Clockability for Ordinal Turing Machines

Merlin Carl^(✉)

Institut für mathematische, naturwissenschaftliche und technische Bildung, Abteilung
für Mathematik und ihre Didaktik. Europa-Universität Flensburg,
Flensburg, Germany
`merlin.carl@uni-konstanz.de`

Abstract. We study clockability for Ordinal Turing Machines (OTMs). In particular, we show that, in contrast to the situation for ITTMs, admissible ordinals can be OTM-clockable, that Σ_2 -admissible ordinals are never OTM-clockable and that gaps in the OTM-clockable ordinals are always started by admissible limits of admissible ordinals. This partially answers two questions in [3].

1 Introduction

In ordinal computability, “clockability” denotes the property of an ordinal that it is the halting time of some program. The term was introduced in [9], which was the paper that triggered the bulk of research in the area of ordinal computability by introducing Infinite Time Turing Machines (ITTMs).¹ By now, a lot is known about clockability for ITTMs. To give a few examples: In [9], it was proved that there are gaps in the ITTM-clockable ordinals, i.e., there are ordinals $\alpha < \beta < \gamma$ such that α and γ are ITTM-clockable, but β is not. Moreover, it is known that no admissible ordinal is ITTM-clockable (Hamkins and Lewis, [9]), that the first ordinal in a gap is always admissible (Welch, [14]), that the supremum λ of the ITTM-writable ordinals (i.e. ordinals coded by a real number that is the output of some halting ITTM-computation) equals the supremum of the ITTM-clockable ordinals (Welch, [14]), that an ITTM-clockable γ has a code that is ITTM-writable in γ many steps (Welch, [14]) and that ITTM-writable ordinals have real codes that are ITTM-writable at the point the next clockable appears. Moreover, it is known that not every admissible below λ starts a gap, there are admissibles properly inside gaps, and occasionally many of them (Carl, Durand, Lafitte, Ouazzani, [6]). And indeed, clockability turned out to be a central topic in ordinal computability; it was, for example, crucial for Welch’s analysis of the computational strength of ITTMs.

Besides ITTMs, clockability was also considered for Infinite Time Register Machines (ITRMs), where the picture turned out to be quite different: In particular, there are no gaps in the ITRM-clockable ordinals (see [5]), and in fact,

¹ As one of our referees pointed out, there are earlier considerations of machine models computing along an ordinal time axis; however, none of them was studied in the detail that ITTMs were.

the ITRM-clockable ordinals are exactly those below $\omega_\omega^{\text{CK}}$, which thus includes ω_n^{CK} for every $n \in \omega$, i.e. the first ω many admissible ordinals.

For other models, clockability received comparably little attention. This work arose out of a question of T. Kihara during the CTFM² conference in 2019 in Wuhan who, after hearing that admissible ordinals are never ITTM-clockable, asked whether the same holds for OTMs. After most of the results of this paper had been proved, we found two questions in the report of the 2007 BIWOC (Bonn International Workshop on Ordinal Computability) [3] concerning this topic: the first (p. 42, question 9), due to J. Reitz, was whether ω_1^{CK} was OTM-clockable, the second, due to J. Hamkins, whether gap-starting ordinals for OTMs can be characterized as “something stronger” than being admissible. In [3], both are considered to be answered by the claim that no admissible ordinal is OTM-clockable, which is attributed to J. Reitz and S. Warner. Upon personal inquiry, Reitz told us that they had a sketch of a proof which, however, did not entirely work; what it does show with a few modifications, though, is that Σ_2 -admissible ordinals are not OTM-clockable, and the argument that Reitz sketched in personal correspondence to us in fact resembles the one of Theorem 6 below. We thus regard Reitz and Warner as the first discoverers of this theorem. Both the argument of Reitz and Warner from 2007 and the one we found during the CTFM in 2019 are adaptations of Welch’s argument that admissible ordinals are not ITTM-clockable.

The statement actually made in [3], is, however, false: As we will show below, ω_n^{CK} is OTM-clockable for any $n \in \omega$. Thus, there are plenty of admissible ordinals that are OTM-clockable, and the answer to the first question is positive. The idea is to use the ITRM-clockability of these ordinals, which follows from Lemma 3 in [5], together with a slightly modified version of the obvious procedure for simulating ITRMs on OTMs. This actually shows that ω_n^{CK} is clockable on an ITTM with tape length α as soon as $\alpha > \omega$. Thus, the strong connection between admissibility and clockability seems to depend rather strongly on the details of the ITTM-architecture. We remark that this is a good example of how the studies of different models of infinitary computability can fruitfully interact: At least for us, it would not have been possible to find this result while only focusing on OTMs.

Moreover, we will answer the second question in the positive as well by showing that, if α starts a gap in the OTM-clockable ordinals, then α is an admissible limit of admissible ordinals.³

Of course, the gap between “admissible limit of admissible ordinals” and “ Σ_2 -admissible” is quite wide. In particular, we do not know whether every gap starting ordinal for OTMs is Σ_2 -admissible, though we conjecture this to be false.

² International Conference on Computability Theory and Foundations of Mathematics.

³ The notion of admissibility will play a prominent role in this paper. Readers unfamiliar with it are referred to Barwise [1].

2 Ordinal Turing Machines

Ordinal Turing Machines (OTMs) were introduced by Koepke in [10] as a kind of “symmetrization” of ITTMs: Instead of having a tape of length ω and the whole class of ordinals as their working time, OTMs have a tape of proper class length On while retaining On as their “working time” structure. We refer to [10] for details.

In contrast to Koepke’s definition but in closer analogy with the setup of ITTMs, we allow finitely many tapes instead of a single one. Each tape has a head, and the heads move independently of each other; the program for such an OTM is simply a program for a (finite) multihead Turing machine. At limit times, the inner state (which is coded by a natural number), the cell contents and the head positions are all determined as the inferior limits of the sequences of the respective earlier values. At successor steps, an OTM-program is carried out as if on a finite Turing machine with the addition that, when a head is moved to the left from a limit position, it is reset to the start of the tape. Though models of ordinal computability generally enjoy a good degree of stability under such variations as far as computational strength is concerned, this often makes a difference when it comes to clockability. Intuitively, simulating several tapes with separate read-write-heads on a single tape requires one to check the various head positions to determine whether the simulated machine has halted, which leads to a delay in halting. For ITTMs, this is e.g. demonstrated in [13]. For OTMs, insisting on a single tape would lead to a theory that is “morally” the same as the one described here, but make the results much less compelling and the proofs more technically involved and harder to follow.⁴ Thus, allowing multiple tapes seems to be a good idea.

An important property of OTMs that will be used below is the existence of an OTM-program P that ‘enumerates L ’; in particular, P will write (a code for) the constructible level L_α on the tape in $< \alpha'$ many steps, where α' is the smallest exponentially closed ordinal $> \alpha$ (this notation will be used throughout the paper).

The following picture of OTM-computations may be useful to some readers: Let us imagine the tape split into ω -blocks. Then an OTM-computation proceeds like this: The head works for a bit in one ω -block, then leaves it to the right, works for a bit in the new ω -portion, again leaves it to the right and so on, until eventually the computation either halts or the head is moved back from a limit position, i.e., goes back to 0 and starts over. Thus, if one imagines an ω -portion as single point, then the head moves from left to right, jumps back to 0, moves right again etc. Moreover, in each ω -portion, we have a classical ITTM-computation (up to the limit rules for the head position and the inner state, which make little difference).

We fix some terminology for the rest of this paper.

⁴ For example, by simulating multitape machines on a single-type machine in a rather straightforward way, one can see that the following holds: If α is exponentially closed and clockable by an OTM, then $\alpha \cdot 2$ is clockable by an OTM using only one tape.

Definition 1. *If M is one of ITRM, ITTM or OTM and α is an ordinal, then α is called M -clockable if and only if there is an M -program that halts at time $\alpha + 1$.⁵ α is called M -writable if and only if there is a real number coding α that is M -computable. An M -clockable gap is an interval $[\alpha, \beta)$ of ordinals such that $\alpha < \beta$, no element of $[\alpha, \beta)$ is M -clockable and $[\alpha, \beta)$ is maximal in the sense that there are cofinally many M -clockable ordinals below α and β is M -clockable. In this case, we say that α “starts” the gap and call α a “gap starting ordinal” or “gap starter” for M .*

3 Basic Observations

We start with some useful observations that can mostly be obtained by easy adaptations of the corresponding results about ITTM-clockability.

We start by noting that the analogue of the speedup-theorem for ITTMs from [9] holds for multitape-OTMs. This is proved by an adaptation of the argument for the speedup-theorems for ITTMs. The main difference is that, in contrast to ITTMs, OTMs do not have their head on position 0 at every limit time and that the head may make long “jumps” when moved to the left from a limit position. This generates a few extra complications.

To simplify the proof, we start by building up a few preliminaries.

For the ITTM-speedup, the following compactness property is used: If P halts in $\delta + n$ many steps and the head is located at position k at time δ , then only the n cells contents before and after the k th one at time δ are relevant for this. Now, this is a fixed string s of $2n$ bits. In [9], a construction is described that achieves that the information whether these $2n$ cells currently contain s at a limit time γ is coded on some extra tapes at time γ . Due to the special limit rules for ITTMs that set the head back to position 0 at every limit time, the Hamkins-Lewis-proof has this information stored at the initial tape cells, but the construction is easily modified to store the respective information on any other tape position.

We will use it in the following way: Suppose that P is an OTM-program that halts at time $\delta + n$, where δ is a limit ordinal and $n \in \omega$. We want to “speed up” P by n steps, i.e. to come up with a program Q that halts in δ many steps. Suppose that P halts with the head on position $\gamma + k$, where γ is a limit ordinal and $k \in \omega$. Let m be $k - n$ if $k - n \geq 0$ and 0, otherwise, and let s be the bit string present on positions $\gamma + m$ until $\gamma + k + n$ at time δ . Then we use the Hamkins-Lewis-construction to ensure that the information whether the bit string present on positions $\eta + m$ until $\eta + k + n$ is equal to s on the $(\eta + k)$ th cells of three extra tapes, for each limit ordinal η .

An extra complication arises from the possibility of a “setback”: Within the n steps from time δ to time $\delta + n$, it may happen that the head is moved left from position δ , thus ending up at the start of the tape. Clearly, it will then take $< n$ many further steps at the start of the tape and only consider the first n

⁵ The $+1$ allows limit ordinals to appear as halting times and thus simplifies the theory.

bits during this time. However, we need to know what these bits are - or rather, whether they are the “right ones”, i.e., the ones present at time δ - while our head is located at position $\delta + k$. The idea is then to store this information in the inner state of the sped-up program. We thus create extra states: The new state $2i$ will represent the old state i together with the information that the first n bits were the “right ones” (i.e. the same ones as at time δ) and $2i + 1$ will represent the old state i together with the information that some of these bits deviated from those at time δ . To achieve this, we use an extra tape T_4 . At the start of Q , a 1 is written to each of the first n cells of T_4 ; after that, the head on T_4 is set back to position 0 and then moved along with the head of P . In this way, we will always know whether the head of P is currently located at one of the first n cells. Whenever this is the case, we insert some intermediate steps to read out the first n bits, update the inner state and move the head back to its original position. (This requires some additional states, but we skip the details). Note that, if η is a limit time and the first n bits have been changed unboundedly often before η , then the head will be located at one of these positions at time η by the liminf-rule and thus, a further update will take place so that the state will correctly represent the configuration afterwards. On the other hand, if the first n bits were only changed boundedly often before time η , then let $\bar{\eta}$ be the supremum of these times. We just saw that the state will represent the configuration correctly finitely many steps after time $\bar{\eta}$, after which the first n cell contents remain unchanged, so that the state is still correct at time η . In each case, updating this information and returning to the original configuration will take only finitely many extra steps and thus not cause a delay at limit times.⁶

In the following construction, we will need to know whether the head is currently located at a cell the index of which is of the form $\delta + k$, where δ is a limit ordinal and k is a fixed natural number. To achieve this, we add three tapes T_0 , T_1 and T_2 to P . The tape T_0 serves as a flag: By having two cells with alternating contents 01 and 10, we can detect a limit time as a time at which both cells contain 0. On T_2 , we move the head along with the head on P and place a 1 on a cell whenever we encounter a cell on which a 0 is written. Thus, the head occupies a certain limit position for the first time if and only if the head on T_1 reads a 0 at a limit time. Finally, on T_2 , we move the head along with the heads on T_1 and the main tape. Whenever the head on T_1 reads a 0 at a limit time, we interrupt the computation, move the head on T_2 for k many steps to the right, write a 1, move the head k many places to the left, and continue. In this way, the head on T_2 will read a 1 if and only if the head on the main tape is at a position of the desired form. As this merely inserts finitely many steps occasionally, running this procedure along with an OTM-program P will still carry out δ many steps of P at time δ whenever δ is a limit ordinal. We will say that the head is “at a $\delta + k$ -position” if the index of the cell where it is

⁶ This leaves us with the case that the head occupies one of the first n tape positions at time δ , in which case even a finite delay would increase our running time. However, in this special case, no setback will take place during the last n steps of the computation, so the construction described in this paragraph can simply be skipped.

currently located is of this form with δ a limit ordinal and, by the construction just described, we can use formulations like “if the head is currently at a $\delta + k$ -position” in describing OTM-programs without affecting the running time at limit ordinals.

Lemma 1. *If $\alpha + n$ is OTM-clockable and $n \in \omega$, then α is OTM-clockable.*

Proof. It is clear that finite ordinals are OTM-clockable and that OTM-clockable ordinals are closed under addition (by simply running one program after the other).⁷ Thus, it suffices to consider the case that α is a limit ordinal. Moreover, we assume for simplicity that P uses only one tape.⁸

Let P be an OTM-program that runs for $\alpha + n$ many steps, where α is a limit ordinal. We want to construct a program Q that runs for α many steps. Let the head position at time α be equal to $\delta + k$, where δ is a limit ordinal and $k \in \omega$. As above, let m be $k - n$ if $k - n \geq 0$ and otherwise let $m = 0$. Let s be the bit string present on the positions $\delta + m$ until $\delta + k + n$ at time α , and let t be the string present on the first n positions.

Using the constructions explained above, Q now works as follows: Run P . At each step, determine whether the head is currently at a location of the form $\eta + k$ with η a limit ordinal and whether one of the two following conditions holds:

1. The head is currently at one of the first n positions and the bit string currently present on the positions $\eta + m$ up to $\eta + k + n$ is equal to s .
2. The head is currently not on one of the first n positions, the bit string currently present on the positions $\eta + m$ up to $\eta + k + n$ is equal to s and whether the bit string currently present on the first n positions is equal to t .

If not, continue with P . Otherwise, halt. As described above, the necessary information can be read off from the various extra tapes and the inner state simultaneously. Now it is clear that, if Q halts at time β , then P will halt at time $\beta + n$. Thus, Q halts at time α , as desired.

Definition 2. *Let σ be the minimal ordinal such that $L_\sigma \prec_{\Sigma_1} L$, i.e. such that L_σ is a Σ_1 -submodel of L .*

Proposition 3. *Every OTM-clockable ordinal is $< \sigma$, and their supremum is σ .*

Proof. The statement ‘The program P halts’ is Σ_1 . Moreover, any halting OTM-computation is contained in L . Consequently, if P halts, its computation is contained in L , and hence in L_σ , and thus, the halting time of P , if it exists, is $< \sigma$.

On the other hand, every real number in L_σ is OTM-computable (see, e.g., [12], proof of Corollary 3), including codes for all ordinals $< \sigma$, and thus we can

⁷ It is folklore (and easy to see) that, for any reasonable model of computation, clockable ordinals are closed under ordinal arithmetic, i.e. under addition, multiplication and exponentiation, see e.g. [9] or [5]. This also holds true for OTMs.

⁸ If P uses several tapes, the construction below is carried out for each of these.

write such a code for any ordinal $\alpha < \sigma$ and then run through this code, which takes at least α many steps. Thus, there is an OTM-clockable ordinal above α for every $\alpha < \sigma$.

Proposition 4. *There are gaps in the OTM-clockable ordinals. That is, there are ordinals $\alpha < \beta < \gamma$ such that α and γ are OTM-clockable, but β is not.*

Proof. This works like the argument in Hamkins and Lewis ([9], Theorem 3.4) for the existence of gaps in the ITTM-clockable ordinals: Take the OTM-program that simultaneously simulates all OTM-programs and halts as soon as it arrives at a level at which no OTM-program halts. If there were no gap, then this program would halt after all OTM-halting times, which is a contradiction.

The following is an OTM-version of Welch’s “quick writing theorem” (see [14], Lemma 48) for ITTMs.

Lemma 2. *If an ordinal α is OTM-clockable, then a real number coding α is OTM-writable in $< \alpha'$ many steps, where α' denotes the next exponentially closed ordinal after α .*

Proof. If α is clocked by some OTM-program P , then $L_{\alpha+\omega}$ believes that P halts. Thus, there is a Σ_1 -statement that becomes true between L_α and $L_{\alpha+\omega}$ for the first time and hence, by finestructure (see [2], Lemma 1), a real number coding $\alpha + 1$ is contained in $L_{\alpha+\omega}$. But the OTM-program Q that enumerates L will have (a code for) $L_{\alpha+\omega}$ on the tape in $< \alpha'$ many steps. So we can simply run this program until we arrive at a code c for a limit L -level that believes that P halts for the first time. Now, we can easily find out the desired real code for α in the code for $L_{\alpha+\omega}$ (by searching the coded structure for an element which it believes to be the halting time of P).

Proposition 5. *If $\beta < \alpha$ is exponentially closed and OTM-clockable and there is a total $\Sigma_1(L_\alpha)$ -function $f : \beta \rightarrow \alpha$ such that f is cofinal in α , then α is OTM-clockable.*

Proof. This works by the same argument as the “only admissibles start gaps”-theorem for ITTMs, see Welch [14]: Suppose for a contradiction that α starts an OTM-gap, but is not admissible.

Pick $\beta < \alpha$ OTM-clockable and $f : \beta \rightarrow \alpha$ such that f is $\Sigma_1(L_\alpha)$ and cofinal in α . Let B be an OTM-program that clocks β . By the last lemma, we can compute a real code for β in $< \beta' \leq \alpha$ many steps. Run the OTM that enumerates L . If β is exponentially closed, then we will have a code for L_β on the tape at time β . In addition, for each new L -level, check which ordinals receive f -images when evaluating the definition of f in that level. Determine the largest ordinal γ such that f is defined on γ . Whenever γ increases, say from γ_0 to γ_1 , let δ be such that $\gamma_0 + \delta = \gamma_1$ and run B for δ many steps. When B halts, all elements of β have images, so we have arrived at time α .

This suffices for an OTM-analogue of Welch’s theorem [14], Theorem 50:

Corollary 1. *If α starts a gap in the OTM-clockable ordinals, then α is admissible.*

Proof. As α starts an OTM-gap, it is exponentially closed.

If α is not admissible, there is a total cofinal $\Sigma_1(L_\alpha)$ -function $f : \beta \rightarrow \alpha$ with $\beta < \alpha$. Pick $\gamma \in (\beta, \alpha)$ OTM-clockable and large enough so that all parameters used in the definition of f are contained in L_γ . By Lemma 2, we can write a real code for L_γ , and thus for all of its elements in time $< \gamma' \leq \alpha$. We can now clock α as in Proposition 5, a contradiction.

4 Σ_2 -admissible Ordinals Are Not OTM-clockable

We now show that no Σ_2 -admissible ordinal α can be the halting time of a parameter-free OTM-computation. The proof is mostly an adaptation of argument in Hamkins and Lewis [9] for the non-clockability of admissible ordinals by ITTMs to the extra subtleties of OTMs.

Theorem 6. *No Σ_2 -admissible ordinal is OTM-clockable.*

Proof. We will show this for the case of a single-tape OTM for the sake of simplicity.

Let α be Σ_2 -admissible and assume for a contradiction that α is the halting time of the parameter-free OTM-program P . At time α , suppose that the read-write-head is at position ρ , the program is in state $s \in \omega$ and the head reads the symbol $z \in \{0, 1\}$. As one cannot move the head more than α many places to the right in α many steps, we have $\rho \leq \alpha$.

By the limit rules, z must have been the symbol on cell ρ cofinally often before time α and similarly, s must have been the program state cofinally often before time α . By recursively building an increasing ‘interleaving’ sequence of ordinals of both kinds, we see that the set S of times at which the program state was s and the symbol on ρ was z , we see that S is closed and unbounded in α .

We now distinguish three cases.

Case 1: $\rho < \alpha$ and the head position ρ was assumed cofinally often before time α .

Let β be the order type of the set of times at which ρ was the head position in the computation of P . We show that $\beta = \alpha$. If not, then $\beta < \alpha$; let $f : \beta \rightarrow \alpha$ be the function sending each $\iota < \beta$ to the ι th time at which ρ was the head position. Then f is Σ_1 over L_α and thus, by admissibility of α , $f[\beta]$ is bounded in α , contradicting the case assumption.

Let T be the set of times at which ρ was the head position. Then, by the limit rules and the case assumption, T is closed and unbounded in α .

As S and T are both Σ_1 over L_α and α is admissible, it follows that $S \cap T$ is also closed and unbounded in α . In particular, there is an element $\gamma < \alpha$ in $S \cap T$, i.e. there is a time $< \alpha$ at which the head was on position ρ , the cell ρ contained the symbol z and the inner state was s . But then, the situation that

prompted P to halt at time α was already given at time $\gamma < \alpha$, so P cannot have run up to time α , a contradiction.

Case 2: $\rho < \alpha$ and the head position ρ was assumed boundedly often before time α .

By the liminf rule for the determination of the head position at time α , this implies that, for every $\iota < \rho$, there is a time $\tau_\iota < \alpha$ such that, from time τ_ι on, the head never occupied a position $< \iota$. The function $f : \iota \mapsto \tau_\iota$ is Π_1 over L_α (we have $f(\iota) = \tau$ if and only if, for all $\beta > \tau$ and all partial P -computations of length β , the head position in the final state of the partial computation was $\geq \iota$) and thus in particular Σ_2 over L_α . By Σ_2 -admissibility of α and the case assumption $\rho < \alpha$, the set $f[\rho]$ must be bounded in α , say by $\gamma < \alpha$. But this implies that, after time γ , all head positions were $\geq \rho$. As ρ was assumed only boundedly often as the head position, this means that, from some time $< \alpha$ on, all head positions were actually $> \rho$. But then, ρ cannot be the inferior limit of the sequence of earlier head positions at time α , contradicting the case assumption that the head is on position ρ at time α .

Case 3: $\rho = \alpha$.

This implies that the head is on position ρ for the first time at time α , so that we must have $z = 0$, as there was no chance to write on the ρ th cell before time α .

Let S be the set of times $< \alpha$ at which some head position was assumed for the first time during the computation of P . By the same reason as above, this newly reached cell will contain 0 at that time. If we can show that there is such a time $< \alpha$ at which the inner state is also s , we are done, because that would mean that the halting situation at time α was already given at an earlier time, contradicting the assumption that P halts at time α .

As $\rho > 0$, there must be an ordinal $\tau < \alpha$ such that the head was never on position 0 after time τ (otherwise, the liminf rule would force the head to be on position 0 at time α). This means that the head was never moved to the left from a limit position after time τ . This further implies that, after time τ , for any position β that the head occupied, all later positions were at most finitely many positions to the left of β and hence that, if β is a limit ordinal, then it never occupied a position $< \beta$ afterwards. In particular, the sequence of limit positions that the head occupied after time τ is increasing. Note that the set of head positions occupied before time τ is bounded in α , say by ξ . Let S' be the set of elements $\iota > \tau$ of S such that, at time ι , the head occupied a limit position $> \xi$ for the first time. Then S' is a closed and unbounded subset of S .

As s is the program state at the limit time α , there must be $\gamma < \alpha$ such that, after time γ , the program state was never $< s$ and moreover, the program state s itself must have occurred cofinally often in α after that time.

But now, building an increasing ω -sequence of times starting with γ that alternately belong to S' and have the program state s , we see that its limit δ is $< \alpha$ and is a time at which the head was reading z and the state was s , we have the desired contradiction.

Since each case leads to a contradiction, our assumption on P must be false; as P was arbitrary, α is not a parameter-free OTM-halting time.

To see now that the theorem holds for any finite number of tapes, consider the argument below for each tape separately, note that we showed above that case 2 cannot occur while cases 1 and 3 both imply that, as far as the tape under consideration is concerned, the halting configuration occurred on a closed unbounded set of times before time α . Thus, one can again build an increasing ‘interleaving’ sequence of times at which each head read the same symbol as in the halting configuration and the inner state was the one in the halting configuration. The supremum of this sequence will be $< \alpha$, leading again to the contradiction that the program must have halted before α .

5 Existence of Admissible OTM-clockable Ordinals

We will now show that at least the first ω many admissible ordinals are OTM-clockable, thus answering the first question mentioned in the introduction positively. To this end, we need some preliminaries about Infinite Time Register Machines (ITRMs). ITRMs were introduced by Koepke in [11]; we sketch their architecture and refer to [11] for further information. An ITRM has finitely many registers, each of which stores one natural number. ITRM-programs are just programs for (classical) register machines. At successor times, an ITRM proceeds like a classical register machine. At limit levels, the active program line index and the register contents are defined to be the inferior limits of the sequences of earlier program line indices and respective register contents. When that limit is not finite in the case of a register content, the new content is defined to be 0, and one speaks of an ‘overflow’ of the respective register.

We recall Lemma 3 from [5]:

Theorem 7. *There are no gaps in the ITRM-clockable ordinals. That is, if $\alpha < \beta$ and β is ITRM-clockable, then α is ITRM-clockable.*

Combining this result with the main result of [11] on the computational strength of ITRMs, we obtain:

Lemma 3. *The ITRM-clockable ordinals are exactly those below ω_ω^{CK} . In particular, ω_n^{CK} is ITRM-clockable for all $n \in \omega$.*

Lemma 4. *Let α be ITRM-clockable. Then α is OTM-clockable.*

Proof. If $\alpha < \omega^2$, this is straightforward. Now let $\alpha \geq \omega^2$.

Let P be an ITRM-program that clocks α . We simulate P by an OTM-program that takes the same running time.

The simulation of ITRMs by OTMs here works like this: Use a tape for each register, have i many 1s, followed by 0s, on a tape to represent that the respective register contains $i \in \omega$; in addition, after a simulation step is finished, the head position on this tape represents the register content, i.e. it is at the first 0 on the tape.

For an ITTM, the simulation takes an extra ω many steps to halt because it takes time to detect an overflow. For an OTM, one can simply use one extra tape for each register, write 1 to their ω th positions at the start of the computation, move their heads along with the heads on the register simulating tapes and know that there is an overflow as soon as one of the heads on the extra tapes reads a 1.⁹ Since $\alpha \geq \omega^2$, the initial placement of 1s on the ω th tape positions does not affect the running time.

Corollary 2. *For every $n \in \omega$, ω_n^{CK} is OTM-clockable.*

This answers the first question mentioned above in the positive. By a relativization of the above argument, we can achieve the same for the second (i.e. whether gap starters for OTMs are something “better” than admissible):

Theorem 8. *Let $\alpha = \beta^+$ be a successor admissible. Then α does not start an OTM-clockable gap.*

Proof. Suppose for a contradiction that $\alpha = \beta^+$ starts an OTM-clockable gap. Then there is an OTM-clockable ordinal $\gamma \in (\beta, \alpha)$; pick one. By Lemma 2 above, a real code c for γ is OTM-writable in $< \alpha$ many steps. Suppose c has been written. Then $\omega_1^{CK, c} \geq \alpha$. Thus, α is ITRM-clockable in the oracle c . But now, α is OTM-clockable by first writing c and then ITRM-clocking α relative to c , contradicting the assumption that α starts a gap.

Corollary 3. *Every gap-starting ordinal for OTMs is an admissible limit of admissible ordinals.*

This allows a considerable strengthening of Corollary 2:

Corollary 4. *Every admissible ordinal up to the first admissible limit of admissible ordinals is OTM-clockable.*

6 Conclusion and Further Work

We showed that OTM-gaps are always started by limits of admissible ordinals and that, while admissible ordinals can be OTM-clockable, Σ_2 -admissible ordinals cannot. This provokes the following questions:

Question: Is every gap-starting ordinal for OTMs Σ_2 -admissible?

⁹ The fact that more tapes are needed the more registers P uses may be seen as a little defect. (Note that, by the results of [11], the halting times of ITRM-programs using n registers are bounded by ω_{n+1}^{CK} so that indeed arbitrarily large numbers of registers - and thus of tapes - are required to make the above construction work for all α_n^{CK} with $n \in \omega$.) It would certainly be nicer to have a uniform bound on the number of required tapes. And indeed, by a slightly refined argument using that only two of the used registers are ultimately relevant for the halting of an ITRM, such a bound can be obtained.

Question: What is the minimal gap-starting ordinal for OTMs? Does it coincide with first Σ_2 -admissible ordinal?

Further worthwhile topics include clockability for OTMs with a fixed ordinal parameter α and for other models of computability, like the “hypermachines” of Friedman and Welch (see [8]), α -ITTMs (see [7]) or α -ITRMs (see [4]), where the main question left open in [4] is to determine the supremum of the α -ITRM-clockable ordinals.

References

1. Barwise, J.: *Admissible Sets and Structures: An Approach to Definability Theory*. Springer, Berlin (1975)
2. Boolos, G., Putnam, H.: Degrees of unsolvability of constructible sets of integers. *J. Symb. Log.* **33**, 497–513 (1968)
3. Dimitriou, I. (ed.): *Bonn International Workshop on Ordinal Computability*. Report Hausdorff Centre for Mathematics, Bonn (2007). <http://www.math.uni-bonn.de/ag/logik/events/biwoc/index.html>
4. Carl, M.: *Taming Koepkes Zoo II: Register Machines*. Preprint (2020). [arXiv:1907.09513v4](https://arxiv.org/abs/1907.09513v4)
5. Carl, M., Fischbach, T., Koepke, P., Miller, R., Nasfi, M., Weckbecker, G.: The basic theory of infinite time register machines. *Arch. Math. Logic* **49**(2), 249–273 (2010)
6. Carl, M., Durand, B., Lafitte, G., Ouazzani, S.: Admissibles in gaps. In: Kari, J., Manea, F., Petre, I. (eds.) *CiE 2017*. LNCS, vol. 10307, pp. 175–186. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58741-7_18
7. Carl, M., Ouazzani, S., Welch, P.: Taming Koepke’s Zoo. In: Manea, F., Miller, R.G., Nowotka, D. (eds.) *CiE 2018*. LNCS, vol. 10936, pp. 126–135. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94418-0_13
8. Friedman, S., Welch, P.: Hypermachines. *J. Symb. Log.* **76**(2), 620–636 (2011)
9. Hamkins, L.: Infinite time turing machines. *J. Symb. Log.* **65**(2), 567–604 (2000)
10. Koepke, P.: Turing computations on ordinals. *Bull. Symb. Log.* **11**, 377–397 (2005)
11. Koepke, P.: Ordinal computability. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) *CiE 2009*. LNCS, vol. 5635, pp. 280–289. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03073-4_29
12. Seyfferth, B., Schlicht, P.: Tree representations via ordinal machines. *Computability* **1**(1), 45–57 (2012)
13. Seabold, D., Hamkins, J.: Infinite time turing machines with only one tape. *Math. Log. Quart.* **47**(2), 271–287 (1999)
14. Welch, P.: Characteristics of discrete transfinite time turing machine models: halting times, stabilization times, and normal form theorems. *Theor. Comput. Sci.* **410**, 426–442 (2009)