Turing Machines with One-sided Advice and the Acceptance of the co-RE Languages

Jan van Leeuwen

Jiří Wiedermann

Technical Report UU-CS-2014-003
February 2014

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
Princetonplein 5
3584 CC Utrecht
The Netherlands

Turing Machines with One-sided Advice and the Acceptance of the co-RE Languages*

Jan van Leeuwen[1] Jiří Wiedermann[2]

[1] Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, the Netherlands
J.vanLeeuwen1@uu.nl

[2] Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
jiri.wiedermann@cs.cas.cz

Abstract. We resolve an old problem, namely to design a 'natural' machine model for accepting the complements of recursively enumerable languages. The model we present is based on Turing machines with 'one-sided' advice, which are Turing machines with advice with a restricted scenario for the use of their advice during computations. We show that Turing machines with one-sided advice accept precisely the co-RE languages, even though the advices can be arbitrary. We prove that Turing machines with one-sided advice are not less powerful than their unrestricted version: for all languages $L \in$ co-RE, L is accepted by an ordinary Turing machine with advice f if and only L is accepted by a Turing machine with one-sided advice f. We show that co-RE filters into an infinite proper hierarchy, based on the complexity ('length') of the advice $f(n)$ a one-sided Turing machine needs for accepting a language. The model and the results dualise to the ordinary RE-languages.

1 Introduction

The recursively enumerable languages (RE) are the widest class of languages that can be accepted by classical Turing machines. A core result of computability theory is that there exist RE-languages (namely all non-recursive ones) whose complement cannot be accepted, and thus not effectively enumerated, by any Turing machine. Is there a natural mechanism for accepting the complements of the r.e. languages anyhow?

The class co-RE of complements of the RE-languages is fundamental in the understanding of the non-computable domain. Co-RE is a so-called full abstract family of languages (cf. [12]) and with the class RE, it is the pillar of the Arithmetical Hierarchy at its first non-recursive level (cf. [8]). This implies e.g. that co-RE is accepted by Π_1-type *Alternating Turing Machines* (cf. [3, 7]), although these machines merely model that words in a co-RE language L should have no

* Version dated February 10, 2014. This research was partially supported by RVO 67985807 and GA ČR grant No. P202/10/1333.

accepting runs in a Turing machine for $\bar{L}$ by a search process. Other iterative mechanisms for co-RE have been proposed in [4, 9].

Clearly any machine model for co-RE will have to be non-effective from a classical point of view. Nevertheless, ideally, a machine model for co-RE should still be *dual* to a machine model for RE (as in the case of Alternating Turing Machines), be sequential in some way and retain a certain closeness to common machine concepts. We resolve this by *injecting* the necessary degree of non-effective information in a classical Turing machine. The challenge will be to do this in such a way that we stay within co-RE and, in the dual case, in RE.

Our starting point will be a regular *non-deterministic* Turing machine with *advice* as introduced by Karp and Lipton [5, 6]. A Turing machine with advice (TM/A) operates exactly like an oracle Turing machine, except that now the oracle is a function f that produces an advice string depending only on the length of the input. (Hence, if the advice is called during a run of the machine, the advice doesn't have to be called again.) We will introduce the notion of *one-sided advice* which is to be called only in runs that are potentially accepting, but never in a run that is concretely rejecting. (Thinking of our goal to obtain a machine for co-RE, it should indeed need no advice in runs that reject word w from co-RE language $\bar{L}$ as we can rely on the Turing machine for L to witness it by accepting runs.) No further special gadgetry will be used.

In Section 2 we define Turing machines with one-sided advice (TM/1A) in more detail and show the duality of the model. In Section 3 we prove that TM/1A's *precisely* accept co-RE. similarly, dual TM/1A's characterise RE. We also prove some special properties that TM/1A's must exhibit when they accept languages in co-RE that are not recursive (REC). In Section 4 we analyse the power of advice in TM/1A's. We show that for all languages $L \in$ co-RE, if L can be accepted by an ordinary Turing machine with advice function f, then L can also be accepted by a TM/1A with advice function f. On the other hand, we show that every $L \in$ co-RE can be accepted by a TM/1A with *linearly bounded* advice, i.e. with an advice function f for which $|f(n)| = O(n)$. We prove that there is an infinite, proper hierarchy 'below linear' inside the class co-RE (or dually, RE), based on the complexity (i.e. the size) of the advice of the accepting TM/1A.

We emphasise that TM/1A's are not ordinary non-deterministic Turing machines even though they resemble them quite closely. Indeed, whether a Turing machine with advice is a TM/1A is undecidable. This phenomenon is inherent to co-RE, as it extends beyond the computable domain. It makes TM/1A's to an interesting class of machines at the *interface*. Note that in general, Turing machines with advice can accept all possible languages. The fact that one-sidedness restricts their power to co-RE gives a new and rich perspective on the working of advice.

2 Turing machines with One-sided Advice

We first define the basic model of Turing machines with advice (TM/A) and with one-sided advice (TM/1A), respectively. We will show that TM/1A's and their

duals satisfy the desired property of characterising co-RE and RE, respectively, in Section 3.

2.1 Advice

The core mechanism we use is that of a non-deterministic Turing machine with advice. TMs with advice were introduced by Karp and Lipton [5, 6] in their seminal study of non-uniform complexity theory. Without loss of generality we will use the same alphabet Σ for advices and inputs throughout.

Definition 1. *An* advice function *is a function $f : \mathbb{N} \to \Sigma^*$. An advice is called $g(n)$-bounded for some function $g : \mathbb{N} \to \mathbb{N}$ if for all n, the length of $f(n)$ is bounded by $g(n)$.*

Technically, a TM M with advice operates on a given input $w \in \Sigma^*$ in much the same way as a standard non-deterministic Turing machine does. However, in any of its runs M can also call its advice, by entering into a special *query state*. After doing so the value of $f(n)$, with $n = |w|$, will 'magically' appear on a special read-only advice tape. We may assume w.l.o.g. that the advice value is available from then on, for the remainder of the particular run.

Notation 1 *For any advice function $f : \mathbb{N} \to \Sigma^*$ we denote its* advice graph *by $R_f = \{< n, f(n) > \mid n \in \mathbb{N}\}$.*

Using advice, M can easily gain super-Turing computing power, because there is no requirement that the advice must be 'computable'. In fact, advice is as powerful as the use of an arbitrary oracle, because one can combine all oracle-values ever queried in computations on inputs of size n into one advice $f(n)$. This is why one normally imposes size-bounds on advice, to keep track of the degree of non-computable information that is injected in computations.

The definition of acceptance easily carries over from classical nondeterministic machines to Turing machines with advice. Thus, for TM/A's, inputs w are accepted if and only if some run leads the machine to halt in an accepting state in finite time. Inputs w are rejected if they lead the machine to halt in a rejecting or undefined state or not to halt at all. (NB In TM/1A's a much stronger convention will be imposed in the latter case.)

Observation 1 *For all languages L, $L \in RE$ if and only L is accepted by a Turing machine with an advice function f with $R_f \in RE$.*

The observation follows by simply noting that $R_f \in$ RE implies that $R_f \in$ REC and thus that f is recursive in this case.

Notation 2 *For bounding functions $g : \mathbb{N} \to \mathbb{N}$, we denote the class of languages accepted by TM/A's with $g(n)$-bounded advice by $TM/A(g)$.*

2.2 One-sided advice

We now describe the assumptions that make a Turing machine M to a machine with *one-sided advice* step by step below. Each assumption will add yet another potentially non-effective ingredient to the model.

First of all, we (only) consider non-deterministic Turing machines M which have their states partitioned into three groups: **accept** states, **reject** states, and **undefined** states. Either of the groups may be empty. If a run of M halts in an accept (reject) state, then the run is called accepting (rejecting). Runs of M that halt in an undefined state or that do not halt at all, are called infinite. (W.l.o.g. we may assume that computations never halt in an undefined state, as we can send the computation into an infinite loop if they do.)

It is perfectly fine for a non-deterministic machine to have both accepting and rejecting runs on the same input. For an acceptor, only accepting runs count and rejecting runs are just viewed equivalent to infinite ones. For TM/1A's we want the power of advice to give us more. We require that every computation is potentially decisive, i.e.

> (R_0): for every input $w \in \Sigma^*$, the machine has at least one run on w that is either accepting or rejecting.

> (R_1): for no input $w \in \Sigma^*$, the machine has both an accepting and a rejecting run on w.

Infinite runs can (and in general will) still occur, but the constraints imply that they can occur only in combination with either accepting or rejecting runs.

The final constraint captures the 'one-sided' use of advice. The constraint is again to be seen as a purely mathematical requirement.

> (R_2): for every input $w \in \Sigma^*$, the machine is only allowed to call its advice during non-rejecting runs.

This constraint is equivalent to saying that a TM/1A is allowed to call its advice *only* during accepting or infinite runs. Note that we do not restrict the advice itself in any way. One may also formulate the following 'dual' version of (R_2).

> $(\bar{R}_2)$: for every input $w \in \Sigma^*$, the machine is only allowed to call its advice during non-accepting runs.

With the basic concepts in place, we can define (dual) Turing machines with one-sided advice and the languages they accept or reject.

Definition 2. *A TM/1A is a non-deterministic Turing machine with advice that satisfies* (R_0), (R_1) *and* (R_2). *A dual TM/1A (or d-TM/1A) is a non-deterministic Turing machine with advice that satisfies* (R_0), (R_1) *and* $(\bar{R}_2)$.

Let M be a (dual) TM/1A.

Definition 3. *The language accepted by M is $L_A(M) = \{w \in \Sigma^* \mid M$ admits an accepting run on $w\}$. The language rejected by M is $L_R(M) = \{w \in \Sigma^* \mid M$ admits a rejecting run on $w\}$.*

By requirements (R_0) and (R_1) for M it follows that inputs are either accepted ('have an accepting run') or rejected ('have a rejecting run'), and thus the following is evident:

Observation 2 *For any (dual) TM/1A M, the languages $L_A(M)$ and $L_R(M)$ are each other's complement.*

Note that TM/1A's may, and in general will, have infinite runs with or without advice as well. These runs hold no clue about acceptance or rejection of the input but they are usually inevitable in the model. In fact, a simple observation characterises the only case in which they are not needed.

Observation 3 *For all language L, $L \in REC$ if and only L is accepted by a TM/1A that has no infinite runs (thus only accepting and rejecting ones).*

The proof is straightforward. It is crucial to note that if a TM/1A without infinite runs calls its advice, then by definition the run must be accepting and the concrete advice isn't needed at all. By Observation 3, infinite runs are unavoidable in any TM/1A that accepts a non-recursive language.

By switching accept and reject states, a TM/1A M is turned into a dual TM/1A $\bar{M}$ which is, unsurprisingly, called its dual (and vice versa).

Observation 4 *For any language L, L is accepted (rejected) by a TM/1A M if and only if L is rejected (accepted) by its dual d-TM/1A $\bar{M}$.*

By means of Observation 4, all results proved for TM/1A's in the sequel can be dualised to results for dual TM/1A and vice versa. We will occasionally mention the dualisations explicitly and usually leave them to the reader to fill in. We will use the following notation, very similar to Notation 2.

Notation 3 *For bounding functions g, we denote the class of languages accepted by TM/1A's with $g(n)$-bounded advice by $TM/1A(g)$.*

3 Characterisations

Turing machines with advice cannot be used to characterise co-RE since these machines are 'too strong' – they recognise more languages than we want. What we need is a class of machines that are more powerful than Turing machines, but less powerful than Turing machines with advice. In this Section we will show that Turing machines with one-sided advice are a plausible solution.

3.1 Accepting co-RE (and RE)

We will show that TM/1A's indeed accept precisely the languages in co-RE. For the result we make *no* special assumptions on the 'one-sided advice' used in the TM/1A's.

Theorem 1. *For all languages L, $L \in$ co-RE if and only if L is accepted by a TM/1A.*

Proof. ($\Rightarrow$) Let $L \in$ co-RE, thus $\bar{L} \in$ RE. Let N be a (classical, deterministic) Turing machine that accepts $\bar{L}$. We may assume that N has accepting states, no rejecting states, that an input w is accepted if N's run on w is accepting, and that w is rejected if N's run on w is infinite. We now design a TM/1A M that accepts L.

Define the advice function f by $f(n) = w_n$, where w_n is the word of length n on which N makes the largest number of steps before accepting it, or a default value when when N does not accept any words of length n. (The choice of w_n is similar to that in [2], Theorem 1.) Now define the operation of M as follows. Note that M is inherently non-deterministic[3]:

input: w (say of length n);
choose (non-deterministically) between thread A and thread B:

A — call advice: $z := f(n)$;
interlace N's computations on w and on z and look for the first of the following possible events to happen:
- (A1) *the computation on w halts first and in an accepting state of N*: then continue the computation in an *infinite* run;
- (A2) *the computation on z halts first*: then stop the computation in an *accepting* state.

B — perform N's run on w and look for the first of the following possible events to happen:
- (B1) *the computation on w halts in an accepting state of N*: then stop the computation in a *rejecting* state;
- (B2) *the computation on w does not halt*: implicitly, continue the computation in its *infinite* run.

We claim that machine M is a TM/1A that accepts precisely the words of $\bar{L}$ and rejects those in L. To prove it, we consider how M operates on arbitrary inputs w. We distinguish between two cases:

– Case I: $w \in L$. Consider how M operates as a non-deterministic machine on input w. If M follows thread A then it calls its advice but, by the choice of the advice function, the interlacing of N's computation on w and the advice necessarily leads to event A2, which halts the computation in an accepting state. If M would follow thread B, the advice is not called but the computation will necessarily lead to event B2, leading the computation into an infinite run.

[3] We describe M's operation in pseudo-code but all steps involved are easily realised by a fixed finite number of Turing machine instructions, using those of N.

– Case II: $w \in \bar{L}$. Again consider how M operates. If M follows thread A then it calls its advice but now, by the choice of the advice function, the computation will necessarily lead to event A1, and it will continue in an infinite run. If M would follow thread B, the advice is not called but the computation will necessarily lead to event B1, leading the computation to halt in a rejecting state.

One easily verifies that M satisfies the requirements $(R_0), (R_1), (R_2)$ and that $L_A(M) = L$. This proves the claim.

($\Leftarrow$) Now consider a TM/1A M, and let $L = L_A(M)$. Let M's advice function be f. (NB f can be any arbitrary function now, possibly very different from the special function f used above.) Note that f can only be called during non-rejecting (i.e. accepting or infinite) runs, if at all. Now consider the following, classical but non-deterministic, Turing machine N that operates as follows:

> input: w (say of length n);
> perform a run of M on w and look for the first of the following possible events to happen:
> – (C1) *the run leads to a call for advice*: then do not call the advice and continue the computation in an infinite run;
> – (C2) *the run stops in an accepting state of* M: then continue the computation in an infinite run;
> – (C3) *the run stops in a rejecting state of* M: then stop the computation in an *accepting* state (of N);
> – (C4) *the run stops in an undefined state of* M: then continue the computation in an infinite run.
> – (C5) *the run does not halt*: implicitly, continue the computation in its infinite run.

Note that machine N does not actually call the advice of M but only observes the situation in which M would. We claim that machine N accepts precisely the words of $\bar{L}$. To prove it, we consider how N operates on arbitrary inputs w. We distinguish between two cases:

– Case I: $w \in L$. Consider how N operates as a non-deterministic Turing machine on input w. As $w \in L$, M cannot have a rejecting run on w and thus only events (C1), (C2), (C4) and (C5) can occur. It follows that all possible runs of N will end up being infinite, in this case.
– Case II: $w \in \bar{L}$. Consider again how N operates. As $w \in \bar{L}$, no run of M can lead to event (C2), but events (C1), (C4) and (C5) are all possible and some run must exist that leads to event (C3). Hence the runs of N will all end up being infinite or leading to an accepting state of N, and the latter will occur for at least one run.

One easily verifies that N is a (classical) non-deterministic Turing machine accepting precisely $\bar{L}$ which is thus RE. Consequently $L = L_A(M) \in co-RE$. □

By Observation 4, we can immediately conclude the following result as well.

Corollary 1. *For all languages L, $L \in RE$ if and only if L is accepted by a dual TM/1A.*

3.2 Some consequences

Theorem 1 is interesting in several ways, for example because the use of arbitrary advice functions still keeps the power of TM/1A's limited to co-RE. We give a number of corollaries to further comprehend the effect of this.

First of all, Theorem 1 and Corollary 1 imply that Turing machines with one-sided advice are indeed weaker than Turing machines with advice in general. This is so because the latter can accept every possible language, if there is no further constraint on the advice function.

Corollary 2. *Turing machines with advice are more powerful than Turing machines with one-sided advice.*

In Section 4 (Theorem 2) we will prove that Corollary 2 does *not* hold when we restrict to the 'home base' of TM/1A's, namely to the class co-RE.

Next, Theorem 1 leads to the following interesting conclusion about the behaviour of TM/1A's, using Observation 3.

Corollary 3. *For all languages L, if $L \in$ co-RE - REC then any TM/1A that accepts L will have to make infinite runs.*

We make a further observation on the crucial dependence on the correct advice in the acceptance of a language, in general. Define a TM/1A M with advice function f to be *insensitive* if, given an input w, M *reaches the same conclusion* on w (accept or reject) regardless of the advice function actually used. M is insensitive if and only if the following properties hold:

- if M has accepting runs, then every run that calls f is accepting or infinite even if the advice is replaced by a different function,
- if M has rejecting runs, then every run that calls f is infinite even if the advice is replaced by a different function.

For insensitive d-TM/1A's the correspondingly dualised properties hold.

Proposition 1. *For all languages L, if $L \in$ co-RE - REC then no TM/1A that accepts L can be insensitive.*

Proof. Let $L \in$ co-RE - REC. Suppose by way of contradiction that L could be accepted by an insensitive TM/1A M. We show that L must be recursive. To prove it, we design a (classical, deterministic) Turing machine N for recognizing L as follows. Fix some standard enumeration of the (advice-) strings over Σ and of the pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$.

Consider an arbitrary input w and let $|w| = n$. Now N operates as follows, explained in 'pseudo-code'. N enumerates the pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$ one after the other, and for each pair (i, j) it *(a)* generates the dovetail (tree) of up to i steps of all possible runs of M on w, using the j-th string of Σ^* as advice if it is needed, *(b)* halts in an accept or a reject state if the dovetail leads M to an accept or a reject state respectively, and *(c)* continues with the next pair if the latter didn't occur.

Clearly, if N halts, it halts with the correct decision on the acceptance of w by the assumed insensitivity. To see that N always halts, note that certainly a pair (i, j) will occur which has i large enough and j the rank number of $f(n)$ (whatever it is) in the enumeration of Σ^* so the dovetail will reveal the accepting or rejecting run of M (respectively). This gives an upperbound on the actual occurrence of case *(b)* above. Thus N is always halting and L is recursive. Contradiction. □

A final observation concerns the relationship between the complexity of the accepted languages and the advice function of a TM/1A. We will study this relationship in more detail in Section 4 and confine ourselves here to a first glimpse of the subject. (The case for dual TM/1A's is similar.)

By Observation 1 it is straightforward that for languages L with $L \in$ co-RE but $L \notin$ RE, any TM/1A accepting L must use an advice f for which $R_f \notin RE$. A further observation can be made. Recall that a set is called *immune* if it is infinite and has no infinite RE-subsets (cf. [8]). Immune sets are not in RE. If $L \in$ co-RE and L is immune, then $\bar{L}$ is called a *simple* set.

Let L be a language, $R_f \subseteq \mathbb{N} \times \Sigma^*$ an advice graph. R_f will be called L-*immune* if for every infinite subset $S = \{< n_1, f(n_1) >, \cdots\} \subseteq R_f$ one has: if S is recursively enumerable, then $L \cap \Sigma^{n_i} = \emptyset$ for infinitely many n_i (i.e. n_i as occurring in the pairs of S).

Proposition 2. *Let $L \in co - RE$. If L is immune, then any TM/1A accepting L must use an advice function f for which R_f is L-immune.*

Proof. Let L be as stated. Let M be a TM/1A accepting L and let f be the advice function used by M. Clearly L is infinite (by definition).

Consider any infinite subset $S = \{< n_1, f(n_1) >, \cdots\} \subseteq R_f$. Assume that S is recursively enumerable. Suppose by way of contradiction that $L \cap \Sigma^{n_i} = \emptyset$ for only *finitely* many n_i. Define the language L' accepted by a (classical, deterministic) Turing machine N as follows. Machine N operates in the following manner:

input: w;
perform the following:
- (D1): determine $n = |w|$;
- (D2): recursively enumerate S; if a tuple $< i, u >$ is encountered with $i = n$ then stop the enumeration of S, set $f(n) := u$ and go to the next step;
- (D3): dovetail the computations (i.e. all runs) of M using advice value $f(n)$ determined in the previous step, until an accepting or rejecting state is reached in the dovetail; if a rejecting state is reached, continue the run in an infinite loop, otherwise go to the next step;
- (D4): accept w and halt.

It is clear that N accepts only words from L and that by the assumption on S, there are infinitely many words $w \in L$ for which control passes from (D2) to (D3) during its computation, thus leading to acceptance of these words in (D4). Hence L' is an infinite and recursively enumerable subset on L, contradicting the fact that L is immune. Thus R_f is L-immune. □

4 Complexity

In Theorem 1 we proved that the languages in co-RE are accepted by Turing machines with one-sided advice. For machines with advice we know that, generally speaking, the machine becomes more powerful the longer (and thus richer) the advice is (cf. Karp and Lipton [5, 6], Verbaan [14]). In this section we will show that Turing machines with one-sided advice have a very similar property, but in a more limited range.

We first observe that the *advice* function of a (dual) TM1/A can be exactly the same as for a general TM/A, if the latter accepts the same language as the former. We also show that the advice function of a TM/1A, although it can be fully arbitrary in general as long as the machine is a valid TM/1A, can remain surprisingly limited. On the other hand we show that below this upper limit, there is an infinite hierarchy based on advice length like in the general case. It follows that there is an infinite hierarchy inside co-RE based entirely on the power of the advice used in a TM/1A. A dual result holds for RE.

4.1 Bounds

We first prove some elementary facts about the power of one-sided advice in general. Let $|\Sigma| = \sigma$.

We first prove that, if a co-RE language L can be accepted by a standard Turing machine with advice $f(n)$, then a TM/1A can accept L with advice f as well, i.e. it doesn't need a more complicated advice. The fact is a generalization of the 'only if'-part of Theorem 1, implementing the core idea of a TM/1A. For completeness, we give the proof in full.

Theorem 2. *For all languages $L \in$ co-RE, if L can be accepted by an ordinary TM/A with advice $f(n)$, then L can be accepted by a TM/1A with advice $f(n)$ as well.*

Proof. Let $L \in$ co-RE, thus $\bar{L} \in$ RE. Let N be a (classical, deterministic) Turing machine that accepts $\bar{L}$, as in the proof of Theorem 1. Let M' be a TM/A with advice f that accepts L. W.l.o.g. we may assume that machine M' is a classical, deterministic machine. We now design a TM/1A M with advice function f that accepts L.

We define M by describing its action on an arbitrary input w. It will be clear that M's operations can be defined by a fixed finite Turing machine program.

input: w (say of length n);
choose (non-deterministically) between thread A and thread B:

A perform the run of M' on w, calling the advice $f(n)$ when needed, and look for the first of the following possible events to happen:

- (A1) *the computation on w halts in an accepting state of M*: then stop the computation, in an *accepting* state;
- (A2) *the computation on w halts in a rejecting state of M*: then continue the computation in an infinite run;

- (A3) *the computation on w halts in an undefined state of M*: then continue the computation in an infinite run;

(Implicitly, if none of these events occurs, the run continues as an infinite run.)

B perform N's run on w and look for the first of the following possible events to happen:

- (B1) *the computation on w halts in an accepting state of N*: then stop the computation in a *rejecting* state;
- (B2) *the computation on w does not halt*: implicitly, continue the computation in its infinite run.

We claim that M is a TM/1A that accepts precisely the words of L. To prove it, we distinguish two cases for any possible input w:

- Case I: $w \in L$. Consider how M operates on input w as a non-deterministic machine. If M follows thread A then it performs a run of M', possibly calling advice. Because M' accepts L, the run must necessarily lead to event (A1), thus to acceptance of w. If M would follow thread B, the advice is not called and the computation will lead to event (B2), i.e. continuing the computation in its infinite run.
- Case II: $w \in \bar{L}$. Again consider how M operates on input w. If M follows thread A then it may call its advice but now the computation will necessarily lead to event (A2) or (A3) or to none of these events at all. In all cases this leads the computation to continue in an infinite run. If M would follow thread B, the advice is not called but the computation will necessarily lead to event B1, leading the computation to halt in a rejecting state.

One easily verifies that M satisfies the requirements of a TM/1A and that $L_A(M) = L$. □

Theorem 2 implies that TM/1A's are as efficient with their advice as standard Turing machines with advice in accepting co-RE languages. Stated differently, if a co-RE language L can be accepted by a TM with $g(n)$-bounded advice, then it can be accepted by a TM/1A with $g(n)$-bounded advice as well.

The next observation follows from the proof of Theorem 1 as well and shows that for accepting co-RE languages, TM/1A's can in fact do with a quite limited type of advice.

Definition 4. *For an arbitrary language L, we let $c_L : \mathbb{N} \to \Sigma^*$ be the* census function *defined by $c_L(n) = \sharp\{w \in L \mid |w| = n\}$ (NB written in σ-ary notation).*

Theorem 3. *L is accepted by a TM/1A if and only if L is accepted by a TM/1A using $\log_\sigma c_L(n)$-bounded advice.*

Proof. We only need to prove the 'only if' part. Thus, let L be accepted by an arbitrary TM/1A M. By Theorem 1 we know that $L \in$ co-RE. Let N be a (classical, deterministic) Turing machine that accepts $\bar{L}$. If we now inspect the first part of the proof of Theorem 1, we see that L can in fact be accepted

by a TM/1A M' with an advice function f defined by $f(n) = w_n$, where w_n is the word of length n on which N makes the largest number of steps before accepting it, or a default value when N does not accept any words of length n (the 'Barzdin advice', cf. Theorem 1).

Now change the advice function M' uses into $c_L(n)$ and insert a little subroutine whenever the 'advice' is called that actually computes w_n from $c_L(n)$, as follows:

- (E1): if $c_L(n) = 0$, then set w_n to its default value (as above);
- (E2): if $c_L(n) = m$ for some 'number' m, then start a dovetail of N's runs on all inputs of length n. Proceed until accepting runs have been found for precisely m different inputs of length n. Then halt the dovetailing and set w_n equal to the word which was last found to be accepted in the dovetail (and which thus has the longest accepting run of N among the words of length n).

Having constructed w_n from $c_L(n)$, the computation of M' can proceed in the old way. □

Observing that $c_L(n) \leq \sigma^n$ for all L, Theorem 3 implies the following. The corollary also follows from the use of the Barzdin'-advice w_n directly.

Corollary 4. *Every language $L \in$ co-RE can be accepted by a TM/1A using linearly bounded advice.*

Corollary 4 dualises for RE as usual. Theorem 2 and Corollary 4 can be combined into the following observation.

Corollary 5. *$TM/A(g) \cap$ co-RE $= TM/1A(g) \subseteq TM/1A(n) =$ co-RE, where g is any advice bounding function.*

An even more special interpretation can be given to Theorem 3. For $L \subseteq \Sigma^*$, let $U(L) \subseteq \{1\}^*$ be the language of 'unary equivalents' of the words of L when these are viewed as σ-ary numbers. Clearly the mapping $U : \Sigma^* \to \{1\}^*$ and its inverse are Turing-computable. Note also that $U(\bar{L}) = U(\bar{L})$.

Theorem 4. *For all languages L, $L \in$ co-RE (RE) if and only if $U(L)$ is accepted by a (dual) TM/1A with its characteristic function as advice.*

Proof. ($\Rightarrow$) Let $L \in$ co-RE. It easily follows that $U(L) \in$ co-RE. By the argument in the proof of Theorem 3, $U(L)$ is accepted by a TM/1A M using $c_{U(L)}(n)$ as advice function. The latter is precisely the characteristic function of $U(L)$ in $\{1\}^*$.

($\Leftarrow$) Conversely, let $U(L)$ is accepted by a TM/1A with its characteristic function as advice. It follows from Theorem 1 that $U(L) \in$ co-RE. From this it easily follows that $L \in$ co-RE. □

(Note that the characteristic function of $U(L)$ cannot just be read off for deciding acceptances. In particular, the TM/1A is not allowed to read it off at all in reject decisions.)

4.2 Hierarchy

We now show that inside co-RE (or RE, respectively) there is an infinite proper hierarchy 'below' the linear advice bound, based on the length of the one-sided advice the TM/1A's need to accept the languages in co-RE: the bigger advice 'below linear' we consider, the more languages can be accepted.

We consider RE and co-RE languages over finite alphabet $\Sigma = \{0, 1, \cdots\}$, with $|\Sigma| = \sigma \geq 2$. We prove the following key result.

Theorem 5. *There are infinite sequences of functions $\{g_i\}_{i\geq 1}$ with $g_1(n) < g_2(n) < \cdots (< n)$ such that for all $i > 1$, TM/1A's with g_i-bounded advice are strictly more powerful, i.e. accept strictly more languages of co-RE, than TM/1A's with g_{i-1}-bounded advice.*

By this theorem, the sets of languages accepted by TM/1A's with g_i-bounded one-sided advice for $i \geq 1$ give us an infinite hierarchy inside co-RE as claimed. By duality, a similar result applies to RE.

Theorem 5 is a direct consequence of the following Lemma, to which we will devote the remainder of this Section.

Lemma 1. *Let g be any recursive function with $g(n) < n$ for all n. Then there is a co-RE language L that can be accepted by a TM/1A with $\min(\sigma^{g(n)+1}, n)$-bounded advice but not by any TM/1A with $g(n)$-bounded advice.*

For the proof of Lemma 1 we will use a diagonal argument, based on the enumeration of a class of relevant acceptors. Notice that the class of TM/1A's itself is not effectively enumerable. Therefore, we will try to enumerate a superset of it instead, namely the class of all non-deterministic Turing machines with $g(n)$-bounded advice over alphabet Σ, as our TM/1A's will certainly occur among them. However, this cannot be done in this form, as the number of different advice functions to enumerate for each machine would be uncountable. To get around it, the concrete advice functions f of the machines will *not* be enumerated explicitly.

Thus, we only enumerate all non-deterministic Turing machines with advice in the right form, including the advice-calling instructions and a partition of the states into **accept** states, **reject** states, and **undefined** states, but no concrete advice function itself. By a slight abuse of terminology, we will refer to these machines simply as TM/A's below. We will use a separate mechanism to consider all possible $g(n)$-bounded values of the advice per machine, for each n. Let $\{M_n\}_{n\geq 1}$ be an effective enumeration of all TM/A's as above.

Proof of Lemma 1. Let g be any recursive function with $g(n) < n$ for all n. We define the language $L \subseteq \Sigma^*$ as the *complement* of the language L' which is accepted by the following (classical, deterministic) Turing machine N. We describe N through its action on inputs $w \in \Sigma^*$: it will be clear that the operations can be realised by a finite Turing machine program for N. Letting $|w| = n$, N acts as follows:

- (F1): if $|w| \leq g(n)$, then N goes into an infinite loop (implying that it 'rejects' w).
- (F2): if $|w| > g(n)$ but w is not of the form $0^*1\alpha$ for $|\alpha| \leq |g(n)|$, then N goes into an infinite loop (implying that it 'rejects' w).
- (F3): if $|w| > g(n)$ and w is of the form $0^*1\alpha$ for some $|\alpha| \leq |g(n)|$, then N works through the machine-enumeration to retrieve M_n and starts dovetailing the runs of M_n on input w, *using* $f(n) = \alpha$ *as advice when advice is called.*
  Let N generate the computation tree of all possible runs level after level, i.e. in breadth-first manner. We may assume w.l.o.g. that no run ends in an 'undefined' state, otherwise just continue such a run in an infinite loop. While generating the levels, N looks for the first of the following events to occur and acts accordingly:
  - (F3:1) *there is a run of* M_n *that halts in the generated level and the leftmost such run halts in an accept state*: then N *halts* and *accepts* w by moving to an accept state (of N).
  - (F3:2) *there is a run of* M_n *that halts in the generated level and the leftmost such run halts in a reject state*: then N sends itself into an infinite loop (thus implicitly rejecting w).

  Note that N keeps generating levels as long as none of the above events occurs. If none occurs, N simply is in an infinite loop on w (and thus rejecting it).

Because N is a straightforward (classical, deterministic) Turing machine, its accepted language L' is recursively enumerable. Hence $L = \bar{L'} \in$ co-RE. Consequently, by Theorem 1 there is TM/1A accepting L.

Claim 1 *L cannot be accepted by a TM/1A with $g(n)$-bounded advice.*

Proof of claim. Suppose L is accepted by TM/1A with $g(n)$-bounded advice. Let the TM/1A occur in the enumeration as machine M_k and have advice function $f(n)$, with $|f(n)| \leq g(n)$ for all n. Consider the action of M_k on input $w = 0\cdots01\alpha$ of length k, with $\alpha = f(k)$. (We use $0\cdots0$ to denote any string of zero or more 0's.)

We now claim that a contradiction occurs. To prove it, we distinguish between two cases.

- Case I: $w \in L$, thus $w \notin L'$. Consider how N acts on input w. By design, N proceeds in part (F3) of its program, using the correct advice value. As $w \in L$, M_k accepts w and thus by the working of a TM/1A, the dovetailing of M_k's runs must lead N to event (F3:1). Then N halts and accepts w, but this contradicts that $w \notin L'$.
- Case II: $w \in \bar{L}$, thus $w \in L'$. As above, N proceeds in part (F3) of its program and dovetails the runs of M_k on w, using the correct advice value. As $w \in \bar{L}$, M_k rejects w and thus by the working of a TM/1A, the dovetailing of M_k's runs must lead N to event (F3:2). Then N sends itself in an infinite loop on w, but this contradicts that $w \in L'$.

Hence a contradiction arises in all possible cases. This proves the claim. □

Claim 2 *L can be accepted by a TM/1A with $\sigma^{g(n)+1}$-bounded advice.*

Proof of claim. Let $h(n) = 1 + \sigma + \cdots + \sigma^{g(n)}$, and fix some standard enumeration of the $h(n)$ strings of length at most $g(n)$. Denote the i-th string in this enumeration by $Str_n(i)$. Note that $h(n) \leq \sigma^{g(n)+1}$. Define the advice function f with $f(n) = b_1 \cdots b_{h(n)} \in \{0,1\}^*$ as follows:

$$b_i = \begin{cases} 1 \text{ if } 0\cdots 01\alpha \in L, \text{ with } |0\cdots 01\alpha| = n \text{ and } Str_n(i) = \alpha \\ 0 \text{ otherwise} \end{cases}$$

We now design a TM/1A M with advice function $f(n)$ as follows. We describe M by its action on an arbitrary input $w \in \Sigma^*$, where it will be clear that the operations can all be defined by a finite Turing machine program 'with advice'. Letting $|w| = n$, M acts as follows:

- (G1): if $|w| \leq g(n)$ then M halts and accepts w in an accept state (without any call for advice).
- (G2): if $|w| > g(n)$ but w is not of the form $0\cdots 01\alpha$ for $|\alpha| \leq |g(n)|$, then M halts and accepts w in an accept state (without any call for advice).
- (G3): if $|w| > g(n)$ and w is of the form $0\cdots 01\alpha$ for some $|\alpha| \leq |g(n)|$, then M chooses (non-deterministically) between thread A and thread B:

  A determine i be such that $\alpha = Str_n(i)$;
  call advice: $f(n) = b_1 \cdots b_{h(n)}$;
  if $b_i = 1$ then M stops and *accepts* w by moving to an accepting state;
  if $b_i = 0$ then M sends itself into an infinite loop.

  B perform N's run on w, necessarily at (F3) by the form of w, and look for the first of the following possible events to happen:
  - (G3:B1) *N runs into event (F3:1) and stops, accepting w*: then M stops and *rejects* w by moving to a rejecting state.
  - (G3:B2) *N runs into event (F3:2) and goes in an infinite loop*: then M goes into an infinite loop as well.

  If N keeps running without reaching any of the previous events, then M simply keeps running as well, making the run to an infinite run.

We now verify that M is a (non-deterministic) TM/1A which precisely accepts L. We need to verify two things: that M satisfies the requirements of a TM/1A, and that M computes the right acceptances. We can restrict ourselves to considering inputs w of the form $0\cdots 01\alpha$ for an $|\alpha| \leq |g(n)|$, as M takes the correct (accepting) decisions for all other inputs, even deterministically and without any call on the advice.

For words $w = 0\cdots 01\alpha$ with $|\alpha| \leq |g(n)|$, M necessarily works in part (G3) of its program. We now distinguish two cases:

- Case I: $w \in L$. If M follows thread A, it determines i such that $\alpha = Str_n(i)$ and calls its advice. It follows from the advice that $b_i = 1$ in this case and M will accept w and move to an accept state. If M follows thread B, it starts

performing N's run on w, i.e. the dovetailing of all possible runs of M_n on w, with advice α (not called but derived from w) and necessarily in part (F3) of its program. Because $w \in L$ and thus $w \notin L'$, N can*not* run into event (F3:1). Thus, in thread B, M can only get into infinite run.

– Case II: $w \notin L$. If M follows thread A, it determines i such that $\alpha = Str_n(i)$ and calls its advice. It follows from the advice that $b_i = 0$ in this case and M will send itself into an infinite loop. If M follows thread B, it starts performing N's run on w, i.e. the dovetailing of all possible runs of M_n on w, with advice α (again, not called but derived from w) and again in part (F3) of its program. Because $w \notin L$ and thus $w \in L'$, N must run into event (F3:1). Thus, in thread B, M will reach event (G3:B1), stop and reject w by moving to a rejecting state (all this without calling its advice).

It follows that M indeed works as a correct TM/1A and that it accepts precisely the language L. □

With the help of Corollary 4, this completes the proof of Lemma 1. □

Lemma 1 immediately implies the existence of an infinite hierarchy of subclasses within co-RE, based on the size of the advice in the relevant TM/1A's. The Lemma also leads to the following strengthening of Corollary 5.

Corollary 6. *Let g be any recursive function with $g(n) < n$ for all n. Then $TM/A(g) \cap$ co-RE $= TM/1A(g) \subsetneq TM/1A(n) =$ co-RE.*

5 Conclusions

In this paper we considered the intriguing question of giving a simple model of hypercomputation for co-RE, i.e. for the class Π_1 of the Arithmetical hierarchy. The model of 'Turing machines with one-sided advice' (TM/1A's) that we proposed satisfies all requirements one can reasonably impose on such a model. We have shown that TM/1A's precisely accept all co-RE languages and that there is an infinite hierarchy inside co-RE based on the complexity (length) of the advice used by the accepting TM/1A's. The results dualise perfectly for RE.

Recursively enumerable languages and their complements are a classical topic within computability theory. The co-RE languages per se have been more of a theoretical than a practical interest, despite their relevant base-position in the Arithmetical hierarchy. They typically arise in connection with undecidability results or in recursion-theoretic studies only. Our results bring new insight into the nature of co-RE languages and of non-RE languages in general.

As a model of hypercomputation, TM/1A's are close enough to ordinary Turing machines to expect that their operation could be studied from similar perspectives. Whereas other models of computation in this domain tend to be search-oriented or non-iterative, Turing machines with one-sided advice stay very close to our intuition, computationally. This could open various interesting lines of further study. Especially sub-recursive (e.g. time-bounded) variants of TM/1A's may be interesting to study further.

References

1. J.L. Balcázar, J. Díaz, J. Gabarró, *Structural Complexity* I, Second Edition, Springer, 1995.
2. Ja.M. Barzdin', Complexity of programs to determine whether natural numbers not greater than n belong to recursively enumerable set, *Soviet Math. Dokl.* 9:5 (1968) 1251-1254.
3. A. Chandra, D. Kozen, L. Stockmeyer, Alternation, *JACM* 28:1 (1981) 114133.
4. A. Ehrenfeucht, G. Rozenberg, K. Ruohonen, A morphic representation of complements of recursively enumerable sets, *JACM* 28:4 (1981) 706-714.
5. R.M. Karp, R.J. Lipton, Some connections between non-uniform and uniform complexity classes, *Proc. Twelfth Ann. ACM Symp. on Theory of Computing*, ACM Press, 1980, pp. 302309.
6. R.M. Karp, R.J. Lipton, Turing machines that take advice, *L'Enseignement Mathématique* II^e Série, Tome XXVIII (1982) 191209. (Extended version of [5].)
7. D. Leivant, Alternating Turing machines and the Analytical Hierarchy, in: A. Voronkov (Ed.), *Turing-100. The Alan Turing Centenary*, EPiC Series vol 10, Easychair, 2012, pp 204-213.
8. H. Rogers Jr, *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967.
9. K. Ruohonen, On machine characterization of nonrecursive hierarchies, *Ann. Univ. Turkuensis*, Ser. A I 186 (1984) 87-101.
10. A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.* 42-2 (1936) 230-265; A correction, ibid., 43-2 (1937) 544-546.
11. A.M. Turing, Systems of logic based on ordinals, *Proc. London Math. Soc.*, Series 2, 45 (1939) pp. 161-228.
12. J.S. Ullian, Three theorems concerning principal AFL's, *Journal of Computer and Systems Sciences* 5 (1971) 304-314.
13. J. van Leeuwen, J. Wiedermann, The Turing machine paradigm in contemporary computing, in: B. Engquist and W. Schmidt (Eds), *Mathematics Unlimited - 2001 and Beyond*, Springer-Verlag, 2001, pp. 1139-1155.
14. P.R.A. Verbaan, *The Computational Complexity of Evolving Systems*, Ph.D. Thesis, Dept of Information and Computing Sciences, Faculty of Science, Utrecht University, 2006.