Archive for Mathematical Logic
https://doi.org/10.1007/s00153-020-00712-z

The noneffectivity of Arslanov's completeness criterion and related theorems

Sebastiaan A. Terwijn[1]

Received: 4 May 2018 / Accepted: 10 January 2020

Abstract
We discuss the (non)effectivity of Arslanov's completeness criterion. In particular, we show that a parameterized version, similar to the recursion theorem with parameters, fails. We also discuss the effectivity of another extension of the recursion theorem, namely Visser's ADN theorem, as well as that of a joint generalization of the ADN theorem and Arslanov's completeness criterion.

Keywords Recursion theorem · ADN theorem · Arslanov completeness criterion · Uniformity

Mathematics Subject Classification 03D25 · 03D28 · 03B40

1 Introduction

Kleene's recursion theorem [8] states that every computable operation on codes of partial computable functions has a fixed point. That is, for every computable function f there exists a number e such that $\varphi_{f(e)} = \varphi_e$. Here φ_e denotes the e-th partial computable function. Kleene actually proved a more general version of this theorem with parameters:

Theorem 1.1 (Recursion theorem with parameters, Kleene [8]) *Let $h(n, x)$ be a computable binary function. Then there exists a computable function f such that for all n, $\varphi_{f(n)} = \varphi_{h(n,f(n))}$.*

This result shows that the recursion theorem is effective, in the sense that the fixed points of a computable sequence of functions can be found in a uniformly computable way. We refer to Moschovakis [10] for an overview of some of the applications of this

✉ Sebastiaan A. Terwijn
terwijn@math.ru.nl

[1] Department of Mathematics, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

Published online: 22 January 2020

classic result. (Note that Kleene referred to Theorem 1.1 as the first recursion theorem, whereas Moschovakis proposes to call it the second recursion theorem, to contrast it with the simpler nonuniform statement.)

The recursion theorem has been extended in several ways. We refer the reader to Soare [12] for a general discussion. In this paper we discuss the effectivity of two extensions, namely Arslanov's completeness criterion (Sects. 2 and 3) and Visser's ADN theorem (Sects. 4 and 5). In particular we show that the parameterized versions of these extensions, analogous to Theorem 1.1, fail. Finally, we discuss a joint generalization of Arslanov's completeness criterion and the ADN theorem from [13]. Though the ADN theorem does not have a parameterized version, it is uniform in certain other respects. In Sect. 6 we show that this uniformity does not hold for the joint generalization.

Our notation from computability theory is mostly standard. Partial computable (p.c.) functions are denoted by lower case Greek letters, and (total) computable functions by lower case Roman letters. ω denotes the natural numbers, φ_e denotes the e-th p.c. function, and W_e denotes the domain of φ_e. We write $\varphi_e(n)\downarrow$ if this computation is defined, and $\varphi_e(n)\uparrow$ otherwise. $\emptyset'$ denotes the halting set. For unexplained notions we refer to Odifreddi [11] or Soare [12].

In the discussion below we will use the following notions from the literature:

- A function f is called *fixed point free*, or simply FPF, if $W_{f(n)} \neq W_n$ for every n. We will also use this terminology for partial functions, see Definition 4.1 below, but by FPF function we will always mean a total function, unless explicitly stated otherwise.
- A function g is called *diagonally noncomputable*, or DNC, if $g(e) \neq \varphi_e(e)$ for every e.

Though the notions of FPF and DNC function are different, it is well-known that they coincide on Turing degrees, cf. Jockusch et al. [5]. Namely, a set computes a FPF function if and only if it computes a DNC function. Moreover, this is also equivalent to computing a function f such that $\varphi_{f(e)} \neq \varphi_e$ for every e.

DNC functions played an important role in Kučera's alternative solution to Post's problem [9]. In the paper by Kjos-Hanssen et al. [7], the notion of DNC function is linked to sets with high initial segment Kolmogorov complexity.

2 Arslanov's completeness criterion

By the recursion theorem, and the equivalence quoted above, no FPF function is computable. It is easy to see that the halting set $\emptyset'$ computes a FPF function, as $\emptyset'$ can list all computable functions. However, by the low basis theorem [6], there also exist FPF functions of low degree. The next result shows that FPF functions cannot have incomplete c.e. degree. (On the other hand, by Kučera [9], any FPF degree below $\emptyset'$ bounds a noncomputable c.e. degree.) This shows that the recursion theorem can be extended from computable functions to functions bounded by an incomplete c.e. degree.

Theorem 2.1 (Arslanov completeness criterion [1]) *A c.e. set A is Turing complete if and only if A computes a* FPF *function.*

Proof Suppose A is c.e. and incomplete, and $f \leqslant_T A$. Then f has a computable approximation $\hat{f}(n, s)$, and there is an A-computable modulus function $m(n)$ such that $\forall s \geqslant m(n)\big(f(n) = \hat{f}(n, s)\big)$. By the recursion theorem with parameters (Theorem 1.1), let h be a computable function such that

$$W_{h(n)} = \begin{cases} W_{\hat{f}(h(n),s_n)} & \text{if } n \in \emptyset' \text{ and } s_n \text{ is minimal such that } n \in \emptyset'_s, \\ \emptyset & \text{otherwise.} \end{cases}$$

Then there exists $n \in \emptyset'$ such that $\hat{f}(h(n), s_n) = f(h(n))$, so that $h(n)$ is a fixed point of f. Namely, if this were not the case, then we would have that for all n, if $n \in \emptyset'$, then $m(h(n)) > s_n$, and hence $n \in \emptyset'_{m(h(n))}$. Thus we would have $\emptyset' \leqslant_T A$, contrary to assumption. □

The proof given here is basically the contrapositive of the proof in Soare [12]. The proof above already suggests that the result is not effective: It does not give a fixed point effectively, but merely produces a c.e. set, namely $\big\{h(n) \mid n \in \emptyset'\big\}$, at least one of the elements of which is a fixed point. That this is necessarily so follows from the result in the next section.

3 The failure of Arslanov's completeness criterion with parameters

Let h be a computable function of two arguments. Since for every fixed n the function $h(n, x)$ is a computable function of x, by the recursion theorem we have

$$\forall n \exists x \; \varphi_x = \varphi_{h(n,x)}.$$

When we Skolemize this formula we obtain:

$$\exists f \forall n \; \varphi_{f(n)} = \varphi_{h(n,f(n))}.$$

The recursion theorem with parameters tells us that we can take f *computable* here. In other words, the recursion theorem holds uniformly.

Now consider the Arslanov completeness criterion. Let A be an incomplete c.e. set, and let $h \leqslant_T A$ be a binary function. By Theorem 2.1 we have

$$\forall n \exists x \; \varphi_x = \varphi_{h(n,x)}$$

and Skolemization gives

$$\exists f \forall n \; \varphi_{f(n)} = \varphi_{h(n,f(n))}.$$

We prove that in general we cannot take f computable in this case. This even fails when A is of low Turing degree. Note that by relativizing the recursion theorem with parameters, there always exists an A-computable Skolem function f.

Theorem 3.1 (Failure of Arslanov with parameters) *There exist a low c.e. set A and an A-computable binary function h such that for every computable f, there exists n with*

$$W_{f(n)} \neq W_{h(n,f(n))}.$$

Proof We build A c.e. and $h \leqslant_T A$ total using a finite injury construction.The requirements for the construction are:

$R_e : f = \{e\}$ is total $\implies \exists n\ W_{f(n)} \neq W_{h(n,f(n))}$,
$L_e : \exists^\infty s\ \{e\}_s^{A_s}(e)\downarrow \implies \{e\}^A(e)\downarrow$.

The requirements L_e guarantee that A is low (cf. Soare [12]), and clearly the requirements R_e are sufficient to prove the theorem. We give the requirements the following priority ordering:

$$L_0 > R_0 > L_1 > R_1 > L_2 > \cdots$$

To satisfy L_e we do not have to enumerate anything into A, we only maintain a restraint function $r(e, s)$ to preserve computations in the usual way. Let us consider the strategy for R_e in isolation. Suppose we have picked n as a potential witness for R_e.

Step 1. Suppose we see at stage s such that $f(n) = \{e\}_s(n)\downarrow$.
If $W_{f(n),s} \neq \emptyset$ we let $W_{h(n,f(n))} = \emptyset$, thus satisfying R_e forever.
If $W_{f(n),s} = \emptyset$ we let $W_{h(n,f(n))} \neq \emptyset$.
Step 2. Suppose that at a later stage $t > s$ we see $W_{f(n),t} = W_{h(n,f(n))} \neq \emptyset$.
Now we *change* $h(n, f(n))$ so that $W_{h(n,f(n))} = \emptyset$ by changing A below the use of h.

Since the definition of $W_{h(n,f(n))}$ needs to be adapted at most twice (from empty to nonempty to empty), we can get by by letting h use only two bits of A. We define h as follows. We use a standard computable pairing function $\langle\cdot,\cdot\rangle$ to denote coded pairs and triples. For ease of notation, we write $A(x, y, z)$ instead of $A(\langle x, y, z\rangle)$. We let h be an A-computable function such that

$$W_{h(n,x)} = \emptyset \iff A(n, x, 0) = A(n, x, 1),$$
$$W_{h(n,x)} \neq \emptyset \iff A(n, x, 0) \neq A(n, x, 1).$$

Clearly such a function h can be defined from A. (As the computation of $h(n, x)$ uses only two bits from A, this is even a btt-reduction.)

We construct A in stages. L_e requires attention at stage s if $e < s$, $\{e\}_s^{A_s}(e)\downarrow$, and $r(e, s) = 0$. (This means that a restraint should be set to preserve the computation.) R_e requires attention at stage s if $e < s$ and one of the following holds:

(a) R_e does not have a witness at stage s, that is, $n_{e,s}$ is undefined. Required action in this case: pick n larger than all current restraints $r(i, s)$, $i \leqslant e$, and also different from all other witnesses $n_{i,s}$ that are currently defined, and let $n_{e,s+1} = n$.

(b) $n = n_{e,s}$ is defined, $f(n) = \{e\}_s(n)\downarrow$, and one of the following subcases applies:

(b.1) $W_{f(n),s} = \emptyset$ and $A_s(n, f(n), 0) = A_s(n, f(n), 1) = 0$. Required action: Define $A(n, f(n), 0) = 1$.

(b.2) $W_{f(n),s} \neq \emptyset$, $A_s(n, f(n), 0) = 1$, and $A_s(n, f(n), 1) = 0$. Required action: Define $A(n, f(n), 1) = 1$.

Also, if either $A(n, f(n), 0) = 1$ or $A(n, f(n), 1) = 1$ is set at stage s, we define $r(i, s+1) = 0$ for all $i > e$.[1]

Construction Initially A is empty: $A_0 = \emptyset$. At stage $s > 0$, pick the highest priority requirement R_e or L_e, if any, that requires attention. If there is none, proceed to the next stage. If L_e is picked, set $r(e, s+1)$ equal to the use of $\{e\}_s^{A_s}(e)$ (this computation converges since L_e requires attention). Also, initialize all lower priority R_i by letting all witnesses $n_{i,s+1}$ with $i \geqslant e$ be undefined, and proceed to the next stage. If R_e is picked, perform the actions indicated above under (a) and (b). This concludes the construction of $A = \bigcup_s A_s$.

Verification We verify that all requirements are satisfied. For L_e, note that the only requirements that can injure it are the R_i with $i < e$, and by induction each of these enumerates at most finitely many numbers into A, so L_e is injured at most finitely often, and hence is eventually satisfied.

For R_e, suppose that $f = \{e\}$ is total. By induction, assume that no higher priority requirement L_i or R_i requires attention after stage t. Let r be the maximum of all higher priority restraints:

$$r = \max_{i \leqslant e} \lim_{s \to \infty} r(i, s).$$

Note that since by assumption every L_i, $i \leqslant e$, acts only finitely often, this limit exists and is finite. By the construction and (a) above, at some stage s after the last stage that a requirement L_i with $i \leqslant e$ acts, $n = n_{e,s} > r$ is defined, which is then never redefined later. We have the following cases.

If $W_{f(n)} = \emptyset$, then R_e acts exactly once after the stage s where n is defined, the clause (b.1) applies at that stage, and we have $A(n, f(n), 0) = 1$ and $A(n, f(n), 1) = 0$. Hence $W_{h(n,f(n))} \neq \emptyset$, and R_e is satisfied.

If $W_{f(n)} \neq \emptyset$ then we have two subcases:

- After the stage s where n is defined, R_e never requires attention. In this case we have $A(n, f(n), 0) = A(n, f(n), 1) = 0$, hence $W_{h(n,f(n))} = \emptyset$, and R_e is satisfied.
- In the opposite case, R_e does require attention after stage s. In this case, R_e will act precisely twice after stage s. The first time, at stage s' say, since $A_s(n, f(n), 0) = 0$ we will have $W_{f(n),s'} = \emptyset$ (for otherwise R_e would not require attention) and case (b.1) will apply. The second time will occur at a stage $s'' > s'$ that is large enough to see that $W_{f(n),s''} \neq \emptyset$. Since now $A_{s''}(n, f(n), 0) = 1$, case (b.2) applies, and we will have $A(n, f(n), 0) = A(n, f(n), 1) = 1$. Hence $W_{h(n,f(n))} = \emptyset$, and R_e is satisfied.

[1] That is, if R_e enumerates an element into A, we drop the restraints of all lower priority requirements L_i. This is overkill since the action may not actually injure all of these, but it is just as easy.

So we see that R_e acts at most twice after the last time it is initialized, and is eventually satisfied. □

4 The ADN theorem

It is well-known that Kleene found the recursion theorem by studying the λ-calculus. (See for example Crossley [4] for some historical comments.) Also motivated by the λ-calculus, arithmetic provability, and the theory of numerations, Visser [14] proved the following generalization of the recursion theorem. It has interesting applications in the theory of numerations, see for example Bernardi and Sorbi [3] and Barendregt [2]. ADN theorem stands for "anti diagonal normalization theorem".

Definition 4.1 We extend the definition of FPF function to partial functions. We call a partial function δ FPF if it is fixed point free on its domain, i.e. for every n,

$$\delta(n)\downarrow \implies W_{\delta(n)} \neq W_n. \tag{4.1}$$

Theorem 4.2 (ADN theorem, Visser [14]) *Suppose that δ is a partial computable FPF function. Then for every partial computable function ψ there exists a computable function f such that for every n,*

$$\psi(n)\downarrow \implies W_{f(n)} = W_{\psi(n)} \tag{4.2}$$
$$\psi(n)\uparrow \implies \delta(f(n))\uparrow \tag{4.3}$$

If (4.2) holds for every n, we say that f *totalizes* ψ, and if in addition (4.3) holds, we say that f *totalizes* ψ *avoiding* δ.

Just as the Arslanov completeness criterion extends the recursion theorem from computable functions to functions computable from any incomplete c.e. degree, Theorem 4.2 can be extended to such degrees. This gives the following joint generalization of the ADN theorem and the Arslanov completeness criterion:

Theorem 4.3 (Joint generalization [13]) *Suppose A is a c.e. set such that $A <_T \emptyset'$. Suppose that δ is a partial A-computable* FPF *function. Then for every partial computable function ψ there exists a computable function f totalizing ψ avoiding δ, i.e. such that for every n* (4.2) *and* (4.3) *above hold.*

Note that Theorem 4.3 implies Theorem 2.1, because if δ were total then (4.3) could not hold. Hence no total FPF function of incomplete c.e. degree can exist.

Thus we have the picture of generalizations of the recursion theorem from Fig. 1. All of these generalizations can be proved using the recursion theorem with parameters (Theorem 1.1). This prompts the question whether any of these generalizations have a parameterized version. The negative answer for Arslanov's completeness criterion was already given in Sect. 3. We discuss the ADN theorem in the next section.

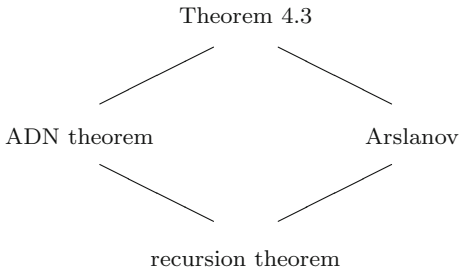

Fig. 1 Generalizations of the recursion theorem

5 The ADN theorem with parameters

The ADN theorem is uniform in codes of ψ, as is easy to see, cf. [14]. In fact, one may assume without loss of generality that the function ψ is universal. Also, from the proof of the ADN theorem from the recursion theorem with parameters, given in [13], it is clear that the code of the function f depends effectively on a code for δ. Hence the result is uniform in both ψ and δ. However, that the result is effective in this sense does not mean it has a parameterized version analogous to the recursion theorem with parameters. As the ADN theorem is a statement about partial FPF functions, and hence in a way a contrapositive of the recursion theorem, it is not even immediately clear what the statement of the ADN theorem with parameters should be. At least it should imply Theorem 1.1.

To formulate the analog of Theorem 1.1 for the ADN theorem, we define the following notion.

Definition 5.1 A partial binary function $\delta(n, x)$ is FPF^+ if for every computable function g there exists n such that either $\delta(n, g(n)) \uparrow$ or $\varphi_{g(n)} \neq \varphi_{\delta(n,g(n))}$.

Note that by negating the property from the definition, δ is *not* FPF^+ if there exists a computable function g such that for every n, $\delta(n, g(n))$ is defined and $\varphi_{g(n)} = \varphi_{\delta(n,g(n))}$. This expresses that g uniformly computes fixed points for the family of functions $\delta(n, x)$. By the recursion theorem with parameters, every *total* computable δ is not FPF^+.

We can now formulate the analog of the recursion theorem with parameters as follows.

Statement 5.2 (ADN theorem with parameters) *Suppose that δ is a binary partial computable* FPF^+ *function. Then for every partial computable function ψ there exists a computable function f such that for every n,*

$$\psi(n)\downarrow \implies W_{f(n)} = W_{\psi(n)} \tag{5.1}$$

$$\psi(n)\uparrow \implies \delta(n, f(n))\uparrow \tag{5.2}$$

To show that this is the proper analog of Theorem 1.1 for the ADN theorem, we show that Statement 5.2 both implies Theorem 1.1 and the ADN theorem. We then proceed to show that it is false.

Statement 5.2 implies Theorem 1.1: Note that for the statement to hold, δ cannot be total (for then (5.2) could not hold in case ψ is nontotal). So if δ is total, it is not FPF$^+$. As already observed above, this means that there is a computable function g such that for every n, $\varphi_{g(n)} = \varphi_{\delta(n,g(n))}$, which is the statement of Theorem 1.1.

Statement 5.2 implies Theorem 4.2: Given a unary p.c. FPF function δ, consider the function defined as $\hat{\delta}(n, x) = \delta(x)$ for every n and x. Note that $\hat{\delta}$ is FPF$^+$: For every computable function g and every n, $\hat{\delta}(n, g(n)) = \delta(g(n))\!\uparrow$ or $\varphi_{g(n)} \neq \varphi_{\delta(g(n))}$ since δ is FPF. Applying Statement 5.2 to $\hat{\delta}$ gives, for a given p.c. ψ, a computable f totalizing ψ such that

$$\psi(n)\!\uparrow \implies \hat{\delta}(n, f(n))\!\uparrow \implies \delta(f(n))\!\uparrow$$

for every n, hence Theorem 4.2 holds for δ.

Proposition 5.3 *Statement 5.2 is false.*

Proof We construct δ p.c. and FPF$^+$ and ψ p.c. to diagonalize against all computable $f = \varphi_e$, ensuring that (5.2) fails. Constructing a FPF$^+$ function is very easy: According to Definition 5.1 we simply have so make sure that for every computable g we have a point n such that $\delta(n, g(n))$ is undefined. The construction is as follows. Let ψ be totally undefined. For every $f = \varphi_e$ pick two witnesses n_e and m_e such that all n_e and m_e are different, e.g. $n_e = 2e$ and $m_e = 2e + 1$. Define δ to be a partial computable function such that

$$\delta(n, x)\!\downarrow \iff n = n_e \wedge \varphi_e(n_e)\!\downarrow = x. \tag{5.3}$$

Now suppose that $f = \varphi_e$ is total. Then $f(n_e)\!\downarrow$, so by (5.3) we have $\delta(n_e, f(n_e))\!\downarrow$. Hence f fails to satisfy (5.2), because $\psi(n_e)\!\uparrow$.

To finish the proof, all that remains is to verify that δ is FPF$^+$. Note that (5.3) implies that $\delta(m_e, x)\!\uparrow$ for every e and x. So if $f = \varphi_e$ is total, we have in particular that $\delta(m_e, f(m_e))\!\uparrow$, which by Definition 5.1 makes δ an FPF$^+$ function. □

6 The nonuniformity of the joint generalization

As remarked in Sect. 5, the dependence of f on ψ and δ in Theorem 4.2 is uniform in codes of ψ and δ. This prompts the question whether a similar uniformity holds for the joint generalization Theorem 4.3. Indeed, Theorem 4.3 is also uniform in ψ, as is easy to check, using the same argument as for Theorem 4.2. As for δ, as this is no longer a p.c. function, we first have to specify what exactly we mean by uniformity in this case. The weakest form of uniformity, using the strongest possible assumption, would be to give f codes of both A and δ, i.e. codes a and d such that $A = W_a$ and $\delta = \{d\}^A$. Uniformity then means that there is a computable function h such that an f as in the theorem is given by

$$f = \varphi_{h(a,d,b)}, \tag{6.1}$$

where b is a code such that $\psi = \varphi_b$. Note that h is total, but f only has to satisfy the theorem in case A is incomplete and δ is FPF. Instead of issuing f with a code b of ψ,

we could alternatively simply assume that ψ is universal. This amounts to the same thing, but in the construction below it will be easier to work with b.

The proof of the joint generalization in [13] is not uniform. For A and δ as in the theorem, the proof provides a family of functions f_x, $x \in \omega$, at least one of which satisfies the theorem. That the proof is necessarily nonuniform is confirmed by the next result.

Theorem 6.1 *Uniformity of Theorem 4.3 in the sense of (6.1) does not hold.*

Proof Assume for a contradiction that a computable function h as in (6.1) exists. We will prove the existence of codes a, d, and b such that $A = W_a$ is Turing incomplete, $\delta = \{d\}^A$ is a partial A-computable FPF function, and $\psi = \varphi_b$ is partial computable, such that $f = \varphi_{h(a,d,b)}$ does not satisfy Theorem 4.3, contradicting the assumption.

The code d for δ will depend effectively on a and b, so that we have only two parameters a and b in the construction. We will construct computable functions p and q such that $A = W_{p(a,b)}$ and $\psi = \varphi_{q(a,b)}$. An application of the double recursion theorem[2] will provide us with codes a and b such that $W_a = W_{p(a,b)}$ and $\varphi_b = \varphi_{q(a,b)}$.

Construction of A, δ, and ψ. We use a coding of δ in A similar to the one used for h in the proof of Theorem 3.1. Namely, we let

$$\begin{aligned} &\delta(x)\uparrow \iff A(x,0) = A(x,1), \\ &\delta(x)\downarrow \ \wedge\ W_{\delta(x)} \neq \emptyset \iff A(x,0) = 1 \wedge A(x,1) = 0. \end{aligned}$$

Note that a code d of δ effectively depends on a code of A. Since $A = W_{p(a,b)}$, there is a computable function d such that $d(a,b)$ is a code of δ.

Our assumption is that $f = \varphi_{h(a,d(a,b),b)}$ satisfies Theorem 4.3. At the beginning of the construction A is empty and ψ is totally undefined.

Step 1. Wait for $f(0)$ to become defined. If this never happens, we automatically win, and do not have to take further action.

Step 2. If $f(0)\downarrow$, we let $\delta(f(0))\downarrow$ such that $W_{\delta(f(0))} \neq \emptyset$ by defining $A(f(0),0) = 1$. This action would kill f by making (4.3) fail, but now there is the threat that $W_{f(0)} = W_{\delta(f(0))}$ so that δ may fail to be FPF, hence we may have to take further action to prevent this.

Step 3. Wait for $W_{f(0)} \neq \emptyset$. We take this as a sign that $W_{f(0)}$ might follow $W_{\delta(f(0))}$, so we redefine $\delta(f(0))\uparrow$, by defining $A(f(0),1) = 1$. Also, we define $\psi(0)$ so that $W_{\psi(0)} = \emptyset$. This kills f by making (4.2) fail.

This completes the construction. Note that the construction depends effectively on the parameters a and b, so that there exist computable functions p and q such that $A = W_{p(a,b)}$ and $\psi = \varphi_{q(a,b)}$. By the double recursion theorem there exist a and b such that $W_a = W_{p(a,b)}$ and $\varphi_b = \varphi_{q(a,b)}$. We verify that $A = W_a$ is incomplete, $\delta = \{d(a,b)\}^A$ is FPF, and that $f = \varphi_{h(a,d(a,b),b)}$ does not satisfy the statement of Theorem 4.3.

[2] The double recursion theorem says that given binary computable functions p and q, there exists codes a and b such that $\varphi_a = \varphi_{p(a,b)}$ and $\varphi_b = \varphi_{q(a,b)}$, cf. Odifreddi [11, p. 155]. Namely, by Theorem 1.1 there exists a computable function g such that $\varphi_{g(x)} = \varphi_{p(g(x),x)}$ for every x. By the recursion theorem there exists b such that $\varphi_b = \varphi_{q(g(b),b)}$. Now we can take $a = g(b)$.

First note that A is finite, since at most the two numbers $\langle f(0), 0\rangle$ and $\langle f(0), 1\rangle$ are enumerated into A. In particular A is Turing incomplete.

If $f(0)$ fails to become defined in step 1, it obviously fails the theorem by not being total, so assume that $f(0)\downarrow$.

In case $W_{f(0)} = \emptyset$, by step 2 we have $W_{\delta(f(0))} \neq \emptyset$, hence δ is FPF (note that it is not defined on any other point). Also, the construction ends with this step, and f fails to satisfy (4.3) since $\psi(0)\uparrow$ and $\delta(f(0))\downarrow$.

If $W_{f(0)} \neq \emptyset$, by step 3 we have $\delta(f(0))\uparrow$, so again δ is FPF. Also, we now have $W_{\psi(0)} = \emptyset$, so f fails to totalize ψ. Thus we see that f fails to satisfy the theorem in every case. □

The set A in the proof above is actually computable, and hence δ is p.c. This does not contradict the fact that Theorem 4.2 is uniform in a code of δ. Namely, $A = W_a$ may be computable, but not via the code a that is provided.

Acknowledgements We would like to thank an anonymous referee for several corrections and improvements.

References

1. Arslanov, M.M.: On some generalizations of the fixed point theorem. Sov. Math. (Izv. VUZ. Mat.) **25**(5), 1–10 (1981)
2. Barendregt, H.: Representing undefined in lambda calculus. J. Funct. Program. **2**(3), 367–374 (1992)
3. Bernardi, C., Sorbi, A.: Classifying positive equivalence relations. J. Symb. Log. **48**(3), 529–538 (1983)
4. Crossley, J.N.: Reminiscences of logicians. In: Crossley, J.N. (ed.) Algebra and Logic, pp. 1–62. Springer, Berlin (1975)
5. Jockusch Jr., C.G., Lerman, M., Soare, R.I., Solovay, R.M.: Recursively enumerable sets modulo iterated jumps and extensions of Arslanov's completeness criterion. J. Symb. Log. **54**(4), 1288–1323 (1989)
6. Jockusch Jr., C.G., Soare, R.I.: Π^0_1 classes and degrees of theories. Trans. Am. Math. Soc. **173**, 33–56 (1972)
7. Kjos-Hanssen, B., Merkle, W., Stephan, F.: Kolmogorov complexity and the recursion theorem. Trans. Am. Math. Soc. **363**, 5465–5480 (2011)
8. Kleene, S.C.: On notation for ordinal numbers. J. Symb. Log. **3**, 150–155 (1938)
9. Kučera, A.: An Alternative, Priority-Free Solution to Post's Problem. Lecture Notes in Computer Science, vol. 233, pp. 493–500. Springer, Berlin (1986)
10. Moschovakis, Y.N.: Kleene's amazing second recursion theorem. Bull. Symb. Log. **16**(2), 189–239 (2010)
11. Odifreddi, P.: Classical Recursion Theory, vol. 1. Studies in Logic and the Foundations of Mathematics, vol. 125. North-Holland, Amsterdam (1989)
12. Soare, R.I.: Recursively Enumerable Sets and Degrees. Springer, Berlin (1987)
13. Terwijn, S.A.: Generalizations of the recursion theorem. J. Symb. Log. **83**(4), 1683–1690 (2018)

14. Visser, A.: Numerations, λ-calculus, and arithmetic. In: Seldin, J.P., Hindley, J.R. (eds.) To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, pp. 259–284. Academic Press, London (1980)