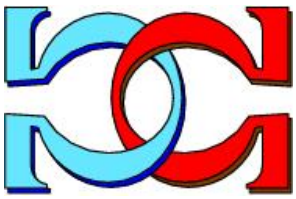
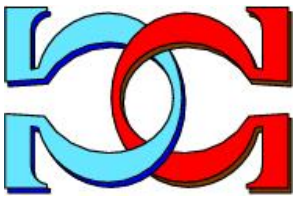
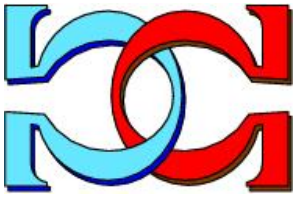
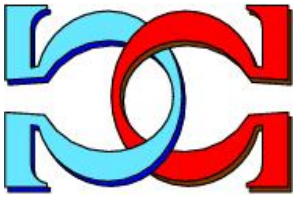


CDMTCS
Research
Report
Series

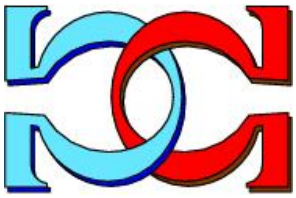




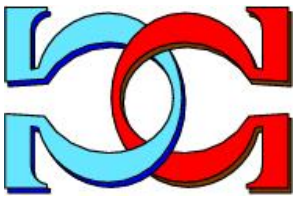
Turing machines with activations of transitions



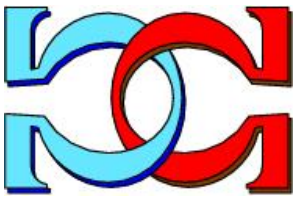
Rudolf Freund
Technische Universität Wien



Ludwig Staiger
Martin-Luther-Universität
Halle-Wittenberg



CDMTCS-534
May 2019



Centre for Discrete Mathematics and
Theoretical Computer Science

Turing machines with activations of transitions

Rudolf Freund[*]
TU Wien, Institut für Logic and Computation
Favoritenstraße 9–11, A-1040 Wien, Austria

and

Ludwig Staiger[†]
Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
von-Seckendorff-Platz 1, D–06099 Halle (Saale), Germany

Abstract

We consider Turing machines which perform infinite sequences of runs, where the (finite) number of computation steps is in each run is defined by the transitions activated in the preceding run when using a transition activated for this computation step in this preceding run. Acceptance is defined by the non-oscillating sequences of runs, i.e., for every tape cell n there exists a run $r(n)$ such that in all runs after run $r(n)$ the contents of tape cell n is not changed any more. In that way, with deterministic Turing machines we can characterize Π_3^0, whereas with non-deterministic Turing machines we obtain Σ_1^1.

[*]email: **rudi@emcc.at**
[†]email: **staiger@informatik.uni-halle.de**

Contents

1 Introduction

In [AFI18b], the idea of activating the rules to be applied in the next steps of a computation of a sequential grammar by the application of rules during the computation has been introduced. Even the concept of backwards applications of rules was discussed in different ways in [AFI18a]. Allowing the computations of the grammar to start from the beginning again having made n derivation steps and in the next "round" making $n+1$ derivation steps with the activations obtained in the preceding "round" allowed to "go beyond Turing" even with the underlying objects being multisets and not strings.

In this paper, we consider Turing machines as accepting devices on strings and ω-words, using the concept of activating transitions to be used in the next "round" of computations in an infinite sequence of finite runs, with the n-th run making exactly n computation steps. The acceptance condition is that the sequence of runs is non-oscillating, i.e., for every tape cell m there exists a run $r(m)$ such that in all runs after run $r(m)$ the contents of tape cell m is not changed any more. Instead of states, labels for the transitions are used; the same transition in δ may have multiple labels.

As a special variant, we consider Turing machines only going from left to right on the tape and infinite sequences of runs of such Turing machines activating the transitions to be used in the next run. Using backwards activations for the next run, i.e., activation of transitions in the previous step, allows for simulating transitions moving to the left, too. In total, even this special variant allows for the simulation of infinite runs of deterministic and non-deterministic Turing machines accepting with non-oscillating runs. As

we consider runs making n computation steps in a row, this requires at least one transition to be activated for every time step. This special variant also allows us to recover the contents of the tape cells visited so far even when starting every run from the initial tape.

If we allow runs where in a computation step nothing happens if no transition is activated, but still the run may continue until n computation steps have been carried out in this relaxed way, we may look at this as another model where in each run exactly the activated transitions are carried out, without having the need to do a complete run of n steps. On the other hand, every run has to start with the tape obtained after the preceding run.

Another interpretation of activating transitions in specific time steps is to consider the transitions assigned to the positions on the tape of the Turing machine, which especially for the variant of Turing machines only going from left to right on the tape turns out to be an equivalent concept. As the rewriting of one tape cell by the activated transition(s) does not depend on what happens to the other tape cells, all activated transitions could even be carried out in parallel – in both models, i.e., with the activations assigned to time steps OR to tape cells.

The rest of the paper is organized as follows: we first recall some definitions and well-known results, especially with respect to (ω-)Turing machines and the accepted families of (ω-)languages. In Section 3 we define the variants of *Turing machines with activations of transitions* as already briefly described above. The proofs for the characterizations of Π_3^0 and Σ_1^1 by deterministic respectively non-deterministic Turing machines with activations of transitions are given in Section 4. A short summary and discussion in Section 5 conclude the paper.

2 Definitions

We assume the reader to be familiar with the underlying notions and concepts from formal language theory, e.g., see [RS97] and [DP89].

2.1 Prerequisites

The set of integers is denoted by $\mathbb{Z}$, the set of positive integers by $\mathbb{N}$, and the set of non-negative integers by $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Given an alphabet V, a finite non-empty set of abstract symbols, the free monoid generated by V under the operation of concatenation is denoted by V^*. The elements of V^* are called strings, the empty string is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. For an arbitrary alphabet $V = \{a_1, \ldots, a_n\}$, the number of occurrences of

a symbol a_i in a string x is denoted by $|x|_{a_i}$, while the length of a string x is denoted by $|x| = \sum_{a_i \in V} |x|_{a_i}$. With respect to a specific order on the elements $a_1, \ldots, a_n$ of the alphabet V, the n-tuple $\left(|x|_{a_1}, \ldots, |x|_{a_n}\right\}$ is called the Parikh vector of $|x|$.

The set of one-sided infinite strings (mappings from $\mathbb{N}$ to V) is denoted by V^ω. Its subsets are referred to as ω-languages.

A finite multiset over an alphabet $V = \{a_1, \ldots, a_n\}$ is a mapping $f : V \to \mathbb{N}_0$ and can be represented by $\left\langle a_1^{f(a_1)}, \ldots, a_n^{f(a_n)} \right\rangle$ or by any string x for which $\left(|x|_{a_1}, \ldots, |x|_{a_n}\right) = (f(a_1), \ldots, f(a_n))$. The set of all multisets over V is denoted by V° or by $Ps(V^*)$.

The families of regular and recursively enumerable string languages are denoted by REG and RE, respectively, and $Co\text{-}RE$ denotes the family of complements of recursively enumerable string languages.

2.2 The Arithmetical Hierarchy

The Arithmetical Hierarchy (e.g., see [Bud06, Rog67] or [Sta97, Section 3.4]) is usually developed with the universal ($\forall$) and existential ($\exists$) quantifiers restricted to the integers. Levels in the Arithmetical Hierarchy are labeled as Σ_n^0 if they can be defined by expressions beginning with a sequence of at most n alternating quantifiers starting with $\exists$; levels are labeled as Π_n^0 if they can be defined by such expressions of at most n alternating quantifiers that start with $\forall$. Σ_0^0 and Π_0^0 are defined as having no quantifiers and are equivalent. Σ_1^0 and Π_1^0 only have the single quantifier $\exists$ and $\forall$, respectively. We only need to consider alternating pairs of the quantifiers $\forall$ and $\exists$ because two quantifiers of the same type occurring together are equivalent to a single quantifier. Moreover, we use the notion Δ_n^0, being defined as to represent $\Pi_n^0 \cap \Sigma_n^0$. In particular, the languages in $\Pi_1^0 \cap \Sigma_1^0$ and Σ_1^0 are the computable languages and the recursively enumerable languages, respectively.

More specifically, we say that a language $W \subseteq V^*$ belongs to the class Σ_n^0 if and only if $W = \{w \mid \exists a_1 \ldots \mathrm{Q}_m a_m : (a_1, \ldots, a_m, w) \in \mathbf{R}_W\}$, $m \leq n$, and an ω-language $F \subseteq V^\omega$ belongs to the class Σ_n^0 if and only if $F = \{\xi \mid \exists a_1 \ldots \mathrm{Q}_m a_m : (a_1, \ldots, a_{m-1}, \xi\lceil m) \in \mathbf{R}_F\}$, $m \leq n$, where $\xi\lceil\ell$ is the prefix of length ℓ of the infinite word ξ, $\mathbf{R}_W \subseteq \mathbb{N}^n \times V^*$ and $\mathbf{R}_F \subseteq \mathbb{N}^{n-1} \times V^*$ are computable relations and Q_i is one of the quantifiers $\forall$ or $\exists$. The class Π_n is defined dually by taking $\forall$ instead of $\exists$ as the first quantifier in all definitions.

If we also quantify over functions from $\mathbb{N}$ to $\mathbb{N}$, i.e., define sets of natural numbers by using second order logic formulas, we get the analytical hierarchy, starting with Σ_1^1 and Π_1^1. For example, an ω-language $F \subseteq V^\omega$ belongs to the class Σ_1^1 if and only if it can be accepted by a non-deterministic Turing

machine (by a complete non-oscillating run).

2.3 Turing Machines

A *Turing machine* (a *TM* for short) is a construct $M = (V, Z_0, B, T, Q, p_0, F, \delta)$ where

- V is the set of the tape alphabet, $\{Z_0, B\} \subset V \setminus T$;
- Z_0 is the left boundary marker of the tape;
- B is the blank symbol;
- $T \subset V$ is the terminal alphabet;
- Q is the set of states;
- $p_0 \in Q$ is the initial state;
- $F \subseteq Q$ is the set of final states;
- $\delta \subseteq Q \times V \times Q \times V \times \{L, R\}$ is the transition function.

The Turing machine M is called *deterministic* if for every pair $(p, X) \in Q \times V$ there is (at most) one triple $(q, Y, D) \in Q \times V \times \{L, R\}$ such that $(p, X, q, Y, D) \in Q \times V \times Q \times V \times \{L, R\}$.

For a finite string $w \in T^*$, acceptance usually is defined by M reaching a finite state when starting with $Z_0 w B^\omega$ on its tape in state p_0 and the read/write head being on Z_0. M making a transition using (p, X, q, Y, D) means that with being in state p and reading X on the current tape cell, M enters state q, rewrites X to Y, and moves its head into direction D, i.e., to the left if $D = L$ and to the right if $D = R$.

For infinite strings, one possible acceptance condition for an ω-word ξ is that it there exists a *non-oscillating run* on it, i.e., for every tape cell n there exists a time $t(n)$ such that for all times $t \geq t(n)$ tape cell n is not visited any more [CG78, FS01]. For finite strings w, we start with $Z_0 w B^\omega$, hence, we obtain a similar definition of acceptance for finite strings.

Remark 2.1 *We observe that every non-oscillating run is* complete*, too, i.e., every tape cell is visited at least once.*

In that way, with deterministic Turing machines we can characterize Π^0_3 (see [FS01]), whereas with non-deterministic Turing machines we obtain Σ^1_1, e.g., see [Sta97, SW78]:

Proposition 2.2 *The family of languages and ω-languages accepted by deterministic and non-deterministic Turing machines with non-oscillating runs equals* Π_3^0 *and* Σ_1^1*, respectively.*

3 Turing Machines with Activations of Transitions

In this section, we describe the variants of Turing machines with activations of transitions as already briefly discussed in Section 1.

A *Turing machine with activations of transitions* (an *ATM)* for short) is a construct

$$M_A = (V, Z_0, B, T, H, L, A_0)$$

where

- V the set of the tape alphabet, $\{Z_0, B\} \subset V \setminus T$;
- Z_0 is the left boundary marker of the tape;
- B is the blank symbol;
- $T \subset V$ the terminal alphabet;
- H is a set of labels;
- L is a set of labeled transitions of the form $r : (X, Y; A)$ with $r \in H$, $X, Y \in V$ and A being a finite subset of $H \times \mathbb{Z}$;
- A_0 is a finite subset of $H \times \mathbb{N}$ describing the initial activations of transitions.

A computation of the ATM M_A in general can be defined as follows:

- In any run, the ATM M_A only moves from left to right, starting on the left boundary marker Z_0 (this symbol must not be overwritten).
- We consider a sequence of runs; in every run, in any time step k, we try to apply one of the labeled transitions activated for this time step k; if none of those is applicable, the computation goes on as long as there is still a time step for which at least one transition is activated.
- The initial activations of transitions in A_0 of the form (r, t) indicate that for time step t of the first run the transition r is activated and thus may be used in this step; observe that different transitions may be activated for the same time step t.

- Executing the transition $r : (X, Y; A)$ in time step k means replacing the contents X of the k-th tape cell by Y (and moving the head one cell to the right for the next computation step of this run); an element (s, t) in A causes the activation of transition s in time step $k + t$ in the next run.

There are several ways to define the final result obtained by a sequence of – finite or infinite – runs of the ATM M_A:

finite runs generating finite strings the first run starts with the "empty tape", i.e., with Z_0B^ω; given some $w \in T^*$ we say that w is generated by M_A if and only if there exists a sequence of runs that ends, i.e., in some run no transition is applicable any more, and the final tape is of the form Z_0wB^ω;

finite runs accepting finite strings the first run starts with Z_0wB^ω for some $w \in T^*$; we say that w is accepted if and only if there exists a sequence of runs that ends, i.e., in a run no transition is applicable;

infinite runs generating ω-words the first run starts with Z_0B^ω; we consider infinite sequences of runs, probably having to fulfill specific conditions with respect to the applied transitions; we here restrict ourselves to the condition of non-oscillating sequences of runs, i.e., for every tape cell n there exists a run $r(n)$ such that in all runs after run $r(n)$ the contents of tape cell n is not changed any more and, if still transitions for time step n are activated, also those will not change any more in runs $r(m)$ with $m > n$;

infinite runs accepting ω-words the first run starts with $Z_0\xi$ for some $\xi \in T^\omega$; we say that ξ is accepted by M_A if and only if there exists a non-oscillating sequence of runs starting with the initial tape $Z_0\xi$;

infinite runs generating finite strings the first run starts with Z_0B^ω; a finite string $w \in T^*$ is said to be generated by M_A if and only if the ω-word wB^ω is generated by M_A;

infinite runs accepting finite strings the first run starts with Z_0wB^ω for some $w \in T^*$; we say that w is accepted by M_A if and only if the ω-word wB^ω is accepted by M_A.

Remark 3.1 *As for TMs we observe that every non-oscillating sequence of runs of an ATM is* complete*, too, i.e., every tape cell is visited at least once.*

We now consider a rather restricted model where in every run in each time step a transition must be executed, i.e., we consider sequences of runs where in the n-th run exactly n steps to the right on the tape are carried out. In step k, $1 \leq k \leq n$, one of the labeled transitions activated for time step k is executed; if none of those is applicable, the computation stops without yielding a result. Such a restricted ATM will be called *continuous* or a *cATM* for short.

Remark 3.2 *If we do not require the ATM to be continuous, we may simplify the final condition of non-oscillating sequences of runs: for every tape cell n there exists a run $r(n)$ such that in all runs after run $r(n)$, no transition is activated for tape cell n any more.*

An accepting ATM M_A is called *deterministic* if in every run of a non-oscillating sequence of runs of M_A never more than one transition is applicable in a time step of a run.

Remark 3.3 *We observe that – in contrast to the feature of a TM to be deterministic, which can syntactically be checked from the description of the TM – the feature of an ATM to be deterministic cannot be checked syntactically; as a dynamic feature of all possible – even infinite – computations it is not decidable.*

This can be verified in the following manner:

We start with a deterministic universal Turing machine U having the single final terminal state s_{fin}. We then construct another Turing machine U' which has an additional new final state s'_{fin}, and to any transition leading to s_{fin}, we add a twin transition leading to this new final state s'_{fin}. Thus the newly defined machine U' is also universal, but is non-deterministic, as for every accepting computation leading to a final state there is a second one leading to the other final state in the last step of the computation.

We first consider the following Turing machines M_k, $k \in \mathbb{N}$: M_k is obtained by simulating the program k on input k on U', that is, the M_k encode the halting problem *K. Then, in view of the invention of the twin final transitions, M_k is non-deterministic, but works in a deterministic way if and only if $k \notin K$ – a problem which is undecidable.*

For every Turing machine M_k, we now construct a non-deterministic ATM AM_k simulating M_k and having an accepting (i.e., non-oscillating) run on input n if and only if the input equals k. To guarantee that its run on input k becomes non-oscillating, AM_k also runs an intermediate subprogram moving the whole tape contents of M_k one cell to the right whenever having simulated a transition of M_k. This guarantees that for every $k \notin K$, AM_k has

a deterministic *non-oscillating run on input k while simulating the computation of M_k on k.*

For $k \in K$, AM_k can simulate the transitions of M_k in a deterministic way until M_k reaches one of its final states s_{fin} and s'_{fin}, yet this last transition has to be chosen in a non-deterministic way. After having reached one of the final states, AM_k just makes an infinite run to the right, thus accepting k by a non-oscillating run.

On the other hand, for any other input $\neq k$, after having checked that the input does not equal k, AM_k immediately starts a complete, but oscillating run, oscillating between going back to the beginning of the tape and visiting another tape cell to the right.

In sum, the ATMs AM_k accept exactly k by (sequences of) non-oscillating runs, but by construction AM_k is deterministic if and only if $k \notin K$, a question which is undecidable as already pointed out above.

As we shall see from the proofs elaborated in Section 4, cATMs have the advantage that we can start every run from the initial tape again, as all information about the contents of the tape cells visited so far can be taken over from the preceding run by using suitable activations for each tape cell n which corresponds to using suitable activations for each time step n. A cATM always starting with the initial tape is called an *initial cATM* or *icATM* for short.

4 Accepting Turing Machines with Activations of Transitions

In this section we consider accepting Turing Machines with activations of transitions and prove that the families of (ω-)languages accepted by deterministic and non-deterministic Turing Machines with activations of transitions accepting with non-oscillating sequences of runs coincides with Π_3^0 and Σ_1^1, respectively.

Before proving our general results we give an example showing how regular languages can be characterized by specific restricted variants of icATMs:

Example 4.1 *A deterministic finite automaton can be seen as a special variant of a deterministic Turing machine $M = (V, Z_0, B, T, Q, p_0, F, \delta)$ accepting finite strings $w \in T^*$, starting with $Z_0 w B^\omega$, having the following restrictions:*

- *$V = T \cup \{Z_0, B\}$, i.e., there are no additional tape symbols;*

- δ *only contains transitions with movements to the right of the form* (p,a,q,a,R), $p,q \in Q$, $a \in T \cup \{Z_0\}$;
- *for each pair* (p,a) *with* $p \in Q$ *and* $a \in T \cup \{Z_0\}$ *there is exactly one state* $q \in Q$ *such that* $(p,a,q,a,R) \in \delta$.

Due to the restricted forms of the transitions, any computation in a deterministic finite automaton halts if it reaches the first blank symbol B*; we say that* w *is accepted by* M *if and only if it is in a final state from* F *when halting. The language accepted by* M*, i.e., the set of all terminal words accepted by* M *in that way is denoted by* $L(M)$.

We now construct an icATM

$$M_A = (V, Z_0, B, T, H, L, A_0)$$

with $V = T \cup \{Z_0, B\}$ *simulating the transitions of* M *in such a way that in the* n*-th run the first* $n-1$ *symbols of the input string* w *are read; in the first step, always* Z_0 *is read. If after* $n+1$ *runs the input string has been parsed completely, the sequence of runs stops as no transition is defined when reading the first blank symbol* B *if* M *has reached this situation in a final state; otherwise, an infinite loop is started moving to the right over the infinite tail* B^ω *on the tape.*

- $A_0 = \{(< p_0, Z_0 >, 1)\}$,
  i.e., for the first step in the first run as the initial transition $< p_0, Z_0 >$ *is activated;*
  $< p_0, Z_0 > : (Z_0, Z_0, \{(< id(Z_0) >, 0)\} \cup \{(< q, a >, 1) \mid (p_0, Z_0, q) \in \delta, a \in T\} \cup \{(< r, B >, 1) \mid (p_0, Z_0, q) \in \delta, q \in Q \setminus F\})$,
  i.e., Z_0 *in the next run is just rewritten to itself, but for the second any valid transition according to the given relation* δ *is activated; as* M *is deterministic, for every terminal symbol* a *to be read exactly one transition will be available; if the end of the input string has been reached, the sequence of runs only continues if* M *has not reached a final state.*
- *Each transition of* M *in state* p *reading the terminal symbol* a *is simulated in* M_A *by*
  $< p, b > : (b, b, \{(id(b), 0)\} \cup \{(< q, a >, 1) \mid (p, b, q) \in \delta, a \in T\} \cup \{(< r, B >, 1) \mid (p, b, q) \in \delta, q \in Q \setminus F\})$,
  i.e., the underlying symbol of the input string on tape cell n *is rewritten to itself, but for the next run any valid transition according to the given relation* δ *is activated for tape cell* $n+1$*; as* M *is deterministic, for*

every terminal symbol a to be read at position $n+1$ exactly one transition will be available; if the end of the input string has been reached, the sequence of runs only continues if M has not reached a final state using the transition labeled by $<r,B>$.

- *For any $b \in T \cup \{B\}$, the identity transition is defined by $<id(b)>:(b,b,\{(<id(b)>,0)\})$.*
- *For doing the infinite loop to the right on the blank symbols, we use $<r,B>:(B,B,\{(<id(B)>,0),(<r,B>,1)\})$.*

Summing up, we have defined

$$
\begin{aligned}
H &= \{<p,b> \mid p \in Q, b \in T \cup \{B\}\} \cup \{<p_0,Z_0>,<r,B>\} \\
&\quad \cup \{<id(b)> \mid b \in T \cup \{B\}\}, \\
L &= \Big\{<p_0,Z_0>:\Big(Z_0,Z_0,\{(<id(Z_0)>,0)\} \cup \\
&\qquad \{(<q,a>,1) \mid (p_0,Z_0,q) \in \delta, a \in T\} \cup \\
&\qquad \{(<r,B>,1) \mid (p_0,Z_0,q) \in \delta, q \in Q \setminus F\}\Big)\Big\} \\
&\quad \cup \Big\{<p,b>:\Big(b,b,\{(id(b),0)\} \cup \\
&\qquad \{(<q,a>,1) \mid (p,b,q) \in \delta, a \in T\} \cup \\
&\qquad \{(<r,B>,1) \mid (p,b,q) \in \delta, q \in Q \setminus F\}\Big) \;\Big|\; p \in Q, b \in T \cup \{B\}\Big\} \\
&\quad \cup \{<id(b)>:(b,b,\{(<id(b)>,0)\}) \mid b \in T \cup \{B\}\} \\
&\quad \cup \{<r,B>:(B,B,\{(<id(B)>,0),(<r,B>,1)\})\}, \\
A_0 &= \{(<p_0,Z_0>,1)\}.
\end{aligned}
$$

Acceptance is obtained by finite sequences of runs, but by replacing $q \in Q \setminus F$ by $q \in F$ we obtain an icATM M'_A accepting by infinite sequences of runs instead.

We may also consider ATMs $\tilde{M}_A$ and $\tilde{M}'_A$ which only use the activations going to the right, i.e., we take the ATM

$$
\begin{aligned}
\tilde{M}_A &= (T \cup \{Z_0,B\},Z_0,B,T,\tilde{H},\tilde{L},\tilde{A}_0) \\
\tilde{H} &= \{<p,b> \mid p \in Q, b \in T \cup \{B\}\} \cup \{<p_0,Z_0>,<r,B>\}, \\
\tilde{L} &= \{<p_0,Z_0>:(Z_0,Z_0,\{(<q,a>,1) \mid (p_0,Z_0,q) \in \delta, a \in T\} \cup \\
&\qquad \{(<r,B>,1) \mid (p_0,Z_0,q) \in \delta, q \in Q \setminus F\})\} \\
&\quad \cup \{<p,b>:(b,b,\{(<q,a>,1) \mid (p,b,q) \in \delta, a \in T\} \cup \\
&\qquad \{(<r,B>,1) \mid (p,b,q) \in \delta, q \in Q \setminus F\}) \mid p \in Q, b \in T \cup \{B\}\} \\
&\quad \cup \{<r,B>:(B,B,\{(<r,B>,1)\})\}, \\
\tilde{A}_0 &= \{(<p_0,Z_0>,1)\}.
\end{aligned}
$$

$\tilde{M}_A$ accepts with finite sequences of runs, and $\tilde{M}'_A$, again obtained from $\tilde{M}_A$ by $q \in Q \setminus F$ by $q \in F$, accepts with infinite sequences of runs.

We now turn our attention to ATMs accepting with infinite runs. In contrast to the finite automata simulated in the example above, in the case of arbitrary TMs we also have to simulate transitions moving to the left.

Theorem 4.2 *The families of string languages and ω-languages accepted by non-deterministic ATMs and cATMs coincide with Σ_1^1.*

Proof. Given a non-deterministic ATM M_A, we can construct a non-deterministic TM M accepting, by non-oscillating runs, the same set of strings or ω-words as accepted by non-oscillating sequences of runs of M_A: M simulates the runs in the sequence one after the other, and stores the activations of transitions for the next run. The main technical detail is to guarantee that the run of M is non-oscillating if and only if the simulated sequences of runs of M_A is non-oscillating. This, for example, can be achieved by using several tracks on the tape cells, one for the initial input, one for the computations to be carried out, and one for returning to the leftmost position where during the simulated run an activated transition has changed the contents of the tape cell. In this way, only the positions corresponding to time steps changing the contents of an underlying tape cell are visited. Hence, for all runs of M_A not changing tape cell n any more, also M does not go left of tape cell n any more. We conclude that a run of M is non-oscillating if and only if it simulates a non-oscillating sequence of runs of M_A.

For the other inclusion, we start with a (non-deterministic) TM

$$M = (V, Z_0, B, T, Q, p_0, F, \delta)$$

and construct an equivalent cATM

$$M_A = (V, Z_0, B, T, H, L, A_0)$$

with

$$\begin{aligned} H \;=\; & \{< p, Z_0, q, Z_0, R > \mid (p, Z_0, q, Z_0, R) \in \delta\} \\ & \cup \{< p, X, q, Y, R > \mid (p, X, q, Y, R) \in \delta, X \neq Z_0\} \\ & \cup \{< p, X, q, Y, L > \mid (p, X, q, Y, L) \in \delta\} \\ & \cup \{< X, R >, < X, 0 >, < X, L > \mid X \in V\}, \end{aligned}$$

$$
\begin{aligned}
L \;=\;& \Big\{ < p,Z_0,q,Z_0,R >: \Big(Z_0,Z_0,\{(< Z_0,L >,0)\} \\
& \quad \cup \{(< q,X',q',Y',D >,1) \mid (q,X',q',Y',D) \in \delta, X',Y' \in V, q' \in Q\}\Big) \\
& \quad \Big| \, (p,Z_0,q,Z_0,R) \in \delta \Big\} \\
& \cup \Big\{ < p,X,q,Y,R >: \Big(X,Y,\{(< Y,L >,0)\} \cup \{(< U,L >,-1)\} \mid U \in V\} \\
& \quad \cup \{(< q,X',q',Y',D >,1) \mid (q,X',q',Y',D) \in \delta, X',Y' \in V, q' \in Q\}\Big) \\
& \quad \Big| \, (p,X,q,Y,R) \in \delta, X \neq Z_0 \Big\} \\
& \cup \Big\{ < p,X,q,Y,L >: \Big(X,Y,\{(< Y,R >,0)\} \cup \{(< U,R >,1)\} \mid U \in V\} \\
& \quad \cup \{(< q,X',q',Y',D >,-1)\} \mid (q,X',q',Y',D) \in \delta, X',Y' \in V, q' \in Q\}\Big) \\
& \quad \Big| \, (p,X,q,Y,L) \in \delta \Big\} \\
& \cup \Big\{ < X,R >: (X,X,\ \{(< Y,R >,1) \mid Y \in V\}) \mid X \in V \Big\} \\
& \cup \Big\{ < X,L >: (X,X,\{(< Y,L >,-1) \mid Y \in V\}) \mid X \in V \setminus \{Z_0\} \Big\} \\
& \cup \{< Z_0,L >: (Z_0,Z_0,\emptyset)\}, \\
A_0 \;=\;& \{(< p_0,Z_0,q,Z_0,R >,1) \mid (p_0,Z_0,q,Z_0,R) \in \delta\}.
\end{aligned}
$$

The simulation of a computation in M starts with the corresponding initial activation $< p_0,Z_0,q,Z_0,R >$, with M_A replacing Z_0 by Z_0 and activating some $< q,X',q',Y',D >$ for the second step of the second run of M_A, whereas for the first step in the second run the rule labeled by $< Z_0,L >$ is activated, only rewriting Z_0 by itself.

In general, as long as M only makes steps to the right, only rules labeled by $< p,X,q,Y,R >$ are activated in the n-th step, replacing X by Y and activating some $< q,X',q',Y',D >$ for step $n+1$ in the run $n+1$ of M_A, whereas for each preceding step in this run rules labeled $< U,L >$ are activated, which rewrite the underlying symbol U by itself, again activating rules labeled $< X,L >$ to the left; this sequence sending activations to the left ends at the position of Z_0, where no rule to the left is activated any more.

As soon as a rule labeled by $< p,X,q,Y,RL >$ is activated in the n-th step, which simulates a computation step of M moving the head to the left, also a sequence of activations to the right is initiated by $< Y,R >,0)$ for the n-th step of the run $n+1$ and $< U,R >$, for some $U \in V$, for step $n+1$ in this run. From that moment on, this sequence of activations to the right ends on a blank symbol in the last step, activating $< B,R >$ for the next step (position on the tape), too.

In general, a transition (p,X,q,Y,R) of M on position m of the tape to the right is simulated by M_A in step m with the rule labeled by $<p,X,q,Y,R>$, which replaces X by Y, activates rules labeled $<q,X',q',Y',D>$ for all possible next transitions of M on position $m+1$. The sequences of rules $<U,L>$ to the left as well as of rules $<U,R>$ to the right guarantee that in the next run of M_A all positions on the tape except $m+1$ rewrite the underlying symbols and again activate these identity rules to the left and to the right, whereas only positions m, $m+1$, and $m+2$ are controlled by the rule labeled by $<q,X',q',Y',D>$.

In the same way, a transition (p,X,q,Y,L) of M on position m of the tape to the right is simulated by M_A in step m with the rule labeled by $<p,X,q,Y,L>$, which replaces X by Y, activates rules labeled $<q,X',q',Y',D>$ for all possible next transitions of M on position $m-1$. The sequences of rules $<U,L>$ to the left as well as of rules $<U,R>$ to the right guarantee that in the next run of M_A all positions on the tape except $m-1$ rewrite the underlying symbols and again activate these identity rules to the left and to the right, whereas only positions m, $m-1$, and $m-2$ (for $m \geq 2$) are controlled by the rule labeled by $<q,X',q',Y',D>$.

With the simulations described above, M_A accepts a finite string w respectively any ω-word ξ by non-oscillating infinite sequences of runs if and only if M accepts w respectively ξ by a complete non-oscillating run, which proves the assertions claimed in the theorem.

We finally observe that the construction of the non-deterministic cATM M_A elaborated above immediately also yields an equivalent ATM

$$M'_A = (V,Z_0,B,T,H',L',A'_0)$$

where we can even omit the activation of the transitions not changing the contents of tape cells from the preceding run:

$$\begin{aligned}
H' &= \{<p,Z_0,q,Z_0,R> \mid (p,Z_0,q,Z_0,R) \in \delta\} \\
&\quad \cup \{<p,X,q,Y,R> \mid (p,X,q,Y,R) \in \delta, X \neq Z_0\} \\
&\quad \cup \{<p,X,q,Y,L> \mid (p,X,q,Y,L) \in \delta\} \\
L' &= \Big\{<p,X,q,Y,R>: \Big(X,Y,\{(<q,X',q',Y',D>,1) \\
&\qquad \mid (q,X',q',Y',D) \in \delta, X',Y' \in V, q' \in Q\}\Big) \,\Big|\, (p,X,q,Y,R) \in \delta\Big\} \\
&\quad \cup \Big\{<p,X,q,Y,L>: \Big(X,Y,\{(<q,X',q',Y',D>,-1) \\
&\qquad \mid (q,X',q',Y',D) \in \delta, X',Y' \in V, q' \in Q\}\Big) \,\Big|\, (p,X,q,Y,L) \in \delta\Big\} \\
A'_0 &= \{(<p_0,Z_0,q,Z_0,R>,1) \mid (p_0,Z_0,q,Z_0,R) \in \delta\}.
\end{aligned}$$

In the runs of M'_A, only the rules $<p,X,q,Y,R>$ or $<p,X,q,Y,L>$ activated at position m are executed, moving the activated position for the next

run to $m+1$ respectively $m-1$. Although the activations are not pushed one position to the right, i.e., initiating one step more, in succeeding runs of M'_A, a sequence of runs in M'_A still is a non-oscillating infinite one if and only if the simulated run of M is a non-oscillating infinite one. ❑

The construction elaborated in the proof of Theorem 4.2 also works in the deterministic case, guaranteeing that the simulations of accepting runs of a deterministic TM correspond to deterministic accepting sequences of runs of the corresponding cATM.

Corollary 4.3 *The families of string languages and ω-languages accepted by deterministic ATMs and cATMs coincide with Π_3^0.*

Proof. As in the proof of Theorem 4.2, we can simulate any accepting non-oscillating computation of the given deterministic TM M by sequences of deterministic runs of the simulating cATM M_A, which therefore, by definition is a deterministic cATM. Yet even the non-accepting sequences of runs of the cATM M_A simulating non-accepting runs of the given TM M are deterministic.

On the other hand, given a deterministic cATM M_A, we can construct a deterministic TM M accepting the same set of strings or ω-words: the accepting sequences of runs of the cATM M_A are deterministic by definition; hence, for all sequences of runs of the cATM M_A, M in addition has to check whether the determinism condition is fulfilled during the runs of M_A; as soon as a non-deterministic activation of transitions is detected by M, it continues with an oscillating computation thus rejecting the input; the second possibility of rejecting an input is by simulating an oscillating deterministic computation of M_A. ❑

Whether the results shown in Theorem 4.2 and Corollary 4.3 for non-deterministic respectively deterministic ATMs and cATMs also holds for icATMs, remains as an open question.

5 Conclusion

In this paper we have considered variants of *Turing machines with activations of transitions*. With each execution of a transition, transitions for the next run of the Turing machine are activated. Accepting with non-oscillating sequences of runs, such deterministic respectively non-deterministic Turing machines with activations of transitions characterize Π_3^0 and Σ_1^1, respectively.

Accepting with non-oscillating runs is the most powerful acceptance mode for languages and ω-languages when using ω-computations on Turing ma-

chines. The results of [CG78, SW78, Sta85] and [FS01], however, show that in the case of non-deterministic machines also the simple red-green machines of [vLW12] achieve the same power.

In the deterministic case, a comparison of results shows that there is a difference in the accepting power between Turing machines accepting ω-languages with arbitrary runs [CG78, Sta97, SW78], complete runs [EH93, Sta99], and complete non-oscillating runs [FS01]. The same is true for the language case where the red-green Turing machines of [vLW12] accept languages in Σ_2^0 and, in a more general setting, see [CS10, Sta85], languages up to the Boolean closure of Σ_2^0 are accepted.

References

[AFI18a] Artiom Alhazov, Rudolf Freund, and Sergiu Ivanov. P systems with activation and blocking of rules. In Susan Stepney and Sergey Verlan, editors, *Unconventional Computation and Natural Computation – 17th International Conference, UCNC 2018, Fontainebleau, France, June 25-29, 2018, Proceedings*, volume 10867 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Cham, 2018.

[AFI18b] Artiom Alhazov, Rudolf Freund, and Sergiu Ivanov. Sequential grammars with activation and blocking of rules. In Jérôme Durand-Lose and Sergey Verlan, editors, *Machines, Computations, and Universality – 8th International Conference, MCU 2018, Fontainebleau, France, June 28-30, 2018, Proceedings*, volume 10881 of *Lecture Notes in Computer Science*, pages 51–68. Springer, Cham, 2018.

[Bud06] P. Budnik. *What Is and What Will Be*. Mountain Math Software, 2006.

[CG78] Rina S. Cohen and Arie Y. Gold. ω-computations on Turing machines. *Theor. Comput. Sci.*, 6:1–23, 1978.

[CS10] Cristian S. Calude and Ludwig Staiger. A note on accelerated Turing machines. *Mathematical Structures in Computer Science*, 20:1011–1017, 2010.

[DP89] Jürgen Dassow and Gheorghe Păun. *Regulated rewriting in formal language theory*, volume 18 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1989.

[EH93] Joost Engelfriet and Hendrik Jan Hoogeboom. X-automata on ω-words. *Theor. Comput. Sci.*, 110(1):1–51, 1993.

[FS01] Rudolf Freund and Ludwig Staiger. Acceptance of ω-languages by communicating deterministic Turing machines. In Carlos Martín-Vide and Victor Mitrana, editors, *Where mathematics, computer science, linguistics and biology meet*, pages 115–125. Kluwer Academic Publishers, Dordrecht, 2001.

[Rog67] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw Hill, New York, 1967.

[RS97] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[Sta85] Ludwig Staiger. ω-computations on Turing machines and the accepted languages. In L. Lovász and E. Szemerédi, editors, *Theory of algorithms (Pécs, 1984)*, volume 44 of *Colloquia Mathematica Societatis János Bolyai*, pages 393–403. North-Holland Publishing Co., Amsterdam, 1985. Papers from the colloquium held in Pécs, July 16–21, 1984.

[Sta97] Ludwig Staiger. ω-languages. In Rozenberg and Salomaa [RS97], pages 339–387 of volume 3.

[Sta99] Ludwig Staiger. On the power of reading the whole infinite input tape. *Grammars*, 2(3):247–257, 1999. Workshop on Formal Languages and Automata (Iasi, 1999).

[SW78] Ludwig Staiger and Klaus Wagner. Rekursive Folgenmengen. I. *Z. Math. Logik Grundlagen Math.*, 24:523–538, 1978.

[vLW12] Jan van Leeuwen and Juraj Wiedermann. Computation as an unbounded process. *Theoretical Computer Science*, 429:202–212, 2012.