Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 221 (2008) 207–217
www.elsevier.com/locate/entcs

Lower Bounds on the Continuation of Holomorphic Functions

Robert Rettinger[1]

Lehrgebiet Algorithmen und Komplexität
FernUniversität Hagen
D-58084 Hagen
Germany

Abstract

We show that continuation of holomorphic functions needs time at least $\Omega(n^2)$ in the uniform case, which is optimal up to a polylogarithmic factor. In the non-uniform case, i.e. assuming f to be computable in time $O(t)$ on an open subset of its domain, we show, that continuation of f cannot be robustly done in time $o(n \cdot t(n))$. This again is optimal up to a polylogarithmic factor.

Keywords: Type-2 theory, continuation, holomorphic functions

1 Introduction

It is well known that a holomorphic function f is uniquely determined by its values on any open subset of its (connected) domain. The question arises, whether a similar property also holds for computability or even complexity aspects of holomorphic functions, i.e. is a given holomorphic function, which is (efficiently) computable on a small open subset, also (efficiently) computable on all of its domain?

This continuation problem can be solved in time essentially $O(n^2)$ (uniformly and up to a polylogarithmic factor) e.g. by following the ideas of [7] (see also [5]) and using FFT methods for interpolation (see e.g. [3],[2]). In the non-uniform case where we assume f to be a holomorphic function f which is computable in time $O(t)$, this gives an upper time bound of essentially $O(n \cdot t(O(n)))$ (again up to a polylogarithmic factor). (We will give detailed definitions below.)

This rises the question, whether there exist further hidden symmetries of analytic functions, which allow faster continuation than stated above. In this paper we will show that this is not true in the general case, and even if we restrict ourself to

[1] Email: robert.rettinger@fernuni-hagen.de

doi:10.1016/j.entcs.2008.12.018

polynomial time computable mappings, such symmetries would have to heavily rely on complexity aspects of the machine model rather than on general analytic properties. The latter is the best we can say about this case, unless we prove some seemingly very hard and long standing conjectures in classical complexity theory. More precisely, we show that the upper bounds above are sharp. Whereas the lower bound in the uniform case follows easily from a combinatoric lemma which we will give in the next section, the non-uniform case is much harder to prove, and we will indeed only show that in this case the lower bound is optimal in a relativized setting. Beside giving the lower bounds in terms of the precision n we will also present lower bounds depending on further parameters.

Next we will introduce the main notions on representations and complexity, using essentially Type-2-Theory of computability and Ko's machine model (see [6]). In Section 2 we will consider holomorphic functions on the unit disc, starting with a combinatorical lemma in Section 2.1 and giving the lower bounds in Section 2.2 afterwards. Finally, in Section 3, we show how these lower bounds can be extended to more general domains.

Let Σ be a finite alphabet, $0,1 \in \Sigma$. Furthermore let $\langle\cdot,\cdot\rangle$ be an efficient pairing function on Σ^*, e.g. $\langle u,v\rangle = 0^{|u|}1uv$. We denote by Σ^{**} the set of all (total) functions $x : \Sigma^* \to \Sigma^*$ and extend the pairing function $\langle\cdot,\cdot\rangle$ to Σ^{**} by $\langle p,q\rangle(\langle u,v\rangle) := \langle p(u),q(v)\rangle$ (for all $p,q \in \Sigma^{**}$ and $u,v \in \Sigma^*$).

In type-2 theory computability and complexity on spaces X are introduced via type-2 Turing machines and representations of X, i.e. surjective (partial) functions $\nu :\subseteq \Sigma^\omega \to X$. Using Cantor space in the definition of representations works perfectly well for computability aspects and/or complexity on sets like $\mathbb{R}$ or $\mathbb{C}$. For function spaces, however, using this definition of representations seems to be unnaturally restricting. For such spaces the oracle machine model used in [6] seems to be much more adequate. We use essentially this second model by using Σ^{**} instead of Σ^ω in the definition of representations, i.e. in the sequel we will consider a representation of a set X to be a surjective function $\nu :\subseteq \Sigma^{**} \to X$ or $\nu :\subseteq \Sigma^* \to X$. Computability and complexity on such objects are then defined as usually once we have defined computability and complexity on Σ^{**}.

To this end we will consider oracle Turing machines, which are (multi-tape) Turing machines with an extra oracle state and two extra oracle tapes. Given an oracle $A : \Sigma^* \to \Sigma^*$, such a machine works as usually until it enters the oracle state. In this case the contents x of the first oracle tape is cleared and the contents of the second oracle tape is replaced by $A(x)$, and the machine continues as usually. In the definition of its time complexity such oracle steps are counted as single steps. Now we say that a function $f :\subseteq \Sigma^{**} \to \Sigma^{**}$ is computable in time $t : \Sigma^{**} \times \mathbb{N} \to \mathbb{N}$ if there exists some oracle Turing machine M so that for each oracle $x \in \mathrm{dom}(f)$ and any $y \in \Sigma^*$ M computes $(f(x))(y) \in \Sigma^*$ in time $t(x,|y|)$.

Given sets X and Y and representations $\nu :\subseteq \Sigma^{**} \to X$, $\nu' :\subseteq \Sigma^{**} \to Y'$ we say that a function $f :\subseteq X \to Y$ is (ν,ν')-computable in time $t : X \times \mathbb{N} \to \mathbb{N}$ iff there exist functions $g :\subseteq \Sigma^{**} \to \Sigma^{**}$ and $T : \Sigma^{**} \times \mathbb{N} \to \mathbb{N}$ so that g is computable in time T and furthermore, for each $x \in \mathrm{dom}(f)$ each $w \in \nu^{-1}(x)$ and each $n \in \mathbb{N}$, we

have $\nu'(g(w)) = f(x)$ and $T(w,n) \leq t(x,n)$.

Next we will fix standard representations for those sets we will use later on (we will frequently implicitly use these representations without further mentioning). To start with, let $\mathbb{Y}$ denote the set of dyadics, i.e. $\mathbb{Y} = \{k \cdot 2^{-l} \mid k,l \in \mathbb{Z}\}$. We take the signed digit representation as our standard representation $\nu_{\mathbb{Y}}$ for $\mathbb{Y}$, i.e. let $\nu_{\mathbb{Y}} :\subseteq \Sigma^* \to \mathbb{Y}$ with $\Sigma = \{-1,0,1,|\}$ be defined by $\nu_{\mathbb{Y}}(a_1...a_n|b_1...b_m) = \sum_{i=1}^{n} a_i 2^{n-i} + \sum_{i=1}^{m} b_i 2^{-i}$ for all $a_1,...a_n,b_1,...,b_m \in \{0,1,-1\}$. Using the pairing function $\langle\cdot,\cdot\rangle$ we get immediately a standard representation $\nu_{\mathbb{Y}[\hat{\imath}]}$ of the complex dyadics $\mathbb{Y}[\hat{\imath}]$ by identifying the complex dyadics by pairs of dyadics as usual ($\hat{\imath}$ denotes the square root of -1). The signed digit representation $\nu_{\mathbb{R}}$ of $\mathbb{R}$ is then defined by $\nu_{\mathbb{R}}(p) = x$ iff for each $u \in \mathrm{dom}(\nu_{\mathbb{Y}})$ we have $|\nu_{\mathbb{Y}}(p(u)) - x| < 2^{-|u|}$ for all $p \in \Sigma^{**}$, $x \in \mathbb{R}$ and $u \in \Sigma^*$. Similarly our standard representation $\nu_{\mathbb{C}}$ of $\mathbb{C}$ is determined by $\nu_{\mathbb{C}}(p) = x$ iff for each $u \in \mathrm{dom}(\nu_{\mathbb{Y}[\hat{\imath}]})$ we have $|\nu_{\mathbb{Y}[\hat{\imath}]}(p(u)) - x| < 2^{-|u|}$ for all $p \in \Sigma^{**}$, $x \in \mathbb{R}$ and $u \in \Sigma^*$.

Finally, we need representations of holomorphic functions. Though defining such representations in general is a difficult task, we can take a straight forward representation in our case. This is mainly due to the fact that in order to treat continuation in a reasonable way one has to restrict the class of holomorphic functions by certain bounds on the absolute values they can take on their domain. As any bounded holomorphic function can be easily translated to a holomorphic function bounded by 1, we will restrict ourself to this case. Given a bounded domain $G \subseteq \mathbb{C}$ let H_G denote the class of holomorphic functions $f : G \to \mathbb{D}$, where $\mathbb{D}$ denotes the unit disk $\mathbb{D} = \{z \in \mathbb{C} \mid |z| < 1\}$. For each domain $G' \subseteq G$ we define a representation $\nu_{G,G'}$ of H_G by means of evaluations, i.e. let $\nu_{G,G'}(p) = f$ iff for each $u \in \Sigma^*$ with $\nu_{\mathbb{Y}[\hat{\imath}]}(u) \in G'$ we have $|\nu_{\mathbb{Y}[\hat{\imath}]}(p(u)) - f(\nu_{\mathbb{Y}[\hat{\imath}]}(u))| < 2^{-|u|}$. In most cases we will choose G' to be some disk $\mathbb{D}_\varepsilon := \{z \in \mathbb{C} \mid |z| < \varepsilon\}$ or disks $\mathbb{D}_\varepsilon^{\delta_G} := \{z \in \mathbb{C} \mid \delta_G(z,0) < \varepsilon\}$ where we assume $0 \in G$ and δ_G is the hyperbolic metric on G.

Before we end this section a few remarks are in common. The oracle machine model we introduced can be easily extended to have multiple oracles. Furthermore we treated oracles as functions whereas in literature often sets are used. It should be clear that these models are equivalent. We introduced computability (complexity) of functions defined on some subset of Σ^{**}. Using the introduced products on this space we can extend these notions to any $(\Sigma^{**})^k$ for any $k \in \mathbb{N}$ and even mixed products of Σ^* and Σ^{**} (by a suitable adaptation of the product $\langle\cdot,\cdot\rangle$). We will use this without further mentioning.

Finally, we will use the O-notation in a sloppy manner. Let t be a function on a product $X_1 \times ... \times X_n$ of spaces (into $\mathbb{R}$). Then we say $t(a_1,...,a_n) \in O(\alpha(a_1,...,a_n))$, where α is some expression in the variables a_1, ...,a_n, iff there exists some constant c so that for all $(x_1,...,x_n)$ in $\mathrm{dom}(f)$ we have $t(x_1,...,x_n) \leq c \cdot \alpha(x_1,..,x_n) + c$. Similarly, the notion $o()$, $\Omega()$ and $\omega()$ are defined, e.g. we say $t(a_1,...,a_n) \in \Omega(\alpha(a_1,...,a_n))$ iff $\alpha(a_1,...,a_n) \in O(t(a_1,...,a_n))$ etc.

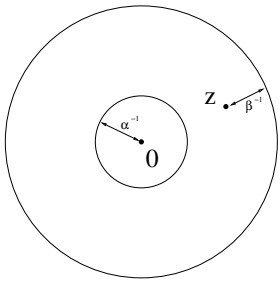

Fig. 1. f is to be evaluated on z from it's values on $\mathbb{D}_{1/\alpha}$

2 Continuation on $\mathbb{D}$

In this section we will restrict ourselves to the unit disc $\mathbb{D}$, first, assuming the following situation (see Figure 1): The holomorphic function f is given by its values on some small disc $\mathbb{D}_{1/\alpha} \subset \mathbb{D}$ around 0 (uniform case) or can be computed in time $O(t)$ on $\mathbb{D}_{1/\alpha}$ (non-uniform case). Our task is to evaluate f with this information at some point $z \in \mathbb{D}$ with $|z| = 1 - 1/\beta$. (Later on we will frequently use $\hat{z}$ instead of z to reuse z to denote other variables.)

2.1 Combinatorics

Let us assume the situation of Figure 1. How many values and what precisions do we need to continue f? The answer will depend on the values α and β (and the precision n). Essentially there are two parameters, which influence the answer to the above question: By $\overline{\log}_\alpha(\beta)$ let us denote

$$\overline{\log}_\alpha(\beta) = \begin{cases} 1 & \text{if } \alpha \geq \beta \\ \lceil \log_\alpha(\beta) \rceil & \text{otherwise} \end{cases}.$$

Furthermore let

$$\Delta_\alpha^\gamma(\beta) = (1 - \frac{1}{\beta})^\gamma - \frac{1}{\alpha^\gamma}$$

for $\gamma \geq 1$, where we omit the index γ in the case of $\gamma = 1$. The influence of the second parameter, $\Delta_\alpha(\beta)$, is quite clear: if the point $\hat{z}$, to which we want to continue f, is near the disc $\mathbb{D}_{1/\alpha}$, then we can evaluate a point near $\hat{z}$ immediately, and thus we know $f(\hat{z})$ to some precision. The influence of the former parameter $\overline{\log}_\alpha(\beta)$ will play some role, if β is quite large and will become clear in Lemma 2.1 below. To simplify things, we will distinguish three cases, where $\Delta_\alpha(\beta)$ is only considered in the first case. In all other cases we will restrict α and β so that this parameter has no influence on the results. Nevertheless, the role of $\Delta_\alpha(\beta)$ can be studied in a similiar fashion for all these cases. More precisely, we consider the following cases, according to the influence of the above parameters on the number of values we have to evaluate f on:

(i) β is quite small. In this case the distance $\Delta_\alpha(\beta)$ between α and $1 - 1/\beta$ will have some influence on the number of values and the precision of evaluations,

whereas we can ignore $\overline{\log}_\alpha(\beta)$.

(ii) β is of medium size, say $2 < \beta < 8$. In this case we get a result, where the precision depends on α, the number of values we have to evaluate depends, however, only on the precision we want to achieve. Notice that the bounds on β are, of course, arbitrary. Actually any compact subintervall of $\mathbb{R}_{>1}$ could be chosen instead.

(iii) β is large, i.e. $\beta \geq 8$. In this case, as in the last one, $\Delta_\alpha(\beta)$ does not have any influence on our results. $\overline{\log}_\alpha(\beta)$ will influence the precision we have to evaluate f to as well as the number of values.

With the above notations we can state our main combinatoric result as follows:

Lemma 2.1 *Let $c := 1/46$, $n \in \mathbb{N}$, $0 < 1/\alpha < 1 - 1/\beta < 1$ and $\hat{z}$ be given so that $\alpha \geq 2$, $(1/2) \cdot \alpha^{\overline{\log}_\alpha(\beta)} \cdot \Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta) =: s > 2$ and $|\hat{z}| = 1 - 1/\beta$. Furthermore let*

$$m \leq c \cdot n \cdot \beta / (\overline{\log}_\alpha(\beta) - \log(\Delta_\alpha(\beta)))$$

and $z_1,...,z_m \in \mathbb{D}_{(1/\alpha)}$ be given.

Then there exists a complex polynomial p so that

(i) $\sup_{z \in p(\mathbb{D})}(|z|) \leq 1$,

(ii) $\sup_{z \in p(\mathbb{D}_{1/\alpha})}(|z|) \leq 2^{-m \cdot (\log(s)-1)-n}$,

(iii) $p(z_1) = p(z_2) = ... = p(z_m) = 0$ *and*

(iv) $p(\hat{z}) = 2^{-n}$.

Furthermore p can be chosen to be computable from $\hat{z}$, $z_1,...z_m$ and α.

Proof. Let α, β, n and m, points $z_1,...,z_m \in \mathbb{D}_{1/\alpha}$ and $\hat{z}$ be given as above. Notice that by the restrictions on α and β we have $\alpha^{\overline{\log}_\alpha(\beta)} > 2$.

To prove the result we essentially take Lagrange's representation of polynomials: Let p be determined by

$$p(z) = 2^{-n} \cdot \prod_{i=1}^{m} (z^{\overline{\log}_\alpha(\beta)} - z_i^{\overline{\log}_\alpha(\beta)}) / (\hat{z}^{\overline{\log}_\alpha(\beta)} - z_i^{\overline{\log}_\alpha(\beta)}).$$

Then items (iii) and (iv) in the statement of the lemma follow immediately. Item (ii) can be seen as follows: For $z \in \mathbb{D}_{1/\alpha}$ we have

$$\begin{aligned} |p(z)| &\leq 2^{-n} \cdot (2 \cdot (1/\alpha)^{\overline{\log}_\alpha(\beta)})^m / ((1 - 1/\beta)^{\overline{\log}_\alpha(\beta)} - 1/\alpha^{\overline{\log}_\alpha(\beta)}))^m \\ &\leq 2^{-n} \cdot (2/s)^m. \end{aligned}$$

To prove item (i) we will distinguish the cases as discussed above:

Case (a) We assume that $\beta \leq 2$. First notice that in this case we have $\overline{\log}_\alpha(\beta) = 1$, so that the definition of p reduces to

$$p(z) = 2^{-n} \cdot \prod_{i=1^m} (z - z_i) / (\hat{z} - z_i).$$

To see item (1), i.e. to prove that $|p(z)|$ is always smaller than 1, we can bound

$$|p(z)| \leq 2^{-n} \cdot (1+1/\alpha)^m/((1-1/\beta-1/\alpha))^m \leq 2^{-n} \cdot \left(\frac{3/2}{\Delta_\alpha(\beta)}\right)^m .$$

Thus item (1) follows from the fact that $m \leq n/(2-\log(\Delta_\alpha(\beta)))$ (i.e. a constant $c = 1/2$ suffices in this case).

Case (b): We assume $2 < \beta < 8$. Let us first bound $\Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta)$: If $\alpha \leq 8$ we have $\Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta)) \geq 1/32$ by the condition on s. If, on the other hand, $\alpha > 8$, we have $1-1/\beta-1/\alpha > (3/8)$.

Thus we get for $z \in \mathbb{D}$

$$|p(z)| \leq 2^{-n} \cdot (1+1/\alpha)^m/(\Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta)))^m \leq 2^{-n} \cdot \left(32 \cdot \frac{3}{2}\right)^m .$$

If $m < n/6$ we get $|p(z)| \leq 1$, which is fulfilled by our condition on m.

Case (c): Let $\beta \geq 8$. Then we have $\overline{\log}_\alpha(\beta)+1 \leq 1/2 \cdot \beta$. Furthermore we get

$$(1+1/\beta)^\beta \leq e \leq (1+1/\beta)^\beta \cdot (1+1/\beta) \leq (1+1/\beta)^\beta \cdot 9/8$$

and

$$\begin{aligned}(1-(\overline{\log}_\alpha(\beta)+1)/\beta)^\beta &\geq e^{-(\overline{\log}_\alpha(\beta)+1)} \\ &\geq (1-(\overline{\log}_\alpha(\beta)+1)/\beta)^\beta \cdot (1-(\overline{\log}_\alpha(\beta)+1)/\beta) \\ &\geq (1-(\overline{\log}_\alpha(\beta)+1)/\beta)^\beta \cdot 1/2.\end{aligned}$$

Thus we get for $\bar{m} = m/\beta$

$$\begin{aligned}|p(z)| &\leq 2^{-n} \cdot (9/8 \cdot e)^{\bar{m}}/((1-1/\beta)^{\overline{\log}_\alpha(\beta)} - 1/\beta)^{\bar{m}} \\ &\leq 2^{-n} \cdot (9/8 \cdot e)^{\bar{m}}/(1/2 \cdot e^{-(\overline{\log}_\alpha(\beta)+1)})^{\bar{m}} \\ &\leq 1\end{aligned}$$

whenever $\bar{m} \leq n/(\log(9/4)+\log(e)+(\overline{\log}_\alpha(\beta)+1)\log(e))$.

□

Notice that the polynomial p in the lemma above can be computed (from $\hat{z}$, z_1,...z_m and α) in polynomial time. We will not use this stronger statement. Furthermore the condition $\alpha \geq 2$ can be easily weakend by either extending case (2) in the proof above or using polynomials $p(z^t)$ where $t = 1/-\log_2(\alpha)$.

2.2 Uniform and Computational Bounds

In this paragraph we will apply the above combinatorical lemma to achieve lower bounds on the complexity needed to continue a holomorphic function in the situation of Figure 1. Whereas the result in the uniform case follows easily from Lemma 2.1, we have to use a finite injury priority argument to prove the existence in the non-uniform case. To simplify things, let for given $\alpha > 0$, $\beta > 0$ and $n \in \mathbb{N}$ the functions $\hat{m}, \hat{p} :\subseteq \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{N} \to \mathbb{R}$ be determined by

$$\hat{m}(\alpha,\beta,n) = c \cdot n \cdot \beta/(\overline{\log}_\alpha(\beta) - \log(\Delta_\alpha(\beta)))$$

and

$$\hat{p}(\alpha,\beta,n) = \hat{m}(\alpha,\beta,n)\cdot(\log(s(\alpha,\beta)-1)+n$$

where $s(\alpha,\beta) = (1/2)\cdot\alpha^{\overline{\log}_\alpha(\beta)}\cdot\Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta)$ (see Lemma 2.1).

Theorem 2.2 *Let $\alpha \in [2;\infty)$ and M be an oracle Turing-machine which $(\nu_{\mathbb{D},\mathbb{D}_{1/\alpha}},\nu_{\mathbb{C}},\nu_{\mathbb{C}})$-computes the function $F_{cont}: H_{\mathbb{D}}\times\mathbb{D}\to\mathbb{D}$ defined by*

$$F_{cont}(f,z) := f(z) \quad (\textit{for all } f\in H_{\mathbb{D}} \textit{ and } z\in\mathbb{D}).$$

Then $t_M(f,z,n)\in\Omega(\hat{m}(\alpha,1/(1-|z|),n)\cdot\hat{p}(\alpha,1/(1-|z|),n))$ for all α and $\beta = 1/(1-|z|)$ fulfilling the conditions of Lemma 2.1 (this means that there exists a common constant in the Ω-notation, which does not dependend on α and β !).

If $q(\alpha,f,z,n)$ denotes the number of questions of precision $2^{-\hat{p}(\alpha,(1-|z|),n)}$ M asks on input (f,z) for $f\in H_{\mathbb{D}}$ and $z\in\mathbb{D}$ with α and $\beta = 1/(1-|z|)$ fulfilling the conditions in Lemma 2.1. Then we have $q(\alpha,f,z,n)\in\Omega(\hat{m}(\alpha,(1-|z|),n))$.

Proof. The first statement clearly follows from the second one. This second statement is a simple consequence of Lemma 2.1: Let α and $\beta = 1/(1-|z|)$ be given, so that the conditions of Lemma 2.1 are fulfilled. Let us first assume that $z\in\mathbb{R}^+$. Then M must on input $p_{\equiv 0}$, which is the constant polynomial 0, ask at least $\hat{m}(\alpha,\beta,n)$ questions of precision $2^{-\hat{p}(\alpha,\beta,n)}$. Otherwise we can replace $p_{\equiv 0}$ by the polynomial p given by Lemma 2.1, where the z_i are those questions of M with precision at least $2^{-\hat{p}(\alpha,\beta,n)}$. Thus M cannot distinguish between $p_{\equiv 0}$ and p for the given precision, whereas $p(z) = 2^{-n}$ and $p_{\equiv 0}(1-1/\beta) = 0$.

Finally, for $z\notin\mathbb{R}^+$, replace in the above proof p by $p\circ e^{\hat{\imath}\phi}$ for suitable $\phi\in(0;2\pi)$ so that $e^{\hat{\imath}\phi}z\in\mathbb{R}^+$.

□

Next to the non-uniform case:

Theorem 2.3 *There exists a computable oracle $A\in\Sigma^{**}$ so that for all rational α, β, which fulfill the condition of Lemma 2.1, and all time-constructible $t:\mathbb{N}\to\mathbb{N}$ there exists a holomorphic function $f:\mathbb{D}\to\mathbb{D}$ which can be computed (with oracle A) in time $O(t)$ on $\mathbb{D}_{1/\alpha}$ but cannot be computed (even with oracle A) in time $O(T)$ on $\mathbb{D}_{1-1/\beta}$ for any T with $n\cdot t(n)\notin O(T)$.*

Proof. Let in the sequel M_0, M_1, ... be an enumeration of all oracle machines. Furthermore let t_0, t_1, ... be the sequence of (partial) computable functions $t_i :\subseteq \mathbb{N}\to\mathbb{N}$ defined by M_i and similarly f_0, f_1, ... be the corresponding sequence of (partial) computable functions $f_i :\subseteq \mathbb{D}\to\mathbb{D}$. We will assume that, given $i,t,x,n\in\mathbb{N}$ and $z\in\mathbb{Y}[\hat{\imath}]$, we can decide, wether $t_{M_i}(x)\le t$ and $t_{M_i}(n,z)\le t$, respectively, which will be denoted by $t_i(x)\downarrow_t$ and $f_i(n,z)\downarrow_t$, respectively. (Here t_M denotes as usually the time complexity of M. Furthermore, by definition of computable functions on $\mathbb{D}$ above, n describes the precision, i.e. 2^{-n}, we want to compute $f_i(z)$ to.)

Let Pol denote the class of complex polynomials. The oracle A will be given by a mapping $A:\mathbb{N}\times((0;1)\cap\mathbb{Y})\times((0;1)\cap\mathbb{Y})\times\mathbb{N}\to\mathrm{Pol}$, which maps a precision n,

two parameters α and β as well as an index j (of t_j) to a polynomial p (given by its values on $\mathbb{D}_{1/\alpha}$). Here we will consider only those parameters α and β, which fulfill the condition of Lemma 2.1, i.e.

$$(1) \qquad 0 < 1/\alpha < 1 - 1/\beta < 1 \text{ and } (1/2) \cdot \alpha^{\overline{\log}_\alpha(\beta)} \cdot \Delta_\alpha^{\overline{\log}_\alpha(\beta)}(\beta) =: s(\alpha,\beta) > 2.$$

Notice that by the standard representations, oracles indeed define functions in Σ^{**}, which are, moreover, computable whenever A is computable. We will thus in the sequel not distinguish between A and the corresponding function in Σ^{**}.

We will construct A so that for all $\alpha, \beta \in ((0;1) \cap \mathbb{Y})$ and j, so that t_j is a total and monotone function, we have

(A) $z \mapsto \lim_{n\to\infty} A(t_j(n)+n, \alpha, \beta, j)(z)$ is well defined and defines a holomorphic function $f_{\alpha,\beta,j} : \mathbb{D}_{1/\alpha} \to \mathbb{D}$, which can be continued to $\mathbb{D}$ and

(B) $|A(t_j(n)+n, \alpha, \beta, j)(z) - f_{\alpha,\beta,j}(z)| < 2^{-n}$ for all $z \in \mathbb{D}_{1/\alpha}$.

Items (A) and (B) above clearly show that the function $f_{\alpha,\beta,j}$ is computable in time $O(n+t_j(n))$ for each $\alpha, \beta \in ((0;1)\cap\mathbb{Y})$ and j so that t_j is a total and monotone function.

To simplify the construction of the oracle A we will fix a few more notations: Let $G :\subseteq \mathbb{N} \times (0;1) \times (0;1) \times \mathbb{C}^* \to \mathrm{Pol}$ be the function, which for given $n \in \mathbb{N}$, reals α and β and a sequence $z_1, ..., z_m \in \mathbb{C}^*$ with $m \le \hat{m}(\alpha,\beta,n)$ determines the polynomial p of Lemma 2.1, i.e. a polynomial p with

- $p(\mathbb{D}) \subseteq \mathbb{D}$,
- $p(z_1) = p(z_2) = ... = p(z) = 0$,
- $p(1-1/\beta) = 2^{-n}$ and
- $\sup_{z\in\mathbb{D}_{1/\alpha}}(|p(z)|) \le 2^{-\hat{m}(\alpha,\beta,n)\cdot(\log(s(\alpha,\beta))-1)-n}$,

whenever α and β fulfill the conditions of this lemma. Furthermore let $\tau : (\mathbb{Y} \cap (0;1))^2 \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a computable bijection (which can, for example, easily be defined by the product $\langle \cdot, ..., \cdot \rangle$).

Next we will give the construction of A, which will be done in stages $k = 0, 1, 2, ...$ In each stage we will construct some oracle A_k with the property that for every $w \in \Sigma^*$ there exists some $k_w \in \mathbb{N}$ so that $A_j(w) = A_{k_w}(w) = A_{k_w}(w)$ for all $j \ge k_w$. We will then take A to be the oracle which maps each $w \in \Sigma^*$ to this stationary result $A_{k_w}(w)$.

Beside the oracle A_k itself (and the values k_w) we will maintain in every stage a sequence $s_0, s_1, ...$ of natural numbers, where $s_r > 0$ in stage k means

(C) if $A_{k+t}(w) = A_k(w)$ for all $|w| \le s_r$ and all $t \in \mathbb{N}$, then there exists a n so that, with $r = \tau(\alpha,\beta,i,j)$, f_i computes not correctly the continuation of $A(t_j(n)+n, \alpha, \beta, j)$ to $1-1/\beta$ in time $\hat{m}(\alpha,\beta,n)\cdot t_j(n)$ and precision 2^{-n}.

Finally, we will construct A so that s_r keeps constant at a certain stage so that any functions in question are covered by condition (C):

(D) Let $i, j \in \mathbb{N}$ and $\alpha, \beta \in \mathbb{Y} \cap (0;1)$ be given so that α and β fulfill the conditions of Lemma 2.1, t_j is monotone increasing and M_i stops on infinitely many inputs

$(n, 1-1/\beta)$ in at most $\hat{m}(\alpha,\beta,n)\cdot t_j(n)$ steps: then there exists a stage k so that $s_r > 0$ keeps constant, where $r = \tau(\alpha,\beta,i,j)$, for all stages $\geq k$.

Notice that items (C) and (D) above guarantee that the functions $f_{\alpha,\beta,j}$ cannot be continued from $\mathbb{D}_{1/\alpha}$ to $\mathbb{D}_{1-1/\beta}$ in time $O(T)$, if $n \cdot t_j(n) \notin O(T)$ (where t_j is a monotone increasing function). Thus once we have constructed A so that (A) to (D) are fulfilled, the theorem is proven.

We will start our construction with $s_r = 0$ and $A_0(n,\alpha,\beta,j) = p_{\equiv 0}$ for all $r,n,j \in \mathbb{N}$ and $\alpha,\beta \in (0;1) \cap \mathbb{Y}$, where $p_{\equiv 0}$ is the polynomial, which everywhere takes the value 0. In step k, $k \geq 0$ we then construct A_k as follows:

Stage k**:** Do for each $r = 1,...,k$ the following computations, where we use the notation $(\alpha,\beta,i,j) = \tau^{-1}(r)$ and $s' = \max_{t\in\mathbb{N}} s_t$:

Step 1: If $s_r > 0$ continue with the next r

Step 2: Compute $\bar{s} = \max_{t<r}(s_r)$

Step 3: Check for all $n = \bar{s}+1,...,k$, wether $\hat{m}(\alpha,\beta,n-2r-3)(\log(s(\alpha,\beta))-1) > 2r+3$, $t_j(n)\downarrow_k$ and $f_i(n,1-1/\beta)\downarrow_{\hat{m}(\alpha,\beta,n)\cdot t_j(n)}$
if this is not the case, continue with the next r

Step 4: Because of the time restriction, f_i can, on input $(n,1-1/\beta)$ only ask $\hat{m}(\alpha,\beta,n)$ questions of the form $A(t,\alpha,\beta,j)(z)$ up to precision 2^{-n} for $t \geq t_j(n)+n$ and $z \in \mathbb{D}_\alpha$.

Let $z_1,...,z_m$ be the inqueried points of this type in $\mathbb{D}_\alpha$ and $p = G(n-2r-2,\alpha,\beta,z_1,...,z_m)$. Notice that this polynomial indeed exists as $\hat{p}(\alpha,\beta,n-2r-2) \leq 2^{-n}$.

Let $s_r := \hat{m}(\alpha,\beta,n)\cdot t_j(n) + n + s'$, $s_l = 0$ for all $l > r$ and $A_k(t,\alpha',\beta',j') = A_{k-1}(t,\alpha',\beta',j')$, whenever $(\alpha',\beta',j') \neq (\alpha,\beta,j)$ or $t < n + t_j(n)$. Furthermore, for $t \geq n + t_j(n)$, let $A_k(t,\alpha,\beta,j)$ be $A_{k-1}(t,\alpha,\beta,j)$, if $|f_i(n,1-1/\beta) - A_{k-1}(t,\alpha,\beta,j)| > 2^{-n}$, and $A_{k-1}(t,\alpha,\beta,j) + 2^{-2r-1}\cdot p$, if $|f_i(n,1-1/\beta) - A_{k-1}(t,\alpha,\beta,j)| \leq 2^{-n}$ in all other cases.

Fixing the values in this way will finish stage k (i.e. the remaining $r's$ are not considered!)

If for none of the $r = 1,...,k$ step 4 above is computed, simply fix $A_k = A_{k-1}$.

To prove the correctness of the above construction we will first show that the construction indeed defines an oracle A:

Claim 2.4 *For every $w \in \Sigma^*$ there exists some $k_w \in \mathbb{N}$ so that $A_j(w) = A_{k_w}(w)$ for all $j \geq k_w$.*

Proof: First notice that by the above construction, s_r is only changed, if $s_r = 0$ and s_r is set to some $s_r > 0$ or vice versa. s_r is only changed to $s_r = 0$ if for some smaller index $l < r$ with $s_l = 0$, s_l is changed to some value $s_l > 0$. Thus each of the cases can happen at most $2^r - 1$ times. This means that for each r there exists a stage k_r, so that none of the values $s_1,...,s_r$ are changed after stage k_r. Furthermore, for each $w \in \Sigma^*$ there exists some r so that $s_r > |w|$ at stage k_r. Choosing k_w to equal such a k_r thus guarantee that A_t and A_{k_w} are equal for all $t \geq k_w$ by our construction.

◇

Finally we have to check that the items (A) to (D) above are fulfilled. Here item (C) and (D) follows from our construction: To prove (D) assume that f_i, t_j, α and β are given as assumed in (D). Then there exists an infinite sequence $(n_t)_{t\in\mathbb{N}} \in \mathbb{N}^{\mathbb{N}}$ so that $f_i(n_t, 1-1/\beta) \downarrow_{\hat{m}(\alpha,\beta,n_t)\cdot t_j(n_t)}$ for all t. There exists a stage k, so that no value s_r with $r < \tau(\alpha,\beta,i,j)$ is changed after step k. Let $\bar{s} = \max\{s_0, ..., s_{\tau(\alpha,\beta,i,j)}\}$ at stage k. For large t with $n_t > \max\{k, \bar{s}\}$ and $\hat{m}(\alpha,\beta,n_t - \tau(\alpha,\beta,i,j))(\log(s(\alpha,\beta)) - 1) > \tau(\alpha,\beta,i,j)$, $\tau(\alpha,\beta,i,j)$ will be considered in Step 4 unless $s_{\tau(\alpha,\beta,i,j)} > 0$ at this time. Thus $s_{\tau(\alpha,\beta,i,j)} > 0$ at some stage $k' > k$ and thus for all stages $\geq k'$. To see (C) let us assume that $s_{\tau(\alpha,\beta,i,j)}$ is set to some value > 0 at some stage k. Then in Step 4, for some $n \in \mathbb{N}$, $A_k(t,\alpha,\beta,j)$, for $t \geq t_j(n) + n$ is chosen so that $|f_i(n, 1 - 1/\beta) - A_k(t,\alpha,\beta,j)| > 2^{-n}$: In the first case, if this already holds for A_{k-1} then the oracle is kept unchanged. In the other case we have $|f_i(n,1-1/\beta) - A_k(t,\alpha,\beta,j)(1-1/\beta)| \geq 2^{-2r-1}2^{-n+2r+3} - |f_i(n,1-1/\beta) - A_{k-1}(t,\alpha,\beta,j)(1-1/\beta)| > 2^{-n}$. However, $f_i(n, 1-1/\beta)$ is supposed to be $A_k(t,\alpha,\beta,j)(1-1/\beta)$ up to precision 2^{-n}.

To see that item (A) holds, let α, β and j be given. Then we add, in the worst case, for each $i \in \mathbb{N}$ and each time s_r is changed, a polynomial $2^{-2r-1}p$, where $r = \tau(\alpha,\beta,i,j)$ and p is a polynomial with $p(\mathbb{D}) \subset \mathbb{D}$. As we have seen above in the proof of Claim 2.4, the values s_r can change at most $2^r - 1$ times to a non-zero value. Thus the sum of all these polynomials at a point in $\mathbb{D}$ is bounded by $\sum_{r=0}^{\infty} 2^r \cdot 2^{-2r-1} \leq 1$. As this convergence is locally uniform, item (A) follows.

To see item (B), we can argue as before, as for each of the added polynomials p for a precision 2^{-n} and for $z \in \mathbb{D}_{1/\alpha}$ we have $\sum_{r=0}^{\infty} 2^r \cdot 2^{-2r-1-n} \leq 2^{-n}$.

□

The result of the previous theorem can be extended in several ways. E.g. it is possible to give oracles so that the time bound T depends on α and β similar to Theorem 2.2.

3 Continuation on $\mathbb{C}$

In this section we will turn to the problem of continuing holomorphic mappings to any point of their domain. This obviously is reasonable only for connected domains. By the Kreiskettenverfahren and Taylor-shifts (see e.g. [8]) we can continue along any path in the domain with roughly the time complexity needed to continue a function on $\mathbb{D}$. Using Riemann mappings similar results are possible without Taylor-shifts. We use this method to give lower bounds, even their dependence on the parameter α and $\beta = 1/(1-|z|)$, on the complexity of continuation on bounded domains. To this end we will replace the parameters $1/(1-|z|)$ and α using the hyperbolic metric δ.

First notice that there exist $c_1 > 0$ and $c_2 > 0$ so that for all $z \in \mathbb{D}$, $z \neq 0$:

$$c_1 \cdot 1/(1-|z|) \leq e^{\delta_{\mathbb{D}}(0,z)} \leq c_2 \cdot 1/(1-|z|).$$

Using this inequality we can reformulate Lemma 2.1 and/or Theorem 2.1. In the uniform case we get the result immediately.

Theorem 3.1 *Let G be a simply connected and bounded subset of $\mathbb{C}$ with $0 \in G$. Furthermore let $\hat{m}(\alpha,\beta,n)$ and $\hat{p}(\alpha,\beta,n)$ be defined as in Section 2.2 and M be an oracle Turing-machine, which $(\nu_{G,\mathbb{D}^{\delta_G}_{1/\alpha}},\nu_{\mathbb{C}},\nu_{\mathbb{C}})$-computes the function $F_{cont,G} : H_G \times \mathbb{C} \to \mathbb{C}$ defined by*

$$F_{cont,G}(f,z) := f(z) \quad (\text{for all } f \in H_G \text{ and } z \in G)$$

in time t_M.

Then $t_M(f,z) \in \Omega(\hat{m}(\alpha, e^{\delta_G(0,z)}, n) \cdot \hat{p}(\alpha, e^{\delta_G(0,z)}, n))$ for all α and $\beta = e^{\delta_G(0,z)}$ fulfilling the condition of Lemma 2.1.

Proof.

Fix a Riemann mapping $\varrho : \mathbb{D} \to G$ and translate Lemma 2.1 and thus Theorem 2.1 via ϱ to G. □

Similarly one can apply the Riemann mapping to the construction in the proof of Theorem 2.2. Instead of altering the construction, however, we can simply add another oracle, say B, first, which allows to compute the Riemann mapping ϱ in linear time. Then we can literally use the construction given in the proof of Theorem 2.2 to get a suitable oracle. Notice, that whenever ϱ is computable, the oracle B is also computable. For conditions on G so that ϱ is computable see [4] (for complexity aspects see [1]).

Theorem 3.2 *Let G be given as in Theorem 3.1. There exists a computable oracle $A \in \Sigma^{**}$ so that for all rational α, β, which fulfill the condition of Lemma 2.1, and all time-constructible $t : \mathbb{N} \to \mathbb{N}$ there exists a holomorphic function $f : G \to \mathbb{D}$ which can be computed (with oracle A) in time $O(t)$ on $\mathbb{D}^{\delta_G}_{1/\alpha}$ but cannot be computed (even with oracle A) in time $O(T)$ on $\mathbb{D}^{\delta_G}_{\beta}$ for any T with $n \cdot t(n) \notin O(T)$.*

References

[1] Binder, I. and Braverman, M. and Yampolsky, M., *On computational complexity of Riemann mapping*, Arkiv for Matematik, 2007, to appear.

[2] Briggs, W. L. and Henson, V. E., *The DFT: An Owner's Manual for the Discrete Fourier Transform*, Society for Industrial and Applied Mathematics, 1995.

[3] Burrus, C. S. and Parks, T. W., *Dft/FFT and Convolution Algorithms*, Wiley, 1985.

[4] Hertling, P., *An effective Riemann Mapping Theorem*, Theoretical Computer Science, 219:225–265, 1999.

[5] Ko, K.-I and Friedman, H., *Computing power series in polynomial time*, Advances in Applied Math., 9:40–50, 1988.

[6] Ko, K.-I, *Complexity Theory of Real Functions*, Progress in Theoretical Computer Science, Birkhäuser, 1991.

[7] Müller, N. Th., *Uniform Computational Complexity of Taylor series*, ICALP, Lecture Notes in Computer Science, 267:435–444, Springer-Verlag, 1987.

[8] Schönhage, A., *Multiplicative Complexity of Taylor Shifts and a New Twist of the Substitution Method*, FOCS, IEEE Computer Society, pp 212–217, 1998.