Logic of Computing with Continuous Data

Franz Brauße
河村彰星
이계식
Pieter Collins
Johannes Kanig
김선영
임동현
박세원
Michal Konečný
Norbert Müller
Eike Neumann
Norbert Preining
Svetlana Selivanova

Martin Ziegler, KAIST

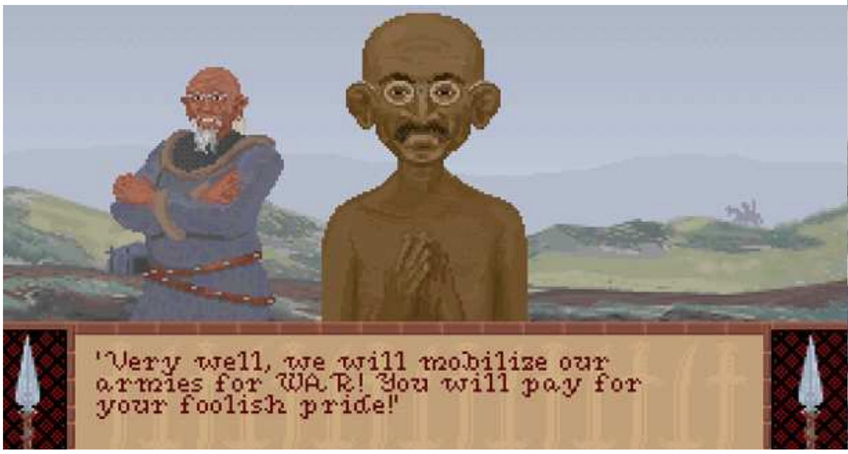
Computer Game *Civilization*

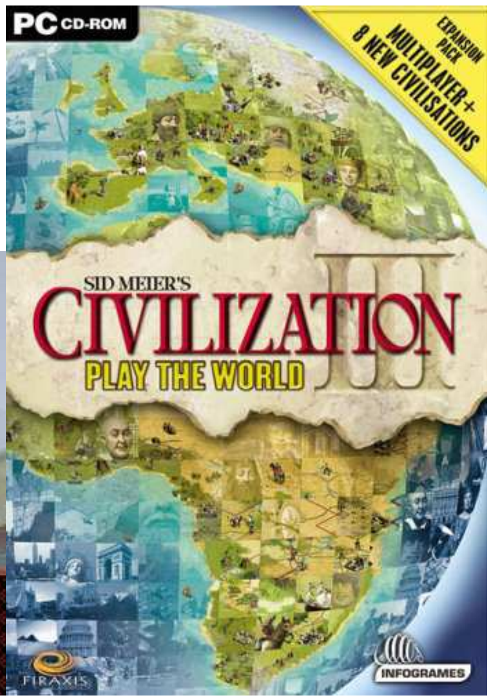
KAIST
Martin Ziegler

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.

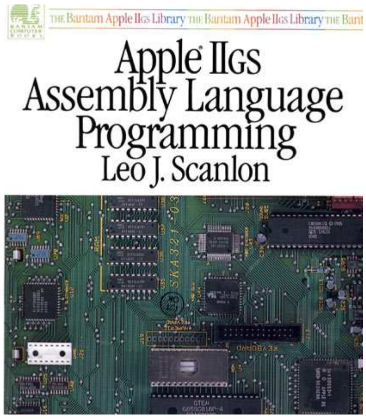




Computer Nostalgia (1980ies)

Martin Ziegler

- *empirical* correctness
- *hardware* data type **`byte/word`** $\neq \mathbb{Z}$
- *In*convenient/ *un*elegant →bugs
- *absolute* performance
- *low*-level prog.language



1 MHz CPU
64kB RAM

Coding, **not** Computer <u>Science</u>

Contemporary Computer <u>Science</u>

Martin Ziegler

- *empirical* correctness
- *hardware* data type **`byte/word`** $\neq \mathbb{Z}$
- *in*convenient/ *un*elegant →bugs
- *absolute* performance
- *low*-level prog.language

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

L O G I C

Abstract Data Type = Structure

Martin Ziegler

- *empirical* correctness
- *hardware* data type `byte/word`
- convenient, elegant → *less* bugs
- ~~*absolute* performance~~
- ~~*low*-level prog.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)

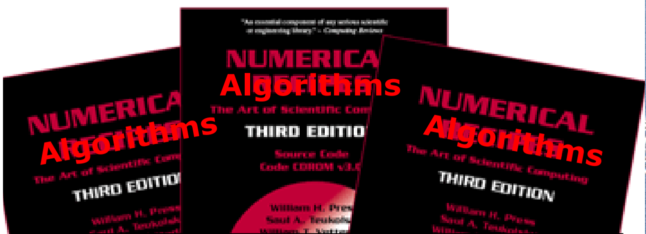
Object-oriented *high*-level programming language:

- *modular* software design
- "*Information hiding*"
- *ideal* data types

Contemporary ***Numerics***

Martin Ziegler

- *empirical* correctness
- *hardware* data type `float/double` ≠$\mathbb{R}$
- *in*convenient/ *un*elegant →bugs
- *absolute* performance



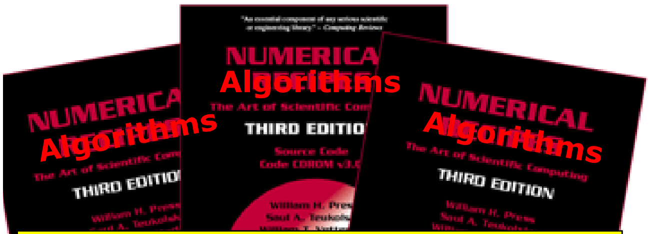
„**nag_opt_one_var_deriv**" *normally* computes a sequence of x values which *tend in the limit* to a minimum of F x

7. Implementation/coding

Medieval Mathematics

KAIST
Martin Ziegler

- *empirical* correctness
- *hardware* data type `float/double` ≠ℝ
- *in*convenient/ *un*elegant & ***in***consistent
- *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

ℚ, as opposed to ℝ, violates:

- intermed. value theorem
- compactness of [0,1]
- fixed point theorems,
- logical completeness,
- and even distributive law

Don't test for equality!

How about *in*equality "**<**" ?

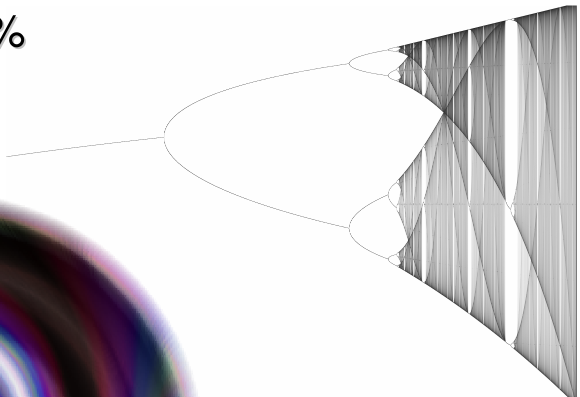
$x=0 \;\Leftrightarrow\; \neg(x<0) \;\wedge\; \neg(x>0)$

IEEE754:
±∞, ±0, **NaN**s, *de*normals

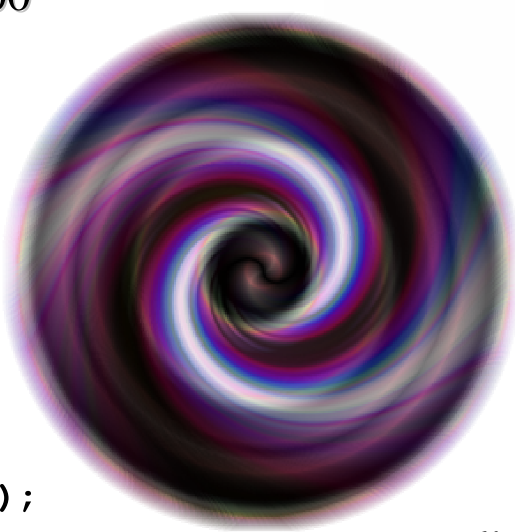
Example Prog. Codes

KAIST
Martin Ziegler

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



`float`: $n\leq 30$, `double`: $n\leq 85$,
`long double`: $n\leq 200$
`rational`: $n\leq 24$



```
MATLAB function
  y=jmmuller(m)
x = vpa(11/2);
y = vpa(61/11);
while m>1;
 t = vpa(111 -
    (1130-3000/x)/y);
 x=y; y=t; m=m-1;
 end; end
```
→ **100**

$x_0 := 11/2, \quad x_1 := 61/11$

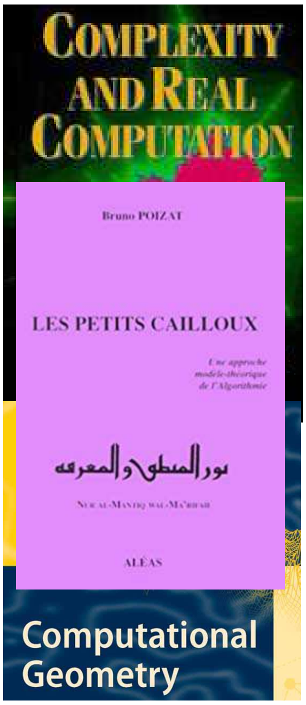
$x_{m+1} := 111 - (1130\text{-}3000/x_{m-1})/x_m \to \mathbf{6}$

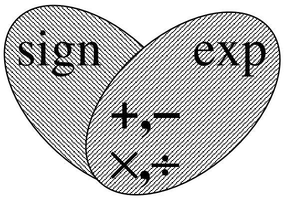
Models of *Real* Computation

KAIST
Martin Ziegler

ℝ, not ℚ nor 𝔸

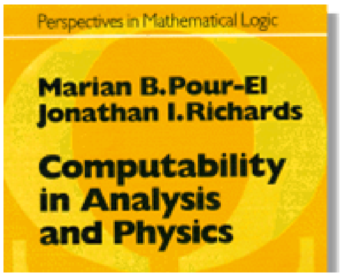
[Turing'37], [Grzegorczyk'57]

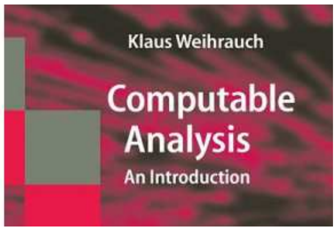




realPCF
[Escardó'96]
(functional)

THIS TALK
(imperative)





Imperative,
+,−, ×, ÷, <

<u>exact</u>

Computing by
streams of approximations

<u>Intermission</u>

KAIST
Martin Ziegler

Floating point numbers (IEEE 754 from 1985)
are a hardware-based, *legacy* data type — like **`byte`**:

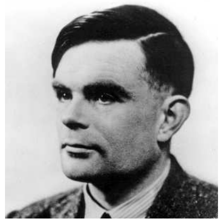
to be *hidden* behind object-oriented *ideal* data types!

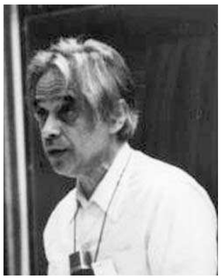
Project: Develop <u>*continuous* abstract data types</u>.
- Fix some continuous structure (example: real numbers).
- Investigate computability of its operations/predicates. (Turing machines, *Computable Analysis*)
- Modify their semantics to *make them computable* and s.t.
- computing *over* this structure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

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

Computing Real Numbers

Martin Ziegler





Fact: For $r\in\mathbb{R}$, the following are equivalent:

Call $r\in\mathbb{R}$ computable if

a) r has a recursive binary expansion

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

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

Ernst Specker (1949): (c)$^{\text{Halting problem}}$ $\Leftrightarrow$ (d)

d) There exists a machine printing $(q_n)\subseteq\mathbb{Q}$ with $q_n\to r$.

$\mathbb{R}$ as Computable Abstract Data Type

Martin Ziegler

- 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: +, -, ×, ÷ are computable

Fact: There is a *computable* sequence $(r_j)\subseteq[0,1]$ such that $\{\,j : r_j\neq 0\,\} = H$, the Halting problem.

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

- +, -, ×, ÷, 0, 1: *exact* operations
- *Partial* semantics of tests: „$x>y$" = ↑ if $x=y$.
- **choose**$(x_1>y_1,\ldots x_d>y_d)$ = *any* j s.t. $x_j>y_j$

provided it exists

Programming over Computable ℝ

KAIST
Martin Ziegler

Example 0: *Fuzzy/soft* test [Chee K. Yap, …]

choose($y+2^{-n} > x$, $x+2^{-n} > y$)

```
REAL Trisection(INTEGER p; REAL→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
```

$(\mathbb{Z},0,1,+,>)$ and $(\mathbb{R},0,1,+,\times,>)$ with $\iota:\mathbb{Z}\ni p\to 2^p\in\mathbb{R}$

Arguments *exact*, value *approximate* to 2^p.

- *Partial* semantics of tests: „**x>y**" = ↑ if $x=y$.
- **choose**($x_1>y_1,\ldots x_d>y_d$) = *any* j s.t. $x_j>y_j$

Logic of Computing over ℝ

KAIST
Martin Ziegler

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!

⇐ [van den Dries'86]
→ formal verification.

c) Generalizing (a) to real function*al*s.

Also "*model complete*"
Wrong for $\iota(n)=1/n$

$(\mathbb{Z},0,1,+,>)$ and $(\mathbb{R},0,1,+,\times,>)$ with $\iota:\mathbb{Z}\ni p\to 2^p\in\mathbb{R}$

Arguments *exact*, value *approximate* to 2^p.

- *Partial* semantics of tests: „**x>y**" = ↑ if $x=y$.
- **choose**($x_1>y_1,\ldots x_d>y_d$) = *any* j s.t. $x_j>y_j$

Computing over Continuous Structures

KAIST
Martin Ziegler

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! ⇐ [van den Dries'86] → formal verification.

c) Generalizing (a) to real functiona***l***s. Also "*model complete*" Wrong for $\iota(n)=1/n$

Continuous Abstract Data Types *in progress*:

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

Intermission

KAIST
Martin Ziegler

Complexity of Computing with Continuous Data

Project: Develop *continuous* abstract data types.

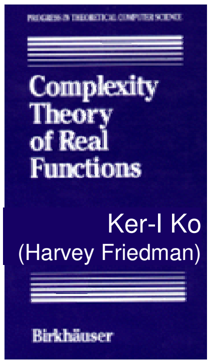
- Fix some continuous structure (example: real numbers).
- Investigate computability of its operations/predicates. (Turing machines, *Computable Analysis*)
- Modify their semantics to *make them computable* and s.t.
- computing *over* this structure becomes Turing-*complete.*

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

Complexity Theory for Numerics

Martin Ziegler

Fix polytime $f:[0;1]\to[0;1]$ ($\Rightarrow$ continuous)



- Max: $f \to \mathrm{Max}(f): x \to \max\{ f(t): t \le x \}$
  $\mathrm{Max}(f)$ computable in $\mathcal{PSPACE}$
  polyn.time-computable **iff** $\mathcal{P}=\mathcal{NP}$
- $\int: f \to \int f: (x \to \int_0^x f(t)\,dt)$ **even** for $f\in C^\infty$
  $\int f$ computable in $\mathcal{PSPACE}$
  polyn.time-computable **iff** $\mathcal{FP}=\#\mathcal{P}$

Polyn.time for analytic $f\in C^\omega$

- odesolve: $C^1([0;1]\times[-1;1])\ni f\to z$: $\dot z(t)=f(t,z)$, $z(0)=0$.
  $\mathcal{PSPACE}$-"complete" [Kawamura 2010]
- Solution to Poisson's Equation is classical and $\#\mathcal{P}$-"complete"

$$\Delta u = f \text{ on } B_2(\mathbf{0},1)$$
$$u = 0 \text{ on } \partial B_2(\mathbf{0},1)$$

[Kawamura+Steinberg+Z., MSCS 2017

Advanced (Complexity) Issues

Martin Ziegler

- Weihrauch Reduction among *continuous* problems

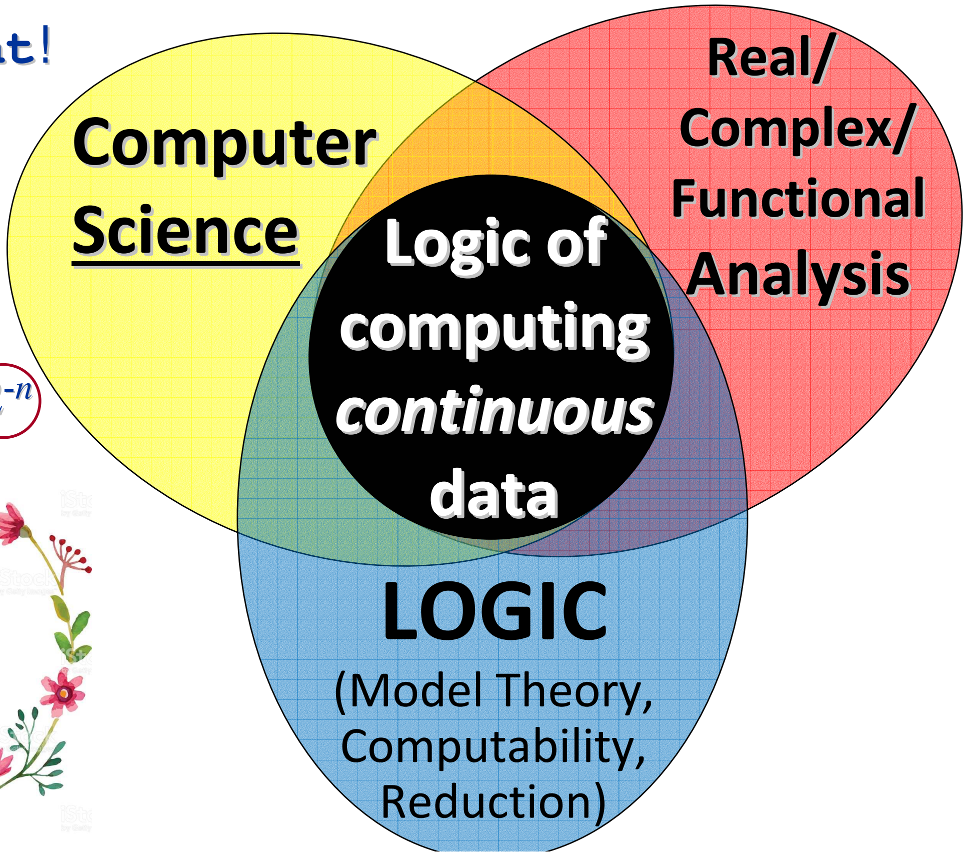
  2nd-order polynomial-time [Kawamura&Cook'2012]

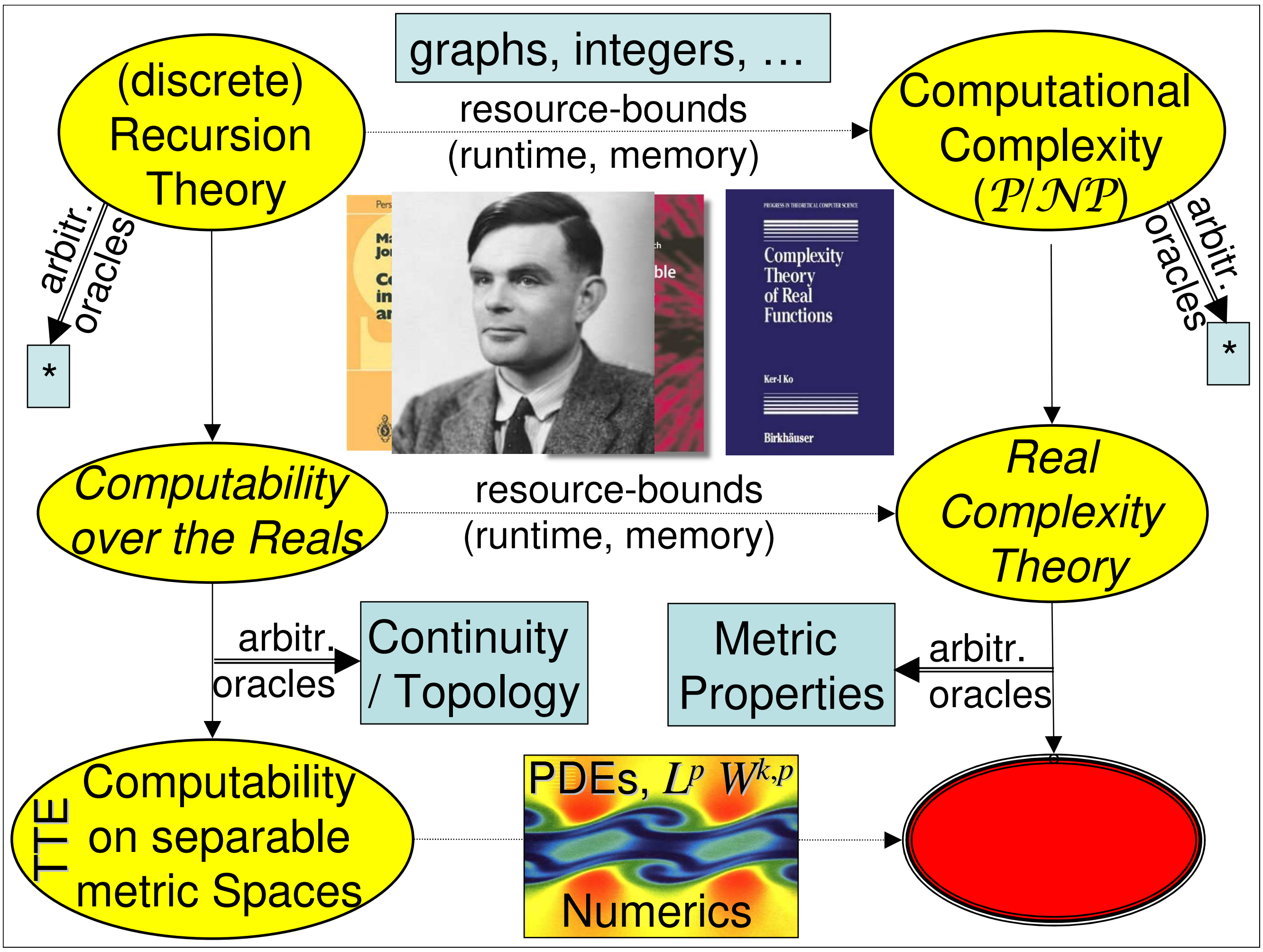
- "Efficiently" encoding ("*representing*") spaces beyond $\mathbb{R}$
  such as Sobolev spaces (theory of PDEs)
- Computational complexity theory of *higher* types
- *Randomization* in continuous data processing
- *Formal verification* of *continuous* abstract data types
- *Software Testing* for *continuous* abstract data types

Logic of Computing with *Continuous* Data: Foundation of Numerical Software Engineering

KAIST
Martin Ziegler

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





Foundations of Software Engineering

KAIST
Martin Ziegler

discrete (integers, graphs, …)

<u>**Implementation/library**</u>

Abstract Data Types

Semantics/Verification
(Scott, Hoare, …)

Complexity
(Hartmanis, Cook, …)

Computability
(Turing, Kleene, …)

Specification

real (numbers, functions, operators)

continuous structures

„A good ***definition*** *is worth 1000 theorems“*
(Doron Zeilberger, Opinion #82)

Data Abstraction and Properties

KAIST
Martin Ziegler

*In*completeness
(Gödel,Robinson)

logic.complete r.e.
1st-order axioms

algebraically complete

distributive law
$x \cdot (y+z) =$
$= x \cdot y + x \cdot z$

Algebraic Geometry

metrically complete

Existence of solutions to PDEs

ring, field

(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` *grids* of **`double`**, ad-hoc

abstract data type **`BigInteger`** (Java) **`iRRAM:REAL`** **Xerox PARC**

Disambiguation

Martin Ziegler

- *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.