How to Compare the Power of Computational Models[⋆]

Udi Boker and Nachum Dershowitz

School of Computer Science, Tel Aviv University
Ramat Aviv, Tel Aviv 69978, Israel
email: `udiboker@tau.ac.il`,
`nachum.dershowitz@cs.tau.ac.il`

Abstract. We argue that there is currently no satisfactory general framework for comparing the extensional computational power of arbitrary computational models operating over arbitrary domains. We propose a conceptual framework for comparison, by linking computational models to hypothetical physical devices. Accordingly, we deduce a mathematical notion of relative computational power, allowing the comparison of arbitrary models over arbitrary domains. In addition, we claim that the method commonly used in the literature for "strictly more powerful" is problematic, as it allows for a model to be more powerful than itself. On the positive side, we prove that Turing machines and the recursive functions are "complete" models, in the sense that they are not susceptible to this anomaly, justifying the standard means of showing that a model is "hypercomputational."

1 Introduction

Our goal is to formalize comparisons of computational models, that is, the determination when one set of partial functions is computationally more powerful than another set. We seek a robust definition of relative power, one that does not depend itself on any notion of computability. It should allow one to compare arbitrary models over arbitrary domains in some quasi-ordering that captures the intuitive concept of computational strength. Such a comparison notion (or notions) should also allow one to prove statements like "analogue machines are strictly more powerful than digital devices," even though the two models operate over domains of different cardinalities.

With a satisfactory comparison notion in place, we look into mathematical relations between computational models, and properties they confer on models. We call a model that is not as powerful as any of its proper expansions "complete." We investigate completeness, and check whether some classical models enjoy this property.

[⋆] This work was carried out in partial fulfillment of the requirements for the Ph.D. degree of the first author.

Extensionality. We are only interested in the computational aspect of computational models (extensionality), that is, which problems can be solved by a model, regardless of the solution's complexity or the model's mechanisms. Hence, a computational model is represented simply by a set of (partial) functions (or multivalued functions) over the domain of its operation.

The Problem. Though model comparison is a common practice in the literature, it is usually done without a formal comparison notion and without justification for the chosen method. To the best of our knowledge, there is currently no satisfactory general means for comparing arbitrary computational models operating over arbitrary domains. A notion is lacking via which one could show, for example, that analogue computers are strictly more powerful than Turing machines, as well as show that finite automata are more powerful than some weak analogue model. In Section 4, we list some of the familiar comparison methods and discuss their ramifications.

The Framework. In Section 2, we propose a general, philosophical, definition of a computational model and of relative computational power. We understand a computational model to be a mathematical modeling and idealization of some hypothetical physical device, from a specific point of view of the world. A model B is at least as powerful as A if it has the potential to do whatever A does, under any possible view of the world. Accordingly, we provide, in Section 3, a method (Definition 3) for comparing arbitrary models over arbitrary domains.

Completeness. In Section 5, we show that the method usually used in the literature for "more powerful" ($\succnsim$) is mathematically problematic, as it allows for a model to be more powerful than itself ($A \succnsim A$). We define a model that is not as powerful as any of its proper expansions to be *complete*. The standard method of comparison is suitable only for such complete models. On the positive side, we prove in Section 6.1 that Turing machines and the recursive functions are complete with respect to the desired comparison notions.

Computability. In Section 6, we show that some of the models known to be of equivalent power to Turing machines (the recursive functions, random access machines and counter machines) are indeed so by our suggested general notion.

Hypercomputation. In Section 6.1, we prove that Turing machines and the recursive functions are complete models. Accordingly, we provide a simpler comparison notion for showing that a model is hypercomputational. This notion provides a justification for the (otherwise improper) comparison method used in the literature for showing that a model is hypercomputational.

Note. We use the Z-standard [3] for function arrows. For example, $\nrightarrow$ denotes a partial function, $\twoheadrightarrow$ is used for a total surjective function, and $\rightarrowtail$ is an injection. We use double-arrows for mappings (multi-valued functions). So $\rightrightarrows$ denotes a total surjective mapping.

Proofs are omitted for lack of space.

2 The Conceptual Framework

We first propose a general, philosophical, definition of a computational model, and—in Section 2.2—of relative computational power. In Section 3, we will formalize these definitions for comparing arbitrary models over arbitrary domains.

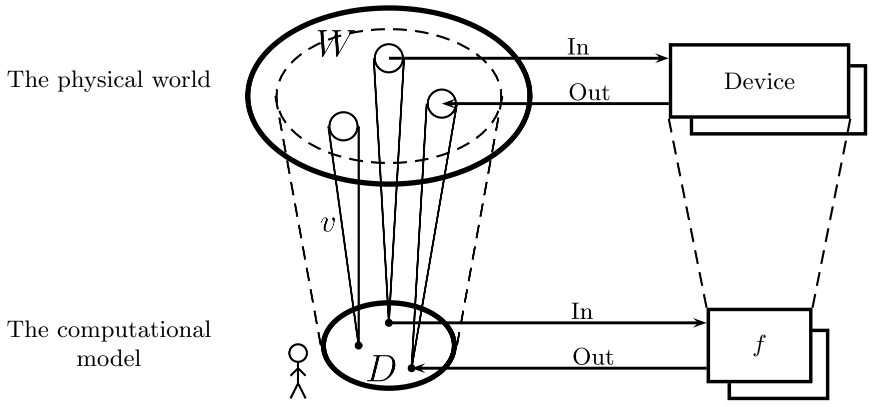

Fig. 1. A computational model is a mathematical modeling of some hypothetical physical devices, from a specific point of view of the world

2.1 What is a Computational Model?

We can think of a computational model as a mathematical modeling and idealization of some hypothetical physical device, from a specific point of view of the world (see Fig. 1).

- A physical device gets a physical input and returns a physical output. For example, an electric device may take some electric voltage at two of its pins as input, and return a voltage at two other pins as output.
- A corresponding computational model takes a specific point of view of the physical world. For example, a model of a digital computer might view a voltage lower than 0.5v as the binary value 0 and of 0.5v or higher as 1. That is, the domain of the model, D, is a "view" of the physical world, W. This view is a partial surjective function $v : W \mapsto\!\!\!\rightarrow D$.
- The device computes a function on world entities (in our example above, $\xi : \mathbf{R} \to \mathbf{R}$), while from the model's point of view it computes a function on its domain (in our example, f :{0, 1} $\to$ {0, 1}).

A computational model, by itself, can be viewed as a "black box," computing a set of partial functions. The domain and range of functions are identical, except that the range is extended with $\perp$, representing "undefined."

The modeling of a hypothetical device from a specific point of view of the world will be at the heart of our method of comparing different models. The world can be chosen to be any set of cardinality at least as large as the cardinality of the model's domain.

The idea that a model encapsulates a point of view of the world is shared by Minsky [9]:

> We use the term "model" in the following sense: To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A. The model relation is inherently ternary. ... It is understood that B's use of a model entails the use of encodings for input and output, both for A and for A*. If A is the world, questions for A are experiments.

Different Domain and Range. There are models with different domain and range, e.g. numeral input and boolean output. A generalized view is to consider the "actual" model's domain to be the union of the original domain and range.

Uniform Computation. It is common to have models with functions of any fixed arity, like the recursive functions, for example. We consider the "actual" domain (and range) to be the set of all finite tuples of elements of the original domain. This is the view taken for Turing machines, in the BSS model [1, pp. 69–70], and implicitly with recursive functions when comparing them to Turing machines.

Computing over Structures. There are models defined over structures, that is, over sets together with "built-in" functions and relations. See, for example, [2, 13, 1]. We consider the structure's set as the domain, and include the structure's functions and relations in the model.

2.2 Comparing Computational Power

We generally say that a model B is at least as powerful as A, written $B \succsim A$, if it can do whatever A does. When both models have the same domain representation, it means "containment": B is at least as powerful as A if it computes all the functions that A does. The question is how one should compare models operating over different domains, as they compute formally-different functions.

We extend the above characterization as follows: B is at least as powerful as A if it has the potential to do whatever A does for every possible user (an abstract user, not necessarily human). In other words, for every view that A has of the world ($v : W \twoheadrightarrow \mathrm{dom}\, A$), there is a view by B of the world ($u : W \twoheadrightarrow \mathrm{dom}\, B$), such that B has the abstraction capabilities of A, and all the functionality of A from A's point of view (see Fig. 2, Definition 2, and Definition 3).

Assumption. We want to allow the world-domain W to be as big as required, as well as the resolution of its elements to be enlarged as much as required. That is, all elements $x \in W$ may be considered as sets of a fixed cardinality.

3 The Formal Comparison Notion

We need to formalize the conceptual framework of the previous section.

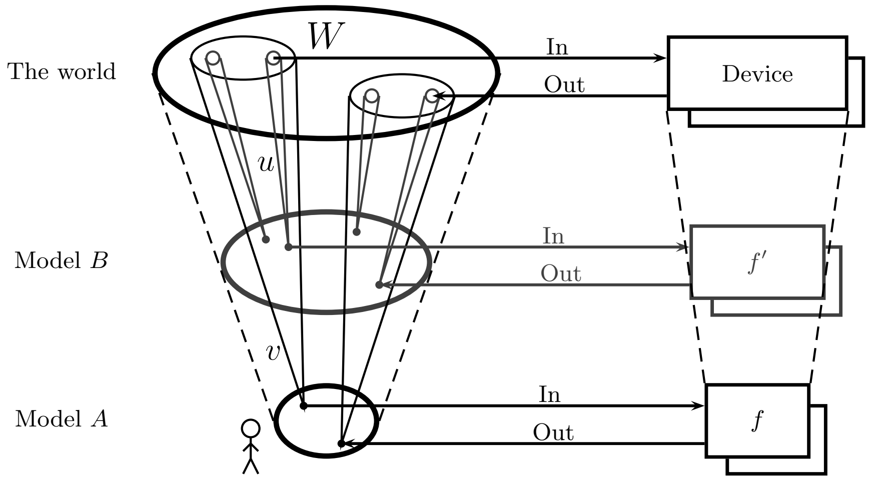

Fig. 2. The "stronger" model, B, should have the potential to provide all the functionality of the "weaker" model, A, from any user point of view

Definition 1 (Computational Model).

- *A* domain *is a nonempty set of elements.*
- *A* computational model *A over domain D is an object that computes a set of partial functions $f : D \mapsto D$, which may be interpreted as total functions $f : D \to D \cup \{\perp\}$.*
- *We write* dom A *for the domain over which model A operates.*
- *The* extensionality *of a model A, denoted* ext A, *is the set of partial functions that A computes.*
- *For models A and B, and a function f we shall write $f \in A$ as shorthand for $f \in \operatorname{ext} A$, and $A \subseteq B$ as short for $\operatorname{ext} A \subseteq \operatorname{ext} B$.*
- *We say that a model B* properly expands *model A if $B \supsetneq A$.*

Some clarifications regarding function notations:

- A (partial) function $f : D \mapsto D'$ can be extended to images of subsets of D, $f : \mathscr{P}(D) \mapsto \mathscr{P}(D')$, in the standard fashion: $f(X) := \{f(x) : x \in X\}$.
- A total surjective mapping $\rho : D \rightrightarrows D'$ is a total function, $\rho : D \to \mathscr{P}(D')$, from D to the subsets of D', such that $\bigcup_{x \in D} \rho(x) = D'$.

Directly formalizing the conceptual characterization of "as powerful" (see Fig. 2), we get the following:

Definition 2 (Conceptual Power Comparison). *Model B is* at least as powerful as *model A if for every domain W (the world) and view $v : W \twoheadrightarrow \operatorname{dom} A$, there are a view $u : W \twoheadrightarrow \operatorname{dom} B$ and abstraction-function $g \in B$, s.t.*

(a) for every function $f \in A$ there is a function $f' \in B$, s.t. $v \circ u^{-1} \circ f' \circ u \circ v^{-1}(x) = \{f(x)\}$ for all $x \in \operatorname{dom} A$,
(b) $g(z) = g(y)$ iff $v \circ u^{-1}(z) = v \circ u^{-1}(y)$ for all $y, z \in \operatorname{dom} B$, and
(c) $v \circ u^{-1} \circ g(y) = v \circ u^{-1}(y)$ for all $y \in \operatorname{dom} B$.

Here "view" is a partial surjective function, and (by the conceptual assumption) all elements $x \in W$ may be considered as sets of a fixed cardinality. The first condition, (a), says that B computes every function of A, up to the mapping between the domains ($v \circ u^{-1}$). Condition (b) says that the function $g \in B$ distinguishes between the equivalence classes generated by the mapping, while (c) says that the distinction is made by choosing a representative element within each class.

Definition 2 may be simplified, omitting the world-domain.

Definition 3 (Power Comparison Notion).

1. *Model B is* (computationally) at least as powerful *as model A, denoted $B \succsim A$, if there are a total surjective mapping $\rho : \operatorname{dom} B \rightrightarrows \operatorname{dom} A$ and function $g \in B$, such that:*
   (a) for every function $f \in A$ there is a function $f' \in B$ such that $\rho \circ f' \circ \rho^{-1}(x) = \{f(x)\}$ for all $x \in \operatorname{dom} A$,
   (b) $g(z) = g(y)$ iff $\rho(z) = \rho(y)$ for all $y, z \in \operatorname{dom} B$, and
   (c) $\rho \circ g(y) = \rho(y)$ for all $y \in \operatorname{dom} B$.
2. *Model B is* (computationally) more powerful *than A, denoted $B \succnsim A$, if $B \succsim A$ but $A \not\succsim B$.*
3. *Models A and B are* (computationally) equivalent *if $A \succsim B \succsim A$, in which case we write $A \approx B$.*

Proposition 1. *The computational power relation $\succsim$ between models is a quasi-order. Computational equivalence $\approx$ is an equivalence relation.*

Theorem 1. *Definitions 2 and 3.1 are equivalent. That is $B \succsim A$ by Definition 3.1 iff B is at least as powerful as A by Definition 2.*

Example 1. Consider a modeling of a simple electric-cable by a model EC, providing only the identity function over the reals. Then $\mathrm{TM} \not\succsim EC$ and $EC \not\succsim \mathrm{TM}$.

Inclusion of the Identity Function. When the "weak" model includes the identity function ($\lambda x.x$), the general comparison notion may be simplified, replacing the surjective mapping (ρ) by a surjective function. If the "stronger" model is closed under functional composition, it may be further simplified, replacing the surjective function with an opposite injection ($\psi : \operatorname{dom} A \rightarrowtail \operatorname{dom} B$). This is similar to the embedding notion (Definition 4 below) with the additional requirement for an abstraction function (g). Comparison via a surjective function resembles the "representation" of [15, p. 33], just that here we insist on a total function.

Theorem 2. *Let A be a computational model with the identity function ($\lambda x.x \in A$). Then a model B, closed under functional composition, is at least as powerful as A ($B \succsim A$) iff there exist an injection $\psi : \operatorname{dom} A \rightarrowtail \operatorname{dom} B$ and a total function $g \in B$ onto $\operatorname{rng} \psi$ ($g : \operatorname{dom} B \twoheadrightarrow \operatorname{rng} \psi$), such that for every function $f \in A$ there is a function $f' \in B$ such that $\psi \circ f(x) = f' \circ \psi(x)$ for all $x \in \operatorname{dom} A$.*

Example 2. Real recursive functions (Rrec) [10], are more powerful than Turing machines (TM). That is $\mathrm{Rrec} \succnsim \mathrm{TM}$. The comparison is done via the injection $\psi : \mathbf{N} \rightarrowtail \mathbf{R}$, where $\psi(n) = n$ [10, p. 18], and the floor function ($\lambda x. \lfloor x \rfloor$) to provide the abstraction capabilities of Rec (the above function g) [10, p. 10].

4 Ramifications of Familiar Notions

Various methods have been used to compare the computational power of competing models.

Extended Domains. It is common to claim that a function is incorporated in any of its extensions. That is, a function $f : D \to D$ is incorporated in $f' : D' \to D'$ if $D \subseteq D'$ and $f = f' \restriction_D$. See, for example, [4, p. 654]: "Here we adopt the convention that a function on $\mathbf{N}$ is in an analog class $\mathcal{C}$ if some extension of it to $\mathbf{R}$ is, i.e. if there is some function $\widetilde{f} \in \mathcal{C}$ that matches f on inputs in $\mathbf{N}$."

By the conceptual framework, "B extends A" can be interpreted as "B having the potential to be at least as powerful as A for a user who has both domain views." For example, one can consider a user who views the world as real numbers, but can identify the natural numbers among them.

This approach is not appropriate as a general power comparison notion, since the extended model B doesn't necessarily have the abstraction capabilities of A. For example, a mathematician working with paper and pencil may consider various physical entities to "be" the symbol 'a' (e.g. a, a, a, a, a). A model that lacks the abstraction of the various 'a's, treating each of them totally differently, is not as powerful.

Embedding. Extending the domain is a special case of embedding. A model B embeds A, if there is an injection from the domain of A to the domain of B, via which B has all the functionality of A *over the range of the injection.*

Definition 4 (Embedding). *A computational model B* embeds *a model A, denoted $B \gtrsim_{\mathrm{E}} A$, if there is an injection $\psi : \mathrm{dom}\ A \rightarrowtail \mathrm{dom}\ B$, s.t. for every function $f \in A$ there is $f' \in B$ such that $f' \circ \psi(x) = \psi \circ f(x)$ for all $x \in \mathrm{dom}\ A$.*

For example, Turing machines and the (untyped) λ-calculus were shown by Church [5], Kleene [8], and Turing [14] to embed the partial recursive functions.

The reasons for the inadequacy of embedding as a generic power comparison notion are analogous to that of domain-extending.

Example 3. Let RE be the recursively enumerable predicates over $\mathbf{N}$. RE may embed an expansion with infinitely many non-r.e. partial predicates $\{h_i\}$. Let

$$h(n) = \begin{cases} 0 & \text{program } n \text{ halts uniformly} \\ 1 & \text{otherwise} \end{cases} \qquad h_i(n) = \begin{cases} 0 & n < i \vee h(n) = 0 \\ \perp & \text{otherwise .} \end{cases}$$

We have that $\mathrm{RE} \gtrsim_{\mathrm{E}} \mathrm{RE} \cup \{h_i\}$, by an injection $\psi(n) = 2n + h(n)$, as

$$h'_i(n) = \begin{cases} 0 & \lfloor n/2 \rfloor < i \text{ or } n \bmod 2 = 0 \\ \perp & \text{otherwise} \end{cases} \qquad f' = \begin{cases} f(\lfloor n/2 \rfloor) & f \in RE \\ h'_i(n) & f = h_i \ . \end{cases}$$

where $f' \in \mathrm{RE}$ and $f' = \psi \circ f \circ \psi^{-1}$ for every $f \in \mathrm{RE} \cup \{h_i\}$. (Without loss of generality, we are supposing that $\psi(0) = h(0) = 0$.)

Effective Encoding. A common approach for comparing models over different domains is to require some manner of effectiveness of the encoding; see [6, p. 21] and [7, p. 290], for example. There are basically two approaches:

1. One can demand informal effectiveness: "The coding is chosen so that it is itself given by an informal algorithm in the unrestricted sense" [11, p. 27].
2. Or one can require encoding effectiveness via a specific model, say, Turing machines: "The Turing-machine characterization is especially convenient for this purpose. It requires only that the expressions of the wider classes be expressible as finite strings in a fixed finite alphabet of basic symbols" [11, p. 28].

By the conceptual framework, an "effective comparison" means that B is at least as powerful as A for a human user, assuming humans are capable of "effective" representations.

Effectivity is a useful notion; however, it is unsuitable as a general power comparison notion. The first, informal approach is too vague, while the second can add computational power when dealing with subrecursive models and is inappropriate when dealing with non-recursive models.

5 When is a Model More Powerful?

In general, the *strict part* $\gnsim^*$ of a quasi-order $\gtrsim^*$ is $\gtrsim^* \cap \not\lesssim^*$. That is, $B \gnsim^* A$ if $B \gtrsim^* A$ but not $A \gtrsim^* B$.

The Common Method. Intuitively, one would expect that a proper expansion of a model (additional functions) is also more powerful, that is, for $B \supsetneq A$ to imply $B \gnsim A$. For example, a model that computes more than Turing machines is considered more powerful (see, e.g., [12]). Hence, the common method of showing that a model B is more powerful than model A, for some comparison notion $\gtrsim^*$, is to show that $B \gtrsim^* C \supsetneq A$.

The Problem. Unfortunately, a proper expansion of a model is not necessarily more powerful. That is, $B \supsetneq A$ does not imply $B \gnsim^* A$, where $\gnsim^*$ may be embedding, our suggested notion, or "containment up to isomorphism" (Theorem 3).

Example 4. Define the set R_2 of "even" recursive functions (Rec):

$$R_2 = \left\{ \lambda n. \begin{cases} 2f(n/2) & n \text{ is even} \\ n & \text{otherwise} \end{cases} : f \in \text{Rec} \right\}$$

R_2 embeds all the recursive functions via the injection $\lambda n.2n$, though $R_2 \subsetneq \text{Rec}$.

See also Example 3, for the embedding of non-r.e. predicates in RE.

Note that the common comparison method (see above) permits a model to be more powerful than itself! For example, one might say that "$R_2 \gnsim_{\text{E}} R_2$," since $R_2 \gtrsim_{\text{E}} \text{Rec} \supsetneq R_2$.

Theorem 3. *There are models isomorphic to proper expansions of themselves. That is, there is a set of functions M over a domain D, and a bijection $\pi : D \rightarrowtail\!\!\!\!\twoheadrightarrow D$, s.t. $\{\pi \circ f \circ \pi^{-1} : f \in M\} \supsetneq M$.*

The Solution. The general solution is to use the strict part of the quasi-order. For example, with embedding one should show that "B may embed A, while there is no injection via which A may embed B."

In addition, one can check whether a specific model is "complete" in the sense that it is not equivalent (with respect to the relevant notion) to any of its proper expansions. For complete models, the common (generally improper) method is suitable, saving the necessity of precluding all possible mappings.

Definition 5 (Complete Models). *Let $\succsim^*$ be a quasi-order (comparison notion). A computational model A is* complete*, with respect to $\succsim^*$, if $A \succsim^* B \supseteq A$ implies $A = B$ for all B.*

Proposition 2. *Let $\succsim^*$ be a quasi-order (comparison notion), and A a complete model w.r.t. $\succsim^*$. Then $B \gnsim^* A$ iff there is a model C, such that $B \succsim^* C \supsetneq A$.*

Theorem 4. *Let A be a model with the identity function, closed under function composition, and complete w.r.t. to embedding ($\succsim_{\mathrm{E}}$), then A is complete w.r.t. power-comparison ($\succsim$).*

Corollary 1. *Let A be a model with the identity function, closed under function composition, and complete w.r.t. to embedding. Then a model B is more powerful than A iff B is at least as powerful as A and embeds some proper expansion C of A. That is, $B \gnsim A$ iff there is a model C s.t. $B \succsim A \subsetneq C \precsim_E B$.*

6 Computability

Some computational models considered to be of equivalent power to Turing machines are still so according to our suggested comparison notion (Definition 3).

Theorem 5. *Turing machines* (TM)*, the recursive functions* (Rec)*, counter machines* (CM)*, and random access machines* (RAM) *are all of the same computational power. That is,* TM $\approx$ Rec $\approx$ CM $\approx$ RAM.

6.1 Hypercomputation

In Section 5, we saw that a proper expansion of a computational model is not necessarily more powerful (by any of the common comparison methods). What does this mean for hypercomputation? Can it be that Turing machines are as powerful as a model that computes additional functions?

We prove that Turing machines and the recursive functions are complete models, thus are not susceptible to such an anomaly. Accordingly, we provide some means to show that a model is hypercomputational.

Definition 6 (Hypercomputation). *A model A is* hypercomputational *if it is more powerful than Turing machines, that is, if $A \gnsim$* TM.

Theorem 6. *The recursive functions* (Rec) *and the partial recursive functions* (PR) *are complete w.r.t. embedding.*

Theorem 7. *Turing machines* (TM) *are complete w.r.t. embedding.*

Corollary 2. *Model A is hypercomputational if any one of the following conditions is satisfied:*

1. $A \gnsim \mathrm{TM}$.
2. $A \supsetneq \mathrm{TM}$.
3. *There is a model C, such that* $A \gtrsim C \supsetneq \mathrm{TM}$.
4. *There is a model C, such that* $A \gtrsim_{\mathrm{E}} C \supsetneq \mathrm{TM}$ *and also* $A \gtrsim \mathrm{TM}$.

References

1. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer-Verlag, New York, 1998.
2. O. Bournez, F. Cucker, P. J. de Naurois, and J.-Y. Marion. Computability over an arbitrary structure. sequential and parallel polynomial time. *FoSSaCS*, pages 185–199, 2003.
3. J. P. Bowen. Glossary of Z notation. *Information and Software Technology*, 37(5–6):333–334, May–June 1995. Available at: `http://staff.washington.edu/~jon/z/glossary.html`.
4. M. L. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16:642–660, 2000.
5. A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.
6. E. Engeler. *Formal Languages: Automata and Structures.* Lectures in Advanced Mathematics. Markham Publishing Company, Chicago, IL, 1968.
7. F. Hennie. *Introduction to Computability.* Addison-Wesley, Reading, MA, 1977.
8. S. C. Kleene. Lambda-definability and recursiveness. *Duke Mathematical Journal*, 2:340–353, 1936.
9. M. L. Minsky. Matter, mind and models. *Proc. International Federation of Information Processing Congress*, 1:45–49, 1965. Available at `http://web.media.mit.edu/~minsky/papers/MatterMindModels.html`.
10. J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 2004. In print.
11. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1966.
12. H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit.* Birkhäuser, Boston, 1998.
13. J. V. Tucker and J. I. Zucker. Abstract versus concrete computation on metric partial algebras. *ACM Transactions on Computational Logic*, 5(4):611–668, 2004.
14. A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936–37.
15. K. Weihrauch. *Computable Analysis — An introduction.* Springer-Verlag, Berlin, 2000.