Designing, Understanding, and Analyzing Unconventional Computation: The Important Role of Logic and Constructive Mathematics

Grigori Mints
Stanford University, gmints@stanford.edu

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu
1. Main Problem: Reminder and Possible Approaches

- **Problem**: computations are often too slow.

- **Traditional approaches**:
  - design faster super-computers (*hardware*);
  - design faster *algorithms*.

- **Limitations of the traditional approaches**:
  - *re hardware*: we use the same physical processes as before;
  - *re algorithms*: we solve the same *exact* problem as before – while often, data are *imprecise*.

- **Possible new approaches**:
  - *hardware*: use unconventional physical (and biological) processes;
  - *algorithms*: perform computations only up to accuracy that matches the input accuracy.
2. Our Main Claim: Use Logic (and Constructive Mathematics)

- Alternative approaches (reminder):
  - use unconventional physical (and biological) processes;
  - perform computations only up to accuracy that matches the input accuracy.
- Our claim: for these approaches to succeed, it is crucial to further develop:
  - the corresponding tools of mathematical logic, and
  - the related methods of constructive mathematics.
3. Why Logic?

- **Claim**: logic is useful on all the stages of *solving a problem*:
  - we *specify* a problem;
  - we design and implement an *algorithm*;
  - we *verify* the corresponding program.

- **Specifying a problem**:
  - sometimes, the problem is to solve an equation;
  - in general, a proper formulation requires quantifiers etc. (stable control, etc.) – i.e., logic.

- **Designing an algorithm**: logic programming transforms a logical specification into an algorithm.

- **Program verification**: logic helps in reasoning about programs; e.g., pre-condition implies post-condition.

- **Proof assistant programs** (based on logic) help to prove.
4. 1st Approach: Computations with Limited Accuracy

- **Objective**: compute $f(x)$ with accuracy $\varepsilon > 0$.
- **Idea**: compute $x$ only with accuracy $\delta > 0$ for which
  
  $$d(x, x') \leq \delta \text{ implies } d(f(x), f(x')) \leq \varepsilon.$$  

- **Constructive math**: a number is $r_n$ s.t. $d(r_n, x) \leq 2^{-n}$; a **constructive function** is a pair of:
  - an algorithm $f : X \to Y$ and
  - an algorithm $\varepsilon \to \delta$.

- **Status**: this is developed only for a few problems.

- **Research Direction I.1**:
  - **develop general constructive mathematics techniques**,  
  - **with a special emphasis on problems requiring intensive computations** (e.g., large-scale PDE).
5. 1st Approach: Need for Logic

- *Decomposition*: solutions to complex problems usually come from combining solutions to subproblems.
- *Logic is needed* for combination: e.g., stable robust control means it is stable $\forall$ possible parameter values.
- *Constructive logic*: we want to preserve constructions:
  - $\exists x \ P(x)$ should mean that we can construct such $x$;
  - $\forall x \exists y \ P(x, y) \leftrightarrow \exists$ algorithm $\varphi : X \rightarrow Y$ s.t. $P(x, \varphi(x))$.
- It was originated by Kolmorogov: e.g., $A \lor \lnot A$ is false.
- *Fact*: constructive logic is used in constructive math.
- *Research Direction I.2*:
  - develop general constructive logic techniques,
  - with a special emphasis on problems requiring intensive computations.
6. Interval Computations

- **Constructive math**: algorithms work for all accuracies.
- **In practice**: we only need one given accuracy.
- **Solution**: interval computations (a.k.a. applied constructive mathematics).
- **Idea**: if we know an estimate $\tilde{x}$ w/accuracy $\Delta$, then
  \[ x \in x = [\tilde{x} - \Delta, \tilde{x} + \Delta]. \]
- **Traditional approach**: we also know probability distribution for $\Delta x \overset{\text{def}}{=} \tilde{x} - x$ (usually Gaussian).
- **Where it comes from**: calibration using standard MI.
- **Problem**: calibration is not possible in:
  - fundamental science – no better Measuring Instr. (MI);
  - manufacturing – too expensive to calibrate.
7. Interval Computations (cont-d)

- **Given**: algorithm \( y = f(x_1, \ldots, x_n) \) and \( x_i = [\underline{x}_i, \overline{x}_i] \).
- **Compute**: the corresponding range of \( y \):
  \[
  y = [\underline{y}, \overline{y}] = \{ f(x_1, \ldots, x_n) \mid x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, x_n \in [\underline{x}_n, \overline{x}_n] \}.
  \]
- **Fact**: this problem is NP-hard even for quadratic \( f \).
- **Challenge**: find a good approximation \( Y \supseteq y \).
- **Applications**: spaceflights, super-colliders, robotics, chemical engineering, nuclear safety, etc.
- **Modal logic** is efficiently used: \( \square (y \in y) \), \( \Diamond (x = \bar{x}) \).
8. Proof Mining

• Historically: first existence proofs were direct (constr.).
• Currently: many proofs are indirect – they prove $\exists x \ P(x)$ without constructing such $x$.
• Meta-results: sometimes, we can extract (“mine”) a constructive proof from a non-constructive one.
• Example (Kohlenbach): uniqueness $\rightarrow$ computability.
• Idea: to find $x$ s.t. $f(x) = 0$, compute $\min_{B_\varepsilon(x_i)} |f(x_i)|$ w/increasing accuracy for $x_i$ from $\varepsilon$-nets, $\varepsilon = 2^{-1}, 2^{-2}, \ldots$
• Research Direction I.4:
  • further develop proof mining,
  • with a special emphasis on its use to develop algorithms for realistic large-scale problems.
9. 2nd Approach: Unconventional Computations

- **Main idea:** use non-standard physical processes to speed up computations.

- **Most well-known example:** quantum computing:
  - search in an un-sorted array of size \( n \) in time \( \sqrt{n} \) (Grover);
  - factoring large integers in polynomial time (Shor).

- **Limitation:** the only provable speed-up is polynomial.

- **Other schemes:** can potentially lead to exponential speed-up.

- **In other words:** we can potentially solve NP-hard problems in polynomial time.

- **Simplest example:** acausal processes – compute and send the result back in time.
10. Potential Use of Acausal Processes (cont-d)

- **Idea** (reminder): compute and send the result back in time.

- **Problem**: paradoxes of time travel – e.g., killing your own grandfather before your father was conceived.

- **Solution**: some low probability event prevented the time traveller from this killing.

- **Consequence**: time travel (TT) can trigger events with probability \( p_0 \ll 1 \).

- **Typical NP-hard problem**: SAT – given a propositional formula \( F(x) \), find \( x = (x_1, \ldots, x_n) \) s.t. \( F(x) \) holds.

- **Usage** (H. Moravec et al.): to solve SAT, generate \( n \) bits \( x \) and if \( \neg F(x) \), launch TT.

- **Why it works**: for \( 2^{-n} \gg p_0 \), TT is statistically improbable.
11. Potential Use of Curved Space-Time

- *Parallelization* – a natural source of speed-up.
- *Claim*: in Euclidean space-time, parallelization only leads to a polynomial speed-up.
- *Fact*: the speed of all the physical processes is bounded by the speed of light $c$.
- *Conclusion*: in time $T$, we can only reach computational units at a distance $\leq R = c \cdot T$.
- The volume $V(R)$ of this area (inside of the sphere of radius $R = c \cdot T$) is proportional to $R^3 \sim T^3$.
- So, we can use $\leq V/\Delta V \sim T^3$ computational elements.
- *Interesting*: in Lobachevsky space-time, $V(R) \sim \exp(R)$.
- Same is true for some more realistic space-time models.
- Hence, we can fit exponentially many processors – and thus get an exponential speed-up.
12. Explicit Use of Kolmogorov Complexity

- **Fact:** it’s often difficult to describe biological processes.

- **Idea (M. Gell-Mann):** physical equations should include terms explicitly depending on complexity.

- **Natural formalization:** Kolmogorov complexity
  \[ K(x) \overset{\text{def}}{=} \min \{ \text{len}(p) : p \text{ generates } x \} \].

- **Conclusion:** by observing physical and biological processes, we can measure the value \( K(x) \).

- **Observation:** \( K(x) \) is not algorithmically computable.

- **Known results:** ability to get non-computable values can speed up computations.

- **Other schemes** are based on:
  - quantum field theory (G. Kreisel),
  - that every theory is approximate, etc.
13. Unconventional Computations (UC) and Constructive Mathematics

- **Above UC schemes**: use or propose a radically new physical process.
- **Fact**: some UC schemes were discovered by analyzing computability of known physical equations.
- **Example** (M. Pour-El et al.): even for wave equation, for some computable $u(x, 0)$, $u(x, T)$ is not computable.
- **Desirable**: extend the existing UC activity to the analysis of what computations can be sped up.

- **Research Direction II.1.** Use constructive mathematics to analyze:
  - how the use of physical processes (described by physically meaningful equations)
  - can speed up computations.
14. References: Why Logic

15. **References: Constructive Mathematics**

16. References: Interval Computations

- website http://www.cs.utep.edu/interval-comp
17. References: Proof Mining

18. References: Unconventional Computations – Use of Curved Space-Time


19. References: Unconventional Computations – Use of Possible Acausal Processes


20. References: Unconventional Computations – Other Schemes


21. References: Constructive Mathematics in Unconventional Computations


