has **decidable** First-Order Theory!

\[ \text{[van den Dries’86]} \quad \rightarrow \quad \text{formal verification.} \]

c) Generalizing (a) to real functionals

\[ \text{Also “model complete” Wrong for } \upsilon(n)=1/n \]

Continuous Abstract Data Types *in progress*:

- Compact Metric Groups
- Grassmannian
- Analytic Functions
- Compact Manifolds

Intermission

Complexity of Computing with Continuous Data

**Project:** Develop continuous abstract data types.

- Fix some continuous structure (example: real numbers).
- Investigate computability of its operations/predicates.
  (Turing machines, *Computable Analysis*)
- Modify their semantics to *make them computable* and s.t. computing *over* this structure becomes Turing-**complete**.

Logic of Computing with Continuous Data:

Foundations of Numerical Software Engineering
Complexity Theory for Numerics

Fix polytime \( f: [0;1] \rightarrow [0;1] \) \((\Rightarrow \text{continuous})\)

- Max: \( f \rightarrow \text{Max}(f): x \rightarrow \max \{ f(t): t \leq x \} \)
  - Max\((f)\) computable in \( \text{PSPACE} \)
  - Polyn.time-computable iif \( \mathbb{P} = \mathbb{NP} \)

- \( \int: f \rightarrow \int f: (x \rightarrow \int_0^x f(t) \ dt) \) **even for \( f \in \mathbb{C}^\infty \)**
  - \( \int f \) computable in \( \text{PSPACE} \)
  - Polyn.time-computable iif \( \mathbb{FP} = \#\mathbb{P} \)

- odesolve: \( C^1([0;1] \times [-1;1]) \ni f \rightarrow z: \dot{z}(t) = f(t,z), \ z(0)=0. \)
  - \( \text{PSPACE} \)-"complete"

- Solution to Poisson's Equation is classical and \( \#\mathbb{P} \)-"complete"
  - \( \Delta u = f \) on \( B_2(0,1) \)
  - \( u = 0 \) on \( \partial B_2(0,1) \)
  - [Kawamura+Steinberg+Z., MSCS 2017]

Advanced (Complexity) Issues

- **Weihrauch Reduction** among **continuous** problems
  - 2nd-order polynomial-time \([\text{Kawamura}&\text{Cook}'2012]\)

- “Efficiently” encoding (“representing”) spaces beyond \( \mathbb{R} \)
  - such as Sobolev spaces (theory of PDEs)

- Computational complexity theory of **higher** types

- **Randomization** in continuous data processing

- **Formal verification** of **continuous** abstract data types

- **Software Testing** for **continuous** abstract data types

---

\[ \text{Kawamura 2010} \]

\[ \text{[Kawamura+Steinberg+Z., MSCS 2017]} \]
Logic of Computing with Continuous Data: Foundation of Numerical Software Engineering

1. Forget float!
2. Continuous abstract data types
3. Complexity: approx. up to $2^{-n}$

Martin Ziegler

Thank you

(discrete) Recursion Theory
Computability over the Reals
Computability on separable metric Spaces

Computational Complexity ($P/NP$)

Real Complexity Theory

Metric Properties
Continuity / Topology

graphs, integers, …
resource-bounds (runtime, memory)

Numerics
PDEs, $L^p W^{k,p}$
Foundations of Software Engineering

Implementation/library
Abstract Data Types
Semantics/Verification
(Scott, Hoare, ...)
Complexity
(Hartmanis, Cook, ...)
Computability
(Turing, Kleene, ...)
Specification

"A good definition is worth 1000 theorems" (Doron Zeilberger, Opinion #82)

Data Abstraction and Properties

Incompleteness
(Gödel, Robinson)
distributive law
\[ x \cdot (y+z) = x \cdot y + x \cdot z \]
(logic.complete r.e.
1st-order axioms)
algebraically complete
metrically complete
Algebraic Geometry
Existence of solutions to PDEs

(finite) sets
\[ \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{A}, \mathbb{R}, \mathbb{C}, \mathbb{P}, \mathbb{H}, \mathbb{W} \]
{0,1}*
byte int double
abstract data type
BigInteger
\texttt{iRRAM: REAL}
Xerox PARC

abstract data type
(Java)
Disambiguation

- Numerical Software Engineering
- Continuous Abstract Data Types
- Closest agreement with structures in Logic
- Rigorous elegant computable specification
- without hassles of actual Turing machines.
- Imperative Turing-complete programming
- Include transcendental computation
- Guided by Newton’s Method: arguments exact, result approx.