Acceptance of ω-Languages by Communicating Deterministic Turing Machines

Rudolf Freund*
Institut für Computersprachen,
Technische Universität Wien,
Karlsplatz 13
A-1040 Wien, Austria

Ludwig Staiger†
Institut für Informatik,
Martin-Luther-Universität,
Kurt-Mothes-Str. 1,
D-06120 Halle, Germany

Abstract

Using a specific model of communicating deterministic Turing machines we prove that the class of ω-languages accepted by deterministic Turing machines via complete non-oscillating (complete oscillating) runs on the input coincides with the class of Π_3-definable (Σ_3-definable, respectively) ω-languages.

*email: rudi@logic.at
†email: staiger@informatik.uni-halle.de

Contents

1 Introduction

In the models found in most papers in the literature (e.g., see the recent surveys [EH93], [St97] or [Th90, 97]), the acceptance of ω-languages by Turing machines is determined by the behaviour of the Turing machines on the input tape as well as by specific final state conditions well-known from the acceptance of ω-languages by finite automata.

Hence, in this paper we consider systems of communicating (deterministic) Turing machines consisting of three components – the input component (where we observe the behaviour on the input tape), the transition component, and the output component (which is checked for the final states condition). The purpose of this decomposition of a deterministic Turing machine in three components is to facilitate the analysis and synthesis of the acceptance behaviour of deterministic Turing machines on infinite words.

The following types of behaviour of deterministic Turing machines on the input tape are found in literature:

Type 1 The approach of [WS77,SW78] (cf. also [St86,97]) does not take into consideration the behaviour of the Turing machine on its input tape. Acceptance is based solely on the infinite sequence of internal states the machine runs through during its infinite computation.

Type 2 For X-automata Engelfriet and Hoogeboom [EH93] require that, in addition to the fulfillment of certain conditions on the infinite sequence of internal states in order to accept an input, the machine has to read the whole infinite input tape. Cohen and Gold [CG77] considered this same type for pushdown automata, too.

Type 3 The most complicated type of acceptance for Turing machines was introduced by Cohen and Gold [CG78,80]. They require – in addition to type 2 – that the machine scans every cell of the input tape only finitely many times. This behaviour is termed as having a complete non-oscillating run.

Type 3' The complementary type of acceptance with respect to type 3 is the acceptance by complete oscillating runs, i.e., almost every cell of the input tape has to be scanned infinitely often.

The investigations carried out in [StMM] for type 1 and type 2 show that taking into account the different possible behaviours of deterministic Turing machines indeed results in different classes of accepted ω-languages. In this paper we focus on type 3 and its complementary type 3'. The class of ω-languages accepted by deterministic Turing machines working with type 3 turns out to coincide with the class $\mathbf{\Pi}_3$ of the arithmetical hierarchy, regardless which of the usual final states conditions we take. We also show that deterministic Turing machines working with the complementary type 3' really accept exactly the ω-languages in the complementary class $\mathbf{\Sigma}_3$.

2 Preliminary definitions

We start with some basic notations. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of natural numbers. We consider the space X^ω of infinite strings (sequences, ω-words) on a finite alphabet of cardinality ≥ 2. By X^* we denote the set (monoid) of finite strings (words) on X, including the *empty* word e. For $w \in X^*$ and $b \in X^* \cup X^\omega$ let $w \cdot b$ be their *concatenation*. This concatenation product extends in an obvious way to subsets $W \subseteq X^*$ and $B \subseteq X^* \cup X^\omega$. As usual we denote subsets of X^* as languages and subsets of X^ω as ω-languages. Furthermore, $|w|$ is the *length* of the word $w \in X^*$. For an ω-word ξ and every $n \in \mathbb{N}$, ξ/n denotes the prefix of ξ of length n.

As usual we define Σ_1*-definable* ω*-languages* $E \subseteq X^\omega$ as

$$E = \{\xi \in X^\omega : \exists n \in \mathbb{N}\, (\xi/n \in W_E)\} \tag{1}$$

where $W_E \subseteq X^*$ is a recursive language, and we define Π_2*-definable* ω*-languages* $F \subseteq X^\omega$ as

$$F = \{\xi \in X^\omega : \forall i \in \mathbb{N}\ \exists n \in \mathbb{N}\, ((i, \xi/n) \in M_F)\} \tag{2}$$

where M_F is a recursive subset of $\mathbb{N} \times X^*$. Σ_2-definable and Π_1-definable ω-languages are defined accordingly.

In the sequel we will consider the following class $\mathfrak{P}$ of ω-languages defined as in condition 1 of lemma 3 in [StMM]:

$$\mathfrak{P} := \{F \subseteq X^\omega : \forall n \in \mathbb{N}\ \exists i \in \mathbb{N}\, ((i, \xi/n) \in M_F)\}\,, \tag{3}$$

where M_F is a recursive subset of $\mathbb{N} \times X^*$. Observe the difference of the ω-languages in $\mathfrak{P}$ to the definition of Π_2-definable ω-languages: For an ω-languages in $\mathfrak{P}$, the first quantifier is related to ξ and the second one to the other variable. The class $\mathfrak{P}$ is a subclass of $\mathbf{\Pi}_2$, the class of all Π_2-definable ω-languages, and it is closed under union and intersection (cf. [St86, StMM]).

Moreover, we define Π_3-definable ω-languages $E \subseteq X^\omega$ as

$$E = \{\xi \in X^\omega : \forall k \in \mathbb{N}\ \exists i \in \mathbb{N}\ \forall n \in \mathbb{N}\ ((k, i, \xi/n) \in M_E)\} \tag{4}$$

where M_E is a recursive subset of $\mathbb{N} \times \mathbb{N} \times X^*$, and we call an ω-language Σ_3-definable, if for some recursive subset $M_F \subseteq \mathbb{N} \times \mathbb{N} \times X^*$ we have

$$F = \{\xi \in X^\omega : \exists k \in \mathbb{N}\ \forall i \in \mathbb{N}\ \exists n \in \mathbb{N}\ ((k, i, \xi/n) \in M_F)\}\,. \tag{5}$$

The higher levels of the arithmetical hierarchy of ω-languages are defined accordingly, see e.g. [Ro67], [St86] or [St97].

We only mention the following well-known closure properties of the classes of the arithmetical hierarchy with respect to boolean operations.

Lemma 1 *Each one of the classes $\mathbf{\Sigma}_i$ and $\mathbf{\Pi}_i$ of Σ_i-definable or Π_i-definable subsets of X^ω, respectively, is closed under union and intersection, and we have $F \in \mathbf{\Sigma}_i$ if and only if $X^\omega \setminus F \in \mathbf{\Pi}_i$.*

3 The model of communicating deterministic Turing machines

For a detailed description of configurations (instantaneous descriptions) and the behaviour (computing process) of Turing machines the reader is referred to the literature (e.g. [Ro67]), for acceptance of ω-languages by Turing machines see [St97] or the papers mentioned in the introduction.

We will describe the construction of (the system of communicating) Turing machines and their behaviour only in an informal manner leaving details to the reader. As indicated in Figure 1 each of the machines of the system has a finite control, a finite number of working tapes and a tape as input where their head can only read. For the Transition and the Output machines the input tapes

are called communication tapes, because they serve also as output tapes for the Input and the Transition machine, respectively.

In addition to the communication from the upper machines to the lower ones via the communication tapes 1 and 2 the system has two feedback lines allowing some information flow from the lower to the upper machines. Formally, we define this as an input to the upper machine providing the current state of the lower machine.

Initially, the input is written on the input tape and the tape-cells of the other tapes are filled with blank symbols. Thus we may require the machines reading on communication tapes 1 and 2 not to read tape-cells where the preceding writing machines have not yet written.

Furthermore, one can easily verify that it is no loss of generality to require that the writing machines do not rewrite the contents of the tape-cells of the communication tapes.

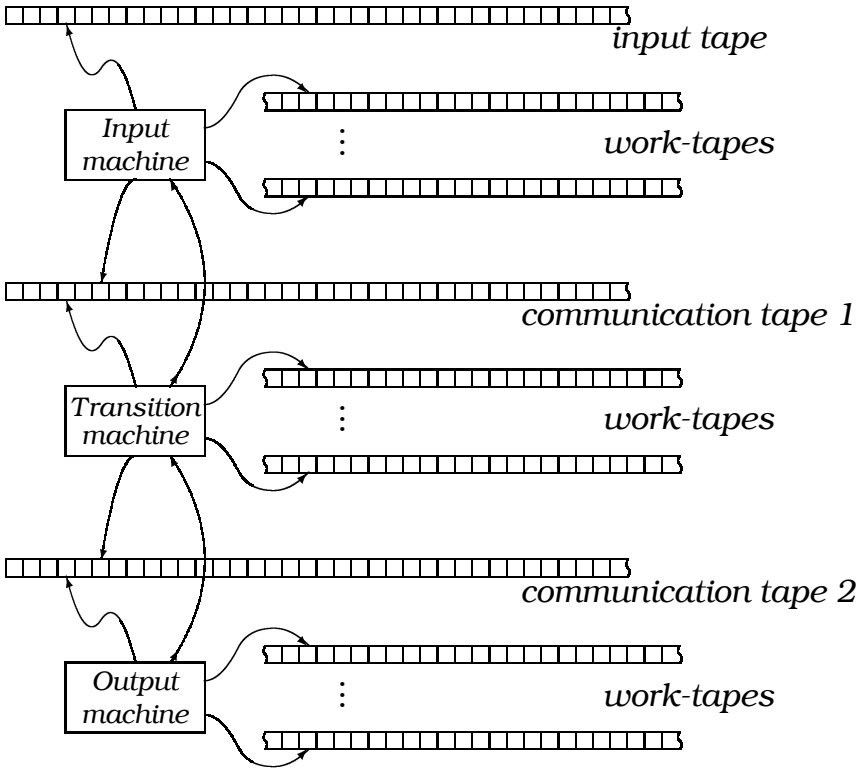

Figure 1: System of communicating Turing machines

It is obvious that the accepting power of a system of communi-

cating Turing machines cannot exceed the power of a single Turing machine. We introduced this splitting of one machine into three components of a system for reasons of better understanding the different items of the accepting process of a Turing machine:

① the behaviour on the input tape,

② the computation process, and

③ the accepting condition.

For machines accepting according to type 1 as defined above a similar splitting was considered in [SW78]. In case of type 1 the input machine is not necessary, and it turned out that the accepting process can easily be described by a composition of a transition machine and a finite automaton as output machine without using any feed-back.

The read-write behaviour on the communication tapes as described above makes it possible to use as machines so-called deterministic Turing transducers (tt-transducers) or strictly deterministic Turing machines as they were called in [Mi71], this type of machine having a read-only input tape, work-tapes and a write-only output tape on which the machine, for every input ω-word, produces (in a continuous or sequential way) an infinite output. These Turing transducer mappings or recursive operators on ω-words are described in more detail in [Mi71], or [St86] (see also [FS96] or [St97]).

We recall here only the following properties useful with respect to Π_i-definable and Σ_i-definable ω-languages.

A first property is the following one.

Lemma 2 *Let $\Psi : X^\omega \to X^\omega$ be a tt-mapping, and let $F \subseteq X^\omega$ be Σ_i-definable and $E \subseteq X^\omega$ be Π_i-definable. Then $\Psi^{-1}(F)$ and $\Psi^{-1}(E)$ are also Σ_i-definable and Π_i-definable, respectively.*

In order to derive the next one we introduce a Cantor triple function, that is, a computable bijection $\langle \cdot, \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Let $\boldsymbol{l} : \mathbb{N} \to \mathbb{N}$, $\boldsymbol{m} : \mathbb{N} \to \mathbb{N}$ and $\boldsymbol{r} : \mathbb{N} \to \mathbb{N}$ be the left, middle and right inverse functions of $\langle \cdot, \cdot, \cdot \rangle$, respectively, defined via the equations

$$\boldsymbol{l}(\langle n, m, k \rangle) := n, \quad \boldsymbol{m}(\langle n, m, k \rangle) := m \quad \text{and} \quad \boldsymbol{r}(\langle n, m, k \rangle) := k \ .$$

For the sake of convenience let $\langle \cdot, \cdot, \cdot \rangle$ be monotone in all of its arguments. Then $\boldsymbol{l}(l), \boldsymbol{m}(l), \boldsymbol{r}(l) \leq l$.

We consider the following ω-languages:

$$P_3 = \{\xi \in \{0,1\}^\omega : \forall n \exists m \forall k \; \xi(\langle n,m,k\rangle) = 1\} \text{ , and} \tag{6}$$

$$S_3 = \{\xi \in \{0,1\}^\omega : \exists n \forall m \exists k \; \xi(\langle n,m,k\rangle) = 1\} \; . \tag{7}$$

In [SW78] it was shown that P_3 is a Π_3-complete ω-language and S_3 is a Σ_3-complete ω-language, that is, the following holds true.

Lemma 3 *For every Π_3-definable ω-language $F \subseteq X^\omega$ there is a tt-mapping $\Phi : X^\omega \to X^\omega$ such that $\Phi(L) \subseteq P_3$ and $\Phi(X^\omega \setminus L) \subseteq \{0,1\}^\omega \setminus P_3$, and for every Σ_3-definable ω-language $E \subseteq X^\omega$ there is a tt-mapping $\Psi : X^\omega \to X^\omega$ such that $\Psi(L) \subseteq S_3$ and $\Psi(X^\omega \setminus L) \subseteq \{0,1\}^\omega \setminus S_3$.*

For more details on Π_i- and Σ_i-complete ω-languages see [SW78] and also [St86].

In Theorem 3.3 of [EH93] the existence of a similar splitting of X-automata into ω-preserving X-transducers and finite automata was proved. Here the notion of ω-preserving X-transducer differs a little bit from the above tt-mapping, because it takes into account the Type 2-acceptance generally assumed in [EH93].

4 Classes of ω-languages accepted by communicating deterministic Turing machines

In this section we investigate the classes of ω-languages accepted by communicating deterministic Turing machines via complete non-oscillating runs and via complete oscillating runs, respectively.

4.1 Acceptance without final states conditions

In this subsection we are interested in the question which ω-languages can be accepted by (communicating) deterministic Turing machines when we disregard the internal behaviour (states) and consider only the reading behaviour of the head on the input tape.

For a (communicating) deterministic Turing machine $\mathfrak{M}$, the ω-language accepted by $\mathfrak{M}$ via complete non-oscillating runs and

complete oscillating runs, respectively, is denoted by $L_{cno}(\mathfrak{M})$ and $L_{co}(\mathfrak{M})$, respectively. The corresponding classes of ω-languages accepted by deterministic Turing machines are denoted by DT_{cno} and DT_{co}.

For any $\mathfrak{M}$, $L_{cno}(\mathfrak{M}) \cup L_{co}(\mathfrak{M})$ is the type 2-accepted ω-language of $\mathfrak{M}$ and therefore in $\mathfrak{P}$ (cf. [StMM]). The complement of $L_{cno}(\mathfrak{M}) \cup L_{co}(\mathfrak{M})$ consists of those ω-words where $\mathfrak{M}$ does not read the whole input.

In fact, it is easy to see that in the case of communicating deterministic Turing machines accepting an ω-word without taking into account the sequence of internal states of the output component, this component can be omitted in our model without changing the power of the model of communicating deterministic Turing machines for accepting ω-languages. As we shall show at the end of this section, the classes of ω-languages accepted by (communicating) deterministic Turing machines via complete non-oscillating (complete oscillating, respectively) runs do not change when we add one of the well-known final states conditions.

We now are going to show that $DT_{cno} = \mathbf{\Pi}_3$.

Lemma 4 $DT_{cno} \subseteq \mathbf{\Pi}_3$.

Proof. Let L be an ω-language accepted by a communicating deterministic Turing machine via type 3; this means that $\xi \in L$ if and only if for every position j on the input tape there is a (smallest) time instance t such that the cell at position j is scanned at time t and for all time instances t' with $t' > t$ the cell at position j is never scanned again. Obviously, this condition is $\mathit{\Pi}_3$-definable.❑

Lemma 5 $P_3 \in DT_{cno}$.

Proof. The actions of a deterministic Turing machine $\mathfrak{M}$ accepting P_3 via complete non-oscillating runs can be sketched as follows:

$\mathfrak{M}$ successively for $l = 0, 1, 2, ...$ computes n, m, k such that $\langle n, m, k\rangle = l$ and reads the contents $c(l)$ of the cell at position l on the input tape.

① In case we have $k = m = 0$ and $c(l) = 1$, $\mathfrak{M}$ adds the pair $(n, 0)$ to the list l_H of hypotheses.

② If $(m,k) \neq (0,0)$ and $c(l) = 1$, $\mathfrak{M}$ proceeds to $l+1$.

③ In case we have $k = m = 0$ and $c(l) = 0$, $\mathfrak{M}$ adds the pair $(n,1)$ to the list l_H of hypotheses, because $(n,1)$ is the first possible hypothesis for fulfilling the required condition. Observe that by the monotonicity of the Cantor numbering we use, $\langle n,1,0\rangle > l$.

④ If $(m,k) \neq (0,0)$ and $c(l) = 0$, $\mathfrak{M}$ has to check whether the pair (n,m) is occurring in the list l_H or not.

(n,m) **is *not* occurring in the list** l_H**:**
$c(l) = 0$ is consistent with the current hypothesis for n, therefore $\mathfrak{M}$ proceeds to $l+1$.

(n,m) **is occurring in the list** l_H**:**
$c(l) = 0$ contradicts the current hypothesis for n, therefore $\mathfrak{M}$ has to replace (n,m) in the list l_H by a new hypothesis (n,m') for n, where m' is computed as follows: For $m' = m+1, m+2, \ldots$ the machine $\mathfrak{M}$ successively checks the condition $c(\langle n,m',k'\rangle) = 1$ for all k' such that $\langle n,m',k'\rangle < l$. If such a minimal number m' with $\langle n,m',0\rangle < l$ can be found, we replace the pair (n,m) in the list l_H by the pair (n,m'), otherwise we compute the minimal number m' such that $\langle n,m',0\rangle > l$ and take this number m' for our new hypothesis for n. Observe that due to the monotonicity of the Cantor numbering, during this procedure $\mathfrak{M}$ never has to read a cell at a position $< \langle n,0,0\rangle$.

Finally, $\mathfrak{M}$ moves its input head back to position $\langle n,0,0\rangle$ and continues with reading the contents of $\langle n,0,0\rangle$ again. This oscillation guarantees that a complete non-oscillating run is possible on the infinite input sequence if and only if it fulfills the defining condition of P_3. ❑

Lemma 6 $DT_{cno} \supseteq \mathbf{\Pi}_3$.

Proof. According to lemma 3, for any ω-language $L \in \mathbf{\Pi}_3$ there exists a deterministic Turing machine $\mathfrak{M}_L$ mapping each ω-word ξ on its input tape to an ω-word $\Psi_L(\xi)$ on its output tape in such a way that no cell on the output tape is rewritten any more.

Moreover, let $\mathfrak{M}_{P_3}$ be a deterministic Turing machine accepting P_3 via complete non-oscillating runs as constructed in lemma 5. Hence, we can construct a communicating deterministic Turing machine $\mathfrak{M}$ accepting L via complete non-oscillating runs in the following way: In essence, $\mathfrak{M}_L$ forms the main part of the input component of $\mathfrak{M}$, whereas $\mathfrak{M}_{P_3}$ builds up the main part of the transition component of $\mathfrak{M}_{P_3}$. $\mathfrak{M}$ simulates $\mathfrak{M}_{P_3}$ and $\mathfrak{M}_L$ step by step in parallel with the constraint that $\mathfrak{M}_{P_3}$ can only read cells $\mathfrak{M}_L$ has already written and via the communication channel to the input component $\mathfrak{M}_L$ has to perform the following additional actions after each computation step simulating one step of $\mathfrak{M}_{P_3}$: $\mathfrak{M}$ moves the head on the input tape to the position currently under consideration on the communication tape between $\mathfrak{M}_L$ and $\mathfrak{M}_{P_3}$ and then moves back the head of the input tape to the cell actually to be read by $\mathfrak{M}_L$. For fulfilling this intermediate task, $\mathfrak{M}$ has to store (for example, on additional working tapes) the actual positions of $\mathfrak{M}$ on the input tape as well as on the communication tape. This intermediate procedure guarantees that $\mathfrak{M}_{P_3}$ has a complete non-oscillating run on the input ξ if and only if $\mathfrak{M}_{P_3}$ has a complete non-oscillating run on $\Psi_L(\xi)$. ❑

Combining lemmas 4 and 6, we obtain

Theorem 1 $DT_{cno} = \mathbf{\Pi}_3$.

Lemma 7 *For any ω-language L with $L \in DT_{cno}$ there exists a deterministic Turing machine $\mathfrak{M}'$ such that every run of $\mathfrak{M}'$ is complete and $L = L_{cno}(\mathfrak{M}')$.*

Proof. Let $\mathfrak{M}$ be a (communicating) deterministic Turing machine accepting L. We now construct a (communicating) deterministic Turing machine $\mathfrak{M}'$ such that every run of $\mathfrak{M}'$ is complete and $L_{cno}(\mathfrak{M}) = L_{cno}(\mathfrak{M}')$. The (communicating) deterministic Turing machine $\mathfrak{M}'$ simulates the (communicating) deterministic Turing machine $\mathfrak{M}$ step by step, yet between two such simulation steps $\mathfrak{M}'$ takes the following intermediate actions: $\mathfrak{M}'$ scans the next cell on the input tape not scanned so far by $\mathfrak{M}'$ itself and then moves back the head of the input tape to the cell actually under consideration by $\mathfrak{M}$. For fulfilling this intermediate task, $\mathfrak{M}'$ has to store (for example, on additional working tapes) the actual position

of $\mathfrak{M}$ on the input tape as well as the last position scanned so far by $\mathfrak{M}'$ itself. In total, this algorithm guarantees that each run of $\mathfrak{M}'$ is complete, but still $L_{cno}(\mathfrak{M}) = L_{cno}(\mathfrak{M}')$. ❑

From the preceding lemma 7, theorem 1 and lemma 1, we immediately infer the following results:

Corollary 1 *For any ω-language L with $L \subseteq X^\omega$ we have:*

1. $L \in DT_{cno}$ *implies* $X^\omega \setminus L \in DT_{co}$.
2. $\mathbf{\Sigma}_3 \subseteq DT_{co}$.

Theorem 2 $DT_{co} = \mathbf{\Sigma}_3$.

Proof. Due to Corollary 1, we only have to prove that $\mathbf{\Pi}_3 \supseteq DT_{co}$. As already mentioned on page 9, for every deterministic Turing machine $\mathfrak{M}$, $L := L_{cno}(\mathfrak{M}) \cup L_{co}(\mathfrak{M}) \in \mathfrak{P} \subseteq \mathbf{\Pi}_2 \subseteq \mathbf{\Sigma}_3$. By definition, $L_{cno}(\mathfrak{M}) \cap L_{co}(\mathfrak{M}) = \emptyset$. As $X^\omega \setminus L_{cno}(\mathfrak{M}) \in \mathbf{\Sigma}_3$ and $\mathbf{\Sigma}_3$ is closed under intersection, $L_{co}(\mathfrak{M}) = L \cap (X^\omega \setminus L_{cno}(\mathfrak{M})) \in \mathbf{\Sigma}_3$. ❑

4.2 State-acceptance conditions for (communicating) deterministic Turing machines

In this part we informally consider the possible influence of the six conditions usually found in literature (cf. [EH93], [St97], [SW78]) as additional conditions for determining the ω-language accepted by a (communicating) deterministic Turing machine. The most powerful final state condition is Muller's condition requiring that the set of states occurring infinitely often in a (complete) run coincides with one of the given final states sets.

In terms of the arithmetical hierarchy, Muller's condition is a Boolean combination of $\forall\exists$-conditions which have to be combined in a conjunctive way with the input condition, which itself is a $\forall\exists\forall$-condition. According to the Tarski-Kuratowski-algorithm, in total we obtain not more than a $\forall\exists\forall$-condition. Hence we can formulate the following corollary:

Corollary 2 *The class of ω-languages accepted by (communicating) deterministic Turing machines via complete non-oscillating (oscillating, respectively) runs fulfilling Muller's condition for the corresponding state sequences is the class of Π_3-definable (Σ_3-definable) ω-languages.*

5 Conclusion

The results shown in [StMM] and in this paper reveal the importance of considering the behaviour of a deterministic Turing machine when reading its input tape for determining the accepted ω-language. As we have proved in this paper, deterministic Turing machines with type 3-acceptance (as well as with the complementary type 3'-acceptance) are strictly more powerful than deterministic Turing machines using type 2-acceptance, which in turn are more powerful than deterministic Turing machines with type 1-acceptance as was shown in [StMM]. In contrast to types 1 and 2, in the case of type 3- and type 3'-acceptance, the resulting classes of ω-languages do not depend on final states conditions. Moreover, separating the behaviour on the input tape from the final state condition in our model of communicating deterministic Turing machines allowed for a transparent analysis of the accepting power of complete non-oscillating runs of deterministic Turing machines.

References

[CG77] R.S. Cohen and A.Y. Gold, Theory of ω-languages I: Characterizations of ω-context-free languages, and II: A study of various models of ω-type generation and recognition, J. Comput. System Sci. 15 (1977)2, 169–184 and 185–208.

[CG78] R.S. Cohen and A.Y. Gold, ω-computations on Turing machines, Theoret. Comput. Sci. 6 (1978), 1–23.

[CG80] R.S. Cohen and A.Y. Gold, On the complexity of ω-type Turing acceptors, Theoret. Comput. Sci. 10 (1980), 249–272.

[EH93] J.Engelfriet and H.J. Hoogeboom, X-automata on ω-words. Theoret. Comput. Sci. 110 (1993) 1, 1–51.

[FS96] R. Freund and L. Staiger, Numbers defined by Turing machines, Collegium Logicum (Annals of the Kurt-Gödel-Society), Vol. 2, Springer-Verlag, Wien 1996, 118–137.

[La69] L.H. Landweber, Decision problems for ω-automata, Math. Syst. Theory 3 (1969) 4, 376–384.

[Mi71] M. L. Minsky: *Berechnung: Endliche und unendliche Maschinen* (Verlag Berliner Union, Stuttgart, 1971).

[Ro67] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw Hill, New York 1967.

[St86] L. Staiger, Hierarchies of recursive ω-languages. J. Inform. Process. Cybernetics EIK 22 (1986) 5/6, 219–241.

[St97] L. Staiger, ω-languages. in: *Handbook of Formal Languages*, Vol. 3, G. Rozenberg and A. Salomaa (Eds.), Springer-Verlag, Berlin. 339–387.

[StMM] L. Staiger, On the power of reading the whole infinite input tape, Grammars, *to appear.*

[SW78] L. Staiger und K. Wagner, Rekursive Folgenmengen I. Zeitschr. Math. Logik u. Grundl. Mathematik 24 (1978) 6, 523–538.

[Th90] W. Thomas, Automata on infinite objects, in: *Handbook of Theoretical Computer Science*, Vol. B, J. Van Leeuwen (Ed.), Elsevier, Amsterdam, 1990. 133–191.

[Th97] W. Thomas, Languages, automata, and logic, in: *Handbook of Formal Languages*, Vol. 3, G. Rozenberg and A. Salomaa (Eds.), Springer-Verlag, Berlin. 389–455.

[WS77] K. Wagner and L. Staiger, Recursive ω-languages. in: *Fundamentals of Computation Theory*, M. Karpiński (Ed.), Lect. Notes Comput. Sci. 56, Springer Verlag, Berlin 1977, 532–537.