Turing Meets Shannon: On the Algorithmic Construction of Channel-Aware Codes

Holger Boche, *Fellow, IEEE*, Rafael F. Schaefer, *Senior Member, IEEE*, and H. Vincent Poor, *Life Fellow, IEEE*

***Abstract*— A capacity result involves two parts: achievability and converse. The achievability proof is usually non-constructive and only the existence of capacity-achieving codes is shown invoking probabilistic techniques. Recently, capacity-achieving codes have been found for several channels demonstrating that such codes can actually be constructed algorithmically. To this end, each construction is designed for a pre-specified channel so that the corresponding algorithm is specifically tailored to it. This paper addresses the general question of whether or not it is possible to find algorithms that can construct capacity-achieving codes for a whole class of channels. To do so, the concept of Turing machines is used which provides the fundamental performance limits of digital computers and therewith fully specifies which tasks are algorithmically feasible in principle. It is shown that there exists no Turing machine that is able to construct capacity-achieving codes for a whole class of channels, where the channel realization from this class is given as an input to the Turing machine. It is further shown that such an algorithmic construction remains impossible when the optimality condition is dropped and codes only need to achieve a fraction of the capacity. Finally, implications on channel-aware transmission, link adaptation, and cross-layer optimization are discussed.**

***Index Terms*— Algorithmic computability, Turing machine, communication system, code construction.**

Manuscript received November 23, 2020; revised June 17, 2021, September 9, 2021, and December 8, 2021; accepted January 14, 2022. Date of publication January 25, 2022; date of current version April 18, 2022. This work of Holger Boche was supported in part by the German Federal Ministry of Education and Research (BMBF) within the national initiative on 6G Communication Systems through the research hub *6G-life* under Grant 16KISK002, within the national initiative for *Post Shannon Communication (NewCom)* under Grant 16KIS1003K, and the project *Hardware Platforms and Computing Models for Neuromorphic Computing (NeuroCM)* under Grant 16ME0442. It has further received funding by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project *6G Future Lab Bavaria* as well as in part by the German Research Foundation (DFG) within Germany's Excellence Strategy EXC-2092 – 390781972. This work of Rafael F. Schaefer was supported in part by the BMBF within NewCom under Grant 16KIS1004 and in part by the DFG under Grant SCHA 1944/6-1. This work of H. Vincent Poor was supported by the U.S. National Science Foundation under Grant CCF-1908308. An earlier version of this paper was presented in part at the IEEE International Conference on Communications (ICC), Montreal, QC, Canada, June 2021 [DOI: 10.1109/ICC42927.2021.9500750] and in part at the VDE-TUM Meeting on 6G, Berlin, Germany, September 2021. The associate editor coordinating the review of this article and approving it for publication was V. Stankovic. *(Corresponding author: Rafael F. Schaefer.)*

Holger Boche is with the Institute of Theoretical Information Technology, Technical University of Munich, 80290 Munich, Germany, also with the BMBF Research Hub 6G-life, 80290 Munich, Germany, and also with the Excellence Cluster Cyber Security in the Age of Large-Scale Adversaries (CASA), Ruhr University Bochum, 44801 Bochum, Germany (e-mail: boche@tum.de).

Rafael F. Schaefer is with the Chair of Communications Engineering and Security, Center for Sensor Systems (ZESS), University of Siegen, 57068 Siegen, Germany (e-mail: rafael.schaefer@uni-siegen.de).

H. Vincent Poor is with the Department of Electrical and Computer Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: poor@princeton.edu).

Digital Object Identifier 10.1109/TCOMM.2022.3146298

I. Introduction

In 1948, Shannon's seminal work "A Mathematical Theory of Communication" [2] established the capacity of the point-to-point channel characterizing the maximum transmission rate at which reliable communication is possible. Since then, capacity results have been established for many (simple) communication scenarios including the multiple access channel, degraded broadcast channel, wiretap channel, compound channel, arbitrarily varying channel and others; see for example [3]–[5] and references therein.

In information theory, proving a capacity result usually involves two parts: achievability and converse which establish matching lower and upper bounds on the capacity. For the achievability part however, the proof is in general non-constructive in the sense that only the existence of good (i.e., capacity-achieving) codes is shown. To make this possible, Shannon developed probabilistic techniques which lay the groundwork for today's so-called random coding technique. In particular, this approach does not impose any requirements on whether or not these codes are effectively, i.e., algorithmically, constructible.

Although the existence of optimal codes for many communication scenarios has been known for a long time, explicit code designs that actually achieve the capacity asymptotically have been found only recently; see for example [6], [7] and references therein. Such good code constructions include turbo codes, Reed-Muller codes, (spatially-coupled) low-density parity check (LDPC) codes, and polar codes. The latter two play a major role in today's fifth generation (5G) mobile networks, where polar codes are used in the control channel with mid-sized blocklengths, whereas LDPC codes are used for transmission with larger blocklengths; cf. for example [8] and [9].

A crucial observation here is that all known effective code constructions (such as [10] for polar codes) show a strong dependency on the underlying channel. In general, the channel of interest is fixed and the code construction is designed beforehand for this particular channel. This means that the corresponding algorithm only gets the blocklength n as the input and then computes the corresponding encoder

and decoder. Its sequence of code rates should approach the capacity asymptotically. However, it would be of practical relevance to further have algorithms that are able to construct such sequences of capacity-achieving codes for a whole class of channels rather than designing all code constructions separately each one tailored to one specific channel realization. To be specific, for a practically relevant and interesting class of channels, we want to have an algorithm that gets both the channel realization (from the class of channels) and the blocklength n as inputs and then computes for that channel and that blocklength a suitable encoder and decoder. This would enable channel-aware transmission, where the coding scheme is algorithmically adapted to the quality of the underlying channel.

To address this issue from a fundamental algorithmic point of view, we use the concept of a *Turing machine* [11]–[13] and the corresponding *computability framework*. The Turing machine is a mathematical model of an abstract machine that manipulates symbols on a strip of tape according to certain given rules. It can simulate any given algorithm and therewith provides a simple but very powerful model of computation. Turing machines have no limitations on computational complexity, unlimited computing capacity and storage, and execute programs completely error-free. They are further equivalent to the von Neumann-architecture without hardware limitations and the theory of recursive functions, cf. also [14]–[18]. Accordingly, Turing machines provide fundamental performance limits for today's digital computers and are the ideal concept to study whether or not such optimal codes can be constructed algorithmically in principle.

Communication from a computability or algorithmic point of view has attracted some attention recently. In [19] the computability of the capacity functions of the wiretap channel under channel uncertainty and adversarial attacks is studied. The computability of the capacity of finite state channels is studied in [20] and of non-i.i.d. channels in [21]. These works have in common that they study capacity functions of various communication scenarios and analyze the algorithmic computability of the capacity function itself. While for discrete memoryless channels (DMCs) the capacity function is a computable continuous function and therewith indeed algorithmically computable [22], [23], this is no longer the case for certain multi-user scenarios or channels with memory. However, they do not consider actual code constructions which, to the best of our knowledge, have not been studied so far from a fundamental algorithmic point of view. In addition, even if the capacity is computable, it is still not clear whether or not it can be algorithmically approximated by capacity-achieving codes (which are discrete objects by definition) making this a non-trivial problem.

In this paper, we show that it is in general impossible to find a Turing machine that is able to construct sequences of capacity-achieving codes (and also sequences of sub-optimal codes that achieve only a fraction of the capacity) for a whole class of channels. To this end, we first introduce the computability framework based on Turing machines in Section II. The system model and precise problem formulation are subsequently given in Section III. Then, in Section IV we demonstrate that already for the class of binary symmetric channels (BSCs) such a Turing machine cannot exist. Subsequently, in Section V this negative result is extended to also hold for general DMCs. As a consequence, it follows that known algorithms or code constructions are not recursive in the channel and need to be specifically designed and tailored to the particular channel of interest. As a consequence, channel-aware transmission schemes with channel-dependent code adaption cannot be algorithmically realized in general. Finally, implications on resource allocation in combination with cross-layer optimization are discussed in Section VI.

A. Notation

Discrete random variables are denoted by capital letters and their realizations and ranges by lower case and script letters, respectively; all logarithms and information quantities are taken to the base 2; $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{R}_c$ are the sets of non-negative integers, rational numbers, real numbers, and computable real numbers, respectively; $\mathcal{P}(\mathcal{X})$ denotes the set of all probability distributions on $\mathcal{X}$ and $\mathcal{CH}(\mathcal{X};\mathcal{Y})$ denotes the set of all stochastic matrices (channels) $\mathcal{X} \to \mathcal{P}(\mathcal{Y})$.

II. Computability Framework and Digital Hardware Platform

We first introduce the computability framework based on Turing machines which provides the needed background. For this we need some basic definitions and concepts of computability which are briefly reviewed. The concept of computability and computable real numbers was first introduced by Turing in [11] and [12].

Recursive functions $f\colon \mathbb{N} \to \mathbb{N}$ map natural numbers into natural numbers and are exactly those functions that are computable by a Turing machine. They are the smallest class of partial functions that includes the primitive functions (i.e., constant function, successor function, and projection function) and is further closed under composition, primitive recursion, and minimization. For a detailed introduction, we refer the reader to [24] and [22]. With this, we call a sequence of rational numbers $\{r_n\}_{n\in\mathbb{N}}$ a *computable sequence* if there exist recursive functions $a, b, s\colon \mathbb{N} \to \mathbb{N}$ with $b(n) \neq 0$ for all $n \in \mathbb{N}$ and

$$r_n = (-1)^{s(n)} \frac{a(n)}{b(n)}, \quad n \in \mathbb{N}; \tag{1}$$

cf. [24, Def. 2.1 and 2.2] for a detailed treatment. A real number x is said to be computable if there exists a computable sequence of rational numbers $\{r_n\}_{n\in\mathbb{N}}$ and a recursive function φ such that we have for all $M \in \mathbb{N}$

$$|x - r_n| < 2^{-M} \tag{2}$$

for all $n \geq \varphi(M)$. Thus, the computable real x is represented by the pair $(\{r_n\}_{n\in\mathbb{N}}, \varphi)$. This form of convergence with a computable control of the approximation error is called *effective convergence*. Note that if a computable sequence of real numbers $\{x_n\}_{n\in\mathbb{N}}$ converges effectively to a limit x, then x is a computable real number, cf. [22]. Furthermore, the set $\mathbb{R}_c$ of all computable real numbers is closed for addition, subtraction, multiplication, and division (excluding the division by zero). A non-computable real number, for

example, is given by the Specker Sequence, which is discussed in greater detail in Example 2, see also [25] for the detailed construction. We denote the set of computable real numbers by $\mathbb{R}_c$. Based on this, we define the set of computable probability distributions $\mathcal{P}_c(\mathcal{X})$ as the set of all probability distributions $P_X \in \mathcal{P}(\mathcal{X})$ such that $P_X(x) \in \mathbb{R}_c$, $x \in \mathcal{X}$. Further, let $\mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be the set of all computable channels, i.e., for a channel $W\colon \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ we have $W(\cdot|x) \in \mathcal{P}_c(\mathcal{Y})$ for every $x \in \mathcal{X}$.

Definition 1: A function $f\colon \mathbb{R}_c \to \mathbb{R}_c$ is called *Borel-Turing computable* if there exists an algorithm that gets for every x an arbitrary representation $(\{r_n\}_{n\in\mathbb{N}}, \varphi)$ for it as input and then computes a representation $(\{\hat{r}_n\}_{n\in\mathbb{N}}, \hat{\varphi})$ for $f(x)$.

Turing's definition of computability conforms to the definition of Borel computability above. Here, we particularly consider the notion of a *computable continuous function*, cf. [22, Def. A].

Definition 2 ([22]): Let $\mathbb{I}_c = [0,1] \cap \mathbb{R}_c$ be the computable unit interval. A function $f\colon \mathbb{I}_c \to [0,1]$ is called *computable continuous* if:

1) f is *sequentially computable*, i.e., f maps every computable sequence $\{x_n\}_{n\in\mathbb{N}}$ of points $x_n \in \mathbb{I}_c$ into a computable sequence $\{f(x_n)\}_{n\in\mathbb{N}}$ of real numbers,
2) f is *effectively uniformly continuous*, i.e., there is a recursive function $d\colon \mathbb{N} \to \mathbb{N}$ such that for all $x, y \in \mathbb{I}_c$ and all $N \in \mathbb{N}$ with $\|x - y\| \leq \frac{1}{d(N)}$ it holds that $|f(x) - f(y)| \leq \frac{1}{2^N}$.

Remark 1: There are other forms of computability such as *Markov computability* and *Banach-Mazur computability*, of which the latter one is the weakest form of computability. In particular, Borel or Markov computability both imply Banach-Mazur computability, but not vice versa. For an overview of the logical relations between different notions of computability we again refer to [14] and, for example, the introductory textbook [13].

Definition 3: A subset $\{W_\lambda\}_{\lambda\in[0,1]} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y})$ is a *computable family of channels* if there exist computable continuous functions $f_{x,y}\colon [0,1] \to [0,1]$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$, such that for all $\lambda \in [0,1]$ we have $W_\lambda(y|x) = f_{x,y}(\lambda)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Some remarks are in order.

1) From this we observe that a computable family of channels is generated by computable continuous functions.
2) For $\lambda \in [0,1] \cap \mathbb{R}_c$ we always have $f_{xy}(\lambda) \in \mathbb{R}_c$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$. Therewith, we obtain $W_\lambda \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ for $\lambda \in [0,1] \cap \mathbb{R}_c$.

We further need the concepts of a recursive set and a recursively enumerable set as, for example, defined in [24].

Definition 4: A set $\mathcal{A} \subset \mathbb{N}$ is called *recursive* if there exists a computable function f such that $f(x) = 1$ if $x \in \mathcal{A}$ and $f(x) = 0$ if $x \notin \mathcal{A}$.

Definition 5: A set $\mathcal{A} \subset \mathbb{N}$ is *recursively enumerable* if there exists a recursive function whose range is exactly $\mathcal{A}$.

We have the following properties which will be crucial later for proving the desired results; cf. also [24] for further details.

- $\mathcal{A}$ is recursive is equivalent to $\mathcal{A}$ is recursively enumerable and $\mathcal{A}^c$ is recursively enumerable.
- There exist recursively enumerable sets $\mathcal{A} \subset \mathbb{N}$ that are not recursive, i.e., $\mathcal{A}^c$ is not recursively enumerable. This means there are no computable, i.e., recursive, functions $f\colon \mathbb{N} \to \mathcal{A}^c$ with $[f(\mathbb{N})] = \mathcal{A}^c$.

Example 1: Here, we provide an example of a set that is not recursive but is recursively enumerable. Let $\{\varphi_n\}_{n\in\mathbb{N}}$ be the set of recursive functions, where φ_n is computed by the Turing machine $\mathfrak{T}_n$. It can be shown that $\{\varphi_n\}_{n\in\mathbb{N}}$ is a computable sequence of recursive functions and that there exists a Turing machine $\mathfrak{T}_*$ that takes $(n, x) \in \mathbb{N}^2$ as inputs and stops if and only if $\varphi_n(x)$ is defined and computes the value $\varphi_n(x)$. Then the set $\mathcal{A} = \{n \in \mathbb{N}\colon \mathfrak{T}_n(n) \text{ stops}\}$ is recursively enumerable, but not recursive. We can find a Turing machine that either stops or runs forever. It stops for $n \in \mathbb{N}$ if and only if $n \in \mathcal{A}$ since for $n \in \mathbb{N}$ we take φ_n, i.e., $\mathfrak{T}_n$, and execute it until $\mathfrak{T}_n(n)$ stops. On the other hand, for the set $\mathcal{A}^c$ such a Turing machine cannot exist since otherwise we would be able to solve the famous halting problem, cf. [11] and [24].

Turing machines are extremely powerful compared to state-of-the-art digital signal processing (DSP) and field gate programmable array (FPGA) platforms and even current supercomputers. It is the most general computing model and is even capable of performing arbitrary exhaustive search tasks on arbitrary large but finite structures. The complexity can even grow faster than double-exponentially with the set of parameters of the underlying communication system (such as time, frequencies, transmit power, modulation scheme, number of antennas, etc.).

III. System Model and Problem Formulation

Here, we introduce the communication scenario of interest and formulate the main question of this work.

A. Communication System Model

We consider the most basic communication scenario of a point-to-point channel with one transmitter and one receiver. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite input and output alphabets. Then the channel is given by a stochastic matrix $W \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$ and the DMC is given by $W^n(y^n|x^n) := \prod_{i=1}^n W(y_i|x_i)$ for all $x^n \in \mathcal{X}^n$ and $y^n \in \mathcal{Y}^n$.

Definition 6: An (M_n, E_n, D_n)*-code* $\mathcal{C}_n(W)$ of block-length $n \in \mathbb{N}$ for the DMC $W \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$ consists of an encoder $E_n\colon \mathcal{M}_n \to \mathcal{X}^n$ at the transmitter with a set of messages $\mathcal{M}_n := \{1, \ldots, M_n\}$ and a decoder $D_n\colon \mathcal{Y}^n \to \mathcal{M}_n$ at the receiver.

The receiver needs to decode the transmitted message reliably. For this purpose, we define the *average error criterion* as

$$\bar{e}_n = \frac{1}{|\mathcal{M}_n|} \sum_{m\in\mathcal{M}_n} \sum_{y^n\colon\ D_n(y^n)\neq m} W^n(y^n|x_m^n) \tag{3}$$

and the *maximum error criterion*, respectively, as

$$e_{\max,n} = \max_{m\in\mathcal{M}_n} \sum_{y^n\colon\ D_n(y^n)\neq m} W^n(y^n|x_m^n) \tag{4}$$

with $x_m^n = E_n(m)$ the codeword for message $m \in \mathcal{M}_n$.

Remark 2: We can now also define an $(M_n, E_n, D_n, \epsilon)$-*code* $\mathcal{C}_n(W, \epsilon)$ of blocklength $n \in \mathbb{N}$ for the DMC $W \in \mathcal{CH}(\mathcal{X}; \mathcal{Y})$ which is an (M_n, E_n, D_n)-code of Definition 6 that further satisfies $\bar{e}_n \leq \epsilon$ (or $e_{\max,n} \leq \epsilon$ respectively), cf. (3) and (4).

Definition 7: A rate $R > 0$ is called *achievable* for the DMC W if there exists a sequence $\{\mathcal{C}_n(W, \epsilon_n)\}_{n\in\mathbb{N}}$ of $(M_n, E_n, D_n, \epsilon_n)$-codes such that we have $\frac{1}{n}\log M_n \geq R$ and $\bar{e}_n \leq \epsilon_n$ (or $e_{\max,n} \leq \epsilon_n$ respectively) with $\epsilon_n \to 0$ as $n \to \infty$. The *capacity* $C(W)$ is given by the supremum of all achievable rates R.

The capacity of the DMC goes back to the seminal work of Shannon [2].

Theorem 1: The capacity $C(W)$ of the DMC W under both average and maximum error criteria is $C(W) = \max_X I(X;Y)$.

The capacity characterizes the maximum transmission rate at which the transmitter can reliably transmit a message to the receiver with vanishing probability of decoding error. Note that for DMCs there is no difference in capacity whether the average probability of error or maximum probability of error criterion is assumed.

Remark 3: Information theoretic formulas such as capacity expressions are known for various communication scenarios including the point-to-point channel, multiple access channel or bidirectional broadcast channel. These provide the indispensable basis for resource allocation and therewith enable cross-layer optimization for modern communication systems [26]–[31].

In the networking community, resource allocation and cross-layer questions are addressed with the help of optimization techniques. The network utility functions of interest are then given by information theoretic quantities specified by the underlying channel parameters of the communication system. In these approaches, it is implicitly assumed that the optimization problem given the desired parameters such as power, blocklength, beamforming coefficients, or frequencies, as well as the channel conditions yields coding schemes that actually achieve the corresponding information theoretic quantities. This means that it is implicitly assumed that an effective algorithmic solution exists for the achievability part. To date, this problem is approached with the help of look-up tables, in which the corresponding coding schemes have been precomputed and stored for specific channel parameters of interest (see also the discussion in Section VI).

B. Problem Formulation

Proving the capacity of a channel usually involves two parts: achievability also known as a coding theorem and a corresponding (strong) converse. These parts establish matching lower and upper bounds on the capacity. In the coding theorem, it is proved that the rate given by the capacity can actually be achieved by a coding scheme. This means that, for sufficiently large blocklength, there are codes whose rates are sufficiently close to the capacity. The most natural approach for this would be the construction of an actual code that achieves capacity.

However, the proof of the coding theorem is usually non-constructive in the sense that only the existence of such coding schemes is shown by invoking probabilistic techniques. In particular for DMCs the following is shown: For fixed error $\epsilon > 0$, the coding theorem only yields the existence of a function F: $\mathcal{CH}(\mathcal{X}; \mathcal{Y}) \times \mathbb{N} \to \mathfrak{C}$ that maps a channel $W \in \mathcal{CH}(\mathcal{X}; \mathcal{Y})$ and a blocklength $n \in \mathbb{N}$ into a code $\mathcal{C}_n(W, \epsilon) \in \mathfrak{C}$ that guarantees the pre-specified error $0 < \epsilon < 1$, i.e., for every $W \in \mathcal{CH}(\mathcal{X}; \mathcal{Y})$ and every $n \in \mathbb{N}$ we have

$$F(W, n) = \mathcal{C}_n(W, \epsilon) \tag{5}$$

with

$$\lim_{n\to\infty} \frac{1}{n} \log |F(W, n)| = C(W). \tag{6}$$

The coding theorem does not only yield the existence of exactly one function F that satisfies (5)-(6), it rather allows the existence of arbitrarily many of these functions. In particular, every coding strategy that achieves the capacity asymptotically may result in a different function F. To this end, also in Shannon's seminal work, only the existence of such functions has been shown and no requirements have been put on whether or not such functions are effectively constructible.

Accordingly, the central question that arises is whether or not this problem can be solved algorithmically. That is, does a function F_* exist for which we can find a Turing machine $\mathfrak{T}_*$ that yields $\mathfrak{T}_*(W, n) = F_*(W, n)$ for all $W \in \mathcal{CH}_c(\mathcal{X}; \mathcal{Y})$ and all $n \in \mathbb{N}$? As we will see later, in general the answer to this question is negative.

But before we continue, it is important to stress the fact that even if some (mathematical) objects exist, this does not immediately imply that they can be constructed or characterized algorithmically; see for example the Specker Sequence [25] or Fekete's Lemma [32]. A detailed discussion can also be found in [33] and we present the example of the Specker Sequence in the following, since it is closely related to what we are interested in.

Example 2 (Specker Sequence): We start with an arbitrary computable sequence $\{a_k\}_{k\in\mathbb{N}}$ of rational numbers with $a_k < a_{k+1}$, $k \in \mathbb{N}$, and $a_k \in [0, 1]$. It is clear that there exists exactly one real number $a_* \in [0, 1]$ such that $\lim_{k\to\infty} a_k = a_*$ holds and the sequence $\{a_k\}_{k\in\mathbb{N}}$ converges monotonically increasingly to a_*. This means the error $a_* - a_k$, $k \in \mathbb{N}$, is monotonically decreasing with k and converges to zero.

Already in 1949, Specker constructed such a sequence $\{a_k\}_{k\in\mathbb{N}}$ for which the limit a_* is not a computable real number [25]. This number a_* exists but is a transcendental number, i.e., it has a unique binary representation

$$a_* = \sum_{n=1}^{\infty} b_n \frac{1}{2^n}$$

with $b_n \in \{0, 1\}$. However, there exists no algorithm that can compute the coefficients b_n, $n \in \mathbb{N}$. The same result is also true for the decimal representation of the number a_*.

This example has the following consequences. Whenever it is proved that every uniformly bounded, monotonically increasing sequence of real numbers has a limit, then this

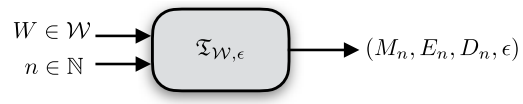


Fig. 1. Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}$ for code construction. For a given tolerated decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, it takes the channel realization $W \in \mathcal{W}$ and blocklength n as inputs and outputs the desired $(M_n, E_n, D_n, \epsilon)$-code $\mathcal{C}_n(W, \epsilon)$.

proof shows only the existence of this limit. Specker's example above demonstrates that it is possible that this limit cannot be algorithmically computed, i.e., there is no algorithm that takes the sequence $\{a_k\}_{k\in\mathbb{N}}$ as input and outputs for any given n of the binary representation of a_* the corresponding coefficients $b_n(a_*)$. Accordingly, such a Turing machine cannot exist.

From a practical point of view it would be desirable to have a universal algorithm for a whole class of DMCs $\mathcal{W} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y})$ that takes the channel realization $W \in \mathcal{W}$ and the blocklength n as inputs and then computes an $(M_n, E_n, D_n, \epsilon)$-code $\mathcal{C}_n(W, \epsilon)$ with $\lim_{n\to\infty} \frac{1}{n} \log M_n = C(W)$ for a given tolerated decoding error $0 < \epsilon < 1$. This is visualized in Fig. 1. Note that the strong converse holds for DMCs so that the specific decoding error ϵ does not play a role asymptotically. As a consequence, allowing a positive, but non-vanishing decoding error does not increase the capacity, i.e., $C(W) = C_\epsilon(W)$ for all $0 < \epsilon < 1$ with C_ϵ the ϵ-capacity tolerating the error ϵ.

From an algorithmic perspective, it is reasonable to restrict the whole analysis to computable channels $\mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ as only such channels are accepted and can be processed by Turing machines. Let $\mathcal{W} \subset \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be a practically relevant class (subset) of channels. We pose the following question:

Question 1: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$ be a rational number and fixed. Is there an algorithm (or Turing machine) $\mathfrak{T}_{\mathcal{W},\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with

$$\lim_{n\to\infty} \frac{1}{n} \log M_n = C(W)? \tag{7}$$

Some discussion is in order:

1) From the coding theorem and the corresponding strong converse result we know that for each channel $W \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ there exists such a sequences of codes with the mapping $W \to \{(M_n, E_n, D_n, \epsilon)\}_{n\in\mathbb{N}}$ that satisfies (7). However, it is not clear which properties this mapping has and the question is now whether or not such a sequence of codes can actually be constructed algorithmically. This question remains valid even if the corresponding capacity functions $C(W)$ are computable, since this does not immediately imply that it can be approximated by actual codes which are discrete mathematical objects.
2) If we consider the special case of $|\mathcal{X}| = |\mathcal{Y}| = 2$ and $\mathcal{W} = \{\text{BSC}(p)\}$ for one fixed computable real $p \in \mathbb{R}_c \cap [0,1]$, i.e., the subset of channels contains only a single BSC, then the answer to the question above is "yes" as capacity-achieving code constructions are known.
3) If we consider the set of BSCs $\mathcal{W} = \bigcup_{p\in\mathbb{R}_c\cap[0,1]} \{\text{BSC}(p)\}$, then we only ask for the construction of capacity-achieving codes for a practically relevant subset of all channels. We will see that even this is algorithmically impossible.

IV. Non-Constructability for BSCs

In this section, we begin with showing that already for simple classes (or sets) of DMCs $\mathcal{W}$ the answer to Question 1 is negative.

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_L\}$ be a finite state set describing L relevant states. Let F be a function that maps a subset $\mathcal{W} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y})$ of channels to $\mathcal{S}$, i.e., $F\colon \mathcal{W} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y}) \to \mathcal{S}$. The function F need not necessarily be defined for all channels. Later, we will consider the set of all BSCs which is a strict subset of all binary input channels. Next, we study Question 1 not only for the set of computable channels, but also for arbitrary computable families of channels according to Definition 3. We need the following definition.

Definition 8: Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary but finite alphabets with $|\mathcal{X}| \geq 2$, $|\mathcal{Y}| \geq 2$, and $\{W_\lambda\}_{\lambda\in[0,1]} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y})$ be an arbitrary computable family of channels. Further, let $F\colon \{W_\lambda\}_{\lambda\in[0,1]} \to \mathcal{S} = \{s_1, s_2, \ldots, s_L\}$ be a function that maps every channel $W \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$ to a corresponding state $F(W) \in \mathcal{S}$. A finite classification problem is said to be *non-trivial* if there exist $\lambda_1, \lambda_2 \in [0,1] \cap \mathbb{R}_c$ and corresponding channels $\hat{W}_1 := W_{\lambda_1}$ and $\hat{W}_2 := W_{\lambda_2}$ with $\hat{W}_1, \hat{W}_2 \in \{W_\lambda\}_{\lambda\in[0,1]} \cap \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ that lead to different states, i.e., $F(\hat{W}_1) \neq F(\hat{W}_2)$.

Remark 4: If a classification task F for a computable family of channels $\{W_\lambda\}_{\lambda\in[0,1]}$ is trivial, then there must exists an $s_* \in \mathcal{S}$ such that for all $\lambda \in [0,1] \cap \mathbb{R}_c$ we always have $F(W_\lambda) = s_*$. Then, there would be nothing for a Turing machine to do, since it would obtain only channels W_λ, $\lambda \in [0,1] \cap \mathbb{R}_c$ as possible inputs, but for these inputs no classification is needed.

For a fixed computable family of channels $\{W_\lambda\}_{\lambda\in[0,1]}$, let $\mathfrak{T}_F$ be a Turing machine for the classification task F, i.e., $\mathfrak{T}_F$ is defined for the set of channels $\{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c}$. Note that $\mathfrak{T}_F$ need not necessarily be defined for channels from $\mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ that are not in the computable family of channels. The following result shows that in general there is no Turing machine $\mathfrak{T}_F$ or algorithm that can solve a non-trivial classification problem F.

Lemma 1: Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary but finite alphabets with $|\mathcal{X}| \geq 2$, $|\mathcal{Y}| \geq 2$, and $\{W_\lambda\}_{\lambda\in[0,1]} \subset \mathcal{CH}(\mathcal{X};\mathcal{Y})$ be an arbitrary computable family of channels. Every non-trivial classification problem $F\colon \{W_\lambda\}_{\lambda\in[0,1]} \to \mathcal{S}$ is not algorithmically solvable, i.e., there exists no Turing machine $\mathfrak{T}_F\colon \{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c} \to \mathcal{S}$ that takes the channel $W \in \{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c}$ as an input and outputs the corresponding state $F(W) \in \mathcal{S}$.

Proof: We prove the desired result by contradiction. We assume that for fixed and finite alphabets $\mathcal{X}$ and $\mathcal{Y}$

there is a computable family $\{W_\lambda\}_{\lambda\in[0,1]}$ of channels $W_\lambda \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$, an $L\in\mathbb{N}$, and a non-trivial classification problem $F\colon \{W_\lambda\}_{\lambda\in[0,1]} \to \{s_1,\dots,s_L\}$ such that there exists a Turing machine $\mathfrak{T}_F$ for which we have $\mathfrak{T}_F(W) = F(W)$ for all $W \in \{W_\lambda\}_{\lambda\in[0,1]} \cap \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$.

Further, for $\mathcal{S} = \{s_1, s_2, \dots, s_L\}$ with $L > 2$ we can construct a Turing machine $\tilde{\mathfrak{T}}_F$ that solves a non-trivial binary classification task as follows. For $L > 2$, there exist some $\lambda_i, \lambda_j \in [0,1]\cap\mathbb{R}_c$ and corresponding $W_i := W_{\lambda_i}$ and $W_j := W_{\lambda_j}$ with $W_i \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ and $W_j \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ and $F(W_i) = s_i \neq s_j = F(W_j)$. Now we choose the state set $\hat{\mathcal{S}} = \mathcal{S}_1 \cup \mathcal{S}_2$ with $\mathcal{S}_1 = \{s_1\}$ and $\mathcal{S}_2 = \bigcup_{j\neq i}\{s_j\}$. We have $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and consider the classification problem $\hat{F}\colon \{W_\lambda\}_{\lambda\in[0,1]} \to \hat{\mathcal{S}}$ with $\hat{F}(W) \in \mathcal{S}_1 \Leftrightarrow F(W) = s_i$ and $\hat{F}(W) \in \mathcal{S}_2 \Leftrightarrow F(W) = s_j$ for $s_j \neq s_i$. This yields a non-trivial classification task for the binary case $L = 2$. Thus, if there would exist a Turing machine $\mathfrak{T}_F$ for $L > 2$, then the modified Turing machine $\hat{\mathfrak{T}}_{\hat{F}}$ with

$$\hat{\mathfrak{T}}_{\hat{F}}(W) = \begin{cases} s_i & \text{if } \mathfrak{T}_F(W) = s_i \\ s_j & \text{if } \mathfrak{T}_F(W) \in \{s_1,\dots,s_L\}\backslash\{s_i\} \end{cases} \tag{8}$$

would solve the corresponding binary problem. Without loss of generality, we accordingly assume only $L = 2$ different states, i.e., $\mathcal{S} = \{s_1, s_2\}$ in the following.

Now, we prove the binary case by contradiction, i.e., we assume that there exist input and output alphabets $\mathcal{X}$ and $\mathcal{Y}$, and a non-trivial classification $F\colon \{W_\lambda\}_{\lambda\in[0,1]} \to \{s_1, s_2\}$ such that there exists a Turing machine $\mathfrak{T}_F$ that correctly outputs $\mathfrak{T}_F(W) = F(W)$ for all $W \in \{W_\lambda\}_{\lambda\in[0,1]} \cap \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$. Since F is non-trivial, there must exist some $\lambda_1, \lambda_2 \in [0,1]\cap\mathbb{R}_c$ and $\hat{W}_1 = W_{\lambda_1}$ and $\hat{W}_2 = W_{\lambda_2}$ with $\hat{W}_1, \hat{W}_2 \in \{W_\lambda\}_{\lambda\in[0,1]} \cap \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ and $F(\hat{W}_1) \neq F(\hat{W}_2)$.

Without loss of generality, we assume $0 \leq \hat{\lambda}_1 < \hat{\lambda}_2 \leq 1$ and consider the computable family of channels $\{W_\lambda\}_{\lambda\in[\hat{\lambda}_1,\hat{\lambda}_2]}$. We inductively construct two computable sequences $\{W_1^{(n)}\}_{n\in\mathbb{N}}$ and $\{W_2^{(n)}\}_{n\in\mathbb{N}}$ of computable channels where the Turing machine $\mathfrak{T}_F$ plays a crucial role. As an initialization step, we set $\tilde{\lambda}^{(1)} = \frac{\hat{\lambda}_1+\hat{\lambda}_2}{2}$. Since $\hat{\lambda}_1 < \hat{\lambda}_2$, it holds that $\hat{\lambda}_1 < \tilde{\lambda}^{(1)} < \hat{\lambda}_2$ and we compute the output $\mathfrak{T}_F(W_{\tilde{\lambda}^{(1)}})$ of the Turing machine $\mathfrak{T}_F$ for which we must have either $\mathfrak{T}_F(W_{\tilde{\lambda}^{(1)}}) = s_1$ or $\mathfrak{T}_F(W_{\tilde{\lambda}^{(1)}}) = s_2$ since we consider the binary case with $\mathcal{S} = \{s_1, s_2\}$. In the former case, we set the first element of the sequences to

$$W_1^{(1)} := W_{\tilde{\lambda}^{(1)}} \text{ and } W_2^{(1)} := W_{\hat{\lambda}_2} \tag{9}$$

and further set

$$\lambda_1^{(1)} := \tilde{\lambda}^{(1)} = \frac{\hat{\lambda}_1+\hat{\lambda}_2}{2} \text{ and } \lambda_2^{(1)} := \hat{\lambda}_2.$$

In the latter case, we set the first element of the sequences to

$$W_1^{(1)} := W_{\hat{\lambda}_1} \text{ and } W_2^{(1)} := W_{\tilde{\lambda}^{(1)}} \tag{10}$$

and further set

$$\lambda_1^{(1)} := \hat{\lambda}_1 \text{ and } \lambda_2^{(1)} := \tilde{\lambda}^{(1)} = \frac{\hat{\lambda}_1+\hat{\lambda}_2}{2}.$$

Note that we always have $\lambda_1^{(1)} < \lambda_2^{(1)}$. Further, these choices satisfy $W_1^{(1)}, W_2^{(1)} \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y}) \cap \{W_\lambda\}_{\lambda\in[0,1]}$ and $\lambda_1^{(1)}, \lambda_2^{(1)} \in \mathbb{R}_c$.

To specify the following elements, we proceed accordingly. Assume that for $n\in\mathbb{N}$ we have already computed the channels $W_1^{(k)}, W_2^{(k)}$ and the numbers $\lambda_1^{(k)}, \lambda_2^{(k)}$ for all $1 \leq k \leq n$. We then set $\tilde{\lambda}^{(n+1)} = \frac{\lambda_1^{(n)}+\lambda_2^{(n)}}{2}$ and compute the output $\mathfrak{T}_F(W_{\tilde{\lambda}^{(n+1)}})$ of the Turing machine $\mathfrak{T}_F$. We either must have $\mathfrak{T}_F(W_{\tilde{\lambda}^{(n+1)}}) = s_1$ or $\mathfrak{T}_F(W_{\tilde{\lambda}^{(n+1)}}) = s_2$. In the former case, we set $W_1^{(n+1)} := W_{\tilde{\lambda}^{(n+1)}}$ and $W_2^{(n+1)} := W_2^{(n)}$ as well as $\lambda_1^{(n+1)} := \tilde{\lambda}^{(n+1)} = \frac{\lambda_1^{(n)}+\lambda_2^{(n)}}{2}$ and $\lambda_2^{(n+1)} := \lambda_2^{(n)}$. In the latter case, we set $W_1^{(n+1)} := W_1^{(n)}$ and $W_2^{(n+1)} := W_{\tilde{\lambda}^{(n+1)}}$ as well as $\lambda_1^{(n+1)} := \lambda_1^{(n)}$ and $\lambda_2^{(n+1)} := \tilde{\lambda}^{(n+1)} = \frac{\lambda_1^{(n)}+\lambda_2^{(n)}}{2}$. Note that we always have $\lambda_1^{(n+1)} < \lambda_2^{(n+1)}$. This procedure defines the sequences $\{W_1^{(n)}\}_{n\in\mathbb{N}}$, $\{W_2^{(n)}\}_{n\in\mathbb{N}}$, $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$, and $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ inductively. Since $\mathfrak{T}_F$ is a Turing machine, $\{W_1^{(n)}\}_{n\in\mathbb{N}}$ and $\{W_2^{(n)}\}_{n\in\mathbb{N}}$ are computable sequences of computable channels and $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$ and $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ are computable sequences of computable real numbers.

We have $\{W_1^{(n)}\}_{n\in\mathbb{N}} \subset \{W_\lambda\}_{\lambda\in[0,1]}$ and $\{W_2^{(n)}\}_{n\in\mathbb{N}} \subset \{W_\lambda\}_{\lambda\in[0,1]}$. For $\lambda \in [0,1]$, the function $\Psi\colon [0,1] \to \mathcal{CH}(\mathcal{X};\mathcal{Y})$ defined by

$$\Psi(\lambda) := \begin{pmatrix} f_{1,1}(\lambda) & \dots & f_{|\mathcal{X}|,1}(\lambda) \\ \vdots & \ddots & \vdots \\ f_{1,|\mathcal{Y}|}(\lambda) & \dots & f_{|\mathcal{X}|,|\mathcal{Y}|}(\lambda) \end{pmatrix}$$

with $f_{x,y}(\lambda) = W_\lambda(y|x)$, cf. Definition 3, is a computable continuous function. Thus, the set $\{W_\lambda\}_{\lambda\in[0,1]}$ is a closed set in $\mathcal{CH}(\mathcal{X};\mathcal{Y})$. Due to the constructions of the sequences $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$ and $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ we have $\lambda_1^{(n)} \leq \lambda_1^{(n+1)}$ and $\lambda_2^{(n+1)} \leq \lambda_2^{(n)}$ for all $n\in\mathbb{N}$. Furthermore, it holds that

$$\begin{aligned} \lambda_2^{(n+1)} - \lambda_1^{(n+1)} &= \max\left(\lambda_2^{(n+1)} - \tilde{\lambda}^{(n+1)}, \tilde{\lambda}^{(n+1)} - \lambda_1^{(n+1)}\right) \\ &= \frac{\lambda_2^{(n)} - \lambda_1^{(n)}}{2} \end{aligned}$$

so that

$$\lambda_2^{(n+1)} - \lambda_1^{(n+1)} \leq \frac{1}{2^n}(\hat{\lambda}_2 - \hat{\lambda}_1) \leq \frac{1}{2^n}.$$

Since $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$ is a monotonically increasing sequence and $\hat{\lambda} \in \mathbb{R}_c$, the sequence $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$ converges effectively to $\hat{\lambda}$. Similar arguments hold for the computable sequence $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$, cf. [22] and [20].

Let $D(W_1, W_2) = \max_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} |W_1(y|x) - W_2(y|x)|$ be the distance between the channels $W_1 \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$ and $W_2 \in \mathcal{CH}(\mathcal{X};\mathcal{Y})$. Since Ψ is a computable continuous function, we have $\Psi(\hat{\lambda}) = W_{\hat{\lambda}}$ and it holds $\lim_{n\to\infty} D(W_{\hat{\lambda}}, W_1^{(n)}) = \lim_{n\to\infty} D(W_{\hat{\lambda}}, W_2^{(n)}) = 0$ where the convergence is effective so that $W_{\hat{\lambda}} \in \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$. We also have $W_{\hat{\lambda}} \in \{W_\lambda\}_{\lambda\in[0,1]}$ so that $W_{\hat{\lambda}}$ is a valid input for $\mathfrak{T}_F$ and we either obtain $\mathfrak{T}_F(W_{\hat{\lambda}}) = s_1$ or $\mathfrak{T}_F(W_{\hat{\lambda}}) = s_2$.

We define the sets

$$\begin{aligned} \mathcal{M}_1 &= \left\{\lambda \in [\hat{\lambda}_1, \hat{\lambda}_2] \cap \mathbb{R}_c\colon \mathfrak{T}_F(W_\lambda) = s_1\right\}, \\ \mathcal{M}_2 &= \left\{\lambda \in [\hat{\lambda}_1, \hat{\lambda}_2] \cap \mathbb{R}_c\colon \mathfrak{T}_F(W_\lambda) = s_2\right\}. \end{aligned}$$

Next, we consider the case for which $\hat{\lambda} \in \mathcal{M}_1$ holds and analyze the sequence $\{\lambda_2^{(n)}\}$. For this purpose, let $\mathcal{A} \subset \mathbb{N}$ be an arbitrary recursively enumerable set such that $\mathcal{A}$ is not recursive, i.e., $\mathcal{A}^c$ is not a recursively enumerable set. With the definition of recursively enumerable sets, cf. Definition 5, we can construct a total function g, i.e., $\text{domain}(g) = \mathbb{N}$, such that the range of g is $\text{range}(g) = \mathcal{A}^c$ and g is recursive and therewith a computable function. Furthermore, without loss of generality, we can require that g: $\mathbb{N} \to \mathcal{A}$ is a one-to-one mapping from $\mathbb{N}$ to $\mathcal{A}$.

Next, we use a similar construction as in [34] and [23] which relies on a construction of Pour-El, cf. Case I on page 336 in [35]. For every $(n, m) \in \mathbb{N} \times \mathbb{N}$ we define the computable function q: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ as

$$q(n,m) = \begin{cases} 2^{m+2} & n \notin \{g(0), \ldots, g(2^{m+2})\} \\ r & n \in \{g(0), \ldots, g(2^{m+2})\} \text{ and } g(r) = n. \end{cases} \tag{11}$$

Note that r above is unique. Since $\mathcal{A}$ is recursively enumerable, the function q is indeed recursive and therewith computable.

For $n \in \mathbb{N}$ we define the sequence $\{\hat{\lambda}^{(n)}\}_{n\in\mathbb{N}}$ with

$$\hat{\lambda}^{(n)} = \begin{cases} \hat{\lambda} & \text{if } n \in \mathcal{A}^c \\ \lambda_2^{(r)} & \text{if } n \in \mathcal{A} \text{ and } g(r) = n. \end{cases}$$

Note that for $n \in \mathcal{A}$ there can only be a single r with $g(r) = n$.

Next, we consider the double sequence $\{\lambda_2^{q(n,m)}\}_{n\in\mathbb{N},m\in\mathbb{N}}$. Note that this is only a suitable variation of the computable sequence $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ which is effectively computable since q is a recursive function. In the following we show that the sequence $\{\lambda_2^{q(n,m)}\}_{n\in\mathbb{N},m\in\mathbb{N}}$ effectively converges in $(n, m) \in \mathbb{N} \times \mathbb{N}$ to $\hat{\lambda}^{(n)} \in \mathbb{R}_c$. In the following, we show that for all $n \in \mathbb{N}$ and $m \in \mathbb{N}$ we always have

$$|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| < \frac{1}{2^m}.$$

This implies then that $\{\hat{\lambda}^{(n)}\}_{n\in\mathbb{N}}$ is indeed a computable sequence of computable real numbers.

For $n \in \mathcal{A}$ let m_0 be the smallest natural number such that $n \in \{g(0), \ldots, g(2^{m_0+2})\}$. Then, $g(r) = n$ is satisfied for exactly one $r \in \mathbb{N}$. Now, for all $m \geq m_0$ we have $q(n, m) = r$ and therewith also $\lambda_2^{q(n,m)} = \lambda_2^{(r)}$ for all $m \geq m_0$. Since $n \in \mathcal{A}$, we have $\lambda_2^{(r)} = \hat{\lambda}^{(n)}$ so that

$$|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| = 0 < \frac{1}{2^m}$$

for all $m \geq m_0$.

If $n \in \mathcal{A}$, but $m < m_0$, then we have $n \notin \{g(0), \ldots, g(2^{m+2})\}$ so that $q(n, m) = 2^{m+2}$ and therewith $\lambda_2^{q(n,m)} = \lambda_2^{(2^{m+2})}$. But this implies

$$\begin{aligned} |\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| &= |\lambda_2^{(r)} - \lambda_2^{(2^{m+2})}| \\ &= |\lambda_2^{(r)} - \hat{\lambda} + \hat{\lambda} - \lambda_2^{(2^{m+2})}| \\ &\leq |\hat{\lambda} - \lambda_2^{(r)}| + |\hat{\lambda} - \lambda_2^{(2^{m+2})}| \\ &\leq \frac{1}{2^{r-1}} + \frac{1}{2^{2^{m+2}-1}}. \end{aligned}$$

Since $n \notin \{g(0), \ldots, g(2^{m+2})\}$ and $g(r) = n$, we must have $r > 2^{m+2}$ for all $m < m_0$. Accordingly, we have

$$\frac{1}{2^{r-1}} + \frac{1}{2^{2^{m+2}-1}} < \frac{2}{2^{2^{m+2}} - 1} = \frac{1}{2^{2^{m+2}-2}} < \frac{1}{2^m}, \tag{12}$$

since for $m \geq 1$ we always have $2^{m+2} - 2 > m$. As a consequence, we always have $|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| < \frac{1}{2^m}$, i.e., for all $n \in \mathcal{A}$ and $m \in \mathbb{N}$ we have

$$|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| < \frac{1}{2^m}. \tag{13}$$

Now we consider the case $n \in \mathcal{A}^c$. Here we have $q(n, m) = 2^{m+2}$ so that

$$|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| \leq \frac{1}{2^{2^{m+2}-2}} < \frac{1}{2^m}. \tag{14}$$

From (13) and (14) we conclude that for all $n \in \mathbb{N}$ and $m \in \mathbb{N}$ we have $|\hat{\lambda}^{(n)} - \lambda_2^{q(n,m)}| < \frac{1}{2^m}$. This implies that the corresponding sequence $\{\hat{\lambda}^{(n)}\}_{n\in\mathbb{N}}$ is computable as well so that $\{W_{\hat{\lambda}^{(n)}}\}_{n\in\mathbb{N}}$ is a computable sequence of computable channels from $\{W_\lambda\}_{\lambda\in[0,1]}$. This further implies that the sequence $\{\mathfrak{T}_F(W_{\hat{\lambda}^{(n)}})\}_{n\in\mathbb{N}}$ of outputs of the Turing machine $\mathfrak{T}_F$ is also a computable sequence with values in $\{s_1, s_2\}$. Based on this, we can now construct an algorithm that decides for every $k \in \mathbb{N}$ whether $k \in \mathcal{A}$ or $k \in \mathcal{A}^c$. This construction is as follows:

We compute $\mathfrak{T}_F(W_{\hat{\lambda}^{(k)}})$. If $\mathfrak{T}_F(W_{\hat{\lambda}^{(k)}}) = s_2$, then we have $k \in \mathcal{A}$. Otherwise, if $\mathfrak{T}_F(W_{\hat{\lambda}^{(k)}}) = s_1$, then we have $k \in \mathcal{A}^c$, since for $k \in \mathcal{A}$ we must have $\hat{\lambda}^{(k)} \in \mathcal{M}_{s_2}$ so that $\mathfrak{T}_F(W_{\hat{\lambda}^{(k)}}) = s_2$. Accordingly, if $k \in \mathcal{A}^c$, we have $\hat{\lambda}^{(k)} \in \mathcal{M}_{s_1}$. This yields the desired algorithm.

Since $\mathcal{A}$ is recursively enumerable but not recursive, such an algorithm cannot exist, which is a contradiction proving the desired result for this case. Note that we actually showed that the function F is not Banach-Mazur computable, which then implies it is also not Turing computable.

We continue with the other case $\hat{\lambda} \in \mathcal{M}_{s_2}$. Here, we consider $\{\lambda_1^{(n)}\}_{n\in\mathbb{N}}$ and follow the same arguments for the first case above which proves the desired result for the second case. ∎

Remark 5: A crucial step in the proof is the construction of the sequence $\{\hat{\lambda}_n\}_{n\in\mathbb{N}}$ that needs to be a computable sequence of computable real numbers in the interval $[0, 1]$. For this purpose, we modified the sequence $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ in a suitable way. In particular, on the recursively enumerable set $\mathcal{A}$, $\hat{\lambda}^{(n)}$ needs to interpolate the sequence $\{\lambda_2^{(n)}\}_{n\in\mathbb{N}}$ appropriately. This is ensured by the second condition in (8). On the set $\mathcal{A}^c$ that is not recursively enumerable, the limit $\hat{\lambda}$ needs to be stable so that the sequence $\{\hat{\lambda}_n\}_{n\in\mathbb{N}}$ actually remains computable. This is ensured by the first condition in (8).

Now, the general result given in Lemma 1 allows us to prove the following result.

Theorem 2: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0, 1)$, be fixed. For $|\mathcal{X}| = |\mathcal{Y}| = 2$ and $\mathcal{W} = \bigcup_{p\in\mathbb{R}_c\cap[0,1]}\{BSC(p)\}$ there is no Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with $\lim_{n\to\infty} \frac{1}{n} \log M_n = C(W)$.

Proof: The desired result is proved by contradiction. Therefore, we assume that for the set $\mathcal{W} = \bigcup_{p\in\mathbb{R}_c\cap[0,1]}$

$\{\text{BSC}(p)\}$ there exists such a Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code. Then, for $n \in \mathbb{N}$ and $W \in \mathcal{W}$ we consider the function $f_n(W) = \frac{1}{n}\log M_n(W)$. Note that the function f_n depends also on ϵ, but ϵ is fixed and we study how the sequence $\{f_n(W)\}_{n\in\mathbb{N}}$ depends on the channel W. Since the strong converse holds, we have

$$\lim_{n\to\infty} f_n(W) = C(W), \quad W \in \mathcal{W}.$$

For $\text{BSC}(0)$ we denote the corresponding channel by W_0 and we have $C(W_0) = 1$. For $\text{BSC}(\frac{1}{2})$ we denote the corresponding channel by $W_{\frac{1}{2}}$ and we have $C(W_{\frac{1}{2}}) = 0$. Therefore, it holds that $\lim_{n\to\infty} f_n(W_0) = 1$ and $\lim_{n\to\infty} f_n(W_{\frac{1}{2}}) = 0$ and, as a consequence, there exists an n_0 such that for all $n \geq n_0$ we have

$$f_n(W_0) > \frac{3}{4} \text{ and } f_n(W_{\frac{1}{2}}) < \frac{1}{4}.$$

For $n \geq n_0$ fixed, we can define the Turing machine $\mathfrak{T}^*_n(W) = \mathfrak{T}_{\mathcal{W},\epsilon}(W, n)$, $W \in \mathcal{W}$, that only obtains the channel $W \in \mathcal{W}$ as input and computes for W an $(M_n, E_n, D_n, \epsilon)$-code. We further define the Turing machine

$$\tilde{\mathfrak{T}}^*_n(W) = M_n(W), \tag{15}$$

which can be seen as a subpart of the Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}(W, n) = (M_n, E_n, D_n, \epsilon)$, $W \in \mathcal{W}$, $n \in \mathbb{N}$, that outputs the first component of $\mathfrak{T}_{\mathcal{W},\epsilon}$. We have $\tilde{\mathfrak{T}}^*_n(W) = 2^{nf_n(W)}$ and with this we obtain $\tilde{\mathfrak{T}}^*_n(W_{\frac{1}{2}}) < 2^{\frac{n}{4}}$ and $\tilde{\mathfrak{T}}^*_n(W_0) > 2^{\frac{3}{4}n}$ so that $\tilde{\mathfrak{T}}^*_n(W_{\frac{1}{2}}) \neq \tilde{\mathfrak{T}}_n(W_0)$. Furthermore, from (15) we see that

$$\tilde{\mathfrak{T}}^*_n \colon \mathcal{W} \to \{1, \ldots, |\mathcal{X}|^n\}, \quad |\mathcal{X}|^n < \infty \tag{16}$$

as the total number of possible sequences in $|\mathcal{X}|^n$ is a trivial upper bound on the number of codewords M_n.

Now, we can apply Lemma 1 to establish the desired contradiction proving the result, since for $n \geq n_0$ the Turing machine $\tilde{\mathfrak{T}}^*_n$ yields a non-trivial classification due to (16). ■

Remark 6: The proof of Theorem 2 can be extended to the case of arbitrary finite alphabets $\mathcal{X}$, $\mathcal{Y}$, and $W \in \mathcal{CH}_c(\mathcal{X}; \mathcal{Y})$.

Remark 7: This result shows that we have the same behavior as for the Specker Sequence and Fekete's Lemma. Thus, we know that for every DMC W there must exist a channel-dependent sequence of codes whose sequence of rates converges to the capacity $C(W)$. However, this sequence of codes cannot be constructed recursively in dependence on W.

Remark 8: Note that this remains true even for a computable family of channels where all channels have the same capacity value. To this end, we can consider a sequence of rates that is capacity-achieving for this computable family of channels. For these, we can accordingly choose for all sufficiently large n encoders and decoders depending on the channel that result in a classification problem as in Lemma 1. More precisely, for fixed n, the set of all possible encodings can be interpreted as the state sets in Lemma 1 and so can the set of all possible decodings. Note that input and output alphabets are finite so that the resulting state set is finite as well. If these mappings do not yield the same codebook for all channels, then the corresponding classification problem is non-trivial and the corresponding encoder and decoder cannot be constructed algorithmically. This shows that the discrete structure of the code (i.e., the encoder and decoder) causes the algorithmic non-constructability.

One may think that the negative result of Theorem 2 and the non-existence of the desired Turing machine stems from the "large" class of channels (in Theorem 2 the set of channels $\mathcal{W}$ actually contains all possible BSCs). However, as we will see in the following theorem, restricting the set of channels does not change the result.

Theorem 3: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, be fixed. Further, let $p_1, p_2 \in \mathbb{R}_c \cap [0,1]$ with $p_1 < p_2$ arbitrary and define the set $\mathcal{W}_{p_1,p_2} = \bigcup_{p\in\mathbb{R}_c\cap[p_1,p_2]}\{\text{BSC}(p)\}$. Then Theorem 2 remains true, i.e., there is no Turing machine $\mathfrak{T}_{\mathcal{W}_{p_1,p_2},\epsilon}$ that takes $W \in \mathcal{W}_{p_1,p_2}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with $\lim_{n\to\infty}\frac{1}{n}\log M_n = C(W)$.

Proof: We first consider the case $0 < p_1 < p_2 \leq \frac{1}{2}$. Let $W_i = \text{BSC}(p_i)$, $i = 1, 2$, and we have $C(W_1) > C(W_2)$. The result is proved by contradiction and, therefore, we assume that there exists a Turing machine $\mathfrak{T}_{\mathcal{W}_{p_1,p_2},\epsilon}$ that can compute the desired code. For $n \in \mathbb{N}$ we consider the function $f_n(W) = \frac{1}{n}\log M_n(W)$, $W \in \mathcal{W}_{p_1,p_2}$. Since $\lim_{n\to\infty} f_n(W_i) = C(W_i)$, $i = 1, 2$, there exists a natural number $n_0 = n_0(p_1, p_2)$ such that for all $n \geq n_0$ we have

$$f_n(W_{p_2}) < C(W_2) + \frac{C(W_1) - C(W_2)}{4}$$

and

$$f_n(W_{p_1}) > C(W_1) - \frac{C(W_1) - C(W_2)}{4}.$$

Now, with the Turing machine $\mathfrak{T}_{\mathcal{W}_{p_1,p_2},\epsilon}$ we can immediately find a Turing machine as in the proof of Theorem 2 that yields a non-trivial classification problem according to Lemma 1. This proves the desired result for the case $0 < p_1 < p_2 \leq \frac{1}{2}$.

For $0 < p_1 < \frac{1}{2}$ and $p_2 > \frac{1}{2}$ we only have to look at the case $0 < p_1 < \frac{1}{2}$ and $p_2^* = \frac{1}{2}$ so that this case is also proved.

If $p \geq \frac{1}{2}$, then with $p_1' = 1 - p_1$ and $p_2' = 1 - p_2$ we can reduce this case to the first case so that this is also proved. ■

As these results provide negative answers to the initial Question 1, it is reasonable to study what happens if the requirements are weakened. Therefore, we ask the following:

Question 2: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, be fixed. Is there an $\alpha \in (0,1)$ such that there exists an algorithm (or Turing machine) $\mathfrak{T}_{\mathcal{W},\alpha,\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with

$$\liminf_{n\to\infty}\frac{1}{n}\log M_n \geq \alpha C(W)? \tag{17}$$

Also for this question, we obtain a negative answer which follows from the following result.

Theorem 4: Let $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, and $\alpha \in (0,1)$ be arbitrary and fixed. For $|\mathcal{X}| = |\mathcal{Y}| = 2$ and $\mathcal{W} = \bigcup_{p\in\mathbb{R}_c\cap[0,1]}\{\text{BSC}(p)\}$ there is no Turing machine $\mathfrak{T}_{\mathcal{W},\alpha,\epsilon}$ that

takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with $\liminf_{n\to\infty} \frac{1}{n} \log M_n \geq \alpha C(W)$.

Proof: The result is proved by contradiction. For this purpose, we assume that there exists an $\hat{\alpha} \in (0,1)$, $\hat{\alpha} \in \mathbb{R}_c$, such that there exists a Turing machine $\mathfrak{T}_{\mathcal{W},\hat{\alpha},\epsilon}$ with the desired properties. We consider the function $f_n(W) = \frac{1}{n} \log M_n(W)$, $W \in \mathcal{W}$, from the proof of Theorem 2. Then it holds that $\lim_{n\to\infty} f_n(W_{\frac{1}{2}}) = 0$ and $\liminf_{n\to\infty} f_n(W) \geq \alpha$ and there exists an $n_0 = n_0(\alpha)$ such that for all $n \geq n_0(\alpha)$ we have

$$f_n(W_{\frac{1}{2}}) < \frac{\alpha}{4} \quad \text{and} \quad f_n(W_0) > \frac{3}{4}\alpha.$$

Now, with the Turing machine $\mathfrak{T}_{\mathcal{W},\hat{\alpha},\epsilon}$ we can proceed exactly as in the proof of Theorem 2 to find a Turing machine that yields a non-trivial classification problem according to Lemma 1, since $\bigcup_{p\in[0,1]}\{\text{BSC}(p)\}$ is a computable family of channels. This is a contradiction which proves the desired result. ∎

V. Non-Constructability for General Channels

Let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary finite input and output alphabets. Further let $\mathcal{W} \subset \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be an arbitrary set of channels of interest for which we would like to find capacity-achieving encoder and decoder algorithmically depending on the actual channel $W \in \mathcal{W}$ and the desired blocklength n. We obtain the following result which generalizes our previous findings for BSCs in Theorem 2.

Theorem 5: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, be fixed. Further, let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary finite alphabets and let $\mathcal{W} \subset \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be an arbitrary set of channels. Suppose there is a computable family of channels $\{W_\lambda\}_{\lambda\in[0,1]}$ with $W_\lambda \in \mathcal{W}$, $\lambda \in [0,1] \cap \mathbb{R}_c$, such that

$$\min_{\lambda\in[0,1]} C(W_\lambda) < \max_{\lambda\in[0,1]} C(W_\lambda). \tag{18}$$

Then, for the set of channels $\mathcal{W}$ there is no Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with $\lim_{n\to\infty} \frac{1}{n} \log M_n = C(W)$.

Proof: We prove the result by contradiction. Therefore, we assume that for the set of channels $\mathcal{W} \subset \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ there exists a Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}$ with the desired properties as stated above. This particularly means that the Turing machine is able to solve the task for the set $\{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c}$.

We consider for $\lambda \in [0,1] \cap \mathbb{R}_c$, W_λ, and $n \in \mathbb{N}$ the corresponding output of the Turing machine $\mathfrak{T}_{\mathcal{W},\epsilon}(W_\lambda, n) = (M_n(\lambda), E_n(\lambda), D_n(\lambda), \epsilon)$. Since (18) holds, the function $g_n(\lambda) = \frac{1}{n} \log M_n(\lambda)$ converges for $n \to \infty$ to $C(W_\lambda)$ for all $\epsilon \in \mathbb{Q} \cap (0,1)$. Note that $q(\lambda) := C(W_\lambda)$, $\lambda \in [0,1]$, is a computable continuous function. Let $\underline{\lambda} \in [0,1]$ and $\overline{\lambda} \in [0,1]$ be such that $C(W_{\underline{\lambda}}) = \min_{\lambda\in[0,1]} C(W_\lambda)$ and $C(W_{\overline{\lambda}}) = \max_{\lambda\in[0,1]} C(W_\lambda)$. Furthermore, we have $\inf_{\lambda\in[0,1]\cap\mathbb{R}_c} C(W_\lambda) = C(W_{\underline{\lambda}})$ and $\sup_{\lambda\in[0,1]\cap\mathbb{R}_c} C(W_\lambda) = C(W_{\overline{\lambda}})$ since $\lambda \in [0,1] \cap \mathbb{R}_c$ is dense in $[0,1]$ and all rational numbers are computable. Then, there exists $\lambda_1, \lambda_2 \in [0,1] \cap \mathbb{R}_c$ with $C(W_{\lambda_1}) < C(W_{\lambda_2})$. In addition, we can find an $n_0 = n_0(\epsilon)$ such that for all $n \geq n_0$ we have

$$g_n(\lambda_1) < C(W_{\lambda_1}) + \frac{C(W_{\lambda_2}) - C(W_{\lambda_1})}{4}$$

and also

$$g_n(\lambda_2) > C(W_{\lambda_2}) - \frac{C(W_{\lambda_2}) - C(W_{\lambda_1})}{4}.$$

Next, we can adapt the proof of Theorem 2 to find a Turing machine $\mathfrak{T}_*$: $[0,1] \cap \mathbb{R}_c \to \mathcal{S}$ with $\mathcal{S}$ being a finite set of natural numbers, that yields a non-trivial mapping. The proof of Lemma 1 further yields that such a non-trivial mapping, which is Turing computable, cannot exist. The establishes the desired contradiction completing the proof. ∎

In the following, we want to study a similar question as in Theorem 4, i.e., we are interested in understanding whether or not it is possible to algorithmically construct codes such that these approach asymptotically at least $\alpha C(W)$ for some $\alpha \in (0,1)$, $\alpha \in \mathbb{R}_c$. We obtain the following negative result.

Theorem 6: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, be fixed. Further, let $\mathcal{X}$ and $\mathcal{Y}$ be arbitrary finite alphabets and let $\mathcal{W} \subset \mathcal{CH}_c(\mathcal{X};\mathcal{Y})$ be an arbitrary set of channels. Suppose there is a computable family of channels $\{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c} \subset \mathcal{W}$ and that

$$1 > \underline{\alpha} = \inf_{\{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c}\subset\mathcal{W}} \frac{\min_{\lambda\in[0,1]} C(W_\lambda)}{\max_{\lambda\in[0,1]} C(W_\lambda)}. \tag{19}$$

Then, for all $\alpha \in (\underline{\alpha}, 1) \cap \mathbb{R}_c$ there is no Turing machine $\mathfrak{T}_{\mathcal{W},\alpha,\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with

$$\liminf_{n\to\infty} \frac{1}{n} \log M_n \geq \alpha C(W). \tag{20}$$

Proof: We prove the result by contradiction. Therefore, we assume that for the set $\mathcal{W}$ there exists an $\hat{\alpha} \in (\underline{\alpha}, 1) \cap \mathbb{R}_c$ such that a corresponding Turing machine $\mathfrak{T}_{\mathcal{W},\hat{\alpha},\epsilon}$ exists for which (20) is true. Then, there exists a computable family $\{\hat{W}_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c} \subset \mathcal{W}$ such that

$$\hat{\alpha} > \frac{\min_{\lambda\in[0,1]} C(\hat{W}_\lambda)}{\max_{\lambda\in[0,1]} C(\hat{W}_\lambda)}.$$

By assumption, for every channel $\hat{W}_\lambda$, $\lambda \in [0,1] \cap \mathbb{R}_c$ we have

$$\mathfrak{T}_{\mathcal{W},\hat{\alpha},\epsilon}(\hat{W}_\lambda, n) = (M_n(\lambda), E_n(\lambda), D_n(\lambda), \epsilon)$$

with

$$\liminf_{n\to\infty} \frac{1}{n} \log M_n(\lambda) \geq \alpha C(\hat{W}_\lambda).$$

Now, let $\delta > 0$ be arbitrary. There exists a $\underline{\lambda}_\delta \in [0,1] \cap \mathbb{R}_c$ with $C(\hat{W}_{\underline{\lambda}_\delta}) < \min_{\lambda\in[0,1]} C(\hat{W}_\lambda) + \delta$ and there exists further a $\overline{\lambda}_\delta \in [0,1] \cap \mathbb{R}_c$ with $C(\hat{W}_{\overline{\lambda}_\delta}) > \max_{\lambda\in[0,1]} C(\hat{W}_\lambda) - \delta$. But this implies that

$$\liminf_{n\to\infty} \frac{1}{n} \log M_n(\overline{\lambda}_\delta) \geq \hat{\alpha} C(\hat{W}_{\overline{\lambda}_\delta}) > \hat{\alpha}\big(\max_{\lambda\in[0,1]} C(\hat{W}_\lambda) - \delta\big).$$

Due to the strong converse, we further have

$$\limsup_{n\to\infty} \frac{1}{n} \log M_n(\underline{\lambda}_\delta) \leq C(\hat{W}_{\underline{\lambda}_\delta}) < \min_{\lambda\in[0,1]} C(\hat{W}_\lambda) + \delta.$$

As a consequence, we can find an n_0 such that

$$\frac{1}{n_0} \log M_{n_0}(\overline{\lambda}_\delta) > \hat{\alpha}\big(\max_{\lambda\in[0,1]} C(\hat{W}_\lambda) - \delta\big)$$

and

$$\frac{1}{n_0}\log M_{n_0}(\underline{\lambda}_\delta) < \min_{\lambda\in[0,1]} C(\hat{W}_\lambda) + \delta$$

hold.

We also have

$$\hat{\alpha}\max_{\lambda\in[0,1]} C(\hat{W}_\lambda) > \min_{\lambda\in[0,1]} C(\hat{W}_\lambda)$$

so that we an find a $\delta_0 > 0$ such that for all $\delta \in (0,\delta_0)$ we have

$$\hat{\alpha}\big(\max_{\lambda\in[0,1]} C(\hat{W}_\lambda) - \delta\big) > \min_{\lambda\in[0,1]} C(\hat{W}_\lambda) + \delta.$$

This means that for $\delta \in (0,\delta_0)$ arbitrary, we have

$$\frac{1}{n_0}\log M_{n_0}(\overline{\lambda}_\delta) > \frac{1}{n_0}\log M_{n_0}(\underline{\lambda}_\delta).$$

Now we are in the position to use the proof of Theorem 4 to prove the desired result. This completes the proof. ■

Remark 9: In Theorem 6 we required that the set of channels $\mathcal{W}$ contains a computable family of channels. Note that the ratio $\frac{\min_{\lambda\in[0,1]} C(W_\lambda)}{\max_{\lambda\in[0,1]} C(W_\lambda)}$ in (19) is always smaller than or equal to 1. It is equal to 1 if the capacity $C(W_\lambda)$ is constant for all computable families of channels $\{W_\lambda\}_{\lambda\in[0,1]}$. We require it to be strictly smaller than 1 which is only true if and only if the capacity $C(W_\lambda)$ is not constant for all computable families of channels $\{W_\lambda\}_{\lambda\in[0,1]}$.

Next, we want to show that the previous result in Theorem 6 is actually sharp, which is done in the following theorem.

Theorem 7: Let the tolerated probability of decoding error $\epsilon \in \mathbb{Q}$, $\epsilon \in (0,1)$, be fixed. For $|\mathcal{X}| = |\mathcal{Y}| = 2$ there exists a set of channels $\mathcal{W} = \{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c}$ with $\{W_\lambda\}_{\lambda\in[0,1]}$ a computable family of channels such that for

$$\alpha = \frac{\min_{\lambda\in[0,1]} C(W_\lambda)}{\max_{\lambda\in[0,1]} C(W_\lambda)}$$

there is a Turing machine $\mathfrak{T}_{\mathcal{W},\alpha,\epsilon}$ that takes $W \in \mathcal{W}$ and $n \in \mathbb{N}$ as inputs and computes an $(M_n, E_n, D_n, \epsilon)$-code with

$$\liminf_{n\to\infty} \frac{1}{n}\log M_n \geq \alpha C(W).$$

Proof: Let $p \in (0,1)\cap\mathbb{Q}$ be a rational number and we consider the sets

$$\mathcal{W}_p = \bigcup_{0\leq\hat{p}\leq p, \hat{p}\in\mathbb{R}_c} \mathrm{BSC}(\hat{p})$$

and

$$\{W_\lambda\}_{\lambda\in[0,1]} = \bigcup_{\lambda\in[0,1]} \mathrm{BSC}(\lambda p).$$

Then we have $\{W_\lambda\}_{\lambda\in[0,1]\cap\mathbb{R}_c} = \mathcal{W}_p$. It holds that

$$\max_{\lambda\in[0,1]} C(W_\lambda) = 1$$

and

$$\min_{\lambda\in[0,1]} C(W_\lambda) = C(W_1) = 1 - h_2(p).$$

For p we can find a Turing machine $\mathfrak{T}_p$ that computes for every $n \in \mathbb{N}$ an $(M_n(p), E_n(p), D_n(p), \epsilon)$-code with $\lim_{n\to\infty}\frac{1}{n}\log M_n(p) = C(W_1) = 1 - h_2(p)$. We observe that the code designed for p also yields a code with average probability of error not larger than ϵ for all channels $W \in \mathcal{W}_p$ since the channel W_1 is always degraded with respect to W. Therefore, we can set $\mathfrak{T}_{\mathcal{W}_p,\alpha,\epsilon}(W,n) = \mathfrak{T}_p(n)$ for $W \in \mathcal{W}_p$. With this, we have $M_n(W) = M_n(p)$ for all $W \in \mathcal{W}_p$ and $n \in \mathbb{N}$. As a consequence, for $W \in \mathcal{W}_p$ arbitrary, we always have

$$\lim_{n\to\infty}\frac{1}{n}\log M_n(W) = C(W_1) = \alpha\max_{\lambda\in[0,1]} C(W_\lambda) \geq \alpha C(W)$$

which proves the desired result. ■

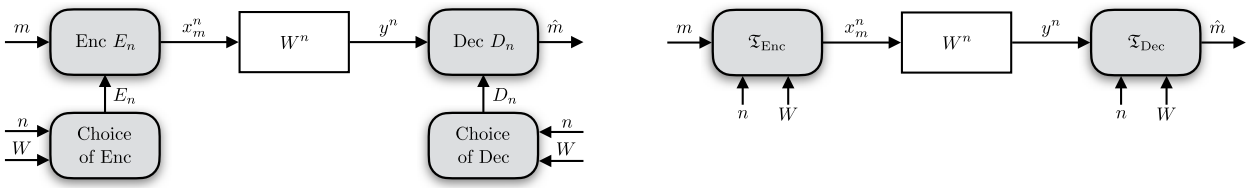
VI. Discussion

To meet demanding requirements on spectral efficiency, modern communication systems aim at adapting the transmission rate according to the current channel quality. For example, resource allocation in orthogonal frequency division multiplexing (OFDM) relies on so-called "bit loading" where each carrier is optimized based on its individual channel quality. In communication scenarios with multiple users, the resource allocation is additionally optimized across the users according to their respective channel qualities (this may even result in the extreme case for some users to be scheduled to not transmit at all). As outlined in Section III-A, this optimization also includes the used coding scheme since this needs to enable the overall rate requirements.

For the simple scenario of a point-to-point DMC with one transmitter and one receiver as introduced in Section III, the task of resource allocation reduces to the design of a coding scheme that is adapted according to the quality of the underlying channel. This means that depending on the desired blocklength, the optimal (or near-optimal) encoding at the transmitter and therewith also the corresponding optimal (or near-optimal) rate as well as the decoding at the receiver is chosen. Today, this is usually done via lookup tables in which a set of previously designed codebooks are stored for a class of channel quality values. This is shown in Fig. 2(a).

In the end, this means that the underlying channel and its quality control the encoder and decoder. In the framework developed above, this can now be mapped to and realized by a Turing machine. The question of interest is then whether or not it is possible to algorithmically construct optimal channel-dependent coding schemes for desired blocklengths as shown in Fig. 2(b).

For a fixed channel, it has been demonstrated in the literature that optimal codes can be algorithmically constructed. Thus, there exists an optimal code for each channel in the whole class of channels. However, in this paper we have shown that the optimal code does not depend recursively on the channel and, as a consequence, there cannot be a universal algorithm that is able to construct capacity-achieving codes for a whole class of channels (which can further be a quite restricted set as demonstrated in Theorem 3). This necessitates a particular code design for each specific channel realization and provides a negative answer to the question of whether or not such a channel-dependent code design as shown in Fig. 2(b) can be algorithmically realized.



(a) Code adaptation via lookup tables

(b) Algorithmic code adaptation based on Turing machines

Fig. 2. (a) Code adaptation in today's communication systems based on lookup tables. Based on the available channel knowledge W and the desired blocklength n, the encoder E_n and decoder D_n are usually chosen from a set of previously designed codebooks. (b) Desired code adaption realized by Turing machines $\mathfrak{T}_{\text{Enc}}$ and $\mathfrak{T}_{\text{Dec}}$. Based on the channel knowledge W and the desired blocklength n, the Turing machines algorithmically determine the optimal coding scheme.

For the channel-dependent code construction we thus observe the same behavior regarding the mathematical existence of certain objects and their algorithmic constructability as for the Specker Sequence discussed in detail in Section III-B. Sequences of capacity-achieving codes exist for all channels, but these cannot algorithmically be computed as a function of the computable channel and the blocklength. The Specker Sequence demonstrates such a behavior for a computable monotonically increasing but bounded sequence of rational numbers that converges to a finite limit, which is not computable in general.

The existence of a channel-aware code construction is a practically relevant question as it would provide a natural extension to today's channel adaptive transmission schemes. However, from the proof of Theorem 2 it follows that such a hypothetical Turing machine solving this problem would also allow one to solve the famous halting problem which is known to be not solvable by a Turing machine.

It is clear that this has further consequences on resource allocation and cross-layer optimization problems. This includes, for example, the stability region of networks, i.e., the region of maximum arrival rates of data packets for different transmitters. Then, a resource allocation is desired that stabilizes the queuing system depending on the channel and interference constraints. The physical transmission must be adapted accordingly already for short time intervals [26]–[30]. As already mentioned, these optimization problems assume that the corresponding coding schemes can be constructed according to the underlying conditions. The results established in this paper however show that such a desired solution is in general not possible, since it is not possible to solve this problem algorithmically based on the channel parameters and the desired parameter requirements.

We observe an interesting effect. Our question on the code construction dependent on the channel as input is a natural and practically relevant question. As discussed above, a hypothetical Turing machine that solves this question would provide a natural extension to today's channel adaptive transmission scheme. From the proof of Theorem 2 follows then if such a Turing machine would exist, it would further solve the Halting problem. This is of course not possible.

This reveals that certain inherent structures in information theory allow the formulation of very natural information theoretic questions that are algorithmically not solvable. In particular the structure of computable channels immediately yields such a problem formulation that is not algorithmically solvable. This is interesting as the logical statement here shows no self-relation as, for example, in the formulation of the halting problem where a Turing machine takes another Turing machine as input and which then has to decide whether or not the Turing machine that has been given as input stops.

ACKNOWLEDGMENT

The author Holger Boche would like to thank Eike Kiltz and Moritz Wiese for valuable discussions about explicit code constructions. He further would like to thank all his project partners and in particular Ullrich Mönich for discussions on the hardware implementation of coding schemes within the projects NewCom, 6G life, NeroCM, and 6G Future Lab Bavaria. The author Rafael F. Schaefer would like to thank Onur Günlü for valuable discussions and insightful comments on capacity-achieving coding schemes.

REFERENCES

[1] H. Boche, R. F. Schaefer, and H. V. Poor, "Turing meets Shannon: Algorithmic constructability of capacity-achieving codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, Jun. 2021, pp. 1–4.
[2] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.
[4] I. Csiszár and J. Körner, *Information Theory: Coding Theorems for Discrete Memoryless Systems*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[5] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[6] D. J. Costello, Jr. and G. D. Forney, "Channel coding: The road to channel capacity," *Proc. IEEE*, vol. 95, no. 6, pp. 1150–1177, Jun. 2007.
[7] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[8] T. J. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 28–34, Mar. 2018.
[9] V. Bioglio, C. Condo, and I. Land, "Design of polar codes in 5G new radio," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 29–40, 1st Quart. 2021.
[10] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct. 2013.
[11] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Soc.*, vol. 2, no. 1, pp. 230–265, 1937.
[12] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem. A correction," *Proc. London Math. Soc.*, vol. 2, no. 43, pp. 544–546, 1937.
[13] K. Weihrauch, *Computable Analysis—An Introduction*. Berlin, Germany: Springer-Verlag, 2000.

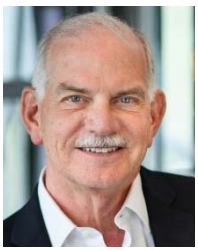
[14] J. Avigad and V. Brattka, "Computability and analysis: The legacy of Alan Turing," in *Turing's Legacy: Developments from Turing's Ideas in Logic*, R. Downey, Ed. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[15] K. Gödel, "Die Vollständigkeit der Axiome des logischen Funktionenkalküls," *Monatshefte für Math.*, vol. 37, no. 1, pp. 349–360, 1930.
[16] K. Gödel, "On undecidable propositions of formal mathematical systems," in *Lectures at the Institute for Advanced Study*, C. Kleene and J. Barkely Rosser, Eds. Princeton, NJ, USA: Princeton, 1934.
[17] S. C. Kleene, *Introduction to Metamathematics*. New York, NY, USA: Van Nostrand, 1952.
[18] M. Minsky, "Recursive unsolvability of Post's problem of 'Tag' and other topics in theory of Turing machines," *Ann. Math.*, vol. 74, no. 3, pp. 437–455, 1961.
[19] H. Boche, R. F. Schaefer, and H. V. Poor, "Secure communication and identification systems—Effective performance evaluation on Turing machines," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1013–1025, 2020.
[20] H. Boche, R. F. Schaefer, and H. Vincent Poor, "Shannon meets Turing: Non-computability of the finite state channel capacity," *Commun. Inf. Syst.*, vol. 20, no. 2, pp. 81–116, 2020.
[21] H. Boche, R. F. Schaefer, and H. V. Poor, "Coding for non-IID sources and channels: Entropic approximations and a question of Ahlswede," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 2019, pp. 1–5.
[22] M. B. Pour-El and J. I. Richards, *Computability in Analysis and Physics*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[23] H. Boche, R. F. Schaefer, S. Baur, and H. V. Poor, "On the algorithmic computability of the secret key and authentication capacity under channel, storage, and privacy leakage constraints," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4636–4648, Sep. 2019.
[24] R. I. Soare, *Recursively Enumerable Sets Degrees*. Berlin, Germany: Springer-Verlag, 1987.
[25] E. Specker, "Nicht konstruktiv beweisbare Sätze der analysis," *J. Symb. Log.*, vol. 14, no. 3, pp. 145–158, Sep. 1949.
[26] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-layer design for wireless networks," *IEEE Commun. Mag.*, vol. 41, no. 10, pp. 74–80, Oct. 2003.
[27] H. Boche and M. Wiczanowski, "Stability-optimal transmission policy for the multiple antenna multiple access channel in the geometric view," *Signal Process.*, vol. 86, no. 8, pp. 1815–1833, Aug. 2006.
[28] E. Yeh and R. Berry, "Throughput optimal control of cooperative relay networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Adelaide, SA, Australia, Sep. 2005, pp. 1206–1210.
[29] T. J. Oechtering and H. Boche, "Stability region of an optimized bidirectional regenerative half-duplex relaying protocol," *IEEE Trans. Commun.*, vol. 56, no. 9, pp. 1519–1529, Sep. 2008.
[30] M. J. Neely, E. Modiano, and C. E. Rohrs, "Power allocation and routing in multibeam satellites with time-varying channels," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 138–152, Feb. 2003.
[31] S. Stanczak, M. Wiczanowski, and H. Boche, *Fundamentals of Resource Allocation in Wireless Networks*, 2nd ed. Berlin, Germany: Springer-Verlag, 2008.
[32] M. Fekete, "über die Verteilung von Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten," *Mathematische Zeitschrift*, vol. 17, no. 1, pp. 228–249, 1923.
[33] H. Boche, R. F. Schaefer, and H. V. Poor, "Communication under channel uncertainty: An algorithmic perspective and effective construction," *IEEE Trans. Signal Process.*, vol. 68, pp. 6224–6239, 2020.
[34] H. Boche, R. F. Schaefer, and H. V. Poor, "Denial-of-service attacks on communication systems: Detectability and jammer knowledge," *IEEE Trans. Signal Process.*, vol. 68, pp. 3754–3768, 2020.
[35] M. B. Pour-El, "A comparison of five 'computable' operators," *Math. Log. Quart.*, vol. 6, nos. 15–22, pp. 325–340, 1960.

Holger Boche (Fellow, IEEE) received the Dipl.-Ing. degree in electrical engineering, the Graduate degree in mathematics, and the Dr.-Ing. degree in electrical engineering from the Technische Universität Dresden, Dresden, Germany, in 1990, 1992, and 1994, respectively, the master's degree from the Friedrich-Schiller Universität Jena, Jena, Germany, in 1997, and the Dr. rer. nat. degree in pure mathematics from the Technische Universität Berlin, Berlin, Germany, in 1998.

In 1997, he joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institute (HHI), Berlin. From 2002 to 2010, he was a Full Professor of mobile communication networks with the Institute for Communications Systems, Technische Universität Berlin. In 2003, he became the Director of the Fraunhofer German-Sino Laboratory for Mobile Communications, Berlin. In 2004, he became the Director of the Fraunhofer Institute for Telecommunications, HHI. He was a Visiting Professor with the ETH Zürich, Zürich, Switzerland, from 2004 to 2006 (Winter), and with KTH Stockholm, Stockholm, Sweden, in 2005 (Summer). He has been a member and an Honorary Fellow of the TUM Institute for Advanced Study, Munich, Germany, since 2014. Since 2018, he has been a Founding Director of the Center for Quantum Engineering, Technische Universität München. Since 2021, he has been leading jointly with Frank Fitzek the BMBF Research Hub 6G-life. He is currently a Full Professor at the Institute of Theoretical Information Technology, Technische Universität München, Munich, Germany, where he joined in October 2010. Among his publications is the recent book *Information Theoretic Security and Privacy of Information Systems* (Cambridge University Press, 2017).

Prof. Boche is a member of the IEEE Signal Processing Society SPCOM and SPTM Technical Committees. He was an Elected Member of the German Academy of Sciences (Leopoldina) in 2008 and to the Berlin Brandenburg Academy of Sciences and Humanities in 2009. He was a recipient of the Research Award "Technische Kommunikation" from the Alcatel SEL Foundation in October 2003, the "Innovation Award" from the Vodafone Foundation in June 2006, and the Gottfried Wilhelm Leibniz Prize from the Deutsche Forschungsgemeinschaft (German Research Foundation) in 2008. He was a co-recipient of the 2006 IEEE Signal Processing Society Best Paper Award and a recipient of the 2007 IEEE Signal Processing Society Best Paper Award. He was the General Chair of the Symposium on Information Theoretic Approaches to Security and Privacy at IEEE GlobalSIP 2016.

Rafael F. Schaefer (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering and computer science from the Technische Universität Berlin, Germany, in 2007, and the Dr.-Ing. degree in electrical engineering from the Technische Universität München, Germany, in 2012. From 2013 to 2015, he was a Post-Doctoral Research Fellow with Princeton University. From 2015 to 2020, he was an Assistant Professor with the Technische Universität Berlin. Since 2021, he has been a Full Professor with the Universität Siegen. Among his publications is the recent book *Information Theoretic Security and Privacy of Information Systems* (Cambridge University Press, 2017). He is a member of the IEEE Information Forensics and Security Technical Committee. He was a recipient of the VDE Johann-Philipp-Reis Award in 2013. He received the Best Paper Award of the German Information Technology Society (ITG-Preis) in 2016. He was one of the Exemplary Reviewers of the IEEE COMMUNICATION LETTERS in 2013. He is currently an Associate Editor of the IEEE TRANSACTIONS OF INFORMATION FORENSICS AND SECURITY and the IEEE TRANSACTIONS OF COMMUNICATIONS.



H. Vincent Poor (Life Fellow, IEEE) received the Ph.D. degree in EECS from Princeton University in 1977. From 1977 to 1990, he was on the faculty of the University of Illinois at Urbana–Champaign. Since 1990, he has been on the faculty at Princeton University, where he is currently the Michael Henry Strater University Professor. From 2006 to 2016, he served as the Dean of Princeton's School of Engineering and Applied Science. He has also held visiting appointments at several other universities, including most recently at Berkeley and Cambridge. His research interests are in the areas of information theory, machine learning and network science, and their applications in wireless networks, energy systems, and related fields. Among his publications in these areas is the forthcoming book *Machine Learning and Wireless Communications* (Cambridge University Press). Dr. Poor is a member of the National Academy of Engineering and the National Academy of Sciences. He is a foreign member of the Chinese Academy of Sciences, the Royal Society, and other national and international academies. He received the IEEE Alexander Graham Bell Medal in 2017.