Embracing the Laws of Physics (Full Presentation)

Roshan P. James Amr Sabry
Indiana University

Computation and Physics. One of the major achievements of Computer Science has been the development of abstract models of computation that shield the discipline from the underlying technology. As effective as these models have been, one must note, however, that they *embody several implicit physical assumptions*. As Toffoli explains:

> Mathematical models of computation are abstract constructions, by their nature unfettered by physical laws. However, if these models are to give indications that are relevant to concrete computing, they must somehow capture, albeit in a selective and stylized way, certain general physical restrictions to which all concrete computing processes are subjected [11].

More specifically, Toffoli and Fredkin [5] argue that at the core of Turing's arguments for the universality of his machine are the following physical assumptions: (i) the speed of propagation of information is bounded; (ii) the amount of information which can be encoded in the state of a finite system is bounded; and (iii) it is possible to construct physical devices which perform in a recognizable and reliable way the logical functions **and**, **not**, and **fanout**. They then argue that some other principles, such as the reversibility of all physical laws, are not reflected in current models of computation.

An even more fundamental issue with the Turing machine abstraction is that it embodies *classical* Physics. As Deutsch notes:

> Turing hoped that his abstracted-paper-tape model was so simple, so transparent and well defined, that it would not depend on any assumptions about physics that could conceivably be falsified, and therefore that it could become the basis of an abstract theory of computation that was independent of the underlying physics. 'He thought,' as Feynman once put it, 'that he understood paper.' But he was mistaken. Real, quantum-mechanical paper is wildly different from the abstract stuff that the Turing machine uses. The Turing machine is entirely classical, and does not allow for the possibility the paper might have different symbols written on it in different universes, and that those might interfere with one another [3, p.252].

Dually, on the other side of the Curry-Howard isomorphism, we find arguments that *logical systems also embody some laws of Physics*. In particular, classical logic is based on the assumption that the truth of a statement is absolute and independent of any reasoning, understanding, or action, and hence that statements are either true or false with no regard to any observer. This absolute view of truth has been critiqued for centuries by philosophers and other scientists. In fact, one might argue that the intuitionistic view that the only truths are those that can be communicated with finite evidence is what led to the birth of Computer Science. More recently, Girard motivates linear logic by stating that it embodies a simple and radical change of viewpoint from other logics and that this change has a physical flavor [6, pp. 6,17]. In fact, Girard clearly argues that the physical laws should guide our conception of logic:

> In other terms, what is so good in logic that quantum physics should obey? Can't we imagine that our conceptions about logic are wrong, so wrong that they are unable to cope with the quantum miracle? [...] Instead of teaching logic to nature, it is more reasonable to learn from her. Instead of interpreting quantum into logic, we shall interpret logic into quantum [7].

New Foundations. As argued above, the "beaten track" of programming language research embodies an idealized, classical, and deterministic worldview that only approximates physical reality. We argue that this approach is fundamentally limiting when dealing with modern applications and problem domains and that, as Milner said in a different context [10], this state of affairs "requires a fresh approach, not merely an extension of the repertoire of [current] entities and constructions."

For concreteness, we propose that the following four physical principles provide a principled approach to the next phase in programming language research:

- Information is physical: it is encoded, processed, stored, and transmitted by physical means.
- Isolated (closed) physical systems evolve in a unitary way which is both reversible and preserves "distances."
- Observing the properties of a physical system requires an interaction which may, probabilistically, disturb the properties of interest.

- A large number of interacting physical systems evolves concurrently, speculatively, and probabilistically.

Translated to a computational language, these principles suggest the following:

- our notion of data should take locality into account;
- our core programming calculi should be reversible and should preserve locality;
- observation of runtime properties should be an integral part of the execution model; and
- higher-level abstractions should explicitly embrace speculative concurrency, nondeterminism, and other irreversible computational effects.

In more detail, being reversible at the lowest level of abstraction implies that — at that level —, all computational effects are accounted for, even those that are related to energy or information [9]. This explicit treatment of *information effects* could provide the conceptual framework for reasoning about security attacks that currently occur "under our abstraction level" (e.g. by analyzing energy signatures) as well as the optimization of energy resources. Furthermore, by taking locality of data into account, i.e., by associating a metric space with our data, several problem domains (continuity analysis of programs [2], differential privacy [4], etc.) could become consequences of the fact that physical computation must preserve distances. Finally, as Baker argued years ago [1], speculative concurrency of closed systems would be trivial.

Embracing nondeterminism and other irreversible computational effects at higher-level of abstractions (open and large scale systems) is consistent with the pragmatic observations that computation changes in qualitative ways at large scales. For example, as the Google experiments show, when one is dealing with petabytes of data, implementing even mundane sequential algorithms like sorting becomes a challenge. Specifically, at large distributed scales, hardware component failure becomes the norm rather than the exception [12]; determinism and consistency become so expensive and impractical to maintain that nondeterminism and "eventual consistency" are often accepted [13]; global synchronization becomes impossible which makes distributed (speculative) algorithms intricate to write and hard to reason about [8]. Building such applications starting with abstract models that are fundamentally deterministic with layers and layers of ad hoc exceptions is certainly not ideal. Much more appealing is a computational model that, from the outset, admits that preconditions and invariants can generally only be approximately verified, that computations can generally only produce probabilistic outcomes, and that consistency can only "eventually" be achieved.

Finally, we note that the physical approach we advocate integrates well with emerging "unconventional" models of computations such as quantum computation and biologically or chemically inspired models of computation and generally makes programming languages "at home" in the universe we live in, able to express computations that interact with all aspects of the "real world." Such physically-inspired programming languages could eventually provide the conceptual framework for many of the themes stated in a recent NSF program: "to exploit computation as a means of achieving deeper understanding in the natural and social sciences and engineering; to simulate and predict complex stochastic or chaotic systems; to explore and model nature's interactions, connections, complex relations, and interdependencies, scaling from sub-particles to galactic, from subcellular to biosphere, and from the individual to the societal."

References

[1] Baker, H. G. NREVERSAL of fortune - the thermodynamics of garbage collection. In *Workshop on Memory Management* (1992), Springer-Verlag.

[2] Chaudhuri, S., Gulwani, S., and Lublinerman, R. Continuity analysis of programs. In *POPL* (2010).

[3] Deutsch, D. *The Fabric of Reality*. Penguin, 1997.

[4] Dwork, C. Differential privacy. In *ICALP (2)'06* (2006).

[5] Fredkin, E., and Toffoli, T. Conservative logic. *International Journal of Theoretical Physics 21*, 3 (1982).

[6] Girard, J.-Y. Linear logic. *TCS 50* (1987).

[7] Girard, J.-Y. Truth, modality and intersubjectivity. *MSCS 17*, 6 (2007).

[8] Guerraoui, R. Foundations of speculative distributed computing. In *Distributed Computing*, vol. 6343 of *LNCS*. Springer, 2010.

[9] James, R. P., and Sabry, A. Information effects. In *POPL* (2012).

[10] Milner, R. Elements of interaction: Turing award lecture. *Commun. ACM 36*, 1 (1993), 78–89.

[11] Toffoli, T. Reversible computing. In *Automata, Languages and Programming* (1980), Springer-Verlag.

[12] Vishwanath, K. V., and Nagappan, N. Characterizing cloud computing hardware reliability. In *Symposium on Cloud computing* (2010).

[13] Vogels, W. Eventually consistent. *Commun. ACM 52* (2009).