The Influence of Domain Interpretations on Computational Models ⋆

Udi Boker and Nachum Dershowitz
School of Computer Science, Tel Aviv University
Ramat Aviv, Tel Aviv 69978, Israel

Abstract

Computational models are usually defined over specific domains. For example, Turing machines are defined over strings, and the recursive functions over the natural numbers. Nevertheless, one often uses one computational model to compute functions over another domain, in which case, one is obliged to employ a representation, mapping elements of one domain into the other. For instance, Turing machines (or modern computers) are understood as computing numerical functions, by interpreting strings as numbers, via a binary or decimal representation, say.

We ask: Is the choice of the domain interpretation important? Clearly, complexity is influenced, but does the representation also affect computability? Can it be that the same model computes strictly more functions via one representation than another? We show that the answer is "yes", and further analyze the influence of domain interpretation on the extensionality of computational models (that is, on the set of functions computed by the model).

We introduce the notion of *interpretation-completeness* for computational models that are basically unaffected by the choice of domain interpretation, and prove that Turing machines and the recursive functions are interpretation-complete, while two-counter machines are incomplete. We continue by examining issues based on model extensionality that are influenced by the domain interpretation. We suggest a notion for comparing computational power of models operating over arbitrary domains, as well as an interpretation of the Church-Turing Thesis over arbitrary domains.

Key words: domain interpretation, domain representation, hypercomputation, Turing machine, computability, computational power, computational models, computational comparison

1. Introduction

We explore the problem of the sensitivity of models to domain interpretation, and the way we propose to handle it. This introductory section parallels the structure of the paper, as illustrated in Figure 1.

Sensitivity to domain interpretation. A computational model is defined over a specific domain. However, we may often use it to compute functions over a different domain. For example, using Turing machines (or modern computers) to compute functions over natural numbers requires a string representation by numbers. Another example becomes apparent when comparing the computational power of two models that operate

⋆ This work was carried out in partial fulfillment of the requirements for the Ph.D. degree of the first author. Research was supported in part by the Israel Science Foundation under grant no. 250/05.

Email address: {udiboker,nachumd}@tau.ac.il (Udi Boker and Nachum Dershowitz).

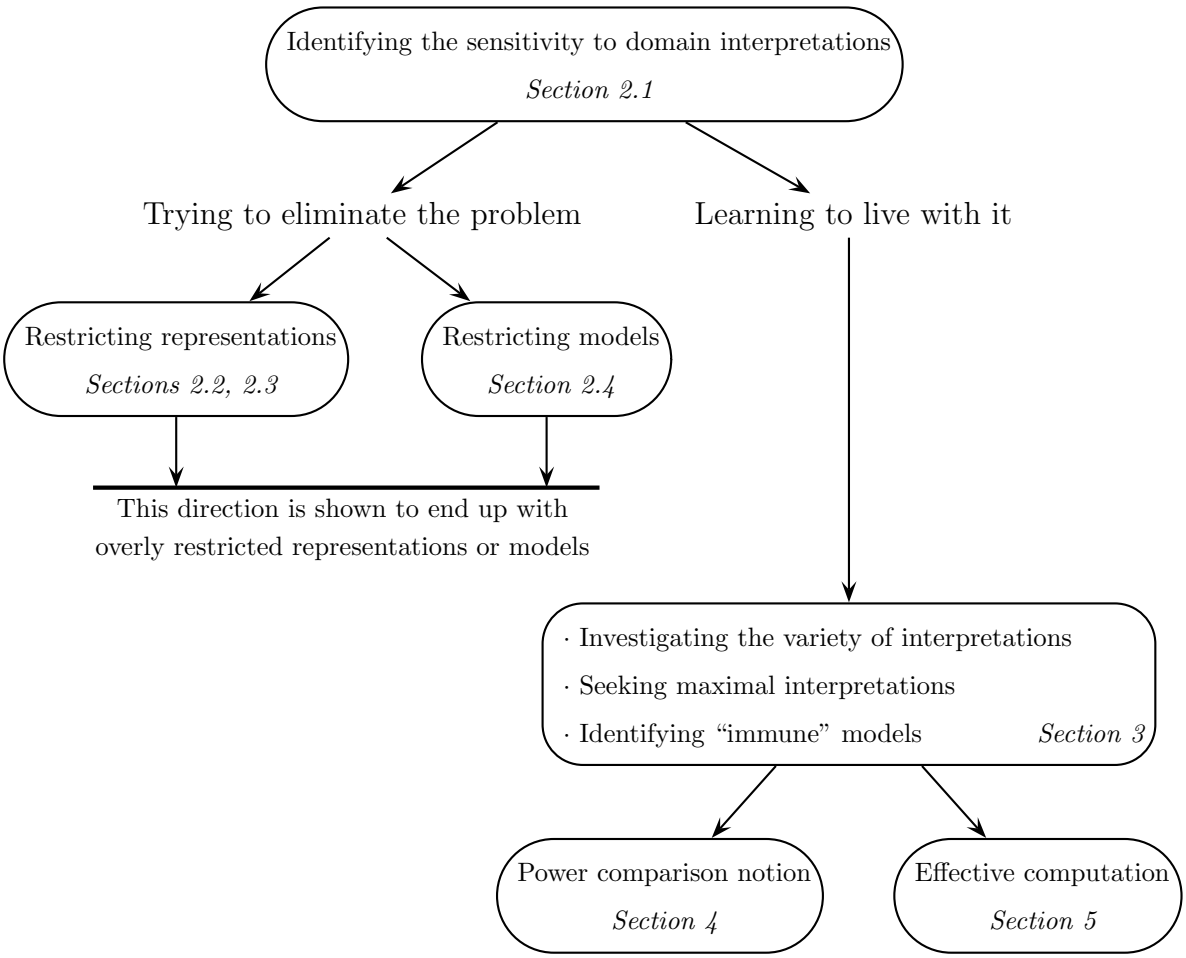


Fig. 1. Structure of this paper

over different domains – we are obliged to represent the domain of one model by the domain of the other. Accordingly, the elements of the original domain are interpreted as elements of the target domain (see illustration in Figure 3). A "representation" is usually allowed to be any mapping from one domain into another, as long as it is injective. That is, every original domain element is mapped to a unique element of the second domain. [1]

It turns out that interpreting the domain allows for the possibility that a model be identified with one of its (strict) supermodels. The interpretation might allow one to "enlarge" the extensionality of a model, adding some "new" functions to it. A study of the sensitivity of models to interpretations via injective representations is undertaken in Section 2.1.

A reasonable response might be to restrict representations to bijections between domains. However, it turns out that there are models that can be identified with a supermodel even with bijective representations. That is, their extensionality is isomorphic to the extensionality of some of their strict supermodels. The case of bijective representations is explored in Section 2.2.

If bijective representations are not stringent enough, which representations are guaranteed not to influence the extensionality of computational models? It turns out that only very limited representations are, namely, "narrow permutations" (Definition 7). A sufficient and necessary criterion for these "harmless representations" is given in Section 2.3. Not only are narrow permutations a very limited family of representations,

[1] A representation may also be a relation (rather than a function), as long as two entities representing the same element behave the same in the relevant context (see, for example, [26]). The generalization of representations to relations does not eliminate the sensitivity of models to the domain interpretation.

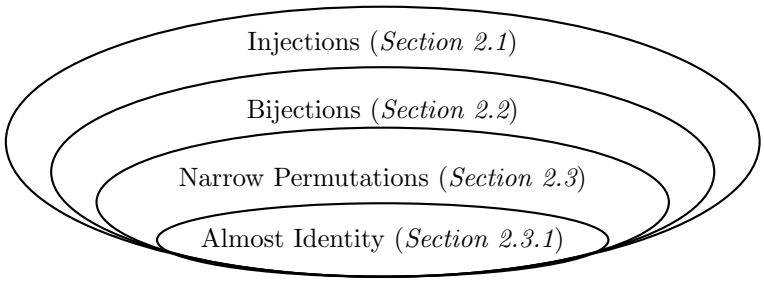


Fig. 2. The hierarchy of mappings involved in seeking harmless representations.

they are also not closed under composition. Thus, seeking harmless representations for comparative purposes would lead to an even more limited family of permutations that are almost the identity (Definition 10). Hence, sticking to harmless representations is not a viable option, as it almost completely evaporates the concept of interpretations. A scheme of the families of representations involved is given in Figure 2.

Another direction for avoiding the influence of representations could be narrowing down the definition of a computational model, insisting – for example – that the set of computed functions is closed under functional composition. It turns out that these standard computational properties are insufficient, as shown in Section 2.4.

The implications. This sensitivity to the domain interpretation places a question mark on some of the main issues that concern model extensionality: How can we compare models over different domains? How should one define effective computation over arbitrary domains? How should one properly represent the natural numbers? Is there always an optimal representation? Are some models immune to the influence of the domain interpretation? These problems are briefly answered below, and more comprehensively addressed in Sections 3, 4 and 5.

Organizing model interpretations. Generally, the various interpretations of a model may be highly varied: they may be larger or smaller than the original; for some models there are maximal interpretations, while for others there are none; and there are models that are already maximal. This variety of possibilities is explored in Section 3.1.

The last property, that a model is already maximally interpreted, is the one that interests us most. We call such a model "interpretation-complete". We also define a weaker property, denoted "interpretation-stable", saying that a model is maximal with respect to bijective representations. When allowing only bijective representations, there are exactly two ways in which the interpretation may influence the model's extensionality: stable models are totally immune, in the sense that they have no better or worse interpretations via bijective representations, while unstable models have no maximum, nor minimum, interpretation via bijective representations (see the illustration in Figure 7). When allowing non-bijective representations, the picture is different – there might be a complete model with interpretations that are strictly contained in its original extensionality (see Figure 8). Interpretation-completeness and interpretation-stability, as well as some means for getting maximal interpretations, are investigated in Section 3.2.

In Section 3.3, we check for the completeness of some standard models. Turing machines and the recursive functions are shown to be complete, while two-counter machines and the untyped lambda calculus (over all lambda terms) are incomplete. As for hypercomputational models, they might be incomplete, though those preserving the closure properties of the recursive functions are ensured to be interpretation-complete.

Comparing computational power. It is common practice to compare the computational power of different models of computation. For example, the recursive functions are considered to be strictly more powerful than the primitive recursive functions, because the latter are a proper subset of the former, which includes Ackermann's function (see, for example, [22, p. 92]). Side-by-side with this "containment" method of measuring power, it is also standard to base comparisons on interpretations over different domains [7,18,21].

For example, one says that the (untyped) lambda calculus is as powerful—computationally speaking—as the partial recursive functions, because the lambda calculus can compute all partial recursive functions by encoding the natural numbers as Church numerals.

The problem is that unbridled use of these two distinct ways of comparing power allows one to show that some computational models are *strictly* stronger than themselves!

We define a comparison notion over arbitrary domains based on model interpretations. With this notion, model B is strictly stronger than A if B has an interpretation that contains A, whereas A cannot contain B under *any* interpretation. We provide, in Section 4.1, three variants of the comparison notion, depending on the allowed interpretations. In Sections 4.1.1 we extend the notion to non-deterministic models. We continue, in Section 4.2, with some results on the relations between power comparison, isomorphism and completeness. In Section 4.3, we use the notion to compare some standard models.

We deal here only with the mathematical aspects of power comparison. Some conceptual discussions and justifications can be found in [2,4].

Effective computation. Let f be some decision function (a Boolean-valued function) over an arbitrary countable domain D. What does one mean by saying that "f is computable"? One most likely means that there is a Turing machine M, such that M computes f, *using some string representation of the domain* D. But what are the allowed string representations? Obviously, allowing an arbitrary representation (any injection from D to Σ^*) is problematic – it will make any decision function "computable". For example, by permuting the domain of machine codes, the halting function can morph into the simple parity function, which returns true when the input number is even, representing a halting machine, and false otherwise. Thus, under a "strange" representation the function becomes eminently "computable" (see Section 5.1). Another approach is to allow only "natural" or "effective" representations. However, in the context of defining computability, one is obliged to resort to a vague and undefined notion of "naturalness" or of "effectiveness", thereby defeating the very purpose of characterizing computability.

Our approach to overcoming the representation problem is to ask about effectiveness of a set of functions over the domain of interest, rather than of a single function (Section 5.1). As Myhill observed [15], undecidability is a property of *classes* of problems, not of individual problems. In this sense, the halting function is undecidable in conjunction with an interpreter (universal machine) for Turing machine programs that uses the same representation. The Church-Turing Thesis, interpreted accordingly, asserts that there is no effective computational model that is more inclusive than Turing machines.

Nonetheless, there might have been a serious problem due the sensitivity of models to the domain interpretation (see Section 2). Fortunately, this cannot be the case with Turing machines (nor with the recursive functions), as they are interpretation-complete (Theorem 24). Hence, the Church-Turing Thesis is well-defined for arbitrary computational models.

Due to this completeness of Turing machines, we can also sensibly define what it means for a string representation of a constructible domain to be "effective" (Section 5.2). Such a representation ρ is effective when the domain's constructing functions are Turing computable via ρ (Definition 48). Hence, one may ask about the effectiveness of a single function over a constructible domain, provided that the means of construction of the domain are defined and are computable.

Equipped with a plausible interpretation of the Church-Turing Thesis over arbitrary domains, one may investigate the general class of "effective computational models". This is done in [5].

Previous work. Usually, the handling of multiple domains in the literature is done by choosing specific representations, like Gödel numbering, Church numerals, unary representation of numbers, etc. This is also true of the usual handling of representations in the context of the Church-Turing Thesis.

A more general approach for comparing the power of different computational models is to allow any representation based on an injective mapping between their domains. This is done, for example, by Rogers [18, p. 27], Sommerhalder [21, p. 30], and Cutland [7, p. 24]. A similar approach is used for defining the effectiveness of an algebraic structure by Froehlich and Shepherdson [8], Rabin [16], and Mal'cev [11]. Our notion of comparing computational power is very similar to this.

Richard Montague [13] raises the problem of representation when applying Turing's notion of computability to other domains, as well as the circularity in choosing a "computable representation".

Stewart Shapiro [20] raises the very same problem of representation when applying computability to number-theoretic functions. He suggests a definition of an "acceptable notation" (string representation of natural numbers), based on some intuitive concepts. We discuss his notion in Section 5.1 and 5.2.

Klaus Weihrauch [25,26] deals heavily with the representation of arbitrary domains by numbers and strings. He defines computability with respect to a representation, and provides justifications for the effectiveness of the standard representations. We elaborate on his justifications in Section 5.2.

Our definition of an "effective representation" resembles Shapiro's notion of "acceptable notation" and goes along the lines of Weihrauch's justifications for the effectiveness of the standard representations.

To the best of our knowledge, our work in [2–4] was the first to point out and handle the possible influence of the representation on the extensionality of computational models. Sections 2, 3, and 4 organize the main results of these papers, while adding some new ones, particularly Proposition 14, Proposition 17, Theorem 21, Theorem 25, Theorem 27, and Theorem 46. Section 5 summarizes the first part of our paper [5].

Terminology. We refer to the natural numbers, denoted $\mathbb{N}$, as including zero, and denote by $\mathbb{N}^+$ the natural numbers excluding zero. When we speak of the recursive functions, denoted $\mathbb{REC}$, we mean the partial recursive functions. Similarly, the set of Turing machines, denoted TM, includes both halting and non-halting machines. We use the term "domain" of a computational model and of a (partial) function to denote the set of elements over which it operates, not only those for which it is defined. By "image", we mean the values that a function actually takes: $\text{Im } f := \{f(x) \mid x \in \text{dom } f\}$.

Proofs are omitted for conciseness reasons.

2. The Sensitivity of Models to the Domain Interpretation

Computational models. Our research concerns computational models. Obviously, a computational model should perform some computation; however demarcating a clear border between what is a computational model and what is not is problematic. Accordingly, for achieving maximum generality, we do not want to limit computational models to any specific mechanism; hence, we allow a model to be any object, as long as it is associated with a set of functions that it computes.

As models may have non-terminating computations, we deal with sets of partial functions. For convenience, we assume that the domain and range (co-domain) of functions are identical. For simplicity, we mainly deal with deterministic computational models. Most of our definitions and theorems can be directly extended to non-determinism, while the more involved ones are handled in Section 4.1.1.

Definition 1 (Computational Model)

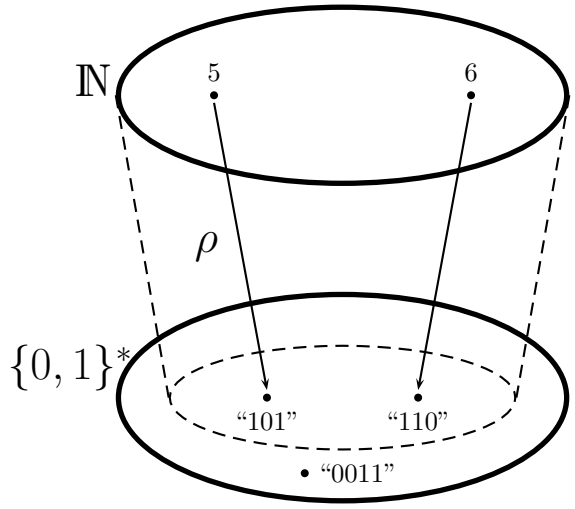
- *A* domain *D is any set of atomic elements.*
- *A* computational model *A over domain D is any object associated with a set of partial functions $f : D^i \to D$, $i \in \mathbb{N}^+$. This set of functions is called the* extensionality *of the computational model, denoted $[\![A]\!]$.*
- *We write* dom *A for the domain over which model A operates.*

In what follows, we often speak of a "submodel" or a "supermodel" of a model, referring to the containment relation between their extensionalities. That is, model A is a *submodel* of B, and B is a *supermodel* of A, if $[\![A]\!] \subseteq [\![B]\!]$. By a "strict submodel" we mean that the containment is proper.

In the following subsections, we explore the sensitivity of models to domain interpretations, ending up with a sufficient and necessary condition for a "harmless representation" (see Figure 2).

2.1. *Injective representations*

Injective representations are the most frequently used ones. The standard decimal and binary notations of the natural numbers are injective (they are not bijective since leading zeros are ignored). Comparisons between computational models are usually done by injective encodings; for example: Church numerals and Gödel numbering are used for comparing the recursive functions and λ-calculus.



$\rho(5) =$"101"

5 is *represented* by "101" via ρ

$[\![$"101"$]\!]_\rho = 5$

"101" is *interpreted* as 5 via ρ

"0011" $\notin \mathrm{Im}\ \rho$

"0011" has no *interpretation* via ρ

Fig. 3. Domain interpretation. Strings are interpreted as natural numbers via the representation ρ.

We begin by defining "representation" to be an injective mapping.

Definition 2 (Representation)

Domain. *Let D_A and D_B be two domains (arbitrary sets of atomic elements). A* representation *of D_A over D_B is an injection $\rho : D_A \to D_B$ (i.e. ρ is total and one-one).*

Function and Relation. *Representations ρ naturally extend to functions and relations f, which are sets of tuples of domain elements: $\rho(f) := \{\langle \rho(x_1), \ldots, \rho(x_n)\rangle \mid \langle x_1, \ldots, x_n\rangle \in f\}$.*

Model. *Representations also naturally extend to (the extensionalities of) computational models, which are sets of functions: $\rho([\![B]\!]) := \{\rho(f) \mid f \in [\![B]\!]\}$.*

An almost dual concept to representation is "interpretation" (see Figures 3 and 4):

Definition 3 (Interpretation) *Assume a representation $\rho : D_A \to D_B$. Then:*

(i) The interpretation of a domain element $b \in D_B$ *via the representation ρ, denoted $[\![b]\!]_\rho$, is the element $\rho^{-1}(b)$ of D_A. If $b \notin \mathrm{Im}\ \rho$ then its interpretation via ρ is undefined.*

(ii) The interpretation of a function g *over D_B via the representation ρ, denoted $[\![g]\!]_\rho$, is the function $\rho^{-1}(g)$ over D_A, which is $\rho^{-1} \circ g \circ \rho$. If $g \circ \rho(a) \notin \mathrm{Im}\ \rho$ for some element $a \in D_A$ then $[\![g]\!]_\rho$ is a partial function, where $[\![g]\!]_\rho(a)$ is undefined.*

(iii) The interpretation of a computational model B *via the representation ρ, denoted $[\![B]\!]_\rho$, is the set of functions $\rho^{-1}([\![B]\!])$, which is $\{[\![g]\!]_\rho \mid g \in [\![B]\!]\}$.*

(iv) When considering only total functions, the interpretation of a total computational model B *via the representation ρ, denoted $[\![B]\!]_\rho$, is the set of functions $\{[\![g]\!]_\rho \mid g \in [\![B]\!]$ and $[\![g]\!]_\rho$ is total$\}$.*

"Interpretation via ρ" is the reverse of "representation via ρ", up to the image of ρ. When the representation ρ is bijective we have that "interpretation via ρ" is exactly as "representation via ρ^{-1}."

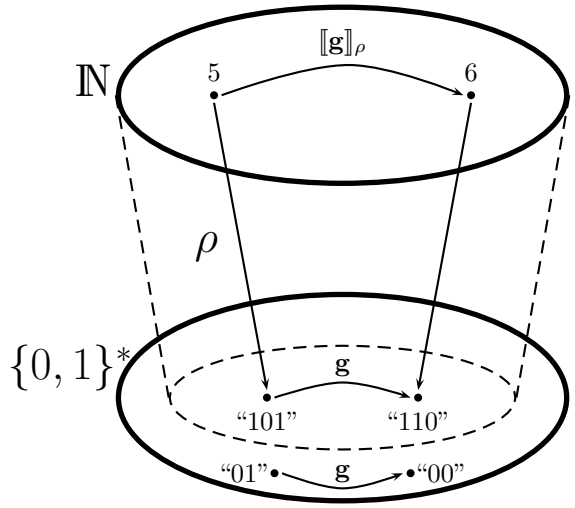
Interpretation and extensionality share the same notation. Indeed, the interpretation of some model B via a representation $\rho : D_A \to D_B$ is its extensionality over the domain D_A, which results from the representation ρ. Note that the extensionality $[\![A]\!]$ of a model A is its interpretation $[\![A]\!]_\iota$ via the identity representation ι.

Sensitivity. Injective representations, however, are prone to hide some computational power. Below is a simple example of such a case.

Example 4 *The set of "even" recursive functions (R_2) can be interpreted as the set of all the recursive functions ($\mathbb{RREC}$), by mapping the original natural numbers into the even numbers*

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

We have that $[\![R_2]\!]_\rho = \mathbb{RREC} \supsetneq R_2$.



The function g is interpreted as $[\![g]\!]_\rho$

$[\![g]\!]_\rho = \rho^{-1} \circ g \circ \rho$

$[\![g]\!]_\rho(5) = 6$

The behavior of g outside of Im ρ does not influence its interpretation via ρ

Fig. 4. Function interpretation. The string functions are interpreted as numeral functions via the representation ρ.

The above anomaly does not appear only with "synthetic" models, but also with some standard ones. An example of such a model is the standard two-counter machines model (see Section 3.3.3).

2.2. *Bijective representations*

The previous subsection demonstrated the sensitivity of models to injective representations. One may ask whether the restriction of representations to bijective mappings might solve the problem. We show that the answer is "no", obtaining that a model might be isomorphic to some of its strict supermodels.

Definition 5 (Isomorphism) *Models A and B (or their extensionalities) are* isomorphic*, denoted $A \cong B$ (or $[\![A]\!] \cong [\![B]\!]$), if there is a bijection π such that $[\![A]\!]_\pi = [\![B]\!]$.*

Theorem 6 ([3]) *There are models isomorphic to a strict supermodel of themselves. That is, there are models A and B, such that $A \cong B$ and $[\![A]\!] \subsetneq [\![B]\!]$.*

A concrete example of such models is given in Example 11 and in [3].

It will be shown, in Section 3, that this process is infinite and symmetric (Theorem 12) – once the model is sensitive to bijective representations, we can always choose a different representation via which we get more, or fewer, functions.

2.3. *Harmless representations*

Are there "harmless representations", via which all models are "protected" from having better and worse interpretations? The answer is "yes", however this family of representations is too limited for being really useful. It is exactly the family of what we call "narrow" permutations:

Definition 7 (Narrow Permutation) *A permutation $\pi : D \to D$ is* narrow *if all its orbits (cycles) are bounded in length by some constant. More precisely, if $\exists k \in \mathbb{N}.\ \forall x \in D.\ |\{\pi^n(x) : n \in \mathbb{N}\}| \leq k$.*

Proposition 8 *A permutation $\pi : D \to D$ is narrow iff there is a positive constant $k \in \mathbb{N}^+$, such that for all $x \in D$ we have $\pi^k(x) = x$. In other words, if $\pi^k = \iota$.*

Theorem 9 ([3]) *For every representation $\rho : D \to D$, there are models A and B such that $[\![A]\!]_\rho = [\![B]\!] \supsetneq [\![A]\!]$, if and only if ρ is not a narrow permutation.*

The family of narrow permutations is very limited and cannot be used as the only mean of interpreting models over different domains. Moreover, this family is not closed under composition. That is, there are narrow permutations π and η such that $\pi \circ \eta$ is not narrow! This situation is very problematic, since interpretations are often used in the context of order relations, for example when saying that two models have the

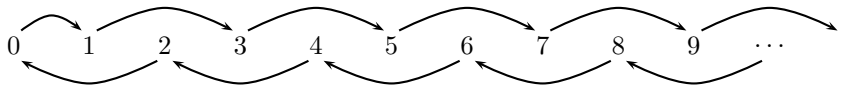


Fig. 5. The permutation π of Example 11.

same extensionality up to the domain interpretation. Any equivalence or order relation should be transitive, which is not the case if representations are limited to narrow permutations.

2.3.1. *Purely harmless representations*

Proceeding in the above direction of seeking a family of representations that would be harmless and closed under functional composition leads us to look for some strict subset of the narrow permutations. However, considering that there are many such (maximal) subsets, which is a reasonable choice?

It turns out that there is a clear distinction between two types of narrow permutations – the "problematic" and the "non-problematic" ones. For every problematic narrow permutation ρ there is a narrow permutation η, such that $\rho \circ \eta$ is not narrow. On the other hand, a non-problematic narrow permutation π guarantees that for every narrow permutation ξ we have that $\xi \circ \pi$ and $\pi \circ \xi$ are narrow.

The family of non-problematic narrow permutations is the "almost identity" permutations (defined bellow), while the rest are problematic.

Definition 10 (Almost Identity) *A permutation $\pi : D \to D$ is* almost identity *if $|\{x \in D \mid \pi(x) \neq x\}| < \infty$.*

The above results suggest that sticking to harmless representations is not a viable direction, as the concept of interpreting models of different domains almost completely evaporates.

2.4. *Models with standard computational properties*

It was shown above that restricting the family of applicable representations cannot solve the sensitivity problem. A different approach is to restrict the definition of a computational model. Nonetheless, in order to allow a variety of internal mechanisms, we seek a restriction on the model's extensionality. We consider four restrictions: (i) closure under functional composition; (ii) inclusion of the identity function; (iii) inclusion of all constant functions; and (iv) the successor function for models operating over $\mathbb{N}$.

In this section we show that the sensitivity problem remains, even when considering only models with all the above properties and allowing only bijective representations.

Closure under functional composition. Denote by $cl(\mathcal{F})$ the set $\mathcal{F}$ of functions closed under functional composition. Considering only models closed under functional composition does not change the sufficient and necessary condition for a harmless representation (Theorem 9), as all models involved in the proof are closed under functional composition.

The identity and constant functions. Adding, or removing, the identity function ι from a model has no influence on its sensitivity to any representation, as $\rho^{-1} \circ (f \circ \iota) \circ \rho = \rho^{-1} \circ (\iota \circ f) \circ \rho = \rho^{-1} \circ f \circ \rho$, for every injection ρ and function f.

Let K be the set of all constant functions over a domain D. Adding, or removing, K from a model A over D, such that $A \cap K \in \{K, \emptyset\}$, has no influence on the sensitivity of A, as $[\![A \circ K]\!]_\rho = [\![K \circ A]\!]_\rho = K$, with respect to total functions, for every injection ρ and model A.

The successor function. It turns out that a model including the successor function and closed under functional composition can still be isomorphic to a strict supermodel of itself:

Example 11 *Define the permutation π over $\mathbb{N}$ (illustrated in Figure 5):*

$$\pi(n) := 1 \text{ if } n = 0; \quad n+2 \text{ if } n \text{ is odd}; \quad \text{and} \quad n-2 \text{ if } n \text{ is even}$$

Let s be the successor function over $\mathbb{N}$, and let A be a computational model with the extensionality $[\![A]\!] := \{\pi^i(s) \mid i \in \mathbb{N}\}$. Let B be the computational model obtained from A by closure under functional composition. That is, $[\![B]\!] := cl([\![A]\!])$. It can be shown that B is isomorphic to a strict supermodel of itself.

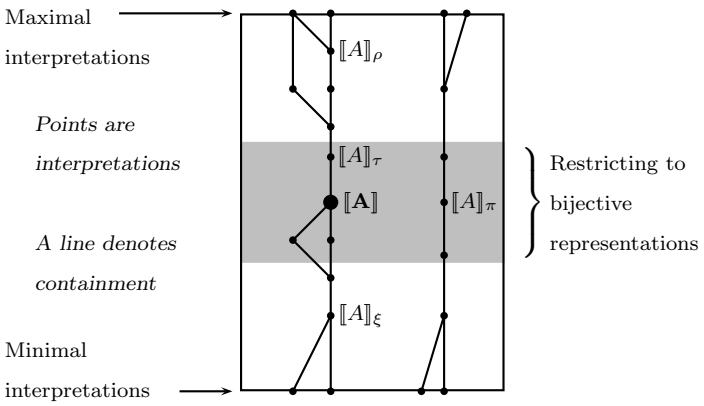


Fig. 6. An illustration of the partially ordered set of interpretations of a model A

3. Organizing Model Interpretations

For examining the influence of the domain interpretation on a model we should compare its different interpretations. Accordingly, we are interested only in the model's interpretations over its original domain. Its interpretations over other domains are isomorphic to those over its original domain, as long as it is not a domain of a lower cardinality. For example, a model has two interpretations with strict containment between them if and only if it has such two interpretations over its original domain.

We are interested in the containment relation between interpretations. That is, examining when some interpretations are better than others in the sense of strictly containing them. Accordingly, the interpretations of a model A form a partially ordered set with respect to containment (illustrated in Figure 6).

Viewing interpretations as a partially ordered set, raises a few natural questions:
– How varied can these partially ordered sets be?
– Are there always maximal interpretations?
– Are there models already in their maximal interpretation (termed "interpretation-complete")?
– How does one choose a proper interpretation?

In what follows we shall shed some light on the subject, considering the above questions and others. In Section 3.1 we answer the first two questions, showing how varied the set of interpretations can be. In Section 3.2 we answer the second pair of questions, dealing with the interpretation-completeness of models.

3.1. *The variety of interpretations*

In general, the set of interpretations may be very varied:
– Some interpretations may be better than the original extensionality while others are worse.
– There might be infinitely many interpretations each contained in the next.
– There might be models with no maximal interpretation!
– Non-bijective interpretations might sometimes add to bijective ones, while in other cases only spoil.
– There are models already having their maximal interpretation ("interpretation-complete") or at least so with respect to bijective representations ("interpretation-stable").

A simple example of how different interpretations may enlarge or decrease the original extensionality is given in Example 4. Interpreting the model via the representation $\lambda n.2n$ enlarges the original extensionality, providing all the recursive functions. On the other hand, interpreting the model via the representation $\lambda n.2n+1$ decreases the original extensionality, leaving only the identity function.

We saw, in Section 2.2, that a model can be isomorphic to a strict supermodel of itself. In such a case, there are infinitely many interpretations enlarging the original extensionality, as well as infinitely many decreasing it, while each is contained in the next. Note that this is true for bijective representations, but not necessarily for injective representations.

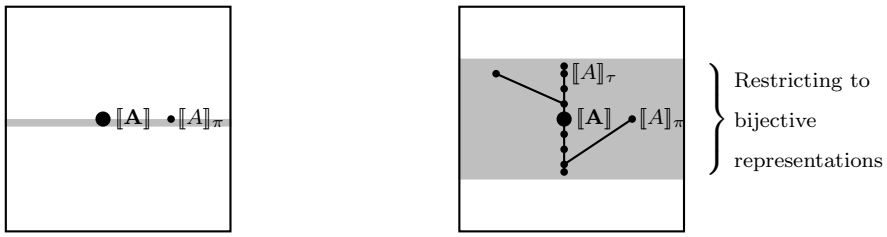


Fig. 7. An illustration of interpretation-stable and unstable models

Theorem 12 *If A is a model and π a bijection such that $[\![A]\!] \subsetneq [\![A]\!]_\pi$, then for every $i \in \mathbb{N}$ we have that $[\![A]\!]_{\pi^i} \subsetneq [\![A]\!]_{\pi^{i+1}}$ and $[\![A]\!]_{\pi^{-i}} \supsetneq [\![A]\!]_{\pi^{-(i+1)}}$.*

Corollary 13 *If A is a model for which there is no bijection π such that $[\![A]\!] \subsetneq [\![A]\!]_\pi$, then there is also no bijection η such that $[\![A]\!] \supsetneq [\![A]\!]_\eta$.*

We see that once a model has a better interpretation via a bijective representation it cannot have a maximal interpretation via a bijective representation. Nevertheless, it might have a maximal interpretation via an injective representation. There are, however, models with no maximal interpretation at all.

Proposition 14 *There are computational models with no maximal interpretation that extends their original extensionality. That is, there is a computational model A, such that for every representation ρ for which $[\![A]\!]_\rho \supseteq [\![A]\!]$ there is a representation η such that $[\![A]\!]_\eta \supsetneq [\![A]\!]_\rho$.*

We can show that model A of Example 11 is such a model. We also get that interpretations via non-bijective representations might sometimes only decrease the model's extensionality, while we saw that in other cases they can further enlarge it on top of bijective ones.

3.2. *Interpretation-completeness and interpretation-stability*

We saw that the extensionality of computational models is sensitive to the domain interpretation. There are, however, models that are already in their maximal interpretation (called "interpretation-complete" or in short "complete"), or at least so with respect to bijective representations (called "interpretation-stable" or in short "stable").

Definition 15 *A model A is* interpretation-complete *if there is no representation $\rho : \mathrm{dom}\ A \to \mathrm{dom}\ A$ such that $[\![A]\!]_\rho \supsetneq [\![A]\!]$.*

Note that when considering only total functions, the interpretation of a model is also defined to contain only total functions. In such a case a model is considered complete even if some interpretations can extend it with partial functions.

Though we generally consider all injective representations, there are also good justifications to stick to bijective representations, as briefly seen in Section 2, and elaborated on in Section 5 and in [2,4]. Accordingly, we also define completeness with respect to bijective representations, called "interpretation-stability":

Definition 16 *A model A is* interpretation-stable *if there is no bijective representation $\pi : \mathrm{dom}\ A \to \mathrm{dom}\ A$ such that $[\![A]\!]_\pi \supsetneq [\![A]\!]$.*

Sticking to bijective representations, there are exactly two options for the representation influence:

– Stable model – totally immune to the influence of bijective representations. No better or worse interpretations are possible via bijective representations.
– Unstable model – there is no maximum, nor minimum, interpretation via bijective representations.

The above is illustrated in Figure 7, and proved in Theorem 12 and Corollary 13.

The general case, allowing non-bijective representations, is much more varied (see Figure 8):

– There are stable models that are incomplete.
– There might be a complete model with worse interpretations.
– Bijective representations preserve completeness.

Completeness obviously implies stability, but the opposite is not true. A simple example is the set of all constant functions except for a single one. A model having this extensionality is stable but incomplete.

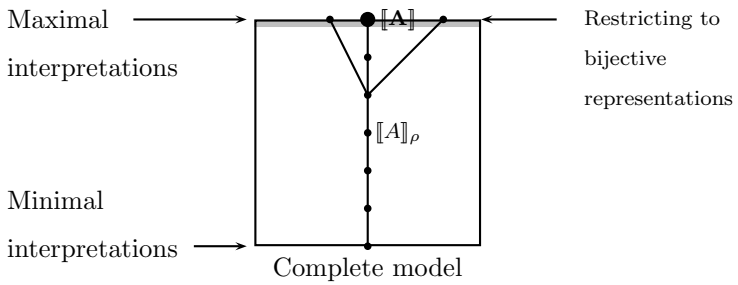


Fig. 8. An illustration of an interpretation-complete model

Completeness assures us that the model cannot have an interpretation better than its original extensionality. However, it might have an interpretation that decreases its original extensionality.

Proposition 17 *An interpretation-complete model might have an interpretation decreasing its extensionality. That is, there is an interpretation-complete model A and a representation ρ such that $[\![A]\!]_\rho \subsetneq [\![A]\!]$.*

Using only bijective representations, we cannot harm the extensionality of a complete model. This follows directly from Corollary 13, as a complete model is also stable.

Corollary 18 *Let A be a model and ρ a representation such that $[\![A]\!]_\rho$ is complete and $[\![A]\!]_\rho \supsetneq [\![A]\!]$. Then there is no bijective representation π such that $[\![A]\!]_\pi = [\![A]\!]_\rho$.*

Corollary 19 *Isomorphism preserves stability and completeness. That is, let A and B be isomorphic models, then A is interpretation-stable iff B is, and A is interpretation-complete iff B is.*

Proposition 20 *A model with a finite extensionality (implementing finitely many functions) is complete.*

Complete models have interesting properties and are generally more convenient to work with. We elaborate on some of the properties concerning power comparison and isomorphism in Section 4.

3.2.1. *Getting maximal interpretations*

A natural question is how to choose a proper representation for getting a maximal interpretation. We should consider two cases, depending on whether the relevant model is complete or not.

For an interpretation-complete model A, we can get a maximal interpretation by one of the following means:

- Its original extensionality.
- Via any bijective representation.
- If A is closed under functional composition: via a representation ρ for which there exists a total injective function $f \in [\![A]\!]$, such that $\mathrm{Im}\ f = \mathrm{Im}\ \rho$ and $f^{-1} \in [\![A]\!]$.

For an incomplete model:

- There is no general known criterion.
- A bijective representation cannot help.
- If one finds a representation via which there is a maximal interpretation, then he can get additional maximal interpretations with the above techniques for complete models.

The special case of proper representations with respect to effectiveness is considered in Section 5.2.

The claims above follow directly from results of previous sections and the following theorem:

Theorem 21 *Let A be a model closed under functional composition, and let $\rho : D \to \mathrm{dom}\ A$ be some representation. Then $[\![A]\!]_\rho$ is isomorphic to $[\![A]\!]$ if there is a total injective function $h \in [\![A]\!]$ such that $\mathrm{Im}\ h = \mathrm{Im}\ \rho$ and $h^{-1}{\restriction_{\mathrm{Im}\ h}} \in [\![A]\!]{\restriction_{\mathrm{Im}\ h}}$.*

3.3. *Specific models*

We turn now to investigate the influence of the domain interpretation on some well known computational models, as well as on hypercomputational models.

3.3.1. *The recursive functions*

The recursive functions (both total and partial) are interpretation-complete! Their completeness is of special importance due to their rôle in the notion of effectiveness. This is elaborated on in Section 5. The completeness is of both the unary recursive functions and of the functions of any arity.

In Section 3.3.2, it will be shown that the recursive functions are isomorphic to the functions computed by Turing machines. They are also isomorphic to 3-counter machines, while being a maximal interpretation of the incomplete 2-counter machines model.

Theorem 22 ([3]) *The unary recursive functions are interpretation-complete.*

Theorem 23 *The partial recursive functions and the total recursive functions are interpretation-complete.*

3.3.2. *Turing machines*

Turing machines are interpretation-complete. As with the recursive functions, this completeness is of special importance due to the rôle of Turing machines in the notion of effectiveness (see Section 5).

Theorem 24 ([3]) *Turing machines are interpretation-complete.*

When seeking a maximal interpretation for Turing machines or for the recursive functions, the criteria of Section 3.2.1 may be extended:

Theorem 25 *An interpretation $[\![\mathrm{TM}]\!]_\rho$ of Turing machines via some injection ρ is maximal if $|\mathrm{Im}\ \rho|$ is infinite and there is a function $h \in [\![\mathrm{TM}]\!]$ such that $\mathrm{Im}\ h = \mathrm{Im}\ \rho$.*

Note that the function h above needs not be total nor injective.

3.3.3. *Counter machines*

The model of two counter machines is very interesting. It was shown independently by Janis Barzdins [1], Rich Schroeppel [19], and Frances Yao that two-counter machines cannot compute the function $\lambda x.2^x$. On the other hand, since two-counter machines can compute all the recursive functions via an injective representation (viz. $n \mapsto 2^n$; see, for example, [12]), it follows that two-counter machines are interpretation-incomplete.

It turns out that the models of one-counter machines as well as of three-or-more counter machines are interpretation-complete.

3.3.4. *Hypercomputational models*

A computational model is generally said to be "hypercomputational" if it computes more than Turing machines or more than the recursive functions (see Definition 47 in Section 5). Due to the completeness of Turing machines and the recursive functions, such a model may indeed be regarded as more powerful. Power comparison is treated in detail in Section 4, and the issue of effective computation over arbitrary domains is treated in detail in Section 5.

Can we conclude from the interpretation-completeness of the recursive functions that every hypercomputational model is interpretation-complete? The answer is, in general, "no". However, if the hypercomputational model preserves the basic closure properties of the recursive functions, then the answer is "yes".

The following example is an incomplete hypercomputational model:

Example 26 *Let h be the (incomputable) halting function. Define:*

$$h_i := \lambda n. \begin{cases} 2^i h(n/2^i) & 2^i \text{ divides } n \\ 0 & \text{otherwise} \end{cases} \qquad \rho := \lambda n.2n$$

Let A be a computational model with the extensionality $[\![A]\!] := \mathbb{RREC} \cup \{h_i \mid i \in \mathbb{N}^+\}$. That is, $[\![A]\!]$ includes all the recursive functions and all functions h_i for $i \geq 1$. We have that $[\![A]\!]_\rho = \mathbb{RREC} \cup \{h_i \mid i \in \mathbb{N}\} \supsetneq [\![A]\!]$.

Yet, the completeness proof of the recursive functions (Theorem 23) may be extended to hypercomputational models, as long as they have the relevant closure properties. It also means that their domain is denumerable, as with higher cardinalities there is no meaning to primitive recursion or minimalization:

Theorem 27 *Let A be a computational model over $\mathbb{N}$ computing all the recursive function and closed under functional composition, primitive recursion and minimalization. Then A is interpretation-complete.*

A special case of such an interpretation-complete hypercomputational model is an oracle Turing machine.

Corollary 28 *An oracle Turing machine is interpretation-complete.*

4. Comparing Computational Power

It is standard in the literature to compare the power of computational models. However, neglecting the possibility of interpretation-incomplete models, it is common to say that model B is strictly stronger than model A when it computes strictly more functions. This might allow one to show that some computational models are *strictly* stronger than themselves (see Section 2).

We define a comparison notion over arbitrary domains based on model interpretations. With this notion, model B is strictly stronger than A if B has an interpretation that contains A, whereas A cannot contain B under *any* interpretation.

We start, in Section 4.1, by providing the mathematical definition of the comparison notion. We give a basic definition, allowing all interpretations, and two additional definitions with some restrictions on the allowed interpretations. In Section 4.1.1 we extend the notion no non-deterministic models.

In Section 4.2 we provide several results with respect to power comparison, isomorphism and completeness. In Section 4.3, we compare between some standard models, as Turing machines, stack machines, etc..

4.1. *The comparison notions*

Since we are only interested here in the extensional quality of a computational model (the set of functions or relations that it computes), not complexity-based comparison or step-by-step simulation, we use model interpretations as the basis for comparison.

We generally say that model B is at least as powerful as model A if it can compute whatever A can. When both models operate over the same domain it simply means containment: B is at least as powerful as A if $[\![B]\!] \supseteq [\![A]\!]$. However, when the models operate over different domains we ought to interpret one model over the domain of the other. Hence, the general comparison notion would say that B is at least as powerful as A if it has an interpretation that contains A.

As one textbook states [21, p. 30]:

> Computability relative to a coding is the basic concept in comparing the power of computation models.... The computational power of the model is represented by the extension of the set of all functions computable according to the model. Thus, we can compare the power of computation models using the concept 'incorporation relative to some suitable coding'.

We provide three notions of comparison, depending on the allowed representations. The basic, most permissive, comparison notion allows any injection, while the firmest notion allows only bijections. In between, we define a notion that allows injections for which the "as-powerful" model can fix their image.

Definition 29 (Power Comparison)

- *Model B is* at least as powerful *as model A, denoted $B \gtrsim A$, if it has an interpretation that contains the extensionality of A. That is, $B \gtrsim A$ iff exists an injection $\rho : \mathrm{dom}\, A \to \mathrm{dom}\, B$, such that $[\![B]\!]_\rho \supseteq [\![A]\!]$.*
- *Model B is* decently at least as powerful *as model A, denoted $B \succsim A$, if it has an interpretation that contains the extensionality of A via a representation for which it can fix its image. That is, $B \succsim A$ iff exists an injection $\rho : \mathrm{dom}\, A \to \mathrm{dom}\, B$, such that $[\![B]\!]_\rho \supseteq [\![A]\!]$ and there is a total function $g \in [\![B]\!]$ for which $\mathrm{Im}\, g = \mathrm{Im}\, \rho$ and such that for every $y \in \mathrm{Im}\, \rho$ we have that $g(y) = y$.*
- *Model B is* bijectively at least as powerful *as model A, denoted $B \gtrapprox A$, if it has a bijective interpretation that contains the extensionality of A. That is, $B \gtrapprox A$ iff exists a bijection $\pi : \mathrm{dom}\, A \to \mathrm{dom}\, B$, such that $[\![B]\!]_\pi \supseteq [\![A]\!]$.*

Proposition 30 *The computational power relations $\gtrsim, \succsim, \gtrapprox$ are quasi-orders.*

Obviously, the two latter comparison notions imply the former one. For the third notion to imply the second one, it is sufficient that the as-powerful model has some surjective function, even the identity function. Additionally, when assuming a little more about the representation and the model, the second notion also implies the third one (see Theorem 21).

Note 31 *When comparing the computational power of models, it should be noted that one function cannot be "better" than another; only a model can be better than another. There is only an equivalence relation between functions, where two partial functions, f and g, over the same domain D are deemed* equal*, denoted $f = g$, if they are defined for exactly the same elements of the domain and have the same value whenever they are defined. For example, a total function f is not better than nor equal to a partial function g that has the same values as f when it converges. This is clearly demonstrated by taking g to be a 'halting function', which returns* 1 *when the input encodes a halting machine and diverges otherwise.*

Our comparison notions go along with some standard approaches, for instance in [7, p. 24], [18, p. 27] and [21, p. 30]. Our decent-comparison notion follows Rabin's definition of a computable group [16, p. 343]:" DEFINITION 3. An *indexing* of a set S is a one to one mapping $i : S \to I$ such that $i(S)$ is a recursive subset of I."

Example 32 *Turing machines are at least as powerful as the recursive functions via a unary representation of the natural numbers. See, for example, [9, p. 147]. Indeed, it is so by all three notions (see Theorem 45).*

One may wonder why we do not require the representation to be Turing-computable. A detailed answer to that is given in Section 5. The main points are:

- When comparing models in the scope of defining effectiveness, the requirement of a Turing-computable representation leads to a circular definition.
- One may wish to compare sub-recursive models or hypercomputational models, for which Turing-computability is not necessarily the proper constraint.

Power equivalence. The power equivalence relation between models follows directly from the power comparison notion. Models A and B are of equivalent power if $A \gtrsim B$ and $B \gtrsim A$.

Definition 33 (Power Equivalence)

- *Models A and B are* power equivalent*, denoted $A \sim B$, if $A \gtrsim B$ and $B \gtrsim A$.*
- *Models A and B are* decently power equivalent*, denoted $A \simeq B$, if $A \gtrsim\!\!\!\!\!\sim\ B$ and $B \gtrsim\!\!\!\!\!\sim\ A$.*
- *Models A and B are* bijectively power equivalent*, denoted $A \approx B$, if $A \gtrapprox B$ and $B \gtrapprox A$.*

Example 34 *The (untyped) λ-calculus is power equivalent to the recursive functions, via Church numerals, on the one hand, and via Gödelization, on the other. However, it is not interpretation-complete, as it cannot even compute the identity function over an arbitrary lambda-term. Hence, it is not bijectively-power equivalent to the recursive functions (otherwise contradicting the completeness of the recursive functions).*

Strictly stronger. We generally think of model B as stronger than model A if it can compute more. However, because of the sensitivity to the domain interpretation (Section 2), proper containment does not imply more computational power. That is, $[\![B]\!] \supsetneq [\![A]\!] \not\Rightarrow B \gtrsim A$. This is so for all three comparison notions.

The proper definition of model B being strictly stronger than model A is that $B \gtrsim A$ while $A \not\gtrsim B$.

Definition 35 (Stronger)

- *Model B is* stronger *than model A, denoted $B \gtrsim\!\!\!\!\!\not\ \ A$, if $B \gtrsim A$ while $A \not\gtrsim B$.*
- *Model B is* decently stronger *than model A, denoted $B \gtrsim\!\!\!\!\!\sim\ A$, if $B \gtrsim\!\!\!\!\!\sim\ A$ while $A \not\gtrsim\!\!\!\!\!\sim\ B$.*
- *Model B is* bijectively stronger *than model A, denoted $B \gtrapprox\!\!\!\!\!\not\ \ A$, if $B \gtrapprox A$ while $A \not\gtrapprox B$.*

Note that for model B to be stronger than model A there should be no injection ρ via which $A \gtrsim_\rho B$. In contradistinction to the "as powerful" case, some model B may be bijectively stronger than a model A, but not stronger than A. However, if A is interpretation-complete, we do have that $B \gtrapprox\!\!\!\!\!\not\ \ A$ implies $B \gtrsim\!\!\!\!\!\not\ \ A$ (Theorem 42).

Example 36 *Real recursive functions [14], operating over $\mathbb{R}^2$, are decently stronger than Turing machines. The comparison is done via an injective representation $\psi : \{0,1\}^* \to \mathbb{R}^2$, defined by $[\![x]\!]_\psi := (0, [\![x]\!]_\rho)$, where $[\![x]\!]_\rho$ is the standard binary interpretation over the natural numbers [14, p. 849]. Since the model computes the floor function ($\lambda x.\lfloor x \rfloor$) [14, p. 843], it follows that it also has a function which fixes the representation image. On the other hand, the real recursive functions are obviously not bijectively stronger than the recursive functions, as their domain is of a higher cardinality.*

Universal machines. Computation is often performed via universal machines, also referred to as "interpreters". That is, a single machine (or computer program) gets as input both the machine to interpret and the latter's input. One may wonder how to compare the computational power of universal machines. From our point of view, computational power is extensionality, meaning the set of computed functions. Hence, when comparing universal machines we compare the sets of functions that they interpret. Accordingly, it is exactly like comparing computational models.

4.1.1. *Non-deterministic models*

The extension of most of the definitions and theorems given so far (in this section and the previous ones) to non-deterministic models is quite straightforward. There are, however, some specific issues concerning the power comparison of non-deterministic models, which will be discussed below.

We begin by defining what we mean by a non-deterministic computational model and its extensionality:

Definition 37 (Non-deterministic Computational Model) *A* non-deterministic computational model *A over domain D is any object associated with a set of (non-unary) relations over $D \cup \perp$. This set of relations is called the* extensionality *of the non-deterministic computational model, denoted $[\![A]\!]$.*

Note that we use the special symbol $\perp$ in the above definition for denoting a non-halting computation, while we did not need it in the definition of a deterministic model (Definition 1). The reason is that a non-deterministic computation might sometimes diverge on a domain element e and sometimes converge to some value v. In such a case, the value of the computation will be both $\perp$ and v, denoted by $\langle e, \perp\rangle, \langle e, v\rangle$ in the (multivalued) function's description. On the other hand, the divergence of a deterministic function on a domain element e, may be simply denoted by not having a tuple with e in the function's description.

As a result of using $\perp$ to denote divergence, we may assume that all functions have at least one value for every domain element.

When investigating the influence of the domain interpretation, we are concerned with the containment relation between the different interpretations of a computational model. Hence, the definitions given in this section and the previous ones apply to both deterministic and non-deterministic models, as in both types of models the interpretations are sets of relations.

When we compare the power of computational models, two functions are considered equal if and only if they are described by the same relation, while a model is as powerful as another if it contains all the functions of the former (see Note 31). This is also the case when we compare between non-deterministic models. There might be cases in which a different approach is required, assuming a different equality notion between functions. For example, two non-deterministic functions might be considered equal if they *may* always produce the same value. That is, a function that sometimes diverges and sometimes converges with the value v is considered equal to a function that always converges with the value v. In such cases, the comparison notions should be adjusted to take into account the special equality notion between functions.

By this comparison approach, a non-deterministic model B may be as powerful as a deterministic model A (if it deterministically computes all the functions of A, in addition to its non-deterministic computations), while the opposite is impossible (when B actually has some non-deterministic computations).

4.2. *Power comparison, isomorphism, and interpretation-completeness*

The general approach for showing that model B is stronger than model A is tedious – we should negate the possibility that $A \gtrsim_\rho B$ for all injections ρ. The situation is much simpler with interpretation-complete models, for which proper containment does imply more power:

Theorem 38 ([3]) *For an interpretation-complete model A and some model B, we have that $B \gtrsim A$ iff there exists an injection ρ such that $[\![B]\!]_\rho \supsetneq [\![A]\!]$.*

Isomorphism and bijective power equivalence are very similar notions, though not the same. However, when sticking to interpretation-stable models they do coincide:

Theorem 39

(i) Isomorphism implies bijective power equivalence.

(ii) Bijective power equivalence does not imply isomorphism.

(iii) For stable models, bijective power equivalence implies isomorphism. That is, if model A is stable, then for every model B, B is isomorphic to A if and only if it is bijectively power equivalent to A.

Isomorphism preserves stability and completeness (Corollary 19). This is also the case, by the above theorem, with bijective power equivalence.

Corollary 40 *Bijective power equivalence preserves stability and completeness. That is, let A and B be bijectively power equivalent models, then A is stable iff B is, and A is complete iff B is.*

The formulation of the previous theorem can be strengthened for complete models:

Lemma 41 ([3]) *If model A is complete and $A \gtrsim B \gtrapprox A$ for some model B, then A and B are isomorphic.*

Interpretation-completeness also helps with showing that some model is stronger than another.

Theorem 42 *If model A is complete, then $B \gtrapprox A$ implies $B \gtrsim A$*

Theorem 43 ([3]) *If model A is complete, $A \approx B$ and $[\![B]\!] \subsetneq [\![C]\!]$, for models B and C, then $C \gtrsim A$.*

4.3. *Comparison of some Standard Models*

As discussed in the beginning of this section, the proper containment between the recursive functions and the primitive recursive functions does not imply that the former are strictly more powerful than the latter. Nonetheless, the recursive functions are indeed strictly more powerful, even by the general comparison notion, allowing all possible interpretations of the primitive recursive functions.

Theorem 44 ([3]) *The primitive recursive functions are strictly weaker than the recursive functions.*

We do not know if the primitive recursive functions are interpretation-complete, Theorem 44 notwithstanding.

It is common to show that Turing machines and the recursive functions are of equivalent computational power. Actually, they are isomorphic. We base the proof on known results, given in [10].

Theorem 45 ([3]) *Turing machines, over a binary alphabet, and the recursive functions are isomorphic.*

From [10,3] we also have that random access machines and counter machines with unlimited number of counters have exactly the same extensionality as the recursive functions.

This is not the case with two-counter machines. They are of equivalent power to the recursive functions, however not isomorphic to them. Indeed, they are bijectively-weaker than them (otherwise contradicting the stability of the recursive functions).

By known results, two-stack machines have the same extensionality as Turing machines.

Due to the closure properties of Turing machines and the recursive functions, they are bijectively at least as powerful as some model A if and only if they are decently at least as powerful as A.

Theorem 46 *Let A be a computational model operating over a denumerable domain. Then Turing machines and the recursive functions are decently at least as powerful as A if and only if they are bijectively at least as powerful as A. That is,* $\mathrm{TM} \succsim A$ *iff* $\mathrm{TM} \gtrapprox A$ *iff* $\mathbb{REC} \succsim A$ *iff* $\mathbb{REC} \gtrapprox A$.

5. Effective Computation

In 1936, Alonzo Church and Alan Turing each formulated a claim that a particular model of computation completely captures the conceptual notion of "effective" computability. Church [6, p. 356] proposed that effective computability of numeric functions be identified with Gödel and Herbrand's general recursive functions, or – equivalently, as it turned out [6] – with Church and Kleene's lambda-definable functions of positive integers.

Turing, on the other hand, explicitly extends the notion of "effective" beyond the natural numbers [23, fn. p. 166] (emphasis added):

> We shall use the expression "computable function" to mean a function calculable by a machine, and we let "effectively calculable" refer to the intuitive idea without particular identification with one of these definitions. *We do not restrict the values taken by a computable function to be natural numbers*; we may for instance have computable propositional functions.

Our purpose, in Section 5.1, is to formalize and analyze the Church-Turing Thesis, referring to functions over arbitrary domains.

Equipped with this definition, and due to the interpretation-completeness of Turing machines, we define, in Section 5.2, effective representations of constructible domains.

5.1. *The Church-Turing Thesis over arbitrary domains*

Simply put, the Church-Turing Thesis is not well defined for arbitrary domains: the choice of domain interpretation might have a significant influence on the outcome. We explore below the importance of the domain interpretation and suggest how to overcome this problem.

Computational model versus single function. A single function over an arbitrary domain cannot be classified as computable or not. Its computability depends on the representation of the domain. [2] For example, the (uncomputable) halting function over the natural numbers (sans the standard order) is isomorphic to the simple parity function, under a permutation of the natural numbers that maps the usual codes of halting Turing machines to strings ending in "0", and the rest of the numbers to strings ending with "1". The result is a computable standalone "halting" function.

An analysis of the classes of number-theoretic functions that are computable relative to different notations (representations) is provided by Shapiro [20, p. 15]:

> It is shown, in particular, that the class of number-theoretic functions which are computable relative to every notation is too narrow, containing only rather trivial functions, and that the class of number-theoretic functions which are computable relative to some notation is too broad containing, for example, every characteristic function.

An intuitive approach is to restrict the representation only to "natural" mappings between the domains. However, when doing so in the scope of defining "effectiveness", one must use a vague and undefined notion.

This problem was already pointed out by Richard Montague on 1960 [13, pp. 430–431]:

> Now Turing's notion of computability applies directly only to functions on and to the set of natural numbers. Even its extension to functions defined on (and with values in) another denumerable set S cannot be accomplished in a completely unobjectionable way. One would be inclined to choose a one-to-one correspondence between S and the set of natural numbers, and to call a function f on S computable if the function of natural numbers induced by f under this correspondence is computable in Turing's sense. But the notion so obtained depends on what correspondence between S and the set of natural numbers is chosen; the sets of computable functions on S correlated with two such correspondences will in general differ. The natural procedure is to restrict consideration to those correspondences which are in some sense 'effective', and hence to characterize a computable function on S as a function f such that, for some effective correspondence between S and the set of natural numbers, the function induced by f under this correspondence is computable in Turing's sense. But the notion of effectiveness remains to be analyzed, and would indeed seem to coincide with computability.

Stewart Shapiro suggests a definition of "acceptable notation", based on intuitive concepts [20, p. 18]:

> This suggests two informal criteria on notations employed by algorithms:
>
> (1) The computist should be able to *write* numbers in the notation. If he has a particular number in mind, he should (in principle) be able to write and identify tokens for the corresponding numeral.
>
> (2) The computist should be able to *read* the notation. If he is given a token for a numeral, he should (in principle) be able to determine what number it denotes.
>
> It is admitted that these conditions are, at best, vague and perhaps obscure.

Michael Rescorla argues that the circularity is inherent in the Church-Turing Thesis [17].

A possible solution is to allow any representation (injection between domains), while checking for the effectiveness of an entire computational model. That is, to check for the computability of a function together with the other functions that are computable by that computational model. The purpose lying behind this idea is to view the domain elements as arbitrary objects, deriving all their meaning from the model's

[2] There are functions that are inherently uncomputable, via all domain representations. For example, a permutation of some countable domain, in which the lengths of the orbits are exactly the standard encodings of the non-halting Turing machines.

functions. For example, it is obvious that the halting function has a meaning only if one knows the order of the elements of its domain. In that case, the successor function provides the meaning for the elements.

Two variants of this solution, corresponding to the variants of our comparison notion (Definition 29), are to either allow only bijective representations, or else allow injections provided that their images are computable.

Adopting the above approach of checking for computability of an entire computational model, we interpret the Church-Turing Thesis as follows:

Thesis A. *All "effective" computational models are of equivalent power to, or weaker than, Turing machines.*

By "effective", in quotes, we mean effective in its intuitive sense.

By a "computational model" we refer to any object that is associated with a set of partial functions (Definition 1). By "equivalent to, or weaker than" we refer to the comparison notions of Definition 29.

A strict supermodel of the recursive functions (or Turing machines) is a "hypercomputational" model.

Definition 47 (Hypercomputational Model) *A computational model H is* hypercomputational *if it is stronger than Turing machines. That is, if $H \gtrsim \text{TM}$. (The corresponding variations, using bijective power comparison or decent power comparison, are $H \gtrapprox \text{TM}$ or $H \gtrsim \text{TM}$.)*

Our interpretation of the Church-Turing Thesis (Thesis A) agrees with Rabin's definition of a computable group [16, p. 343], as well as with its generalization, by Lambert [24, p. 594], to any algebraic structure. Similar notions were also presented by Froehlich and Shepherdson [8] and Mal'cev [11].

Influence of representations. The Church-Turing Thesis, as stated above (Thesis A), matches the intuitive understanding only due to the interpretation-completeness of Turing machines (Theorem 24). Were the thesis defined in terms of two-counter machines (2CM), for example, it would make no sense: a computational model is not necessarily stronger than 2CM even if it computes strictly more functions.

5.2. *Effective representations*

What is an effective representation? We argued above that a "natural representation" must be a vague notion when used in the context of defining effectiveness. We avoided the need of restricting the representation by checking the effectiveness of entire computational models. But what if we adopt the Church-Turing Thesis; can we then define what is an effective string representation?

Simply put, there is a problem here. Turing machines operate only over strings. Thus a string representation, which is an injection from some domain D to Σ^*, is not itself computable by a Turing machine. All the same, when we consider, for example, string representations of natural numbers, we can obviously say regarding some of them that they are effective. How is that possible? The point is that we look at the natural numbers as having some structure, usually assuming their standard order. A function over the natural numbers without their order is not really well-defined. As we saw, the halting function and the simple parity function are exactly the same (isomorphic) function when numbers are unordered.

Hence, even when adopting the Church-Turing Thesis, a domain without any structure cannot have an effective representation. It is just a set of arbitrary elements. However, if the domain comes with a generating mechanism (as the natural numbers come with the successor) we can consider effective representations.

Due to the interpretation-completeness of the recursive functions and Turing machines, we can define what is an effective string representation of the natural numbers (with their standard structure). A similar definition can be given for other domains, provided that they come with some finite means of generating them all, akin to successor for the naturals.

Definition 48 *An* effective representation *of the natural numbers by strings is an injection $\rho : \mathbb{N} \to \Sigma^*$, such that $\rho(s)$ is Turing-computable ($\rho(s) \in [\![\text{TM}]\!]$), where s is the successor function over $\mathbb{N}$.*

That is, a representation of the natural numbers is effective if the successor function is Turing-computable via this representation.

Note 49 *One may also require that the image of the representation ρ is Turing computable, along the decent power comparison notion. In such a case, there would also be a corresponding bijective representation (see Theorem 21 and 25).*

We justify the above definition of an effective representation by showing that: (a) every recursive function is Turing-computable via any effective representation; (b) every non-recursive function is not Turing-computable via any effective representation; and (c) for every non-effective representation there is a recursive function that is not Turing-computable via it.

Theorem 50 ([5])

(a) Let f be a recursive function and $\rho : \mathbb{N} \to \Sigma^*$ an effective representation. Then $\rho(f) \in [\![\mathrm{TM}]\!]$.

(b) Let g be a non-recursive function and $\rho : \mathbb{N} \to \Sigma^*$ an effective representation. Then $\rho(g) \notin [\![\mathrm{TM}]\!]$.

(c) Let $\eta : \mathbb{N} \to \Sigma^*$ be a non-effective representation. Then there is a recursive function f, such that $\eta(f) \notin [\![\mathrm{TM}]\!]$.

To see the importance of the interpretation-completeness for the definition of an effective representation, one can check that an analogous definition cannot be provided with two-counter machines as the yardstick.

Our definition of an effective representation resembles Shapiro's notion of an "acceptable notation" [20, p. 19] and goes along with Weihrauch's justifications for the effectiveness of the "standard numberings" (representations by natural numbers) [25, p. 80–81]. A description of the correlation between our notion and Shapiro's and Weihrauch's notions can be found in [5].

6. Discussion

Key points. In this paper, we have explored various aspects of the influence of domain interpretations on the extensionality of computational models, and suggested how we believe they should be handled. Some of the key points are:

- The extensionality of computational models is a very interesting set (of functions) – it varies from "fluid" sets for which containment does not mean more power, as with ordinary sets, to explicit, complete, sets for which nothing can be added without additional power.
- Get to know your model: Is it interpretation-stable or interpretation-complete?
- Compare computational power properly.
- Turing machines and the recursive functions are shown to be robust, this time from the representation/interpretation point of view.
- Effectiveness does not apply to a single function; it is a set of functions, or a function together with its domain constructors, that may be deemed effective.

Domain Representation. One might think that our study is a part of the "domain representation" research area. This is not the case. Indeed, the basic concept of "representation" is the same: one set of elements is represented by a subset of another set. Nevertheless, the issues studied are very different. *Domain representation* concerns mapping of some (possibly topological/metric) sets into *domains* (partially ordered sets with some properties), and studies the (topological/metric) properties preserved via the mappings.

Further research. A central issue, yet to be understood, is the relation between the internal mechanism of a computational model and its extensional properties of interpretation-completeness and interpretation-stability. With the recursive functions, this relation is apparent, shown in the proof of their interpretation-completeness. However, we do not know how to relate, in general, the completeness or stability of a computational model to its internal mechanism. A more specific open question is whether there is some standard, well known, computational model that is unstable.

References

[1] J. Barzdins. About one class of Turing machines (Minsky machines). *Algebra and Logic (seminar)*, 1(6):42–51, 1962.

[2] U. Boker and N. Dershowitz. How to compare the power of computational models. In *Computability in Europe 2005: New Computational Paradigms (Amsterdam), S. Barry Cooper, Benedikt Löwe, Leen Torenvliet, eds.*, volume 3526 of *Lecture Notes in Computer Science*, pages 54–64, Berlin, Germany, 2005. Springer-Verlag.
[3] U. Boker and N. Dershowitz. Comparing computational power. *Logic Journal of the IGPL*, 14(5):633–648, 2006.
[4] U. Boker and N. Dershowitz. A hypercomputational alien. *Applied Mathematics and Computation*, 178(1):44–57, 2006.
[5] U. Boker and N. Dershowitz. The Church-Turing thesis over arbitrary domains. In A. Avron, N. Dershowitz, and A. Rabinovich, editors, *Pillars of Computer Science*, volume 4800 of *Lecture Notes in Computer Science*, pages 199–229. Springer, 2008.
[6] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.
[7] N. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, Cambridge, 1980.
[8] A. Froehlich and J. Shepherdson. Effective procedures in field theory. *Philosophical Transactions of the Royal Society of London*, 248:14–20, 1956.
[9] F. Hennie. *Introduction to Computability*. Addison-Wesley, Reading, MA, 1977.
[10] N. D. Jones. *Computability and Complexity from a Programming Perspective*. The MIT Press, Cambridge, Massachusetts, 1997.
[11] A. Mal'cev. Constructive algebras I. *Russian Mathematical Surveys*, 16:77–129, 1961.
[12] M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, N.J., 1967.
[13] R. Montague. Towards a general theory of computability. *Synthese*, 12(4):429–438, 1960.
[14] J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857, 2004.
[15] J. Myhill. Some philosophical implications of mathematical logic. Three classes of ideas. *The Review of Metaphysics*, 6(2):165–198, 1952.
[16] M. O. Rabin. Computable algebra, general theory and theory of computable fields. *Transactions of the American Mathematical Society*, 95(2):341–360, 1960.
[17] M. Rescorla. Church's thesis and the conceptual analysis of computability. *Notre Dame Journal of Formal Logic*, 48(2):253–280, 2007.
[18] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, 1966.
[19] R. Schroeppel. A two counter machine cannot calculate 2^N. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1972. available at: ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-257.pdf.
[20] S. Shapiro. Acceptable notation. *Notre Dame Journal of Formal Logic*, 23(1):14–20, 1982.
[21] R. Sommerhalder and S. C. van Westrhenen. *The Theory of Computability: Programs, Machines, Effectiveness and Feasibility*. Addison-Wesley, Workingham, England, 1988.
[22] G. J. Tourlakis. *Computability*. Reston Publishing Company, Reston, VA, 1984.
[23] A. M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45:161–228, 1939.
[24] J. W. M. Lambert. A notion of effectiveness in arbitrary structures. *The Journal of Symbolic Logic*, 33(4):577–602, 1968.
[25] K. Weihrauch. *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1987.
[26] K. Weihrauch. *Computable Analysis — An introduction*. Springer-Verlag, Berlin, 2000.