On Approximating Real-World Halting Problems

Sven Köhler[2], Christian Schindelhauer[1], and Martin Ziegler[3]

[1] Heinz Nixdorf Institute, University of Paderborn
schindel@uni-paderborn.de
[2] University of Paderborn
skoehler@uni-paderborn.de
[3] University of Southern Denmark
ziegler@imada.sdu.dk

Abstract. No algorithm can of course solve the Halting Problem, that is, decide within finite time always correctly whether a given program halts on a certain given input. It might however be able to give correct answers for 'most' instances and thus solve it at least approximately. Whether and how well such approximations are feasible highly depends on the underlying encodings and in particular the Gödelization (programming system) which in practice usually arises from some programming language.

We consider `BrainF*ck` (BF), a simple yet Turing-complete real-world programming language over an eight letter alphabet, and prove that the natural enumeration of its syntactically correct sources codes induces a both efficient and dense Gödelization in the sense of [Jakoby&Schindelhauer'99]. It follows that any algorithm M approximating the Halting Problem for BF errs on at least a constant fraction $\varepsilon_M > 0$ of all instances of size n for infinitely many n.

Next we improve this result by showing that, in every dense Gödelization, this constant lower bound ε to be independent of M; while, the other hand, the Halting Problem does admit approximation up to arbitrary fraction $\delta > 0$ by an appropriate algorithm M_δ handling instances of size n for infinitely many n. The last two results complement work by [Lynch'74].

1 Introduction

In 1931, the logician KURT GÖDEL constructed a mathematical predicate which could neither be proven nor falsified. In 1936, ALAN TURING introduced and showed the Halting Problem H to be undecidable by a Turing machine. This was considered a strengthening of Gödel's result regarding that, at this time and preceding AIKEN's Mark I and ZUSE's Z3, the Turing machine was meant as an idealization of an average mathematician.

Nowadays the Halting Problem is usually seen from a quite different perspective. Indeed with increasing reliance on high speed digital computers and huge software systems running on them, source code verification or at least the detection of stalling behaviour becomes even more important. In fact, by RICE's Theorem, this is equivalent to many other real-world problems arising from goals like automatized software engineering, optimizing compilers, formal proof systems and so on. Thus, the Halting problem is a very practical one which has to be dealt with some way or another.

M. Liśkiewicz and R. Reischuk (Eds.): FCT 2005, LNCS 3623, pp. 454–466, 2005.

One direction of research considered and investigated the capabilities of extended Turing machines equipped with some kind of external device solving the Halting problem. While the physical realizability of such kinds of super-Turing computers is questionable and in fact denied by the Church-Turing Hypothesis, the current field of Hypercomputation puts in turn this hypothesis into question. On the theoretical side, these considerations led to the notion of relativized computability and the Arithmetical Hierarchy which have become standard topics in Recursion Theory [Soar87].

1.1 Approximate Problem Solving

Another approach weakens the usual notion of algorithmic solution from strict to approximate or from worst-case to average case. The first arises from the fact that many optimization problems are $\mathcal{NP}$-complete only if requiring the solution to exactly attain the, say, minimum whereas they become computationally much easier when asking for a solution only within a certain factor of the optimum.

Regarding decision problems, a notion of approximate solution has been established in Property Testing [Gold97]. Here for input $x \in \Sigma^n$, the answer "$x \in L$" is considered acceptable even for $x \notin L$ *provided* that $y \in L$ holds for some $y \in \Sigma^n$ with (edit or Hamming) distance $d(x,y) \leq \varepsilon n$. Observe that this notion of approximation strictly speaking refers to the arguments x to the problem rather than the problem L itself. Also, any program source x is within constant distance from the terminating one y obtained by changing the first command(s) in x by a `halt` instruction.

Average case analysis is an approach based on the observation that the hard instances which make a certain problem difficult might occur only rarely in practice whereas most 'typical' instances might turn out as easy. So, although for example $\mathcal{NP}$-complete, an algorithm would be able to correctly and efficiently solve this problem in, say, 99.9% of all cases while possibly failing on some few and unimportant others. In this example, $\varepsilon = 1/1000$ is called the error rate of the problem under consideration with respect to a certain probability distribution or encoding of its instances.

Such approaches have previously been mainly applied in order to deal with important problems where the practitioner cannot be silenced by simply remarking that they are $\mathcal{NP}$-complete, that is, within complexity theory. However the same makes sense, too, for important undecidable problems such as Halting: even when possibly erring on, say, every 10th instance, detecting the other 90% of stalling programs would have prevented many buggy versions of a certain operation system from being released prematurely.

1.2 The Error Complexity

So instead of deciding some (e.g., hard or even non-recursive) problem L, one is satisfied with solving some problem S which approximates L in the sense that the symmetric[1] set difference $A := L \triangle S := (L \setminus S) \cup (S \setminus L)$ is 'small'. For $L \subseteq \Sigma^*$ (with an at least two-letter alphabet Σ) this is formalized, analogously to the error complexity from average analysis, [JaSc99, DEFINITION 1] as the asymptotic behavior of $\mu\{\bar{x} \in (L \triangle S) | \bar{x} \in \Sigma^n\}$ for a fixed probability measure $\mu : \Sigma^* \to [0,1]$; if this quantity

[1] The error to the Halting problem can in fact be made one-sided, see Corollary 22 below.

tends to 0 as $n \to \infty$ it basically means that, for (μ-) average instances, S ultimately equals L. In the case of μ denoting the counting measure, this amounts [Papa95, §14.2, p.336],[RoUl63] to the following

Definition 1. *For $A \subseteq \Sigma^*$, let* $\text{density}(A,{=}n) := \#(A \cap \Sigma^n)/\#\Sigma^n$ *and* $\text{density}(A,{<}n) := \#(A \cap \Sigma^{<n})/\#(\Sigma^{<n})$ *where* $\Sigma^{<n} = \bigcup_{j=0}^{n-1} \Sigma^j$.
For $A \subseteq \mathbb{N}$, let $\text{Density}(A,N) := \#(A \cap \{0,\ldots,N-1\})/N$.

The latter formalization has been considered independently in [RoUl63, Lync74][2] for approximating decision problems $L \subseteq \mathbb{N}$. The notions are related as follows:

Lemma 2. *For $x \in \mathbb{N}$, let $\bar{x}$ denote the x-th string in Σ^* ordered with respect to length (ties broken arbitrarily). For $A \subseteq \mathbb{N}$, $\tilde{A} := \{\bar{x} : x \in A\}$, and $0 \le \varepsilon \le 1$ it holds:*

a) $\text{density}(\tilde{A},{=}n) \le \varepsilon \ \forall n \quad \Rightarrow \quad \text{density}(\tilde{A},{<}n) \le \varepsilon \ \forall n.$
b) $\text{Density}(A,N) \le \varepsilon \ \forall N \quad \Rightarrow \quad \text{density}(\tilde{A},{<}n) \le \varepsilon \ \forall n.$
c) $\text{density}(\tilde{A},{<}n) \le \varepsilon \ \forall n \quad \Rightarrow \quad \text{Density}(A,N) \le \varepsilon' \ \forall N \quad$ *where* $\varepsilon' := \varepsilon \cdot (2-\varepsilon)$.

Taking complements yields similar claims for reversed inequalities "$\ge \varepsilon$".

Since $0 < \varepsilon' < 1$ whenever $0 < \varepsilon < 1$ in b+c), both densities are essentially equivalent up to constants in that one tends to 0/1 iff so does the other. This allows us to deliberately switch in the sequel between $A \subseteq \Sigma^*$ encoded over one alphabet Σ (say, the decimals $\{0,1,\ldots,9\}$) and its re-coding over some other (e.g., binary or hexadecimal) finite Σ'.

Proof (Lemma 2). a) is obvious; for b) observe $\text{density}(\tilde{A},{<}n) = \text{Density}\big(A,\#(\Sigma^{<n})\big)$. This also establishes c) in case $N = \#(\Sigma^{<n}) = \frac{\#\Sigma^n-1}{\#\Sigma-1}$ with $\#(A \cap \{0,\ldots,N-1\}) \le \varepsilon \cdot \#(\Sigma^{<n})$, whereas the worst-case occurs for $N = \#(\Sigma^{<n}) + \varepsilon \cdot \#\Sigma^n$ with $\#(A \cap \{0,\ldots,N-1\}) = \varepsilon \cdot \#(\Sigma^{<n}) + \varepsilon \cdot \#\Sigma^n$. Then and thus,

$$\text{Density}(A,N) \ \le \ \frac{\varepsilon \cdot \frac{\#\Sigma^{n+1}-1}{\#\Sigma-1}}{\frac{\#\Sigma^n-1}{\#\Sigma-1} + \varepsilon \cdot \#\Sigma^n} \ = \ \varepsilon \cdot \left(1 + (1-\varepsilon) \cdot \frac{\#\Sigma-1}{\#\Sigma - \frac{1}{\#\Sigma^n}}\right) \ \le \ \varepsilon \cdot \big(1 + (1-\varepsilon)\big)$$

□

For a good approximation S of L, one wants the density of $A = L \triangle S$ to eventually drop below some prescribed ε; that is satisfy, e.g., $\exists n_0 \, \forall n \ge n_0 : \text{density}(A,n) \le \varepsilon$.

Definition 3. *An inequality "$f(n) \le g(n)$" depending on $n \in \mathbb{N}$ holds* **almost everywhere**, *denoted by "$f(n) \le_{ae} g(n)$", iff $\exists n_0 \forall n \ge n_0 : f(n) \le g(n)$.*
It holds **infinitely often** *("$f(n) \le_{io} g(n)$") iff $\forall n_0 \exists n \ge n_0 : f(n) \le g(n)$.*

So if "$\text{density}(A,n) \le_{ae} \varepsilon$" fails, one may try for the weaker "$\text{density}(A,n) \le_{io} \varepsilon$".

1.3 The Halting Problem

The halting problem is defined with respect to an (often implicitly chosen) programming system. Here we follow the notation of [Roge67, Soar87, Smit94].

Definition 4. *A* **Gödelization** φ *is a sequence of all partial recursive functions s.t.*
– *there exists a partial universal program u with* $\varphi_u(\langle i,x\rangle) = \varphi_i(x)$ *(UTM)*

[2] We are considerably grateful to an anonymous referee for pointing out the work of N. LYNCH.

– *and a total program s with* $\varphi_{s(\langle i,x\rangle)}(y) = \varphi_i(\langle x,y\rangle)$ *(SMN)*
– *for a bijective computable function* $\langle\cdot,\cdot\rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ *or* $\langle\cdot,\cdot\rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

called **pairing function**. *The* **Halting problem** *for* φ *is* $H_\varphi = \{\langle i,x\rangle : x \in \mathrm{dom}(\varphi_i)\}$.

The Halting problem is sometimes alternatively defined as the task $\tilde{H}_\varphi$ of deciding whether a given program i terminates on the empty input, that is, whether $\lambda \in \mathrm{dom}(\varphi_i)$; or the question whether $i \in \mathrm{dom}(\varphi_i)$. Based on RICE's Theorem, all three versions can be reduced to one another and are thus equivalent from the point of view of strict computability but in generally *not* concerning approximations; see Example 24.

Similarly, strict undecidability of H_φ holds independently of the underlying programming system whereas a change in φ may sensitively affect its error complexity. In fact one can artificially 'blow up and pad' any Gödelization to obtain one where already a constant answer yields exponentially small error to the Halting problem [Lync74, PROPOSITION 1]. While the *Padding Lemma* of Recursion Theory asserts any programming system to repeat each computable function an infinite number of times, these repetitions should occur in a 'balanced' way for the Gödelization to be reasonable.

Definition 5. *Gödelization* φ *is* **dense** *iff* $\forall i\, \exists c > 0 :\ \mathrm{density}\left(\{j : \varphi_i = \varphi_j\}, n\right) \geq_{ae} c$.

Another influence to the complexity of the Halting problem arises from the pairing function under consideration. Again, in order to avoid trivial approximations, we restrict to pairing functions which are **pair-fair** in the sense of [JaSc99, DEFINITION 5] and recall that for instance the standard pairing $\langle x,y\rangle = x + \frac{(x+y)(x+y+1)}{2}$ satisfies this condition. It has been proven that, under these natural restrictions, every heuristic claiming to solve the Halting problem makes at least a constant fraction of errors:

Theorem 6 ([JaSc99, THEOREM 4])**.** *Let* $\mathcal{REC}$ *denote the class of recursive languages and* φ *a dense Gödelization. Then* $\forall S \in \mathcal{REC}\ \exists \varepsilon > 0 :\ \mathrm{density}(H_\varphi \triangle S, n) \geq_{ae} \varepsilon$.

1.4 Own and Related Contributions

Observe that the lower approximation bound ε in Theorem 6 may in general depend on S; it seems thus still conceivable that H_φ admits an approximation *scheme* in the sense that better and better algorithms achieve smaller and smaller error densities. In fact the question whether or not there exists a *universal* constant lower bound was open for half a decade [JaSc99, bottom of p. 402].

The present paper gives both a positive and a negative answer to this question:

Theorem 7. *For any dense Gödelization* φ *it holds*
a) $\exists \varepsilon > 0\ \forall S \in \mathcal{REC} :\ \mathrm{density}(H_\varphi \triangle S, n) \geq_{io} \varepsilon$.
b) $\forall \varepsilon > 0\ \exists S \in \mathcal{REC} :\ \mathrm{density}(H_\varphi \triangle S, n) \leq_{io} \varepsilon$.

This complements [Lync74, PROPOSITION 6][5] where *ae*- rather than *io*-approximation is considered. In addition, our work differs from [Lync74] in treating the Halting problem H_φ *with* inputs as opposed to $\tilde{H}_\varphi$; see the discussion following Definition 4. Thirdly, we consider *dense* programming systems whereas [Lync74, p.147] requires them to be *optimal* in the sense of [Schn75] — a strictly[6] stronger condition:

Lemma 8. *Any optimal Gödelization* φ *is* dense *according to Definition 5.*

Proof. Start with some dense Gödelization φ'. In φ, fix an arbitrary index $i \in \mathbb{N}$. Thus $\varphi_i = \varphi'_{i'}$ for some $i' \in \mathbb{N}$. φ' being dense, the set $J' := \{j' : \varphi'_{i'} = \varphi'_{j'}\}$ has Density$(J', N) \geq_{ae} c$ for some $c > 0$. By definition of optimality there exists $C \in \mathbb{N}$ and to each j' some $j \leq C \cdot j'$ such that $\varphi_j = \varphi'_{j'}$. Hence, Density$(\{j : \varphi_i = \varphi_j\}, N) \geq_{ae} c/C$. □

The above differences (H_φ rather than $\tilde{H}_\varphi$, dense rather than optimal Gödel numberings) to [Lync74] are due to our interest in the Halting problem as arising *in practice*, that is, for real programming languages; see Sections 1.5 and 2. We focus on mere computability of according approximations; in particular our work is not related to the *restricted* Halting problem — *Given* (i,t)*, does Turing machine #i terminate after* $\leq t$ *steps?* — considered in [Mach78, SECTION 6.1] for complexity purposes.

1.5 Omega Numbers

Approximations to the Halting problem have been treated by encoding H into a single real $r \in \mathbb{R}$ and then considering computational approximations to this r. A first encoding goes back to [Spec49] in terms of the number $x[H] = \sum_{n \in H} 2^{-n}$ whose binary digits are obviously not decidable but semi-decidable, i.e., any 1 can be verified within finite time.

CHAITIN's Omega-Number [Chai87, LiVi97] gives another way of encoding the entire Halting problem into a single real $\Omega_U = \sum_{\bar{x} \in \mathrm{dom}(U)} 2^{-|\bar{x}|}$ where U denotes a universal Turing machine which is required to be self-delimiting. This implies by KRAFT's inequality that $\Omega_U \leq 1$ can be interpreted as the probability for U to terminate upon input of a random program. Ω_U is considered 'denser' and more difficult to approximate than $x[H]$ because its binary digits are not even semi-decidable; see, e.g., [LiVi97, CHKW01]. At first, it has therefore received noticable attention when [CDS01] did succeed in determining the first 64 bits of Ω_U.

However this approximation had been significantly simplified by the observation that, for the specific U considered in [CDS01], an overwhelming fraction of all instances do not contain a `halt` instruction at all and thus stall trivially. In other words, those program sources arising in practice form only a very sparse subset within the programming system treated there. In order to avoid such trivialities and instead obtain meaningful results about the possibility or impossibility of approximations to the Halting problem, we now present:

2 A Particularly Compact, Practical Dense Programming System

Concerning the applicability of Theorem 6, its prerequisite is satisfied by every Turing-complete programming language over alphabet Σ with some kind of *end-of-string* (eof) indicator. More generally it holds:

Example 9. Let $\varphi = (\varphi_{\bar{x}})_{\bar{x} \in \Sigma^*}$ denote a Gödelization which is *self-delimiting* in the sense that, whenever $\varphi_{\bar{x}}$ does not identically diverge, it holds $\varphi_{\bar{x}} = \varphi_{\bar{x} \circ \bar{y}}$ for all $\bar{y}$. Then, φ is dense. This includes, for arbitary Gödelization $\psi = (\psi_n)_{n \in \mathbb{N}}$, the 'tally'[3] re-coding

[3] See [Book74]. Also, this dense Gödelization is obviously non-*optimal* in the sense of [Schn75].

$$\varphi_{1^n0} := \psi_n, \qquad \varphi_{1^n0\bar{x}} := \psi_n \text{ for } \bar{x} \in \{0,1\}^*, \qquad \varphi_{\bar{x}} :\equiv \bot \text{ for } \bar{x} \in \{1\}^* \ .$$

While a special symbol ⟨eof⟩ may always be added to Σ, we consider this cheating. Also from the practical side, compilers for programming languages nowadays rely on the end-of-file being indicated by the operating system (e.g., via `feof`) as opposed to the out-dated detection of characters like nul, `^D`, or `^Z`. In the present section we analyze and establish a practical, *non*-self delimiting programming language to be dense.

2.1 The BF Programming Language

BF ('BrainF*ck') was designed in 1993 by URBAN MÜLLER and has since then spread the Internet for its shrewd simplicity [Wiki05]. It is a Turing-complete programming language over the eight letter alphabet $\Sigma_{\text{BF}} = \{\boxed{<}, \boxed{>}, \boxed{+}, \boxed{-}, \boxed{,}, \boxed{.}, \boxed{[}, \boxed{]}\}$. The first six characters represent commands, the remaining two brackets are used to construct simple loops.

A BF-program stores data on a tape similar to that of a Turing-Machine. Each cell of the tape may contain an integer between 0 and 255, that is, one byte. The current cell may be incremented using $\boxed{+}$ and decremented with $\boxed{-}$; (incrementing 255 will result in 0, decrementing 0 will result in 255). Other cells can be accessed by moving the read/write-head either to the left $\boxed{<}$ or to the right $\boxed{>}$. Initially, all cells are set to 0. The two commands $\boxed{,}$ and $\boxed{.}$ are for input and output: $\boxed{,}$ will fetch a byte from the input stream and store it into the current cell; $\boxed{.}$ appends the byte in the current cell to the output stream.

Loops are formed by putting commands inbetween the two bracket symbols $\boxed{[}$ and $\boxed{]}$. Each time the loop is about to be executed, the current cell is checked whether it contains a value other than 0. If so, the loop is executed again. The commands in the loop are skipped, if the current cell was 0. Note, that after each round of the loop, another cell could have been made current by the commands within the loop.

Definition 10. *Let $\mathcal{BF}_n \subseteq \Sigma_{\text{BF}}^n$ denote the set of strings $\bar{p}$ of length n representing a syntactically correct BF source code and $\mathcal{BF} = \bigcup_n \mathcal{BF}_n$.*

Observe that the syntax of this programming language is quite simple, the only requirement being that opening and closing brackets are nested correctly.

Remark 11. BF is sometimes refered to with a fixed tape of 30.000 cells size. However the level of standardization is not very advanced yet. In order to obtain a Turing-complete system, we shall assume an unbounded tape.

2.2 Naive Encoding of BF

This straight-forward idea takes BF source codes as Gödel indices:

Definition 12. *For $\bar{p} \in \Sigma_{\text{BF}}^*$, let $\psi_{\bar{p}}$ denote the function obtained by interpreting $\bar{p}$ as source code for some BF program; $\psi_{\bar{p}} :\equiv \bot$ in case $\bar{p}$ lacks syntactical correctness.*

However, closer analysis reveals that this programming system is *not* dense:

Theorem 13. *Let* $\mathrm{BF}_n := |\mathcal{BF}_n|$ *denote the number of correct* BF*-sources of length n.*

a) $\mathrm{BF}_{n+1} = 6 \cdot \mathrm{BF}_n + \sum_{i=0}^{n-1} \mathrm{BF}_i \cdot \mathrm{BF}_{n-1-i}$.

b) $\mathrm{BF}_n = \sum_{k=0}^{n/2} C_k \cdot \binom{n}{2k} \cdot 6^{n-2k}$, *where* $C_k = \frac{1}{k+1} \cdot \binom{2k}{k}$ *denotes* CATALAN*'s number.*

c) $(n+3) \cdot \mathrm{BF}_{n+1} = (12n+18) \cdot \mathrm{BF}_n - 32n \cdot \mathrm{BF}_{n-1}$.

d) $\mathrm{BF}_{n+1} \leq 8 \cdot \mathrm{BF}_n$.

e) $\mathrm{BF}_n \leq O(8^n/\sqrt{n})$.

Thus among all 8^n strings $\bar{p} \in \Sigma_{\mathrm{BF}}^n$, the fraction of syntactically correct sources tends to zero, permitting for H_ψ a trivial $O(\frac{1}{\sqrt{n}})$-approximation.

Proof. a) A syntactically correct BF program of length $n+1$ either consists of one (out of 6 possible) non-loop character followed by an, again syntactically correct, program of length n; or, in case it begins with the loop character [, it consists of a loop (whose body is a syntactically correct program of length i for some $i < n$) followed by some other syntactically correct source of length $n-1-i$.

b) Consider the collection $\mathcal{BF}_{n,k} \subseteq \mathcal{BF}_n$ of BF programs $\bar{p} \in \Sigma_{\mathrm{BF}}^n$ of length n with $0 \leq k \leq n/2$ occurrences of [or, equivalently, of]. Then, C_k equals the number of correct ways of nesting $2k$ brackets [Bail96]. Any $\bar{p} \in \mathcal{BF}_{n,k}$ can be obtained in a unique way by choosing $2k$ out of n positions in $\bar{p}$ for placing these brackets and by filling each of the remaining $n-2k$ positions independently with one out of the 6 non-bracket characters in Σ_{BF}.

c) follows from a) and b) by induction.

d) Claim c) immediately yields $\mathrm{BF}_{n+1} \leq c_0 \cdot \mathrm{BF}_n$ by induction, where $c_0 := 12$. Repeated application of c) establishes a sequence of improved bounds $\mathrm{BF}_{n+1} \leq c_k \cdot \mathrm{BF}_n \ \forall n$ with (c_k) decreasing down to 8.

e) Combining c+d), obtain $BF_{n+1} \leq 8 \cdot \frac{n+\frac{9}{4}}{n+3} \cdot \mathrm{BF}_n \leq 8^n \cdot \prod_{i=1}^{n} \frac{i+\frac{9}{4}}{i+3}$,

$$\prod_{i=-2}^{n} \frac{i+\frac{9}{4}}{i+3} = \prod_{j=1}^{n+3} \frac{j-\frac{3}{4}}{j} \leq \prod_{j=1}^{n+3} \frac{j-\frac{1}{2}}{j}, \quad \Big(\prod_{j=1}^{n} \frac{2j-1}{2j}\Big)^2 \leq \Big(\prod_{j=1}^{n} \frac{2j-1}{2j}\Big) \cdot \Big(\prod_{j=1}^{n} \frac{2j}{2j+1}\Big) = \frac{1}{2n+1}$$

□

2.3 Compact Encoding of BF

Regarding Theorem 13, a dense programming system based on BF better avoids enumerating syntactically incorrect sources. This leads to the following

Definition 14. *Define* φ_N *to denote the function computed by the N-th syntactically correct* BF *program* $\bar{p}_N$*. More formally, let* $\mathcal{BF}$ *be ordered primarily with respect to length n and secondarily according to the enumeration given by recursive application of Theorem 13a), that is, by first listing the* $6 \cdot \mathrm{BF}_n$ *programs starting with no loop and then listing, recursively and for each* $i = 0 \ldots n-1$*, the loop bodies and loop tails as* BF *sources of length i and* $n-1-i$*, respectively.*

Although this programming system is *not* self-delimiting, it holds:

Theorem 15. *The Gödelization* φ *from Definition 14 is dense.*

We emphasize that this is by no means a consequence of syntactical correctness alone!

Proof. Fix a partial recursive function computed by some BF source $\bar{p} \in \mathcal{BF}_m$ of length $m = |\bar{p}|$. For $n \geq m+2$, we construct BF_{n-m-2} equivalent programs $\bar{p}' \in \mathcal{BF}_n$. To this end preced $\bar{p}$ with a loop, i.e., let $\bar{p}' := \boxed{[} \circ \bar{q} \circ \boxed{]} \circ \bar{p}$ for an arbitrary syntactically correct source $\bar{q}$ of length $n-m-2$. Since, upon start of execution, the current cell is initialized to 0, this loop gets skipped anyway and $\bar{p}'$ thus behaves like $\bar{p}$, indeed. The thus obtained sources $\bar{p}'$ constitute, in relation to BF_n and by Theorem 13d), a fraction

$$\frac{\mathrm{BF}_{n-m-2}}{\mathrm{BF}_n} \;\geq\; \frac{\mathrm{BF}_{n-m-2}}{8^{m+2} \cdot \mathrm{BF}_{n-m-2}} \;=\; 8^{-m-2} \;=:\; c \;>\; 0$$

among all programs of length $n \geq n_0 := m+2$. Now proceed as in Lemma 2c). □

Conversion between BF sources and Gödel indices is a central part of efficient (rather than merely computable) SMN- and UTM-properties according to Definition 4. A naive approach enumerates *all* strings $\bar{p} \in \Sigma_{\mathrm{BF}}^n$ and counts the syntactically correct ones in order to obtain $\bar{p}_N$. This, however, gives rise to exponential time in $\log N$. The following result improves to running time polynomial in the input size, that is, $|\bar{p}|$ or $\log N$:

Theorem 16. *Given a program $\bar{p} = \bar{p}_N \in \mathcal{BF}_n$ of length n, one can calculate its index $N \in \mathbb{N}$ according to Definition 14 within time $O(n^3 \cdot \log n \cdot \log\log n)$. Conversely, from N, the according $\bar{p}_N$ is computable using $O(n^3 \cdot \log n \cdot \log\log n)$ steps where $n = \log N$. Both algorithms use memory of size $O(n^2)$. See also* `http://www.upb.de/cs/bf`

To conclude, the Gödelization introduced in this section is practical, efficient, and dense. It even seems plausible to satisfy the stronger condition of *optimality*; recall Lemma 8. To this end one might establish a sparse SMN-property for BF as required in

Lemma 17. *Let $\varphi = (\varphi_{\bar{p}})_{\bar{p} \in \Sigma^*}$ denote a Gödelization and SMN-function $s : \Sigma^* \times \Sigma^* \to \Sigma^*$ according to Definition 4 satisfying $|s(\bar{p},\bar{x})| \leq c(\bar{p}) + |\bar{x}|$ for all $\bar{p},\bar{x} \in \Sigma^*$ with arbitrary $c : \Sigma^* \to \mathbb{N}$. Then, φ is* optimal *in the sense of* [Schn75].

Proof. Fix some other Gödelization Φ. Consider its UTM-function Φ_U and let u' denote the index of Φ_U *in* φ; i.e., $\forall \bar{x} \in \Sigma^* : \quad \Phi_{\bar{P}}(\bar{x}) = \Phi_U(\langle \bar{P}, \bar{x} \rangle) = \varphi_{u'}(\langle \bar{P}, \bar{x} \rangle) = \varphi_{\bar{p}}(\bar{x})$ where $\bar{p} := s(u', \bar{P})$ has by prerequisite length $|\bar{p}| \leq c(u') + |\bar{P}| = c_0 + |\bar{P}|$. □

3 The Error Complexity of Dense Programming Systems

In the last section we showed that a natural encoding of BF is dense. From Theorem 6 it follows that every algorithm A trying to solve the Halting problem of such a dense programming system errs on at least a constant fraction $\varepsilon_A > 0$. This constant fraction $\varepsilon_A > 0$ may depend on the algorithm A and can be arbitrarily small. In this section we will show that there is a universal constant $\varepsilon_0 > 0$ lower bounds the error made by any heuristic trying to approximate the Halting problem for a dense Gödelization.

3.1 Halting Ratio

A straight-forward implication of Theorem 6 is that neither nearly all programs halt nor do nearly all of them stall. This is formalized as follows:

Definition 18. *Call* $h_\varphi : N \mapsto \text{Density}\big(\{\langle i,x\rangle : x \in \text{dom}(\varphi_i)\}, N\big)$ *the* ***halting ratio****.*

Like Ω_U (see Section 1.5), h_φ describes a probability for a random instance to halt.

Corollary 19. *For every dense Gödelization* φ, $\exists c > 0 : \; c \leq_{ae} h_\varphi \leq_{ae} 1 - c$.

Proof. Consider two indices i, j with $\text{dom}(\varphi_i) = \Sigma^*$ and $\text{dom}(\varphi_j) = \emptyset$. Because of the dense programming system and the pair-fair pairing, these indices alone induce a constant fraction of halting and non-halting indices. □

It seems desirable, again similarly to Section 1.5, to investigate the real number $r_\varphi := \lim_{n\to\infty} h_\varphi(n)$. However in many cases h_φ fails to converge:

Example 20. Take any programming system $\psi = (\psi_i)_{i\in\mathbb{N}}$ and define $\varphi = (\varphi_I)_{I\in\mathbb{N}}$ by

$$\varphi \quad = \quad (\psi_1, \;\; \psi_2, \psi_2, \psi_2, \psi_2, \;\; \ldots\ldots, \; \underbrace{\psi_i, \psi_i, \psi_i, \ldots, \psi_i, \psi_i,}_{i^i \text{ times}} \;\; \ldots\ldots\ldots) \; .$$

Obviously φ_I behaves identically for all I within a block arising from the same ψ_i. Since the size i^i of such a block dominates by far those of all previous blocks together, namely $N_i = 1+4+\ldots+(i-1)^{i-1} \leq (i-1)^0+(i-1)^1+\ldots+(i-1)^{i-1} = \frac{(i-1)^i-1}{i-2} \leq \frac{1}{i-2}\cdot i^i$,

a) termination of ψ_i determines whether $h_\varphi(N_i)$ is (arbitarily close to) 1 or 0. In particular, h_φ is an oscillating function and fails to converge for $N \to \infty$.
b) As infinitely many instances of H_ψ are undecidable, so is almost every entire block of H_φ. In particular, $\forall \varepsilon > 0 \, \forall S \in \mathcal{REC} : \; \text{Density}(H_\varphi \triangle S, N) \geq_{io} 1 - \varepsilon$.
c) On the other hand, $S := \emptyset \in \mathcal{REC}$ satisfies $\forall \varepsilon > 0 : \; \text{Density}(H_\varphi \triangle S, N) \leq_{io} \varepsilon$. Indeed, each of the infinitely many i corresponding to stalling instances of H_ψ yields an entire block of them in H_φ, dominating $\text{Density}(H_\varphi, N_i + i^i) \leq \frac{1}{i-2}$ as above. □

With the last two properties, this specific Gödelization concretizes the REMARK on top of p.147 in [Lync74]. Compare them to *io*-approximations of arbitrary dense programming systems according to Theorem 7.

3.2 Relation Between Two Approximations

Consider the question of approximating the function $h_\varphi : \mathbb{N} \to \mathbb{Q}$. This is related to the approximation of the Halting problem in the sense of Section 1.2 as follows:

Lemma 21. *Fix Gödelization* φ *with Halting problem* $H = H_\varphi$ *and halting ratio* $h = h_\varphi$.

a) Given $N \in \mathbb{N}$, $\varepsilon \in \mathbb{Q}$, *and* $b \in \mathbb{Q}$ *with* $|b - h(N)| \leq \varepsilon$, *one can compute a list* $H'_N \subseteq H \cap \{0,1,\ldots,N-1\}$ *of halting instances satisfying* $\text{Density}(H \triangle H'_N, N) \leq \varepsilon$.

b) Let $S \subseteq \mathbb{N}$ *be arbitrary. Given* $N \in \mathbb{N}$ *and* $\varepsilon \in \mathbb{Q}$ *such that* $\text{Density}(H \triangle S, N) \leq \varepsilon$, *an* S*-oracle machine can compute* $b \in \mathbb{Q}$ *with* $|b - h(N)| \leq \varepsilon$.

In particular for this φ *and any* $\varepsilon > 0$, *the Halting problem* H_φ *can* ***ae*** *be* ε*-approximated iff the halting ratio* h_φ *can* ***ae*** *be* ε*-approximated; analogously for approximating* ***io****.*

Proof. a) Recursively enumerate elements $x \in H \cap \{0,1,\ldots,N-1\}$ until having obtained a collection $H'_N \subseteq H$ of cardinality $\#H'_N \geq (b-\varepsilon)\cdot N$.

b) By repeatedly quering the oracle S for all finitely many $x \in \{0,1,\ldots,N-1\}$, calculate the number $b := \#(S \cap \{0,1,\ldots,N-1\})/N$. □

Corollary 22. *Fix computable $f : \mathbb{N} \to \mathbb{Q}$ and recursively enumerable $L \subseteq \mathbb{N}$ admitting (ae/io) an $f(N)$-approximation with two-sided error. Then L can (ae/io) be $f(N)$-approximated with one-sided error.*

Proof. W.l.o.g. $L = H_\varphi = H$ for some φ. Let $S \subseteq \mathbb{N}$ denote a recursive two-sided $f(N)$-approximation of H. Upon input of x, compute $\varepsilon := f(N)$, $N \geq |x|$; then obtain an approximation $b \in \mathbb{Q}$ for $h(N)$ by virtue of Lemma 21b), observing that oracle queries to S can be decided by presumption. Then apply Lemma 21a) to get a some $H'_N \subseteq H \cap \{0,\ldots,N-1\}$ with $\mathrm{Density}(H \triangle H'_N, N) \leq \varepsilon$, i.e., one-sided ε-approximation. □

3.3 Approximating the Halting Ratio

We now reveal that the Halting ratio of a dense programming system infinitely often admits a well approximation and infinitely often it does not.

Lemma 23. *Fix a dense programming system φ.*

a) For all $\varepsilon > 0$, there exists a TM M such that $|M(n) - h_\varphi(n)| \leq_{io} \varepsilon$.

b) There exists $\varepsilon > 0$ such that all TMs M have $|M(n) - h_\varphi(n)| \geq_{io} \varepsilon$.

Proof. a) For fixed $\varepsilon > 0$ consider $k := \lceil 1/\varepsilon \rceil$ and the $k+1$ constant (trivially computable) functions $0, \frac{1}{k}, \frac{2}{k}, \frac{3}{k}, \ldots, 1$. For every input length n, at least one of these values differs from $h_\varphi(n) \in [0,1]$ by at most ε. A second application of pidgeon-hole's principle yields that some of these constant functions is close to h_φ for infinitely many n.

b) For a fixed rational $\varepsilon > 0$ (whose actual value we determine later) we assume $\varphi_i(n)$ for some $i \in \mathbb{N}$ is a candidate for computing $h_\varphi(n)$.
Now an algorithm A computes on input $z \in \Sigma^n$ the following. First it computes $b = \varphi_n(n)$ as an approximation to $h_\varphi(n)$ on input $z = \langle i,x \rangle \in \Sigma^n$. Let $f_1(z) = i$ and $f_2(z) = x$ be the decoding functions of $z = \langle i,x \rangle$. Let i^* be an index of A. According to the Recursion Theorem, A may know its own index and $\varepsilon \in \mathbb{Q}$. Then the algorithm simulates all inputs to H_φ of length n in parallel step by step until $(b-\varepsilon)|\Sigma|^n$ strings $y \in \Sigma^n$ have been found with $f_2(y) \in \mathrm{dom}(\varphi_{f_1(y)})$ and $f_1(y) \neq i^*$. Let s denote this number of halting inputs $\langle i,x \rangle \in \Sigma^n$ with $i \neq i^*$ found by A. If $s \geq (b-\varepsilon)|\Sigma|^n$ then the algorithm halts, else the algorithm does not halt.
There is a chance that the algorithm does not halt before this last condition, which means that $n \notin \mathrm{dom}(\varphi_n)$ or less than $(b-\varepsilon)|\Sigma|^n$ strings of length n corresponding to halting instances exist. In both cases φ_n was proven not to compute h_φ within the error margin ε.
Recall that i^* is an index for algorithm A and that the repetition rate of i^* is constant. The diagonalization argument is that all inputs $\langle i^*,x \rangle$ will result in $a(i^*,n) := |\{\langle i^*,x \rangle \in \Sigma^n\}|$ additional halting inputs of length n compared to $(b-\varepsilon)|\Sigma|^n$ where φ_n predicted at most $(b+\varepsilon)|\Sigma|^n$ halting instances. For large enough n, this number $a(i^*,n)$ is lower bounded by $\Omega(|\Sigma|^n)$, i.e. $\exists c > 0 : \forall_{ae} n : a(i^*,n) > c|\Sigma|^n$, because of the pair-fair property of $\langle \cdot,\cdot \rangle$.

Now if $s \geq (b-\varepsilon)|\Sigma|^n$ then there are at least $(b-\varepsilon+c)|\Sigma|^n$ halting instances for almost all input lengths n where $\varphi_n(n)$ is a candidate for $h_\varphi(n)$. Note that there are infinite many equivalent machines $n_1, n_2, \ldots$, with $\varphi_{n_i} = \varphi_n$. For $\varepsilon < c/2$ this implies that φ_n errs infinitely often on these inputs of length n_i with an error margin of at least ε (which can be determined independent from n).
Therefore for all machines $M = \varphi_n$ there are infinitely many input lengths n such that $M(n)$ does not approximate $h_\varphi(n)$ by an additional error term of ε. □

3.4 The Halting Problem is *ae*-hard and *io*-easy

Combining Lemma 21 and Lemma 23 establishes the already announced

Theorem 7. *For any dense Gödelization* φ *it holds*
a) $\exists \varepsilon > 0 \ \forall S \in \mathcal{RTC} : \ \text{density}(H_\varphi \triangle S, n) \geq_{io} \varepsilon$.
b) $\forall \varepsilon > 0 \ \exists S \in \mathcal{RTC} : \ \text{density}(H_\varphi \triangle S, n) \leq_{io} \varepsilon$.

This solves the open problem stated in [JaSc99] for the case of dense programming system. In particular, it shows that the dense encoding of BF from Section 2.3 provides a natural hard problem which cannot be approximated better than up to a constant factor.

Furthermore, Theorem 7 nicely complements [Lync74, PROPOSITION 6]. Observe that Claim b) there only seems to be stronger than our Theorem 7a) because of the more restrictive presumption that the Gödelization φ under consideration be *optimal* in the sense of [Schn75] rather just dense.

In addition, [Lync74] refers to the Halting problem as termination of φ_i on the special input i (that is, in our notation, to $\tilde{H}_\varphi$; see Definition 4) whereas we treat the more general and practically relevant H_φ, i.e., termination of φ_i on given input x. Although both problems are equivalent with respect to exact computability, their behaviour concerning approximations differs significantly. This can be observed already in the proof of Lemma 23b) which heavily relies on the described algorithm A's behaviour to depend on (the length of) its input. More explicitly, we have the following

Example 24. Consider the dense tally Gödelization φ in Example 9. There, any ψ_n gives rise to an asymptotic 2^{-n-1}-fraction of equivalent instances $\varphi_{\bar{x}}$. Thus, storing the solutions to H_ψ for the first N inputs $\psi_1, \ldots, \psi_N$ allows for *ae* answering correctly a fraction $\varepsilon_N = \sum_{n=1}^{N} n \cdot 2^{-n-1}$ of instances to $\tilde{H}_\varphi$ with $\varepsilon_N \to 1$ as $N \to \infty$.

4 Conclusion

Since the Halting problem is of practical importance yet cannot be solved in the strict sense, we considered the possibility of approximating it. Similarly to the average-case theory of complexity, this depends crucially on the encoding of the problem, that is here, the programming system under consideration.

Many practical programming languages lacking density in fact do admit such an approximation with asymptotically vanishing relative error for the simple reason that the fraction of syntactically incorrect instances tends to 1. This was exemplified by a combinatorial analysis of the Turing-complete formal language BF. Here and in similar

cases, the question for approximation the Halting problem is equivalent to a mere syntax check and thus becomes trivial and vain.

On the other hand, considering only syntactically correct sources was established to yield an efficient *and* dense programming system in the case of BF. For any such system, we proved a universal constant lower bound on relative approximations to the Halting problem even in the weak *io*-sense. Our third contribution establishes that, conversely, any constant relative error $\varepsilon > 0$ is *io* feasible by an appropriate machine M.

Question 25. Is there some optimal (but necessarily non-dense) programming system φ whose Halting problem H_φ satisfies the following even stronger inapproximability property similar to [Lync74, PROPOSITION 2]

$$\forall S \in \mathcal{REC} \quad \forall \varepsilon > 0 : \quad \text{density}(H_\varphi \triangle S, n) \geq_{io} 1 - \varepsilon \qquad \text{or even} \quad \geq_{ae} 1 - \varepsilon \ ?$$

Observe that [Lync74, PROPOSITION 6] reveals the answer to be negative concerning the Halting problem $\tilde{H}_\varphi$ *without* input which, regarding Example 24, tends to be strictly easier to approximate than H_φ anyway.

Another open problem, it remains whether BF leads in Section 2.3 to an even optimal (rather than just dense) Gödelization; cf. Lemma 8. Furthermore it is conceivable — although by no means obvious — that the programming system Jot by C. BARKER is dense as well; see `http://ling.ucsd.edu/~barker/Iota/#Goedel`.

References

[Bail96] D.F. BAILEY: "Counting Arrangements of 1's and -1's", pp.128–131 in *Mathematics Magazine* vol.**69** (1996).

[Book74] R.V. BOOK: "Telly languages and complexity classes", pp.186–193 in *Information and Control* vol.**26:2** (1974).

[CDS01] C.S. CALUDE, M.J. DINNEEN, C.-K. SHU: "Computing a Glimpse of Randomness", pp.361–370 in *Experimental Mathematics* vol.**11:3** (2001).

[CHKW01] C.S. CALUDE, P. HERTLING, B. KHOUSSAINOV, Y. WANG: "Recursively enumerable reals and Chaitin Ω numbers", pp.125–149 in *Theoretical Computer Science* vol.**255** (2001).

[Chai87] G.J. CHAITIN: *Algorithmic Information Theory*, Cambridge University Press (1987).

[Gold97] O. GOLDREICH: "Combinatorial property testing (a survey)", pp.45–59 in *Proc. DIMACS Workshop in Randomized Methods in Algorithm Design* (1997).

[JaSc99] A. JAKOBY, C. SCHINDELHAUER: "The Non-Recursive Power of Erroneous Computation", pp.394–406 in *Foundations of Software Technology and Theoretical Computer Science* (FSTTCS 1999), Springer LNCS vol.**1738**.

[Koeh04] S. KÖHLER: "*Zur Approximierbarkeit des Halteproblems in einer praktischen Gödelisierung*", Bachelor's Thesis, University of Paderborn (2004).

[LiVi97] M. LI, P. VITÁNI: *An Introduction to Kolmogorov Complexity and its Application*, 2nd Edition, Springer (1997).

[Lync74] N. LYNCH: *Approximations to the Halting Problem*, pp.143–150 in *J. Computer and System Sciences* vol.**9** (1974).

[Mach78] M. MACHTEY, P. YOUNG: *An Introduction to the General Theory of Algorithms*, The Computer Science Library (1978).

[Papa95] C.H. PAPADIMITRIOU: *Computational Complexity*, Addison-Wesley (1995).
[Roge67] H. ROGERS JR: *Theory of Recursive Functions and Effective Computability*, McGraw Hill (1967).
[RoUl63] G.F. ROSE, J.S. ULLIAN: "Approximation of Functions on the Integers", pp.693–701 in *Pacific Journal of Mathematics* vol.**13:2** (1963).
[Schn75] C.P. SCHNORR: "Optimal Enumerations and Optimal Gödel Numberings", pp.182–191 in *Mathematical Systems Theory* vol.**8:2** (1975).
[Smit94] C. SMITH: *A Recursive Introduction to the Theory of Computation*, Springer (1994).
[Soar87] R.I. SOARE: *Recursively Enumerable Sets and Degrees*, Springer (1987).
[Spec49] E. SPECKER: "Nicht konstruktiv beweisbare Sätze der Analysis", pp.145–158 in *J. Symbolic Logic* vol.**14:3** (1949).
[Wiki05] `http://wikipedia.org/wiki/BrainFuck`; Wikipedia, the free encyclopedia (2005)