RAIRO-Theor. Inf. Appl. 60 (2026) 2
https://doi.org/10.1051/ita/2026004

**RAIRO - Theoretical Informatics and Applications**
www.rairo-ita.org

# QUASI-DETERMINISTIC FINITE ONE-HEAD AND TWO-HEAD AUTOMATA

Benedek Nagy[1,2,*]

**Abstract.** Determinism is a central concept of computations. In some computational models, *e.g.*, (traditional one-head) finite automata, deterministic and nondeterministic variants characterize the same language class. Considering other models, *e.g.*, the 2-head finite automata known as sensing $5' \to 3'$ Watson–Crick automata, the deterministic variants are weaker than the nondeterministic ones, *i.e.*, the former variants characterize a proper subset of the language class accepted by the latter ones. Watson–Crick finite automata emerged as a model of DNA computing: they work on a Watson–Crick tape, on double stranded DNA molecules, thus they have two heads, one for each strand. The heads of sensing $5' \to 3'$ Watson–Crick automata start from the two extremes of the input, read the input in opposite direction and the computation finishes when the heads meet. Variants based on restrictions on the set of states (*e.g.*, stateless) and on the (head movements in) transitions (*e.g.*, 1-limited), as well as, nondeterministic, deterministic and the recently investigated state-deterministic variants have been studied. In this paper a new concept, quasi-determinism is investigated, that is, in each configuration of a computation (if it is not finished yet), the next state is uniquely determined although the next configuration may not be, in case various transitions are enabled at the same time. We prove that this new concept is a generalisation of both of usual determinism and state-determinism both for finite one-head and two-head automata. More precisely, the class of quasi-deterministic sensing $5' \to 3'$ Watson–Crick automata is a superclass of both of the mentioned other classes of sensing $5' \to 3'$ Watson–Crick automata. The new sublinear class of languages characterized by the new model is not closed under the regular operations, under intersection, nor under complement. Hierarchy of language classes accepted by various subclasses of quasi-deterministic sensing $5' \to 3'$ Watson–Crick automata and also some other well-known classes is presented.

**Mathematics Subject Classification.** 68Q45.

Received January 28, 2023. Accepted January 8, 2026.

## 1. Introduction

Finite automata and formal languages belong to the bases of computer science, both in theory and practice [1, 2]. DNA computing [3–5] emerged in the last decade of the 20th century and it constitutes some of the most known natural computing paradigms. Watson–Crick automata belong to the DNA computing models initiated

*Keywords and phrases:* Watson–Crick automata, $5' \to 3'$ WK automata, finite automata, 2-head automata, linear context-free languages, hierarchy, determinism, ambiguity, sublinear language classes.

[1] Department of Mathematics, Faculty of Arts and Sciences, Eastern Mediterranean University, Famagusta, North Cyprus, via Mersin-10, Türkiye.
[2] Institute of Mathematics and Informatics, Eszterházy Károly Catholic University, Eger, Hungary.

* Corresponding author: nbenedek.inf@gmail.com

by the structure of the DNA. In fact, they are finite state machines that are working on double stranded tape (like a DNA molecule) and thus, these automata have two heads. From biological motivation the heads may read strings (on their strands) in a computation step. The symbols located in the same position of the double-stranded tapes are related by the Watson–Crick complementarity relation. Watson–Crick automata are abbreviated as WK automata, for short, by the first and last letters of the names of the Nobel prize winners. There are various models of WK-automata, in the original models [5–9], the two heads belonging to the two strands are starting from the same end of the input molecule and moving to the same physical direction. However, the two strands of a DNA molecule have opposite $5' \to 3'$ orientation, *i.e.*, their biochemical direction is opposite. The reverse and the $5' \to 3'$ variants of WK automata are more realistic in the sense, that both heads use the same biochemical (that is opposite physical) direction [7, 10–12]. Some variants of the reverse Watson–Crick automaton with sensing parameter which tells whether the upper and the lower heads are within a fixed distance (or meet at the same position) are discussed in [12–15]. In [16, 17], it was shown that the sensing $5' \to 3'$ WK automata with sensing parameter are equivalent to a newer model without the sensing parameter: both models characterise the class LIN of linear context-free languages.

There are three concepts of determinism introduced to WK automata [18–20], due to the possible properties of the used Watson–Crick complementary relations, each of those concepts differs from the other two. According to the definition, in a *weakly deterministic* WK-automaton in any configuration that could occur in any computation, there is at most one way to continue the computation. From our point of view, this concept seems to be the closest to the original idea of determinism. However, in traditional WK automata, other related concepts may also be used. The second concept, the *deterministic* WK-automata, have a stronger constraint than the former weakly deterministic ones, namely: (the formal description will be explained in the next section) if there are two transitions from the same state, *i.e.*, the automaton has both $p_1 \in \delta(q, u_1, v_1)$ and $p_2 \in \delta(q, u_2, v_2)$ with some states $q, p_1, p_2$ and strings $u_1, u_2, v_1, v_2$, then none of $u_1$ and $u_2$ are prefix of each other and none of $v_1$ and $v_2$ are prefix of each other. Finally, the third concept is defined as follows. A WK-automaton is *strongly deterministic* if it is deterministic (as above), moreover the identity is used as Watson–Crick complementarity relation.

In sensing $5' \to 3'$ WK automata the computation on the input finishes when the heads meet, thus each position of the Watson–Crick tape is read by only one of the heads. Due to this fact, the complementarity relation does not have any importance in these models. On the other hand, the complementarity can also be excluded from the classical models [21], however, as we have mentioned it may play an important role in defining various types of determinism and also from the complexity point of view. Deterministic variants of $5' \to 3'$ WK automata are described in detail in [22, 23], they accept the sublinear class of languages 2detLIN. Also in this case, the earlier model using sensing parameter was proven to be equivalent to the new model excluding the sensing parameter. As the sensing $5' \to 3'$ WK automata are 2-head automata models, and they accept the linear languages, the class 2detLIN is named in this way as it is the deterministic counterpart of the class LIN of linear languages by this 2-head model. Moreover, 2detLIN and detLIN (the class of linear languages accepted by deterministic one-turn pushdown automata) are incomparable under set-theoretic inclusion.

Recently, *state-determinism* has been investigated for finite and sensing $5' \to 3'$ WK automata [24]. Here, we present a new concept, the *quasi-determinism*. In these new models, the state of the next configuration is determined, but the next configuration may not be. In the state-deterministic automata, the state of the next configuration depends only on the actual state, while in the quasi-deterministic automata (as we will define it formally), it may also depend on the part of the input being read.

WK automata have restricted variants based on restrictions on the states and/or on the transitions. These restrictions are somewhat orthogonal to the concepts of determinism, therefore it is also interesting to see the new concept of determinism combined with them. In this paper, we consider the concept of quasi-determinism, which is closely related to the usual determinism in the case of (traditional one-head) finite automata. Actually, as we will see, $\lambda$-transition free quasi-deterministic NFA = DFA. On the other hand, already for NFA with $\lambda$-transitions we may allow some non-determinism that is similar to the earlier introduced state-determinism. We should note here that our new concept can be applied in various other models of automata, however, it may not give anything new in those cases where stateless variants suffice, as we discuss it in the conclusion.

We also show that quasi-deterministic sensing $5' \to 3'$ WK automata are strictly more powerful than deterministic sensing $5' \to 3'$ WK automata and state-deterministic sensing $5' \to 3'$ WK automata. However, they are not as powerful as nondeterministic sensing $5' \to 3'$ WK automata. Closure properties for the most basic language operations are also investigated, and we prove that the class of languages accepted by quasi-deterministic sensing $5' \to 3'$ WK automata is not closed under them. Further, several hierarchy results among the language classes defined by the known restricted variants such as all-final, simple, 1-limited, and stateless $5' \to 3'$ Watson–Crick automata are shown together with some relations to REG of regular, LIN of linear context-free languages, and 2detLIN of the class accepted by deterministic sensing $5' \to 3'$ WK automata.

We also show an interesting ambiguity of deterministic computations in the case of unary languages and relate it also to such context-free derivations that have multiple derivation-trees.

## 2. Basic definitions

We assume that the reader is familiar with basic concepts of formal languages and automata theory, the language classes of the Chomsky hierarchy and regular expressions, otherwise, she or he is referred to [1, 2, 25]. In this paper, we denote the alphabet by $T$ and the empty word by $\lambda$.

We briefly recall the concepts of finite automata and sensing $5' \to 3'$ Watson–Crick automata.

A traditional finite automaton is a 5-tuple $A = (T, Q, q_0, F, \delta)$, where:

- $T$ is the (input) alphabet,
- $Q$ is the finite set of states,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final (also called accepting) states and
- $\delta$ is the transition mapping.

If $\delta$ is written in the form $\delta : Q \times (T \cup \{\lambda\}) \to 2^Q$, then $A$ is a nondeterministic finite automaton with (allowed) transitions by the empty word (NFA+$\lambda$, for short). If $\delta$ is also in the form $\delta : Q \times T \to 2^Q$, then $A$ is nondeterministic finite automaton without transitions by the empty word (NFA for short). Further, if $\delta : Q \times T \to Q$ is a possibly partially defined function, then $A$ is a deterministic finite automaton (DFA). Since this model has one reading head, sometimes we refer to it as one-head finite automata to differentiate it from the 2-head model we are also considering in this paper.

The initial configuration of a finite automaton $A$ consists of the initial state and the input word $w \in T^*$ as a pair $(q_0, w)$. The computation on an input word goes by configurations according to the transition mapping as follows: $(q, au) \Rightarrow (q', u)$ if $q' \in \delta(q, a)$. A word $w$ is accepted if $(q_0, w) \Rightarrow^* (q_F, \lambda)$ for a state $q_F \in F$, where $\Rightarrow^*$ is a reflexive and transitive closure of $\Rightarrow$. The accepted language is defined as $L(A) = \{w \mid (q_0, w) \Rightarrow^* (q_F, \lambda)$ for a state $q_F \in F\}$.

Notice that, as usual, finite automata process the input from left to right and (at most) one letter is being read in each transition (depending on the type, see above). However, in the literature there are various cases, when string reading is allowed, *i.e.*, from a state a finite set of words are given for which transitions are allowed. It is well-known that this feature does not increase the "accepting power" of finite automata, still exactly the class REG of regular languages can be accepted, however, it may have some effects on complexity measures, *etc.* Also there are various finite state automata models, when the automaton may read and process the input not in the strict left to right manner, *e.g.*, automata with translucent letters [26–29] and jumping automata [30, 31]. For the former model state-deterministic variants are studied in [32].

In this paper, we focus also on a 2-head finite automata model [33, 34], namely, sensing $5' \to 3'$ WK automata [12]. As we have already mentioned, the two strands of the DNA molecule have opposite $5' \to 3'$ orientations. Therefore, it is very natural to consider Watson–Crick finite automata that parse the two strands of the Watson–Crick tape in opposite (physical) directions. Since in sensing $5' \to 3'$ WK automaton the heads sense that they are meeting and the process on the input is finished (at the latest) in this position of the heads, the complementarity relation does not play any role in this model, w.l.o.g., the identity can be used. For this reason, we present a simplified (but equivalent) model in which a normal tape is used for the input. Moreover, since the head starting

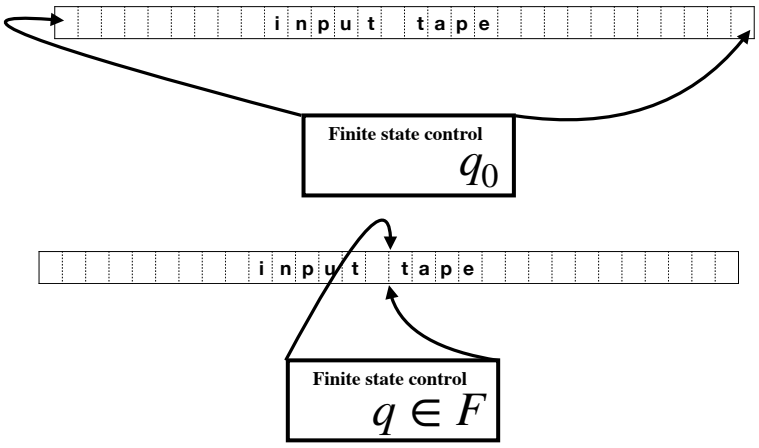


FIGURE 1. Schematic pictures of a sensing $5' \to 3'$ WK automaton in the initial configuration (top) and in an accepting configuration (bottom, with a final state $q$).

from the left end is always to the left of the other head, but the meeting final position, we may refer to the heads as left and right heads, or alternatively, as first and second heads.

Formally, a Watson–Crick automaton is considered to be a 5-tuple $A = (T, Q, q_0, F, \delta)$ similarly to the finite automata, with

- the (input) alphabet $T$, originally the letters standing for possible bases of the nucleotides,
- the finite set of states $Q$, the initial state $q_0 \in Q$ and the set of final states $F \subseteq Q$,
- the transition mapping $\delta$ is of the form $\delta : Q \times T^* \times T^* \to 2^Q$, such that it is defined only for finitely many triplets $(q, u, v), q \in Q, u, v \in T^*$.

Notice that there are two main differences between finite one-head automata and Watson–Crick automata, and both of these differences are in their transition mappings: Watson–Crick automata have two reading heads, and they may read strings in a transition. Actually, the main difference is caused by the two heads: the input is processed from its two extremes as we formally define below. The string-reading feature does not really give additional power in case there is no restriction used on the set of states [16, 22].

A configuration of a Watson–Crick automaton is a pair $(q, w)$ where $q$ is the current state and $w$ is the part of the input word which has not been processed (read) yet. In sensing $5' \to 3'$ WK automata, for any $w', x, y \in T^*$, $q, q' \in Q$, we write a transition (step of the computation) between two configurations as follows: $(q, xw'y) \Rightarrow (q', w')$ if and only if $q' \in \delta(q, x, y)$. We denote the reflexive and transitive closure of the relation $\Rightarrow$ by $\Rightarrow^*$. Therefore, for a given input $w \in T^*$, an accepting computation is a sequence of transitions $(q_0, w) \Rightarrow^* (q_F, \lambda)$, starting from the initial state and ending in a final state.

The language accepted by a WK automaton $A$ is: $L(A) = \{w \in T^* \mid (q_0, w) \Rightarrow^* (q_F, \lambda), q_F \in F\}$.

Figure 1 shows the schematic representations of the initial configuration and an accepting configuration of a sensing $5' \to 3'$ WK automaton.

Since in DNA computing the empty word does not represent any molecule, in this paper, as usual in this field, we do not distinguish languages that differ only in the empty word, that is, we call two automata equivalent if they accept the same language modulo the empty word. It is known that the class of languages accepted by sensing $5' \to 3'$ WK automata is exactly the class of linear context-free languages of the Chomsky hierarchy ([12, 13, 15–17]).

There are restricted variants of WK automata which are defined as follows. We say that a WK automaton is

- *stateless*, *i.e.*, with only one state, if $Q = F = \{q_0\}$. This class of WK automata is denoted by the letter N, as **N**o-state.

- *all-final*, *i.e.*, with only final states, if $Q = F$. This class of WK automata is denoted by the letter F, as all-**F**inal.
- *simple*, when at most one head moves in a step, formally: $\delta : (Q \times ((\lambda, T^*) \cup (T^*, \lambda))) \to 2^Q$. This class of WK automata is denoted by the letter S, as **S**imple.
- 1-*limited*, when exactly one letter is being read in each step of the computations, *i.e.*, $\delta : (Q \times ((\lambda, T) \cup (T, \lambda))) \to 2^Q$. This class of WK automata is denoted by 1, as **1**-limited.

Obviously, by definition, every N sensing $5' \to 3'$ WK automaton is an F sensing $5' \to 3'$ WK automaton, and also, every 1 sensing $5' \to 3'$ WK automaton is an S sensing $5' \to 3'$ WK automaton. However, some of the restrictions are independent, and thus, their combinations yield F1, N1, FS, NS WK automata. Stateless variants were analyzed in details, *e.g.*, in [35].

There is one usual definition of (classical) determinism in the case of sensing $5' \to 3'$ WK automata: We say that a $5' \to 3'$ WK automaton is *deterministic*, if there is at most one transition (computation step) applicable in each configuration. It is easy to see that this condition implies the fact that each possible configuration $(q, w)$ in any computation $(q \in Q, w \in T^*)$, there is at most one configuration $(p, u)$ such that $(q, w) \Rightarrow (p, u)$. However, as we shall see later, these two conditions are not in an 'if and only if' relation. Somewhat similarly, as at DFA, when deterministic WK automata are mentioned, we may use the letter D to denote this fact. Note that the language class accepted by deterministic sensing $5' \to 3'$ WK automata is a proper subset of LIN and denoted by 2detLIN [22, 23]. These deterministic models are characterized by Myhill-Nerode type theorems and used also in active learning [36–38]. Backward deterministic 2-head automata, as well as, reversible variants are studied in [39].

The concept of state-determinism is recalled from [24, 32]. An automaton (one-head or two-head) is *state-deterministic* if for each of its state $q \in Q$, if there is a transition from $q$ and it goes to state $p$ (*i.e.*, $p \in \delta(q, a)$ in the case of finite automata with $a \in T \cup \{\lambda\}$ and $p \in \delta(q, u, v)$ in case of WK-automata with $u, v \in T^*$), then every transition from $q$ goes to $p$. We may say that if an automaton (one-head or Watson–Crick) has state $q$ in its configuration, then, if the process continues, the state of the next configuration must be $p$. State-deterministic automata will be denoted by the prefix sD.

Now, we are ready to define our new concept that plays a central role in this paper. An automaton (one-head or Watson–Crick) is *quasi-deterministic* if the following condition holds. For each of its possible configurations, $(q, w)$ (with $q \in Q, w \in T^*$), if $(q, w) \Rightarrow (p, u)$ and $(q, w) \Rightarrow (r, v)$ then $p = r$ must hold. In other words, there is a unique state $p$ such that if there is a transition allowed in the configuration $(q, w)$, then it must go to a configuration in which the state is $p$. For quasi-deterministic automata, we use the qD prefix in the sequel.

Further in this paper, we use boldface writing to denote the language class accepted by a model described above, *e.g.*, **qDF1WK** denotes the class of languages accepted by quasi-deterministic all-final 1-limited $5' \to 3'$ Watson–Crick automata.

Automata are usually represented by their graphs, thus we may freely use graph theoretical concepts here. Further, we assume that each state of the automaton appears in an accepting path of a word of the accepted language. A transition from a state to itself is represented by a loop edge. Clearly, in a stateless automaton, there are only loop edges (if any).

## 3. On quasi-deterministic finite automata

In this section, our aim is to show how this new concept appears at the case of classical one-head finite automata.

The next proposition shows how quasi-determinism relates to determinism when finite one-head automata are considered.

**Proposition 3.1.** *Let $A$ be a $\lambda$-transition free NFA. If $A$ is quasi-deterministic, then it is deterministic. Moreover, each DFA is a $\lambda$-free NFA that is quasi-deterministic.*

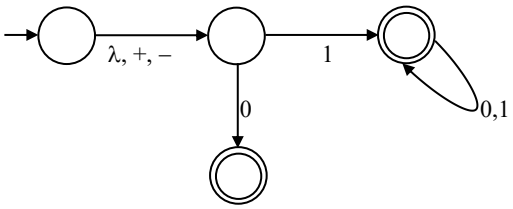


FIGURE 2. A finite automaton that is not deterministic, but quasi-deterministic.

*Proof.* Since $\lambda$-transition free NFA read exactly one input letter in each transition, for a quasi-deterministic NFA, the first letter of the remaining input in the configuration and the actual state together uniquely determines the next state exactly in the same way as in a DFA by definition. □

The new concept also relates to state-determinism in finite one-head automata:

**Proposition 3.2.** *If $A$ is a state-deterministic NFA+$\lambda$, then $A$ is also quasi-deterministic.*

*Proof.* Obvious from the definitions of these models. □

As we have seen, quasi-determinism is a close connection to determinism and also to state-determinism. Now to show that they are not exactly the same, we present an example.

**Example 3.3.** Consider the finite automaton shown in Figure 2. It accepts the language of binary integers over the alphabet $\{+, -, 0, 1\}$. It is not a DFA due to the transition by $\lambda$ from its initial state. Moreover, it is not state-deterministic, as there is a state for which reading 0 and 1 lead to different states. On the other hand, it is easy to check that it is quasi-deterministic: from the initial state, all the three possible transitions (including the one with $\lambda$) go to the same state. From the other states, the transitions are deterministic transitions.

Now, let us give a technical result about quasi-deterministic finite automata. In fact, their states behave similarly to the states of the deterministic or to the states of state-deterministic automata.

**Lemma 3.4.** *Let $A = (T, Q, q_0, F, \delta)$ be a quasi-deterministic finite NFA+$\lambda$ automaton. Then, the set of states can be partitioned into two sets as $Q = Q_d \cup Q_s$ ($Q_d \cap Q_s = \emptyset$) such that*

$$Q_d = \{q \mid \emptyset = \delta(q, \lambda) \text{ and for any } a \in T, \text{ there is at most one state } p \text{ such that } p \in \delta(q, a)\}$$

*and*

$$Q_s = \{q \mid \{p\} = \delta(q, \lambda) \text{ and if } q' \in \delta(q, a) \text{ for some } a \in T, \text{ then } q' = p\}.$$

*Proof.* Let $A$ be an NFA+$\lambda$ automaton. Then, for any of its states $q \in Q$ there are two options, either $\emptyset \neq \delta(q, \lambda)$ or $\emptyset = \delta(q, \lambda)$. In the latter case, there could be transitions from $q$ only by reading an input letter. If $A$ is quasi-deterministic, then from any configuration (with state $q$ having the first letter $a$ of the remaining input), the computation should go to a determined state (to $p$ in this case), if any (otherwise, the computation gets stuck and not accepting). In the former case, by not reading any input letter, $A$ may enter a configuration $(p, w)$ from $(q, w)$ if $p \in \delta(q, \lambda)$. Since $A$ is quasi-deterministic, it may not happen that there is a $p' \in Q$ such that $p' \in \delta(q, \lambda)$ and $p \neq p'$. Further, if there are transitions from $q$ by reading a letter, let us say $a \in T$, it must also go to state $p$. □

If $Q = Q_d$ in the previous lemma, then, in fact, the quasi-deterministic $A$ is also deterministic. Moreover, if $A$ is deterministic, then it is a quasi-deterministic finite automaton with the property $Q = Q_d$. On the other hand, if $A$ is quasi-deterministic with $Q = Q_s$, then $A$ is also state-deterministic. However, for a state-deterministic $A$, it may happen that it is quasi-deterministic, but $Q_d \neq \emptyset$. In this case, there are some states $q$ such that $\lambda$-transitions are not defined on those, but all the transitions from $q$ go to a unique state $p$ of the automaton.

As both the classes of DFA and NFA+$\lambda$ accept exactly the class of regular languages, it is straightforward to see that the class of quasi-deterministic finite automata, denoted by qDFA, is also characterizing the same class of languages. From this point of view, the new model does not seem to give a new power. However, from a complexity point of view, the classes of DFA and NFA have a remarkable difference, as in some cases, exponentially more states are needed for a DFA to accept the same regular language as an NFA needs. As qDFA allows a little non-determinism, it is an interesting challenge to see how the descriptional complexities of DFA, qDFA, NFA and NFA+$\lambda$ are related to each other. We leave this topic for a future work, and in this paper, we concentrate on $5' \to 3'$ WK automata, in which the quasi-determinism gives more freedom due to the two heads.

Observe the following:

**Fact 3.5.** *If a sensing $5' \to 3'$ WK automaton has transitions only with its left head (the right head is always reading $\lambda$), then, in fact, it is equivalent to a finite automaton. If this finite automaton is quasi-deterministic, then so is the $5' \to 3'$ WK automaton.*

We may use this link between finite automata and WK automata in the next sections as well.

## 4. The case of unary alphabet

Here, we investigate the case of unary alphabet, and show that neither the number of heads is important, nor the new concept is really interesting. However, from another point of view, a kind of ambiguity in deterministic computations is discussed.

Let us now consider the case of the unary alphabet. In this part of the paper, let the alphabet always be $\{a\}$. In this case only the number of the letters of the words are counted. It is well-known that over the unary alphabet every semi-regular language is regular and the language classes of regular, linear and context-free languages coincide [2, 40]. Actually, in the case of unary alphabet a language is semi-linear if the set of the lengths of the words of the language can be written as a finite union of linear sets, where a set is linear if it can be defined by finitely many fixed nonnegative integers $z_0, z_1, \ldots, z_n$ as follows: $\{z_0 + x_1 z_1 + \cdots + x_n z_n \mid x_1, \ldots, x_n$ are nonnegative integers$\}$. Therefore, DFA and nondeterministic sensing $5' \to 3'$ WK automata have the same power in the sense that the same class of languages are accepted by them. Moreover, any models which are at least as powerful as DFA and at most as powerful as sensing $5' \to 3'$ WK automata accept also the same language class. Thus, both quasi-deterministic one-head finite automata, and by Fact 3.5, quasi-deterministic sensing $5' \to 3'$ WK automata accept also the same class. In this way, neither the second head, nor the string-reading feature really add anything. We have a very interesting scenario considering determinism and deterministic computations.

**Remark 4.1.** Let us consider a 2-head automaton with transitions from state $p$ to $q$ where either the first head reads an $a$ or the second head reads an $a$: even if we have more than one transitions allowed, they both lead to the same configuration. Theoretically, by our definition, this model is not deterministic. On the other hand, technically and practically, it behaves as a deterministic machine, there is a unique computation for any input, as the sequence of configurations is uniquely determined (and practically, this is also a usual way to define determinism). However, if we consider the same automaton (or its part) but changing the alphabet to allow more letters, the automaton becomes not deterministic any more, as the different transitions, the movement of the first or the second head may lead to different configurations in this case.

This type of ambiguity with determinism is in close relation to the following issue concerning context-free derivations. Let us have the grammar with only one nonterminal $S$ and the production $S \to SS$, then in the second step of the derivation applying this rule, we have $SS \Rightarrow SSS$ independently which $S$ (the left or the right) of the sentential form is rewritten, and thus, this derivation step refers to the construction of two different derivation trees.

As we have discussed above, the second head does not really contribute in case of languages over the unary alphabet. This can be stated in the formal was as follows.

**Proposition 4.2.** *Let $A$ be a sensing $5' \to 3'$ WK automaton that accepts the unary language $L$. Then, there is a sensing $5' \to 3'$ WK automaton $A'$ accepting the same language $L$ such that in every transition of $A'$, the second head reads the empty word $\lambda$. Moreover, if $A$ is quasi-deterministic, then so is $A'$.*

*Proof.* The proof is constructive, consider $A$ and replace each of its transitions, $p \in \delta(q, u, v)$ (where $p, q$ states and $u, v \in a^*$) by the transition $p \in \delta(q, uv, \lambda)$. Obviously, the accepted language is the same. Moreover, by the definition of quasi-deterministic sensing $5' \to 3'$ WK automaton, this property of $A$ is kept in the construction and thus $A'$ also has it, if $A$ had. □

The previously constructed automaton $A'$ is, in fact, equivalent to a one-head finite automaton with string-reading feature. For these automata, the quasi-determinism allows a wider range of nondeterminism than for NFA+$\lambda$. Here, an automaton may have multiple transitions allowed at the same configuration by reading various length parts of the input. However, if more than one transitions are allowed, then they must go to the same state. In this way we can establish the following relationship.

**Theorem 4.3.** *Let $A$ be a sensing $5' \to 3'$ WK automaton that accepts the unary language $L$. If $A$ is quasi-deterministic, then it is also state-deterministic.*

*Proof.* Let $A$ be a quasi-deterministic sensing $5' \to 3'$ WK automaton over the unary alphabet. Consider computations by $A$. Clearly, any input string is of the form $a^*$, and thus, if the input is long enough, every transition from the actual state are allowed at the same time, thus all transitions must go to the same state. □

The previous theorem applies also to the special case, when only the first head is used and the automaton reads at most one letter in a transition, *i.e.*, we state the following.

**Corollary 4.4.** *In the case of unary one-head automata the concepts of state-determinism and quasi-determinism coincide.*

## 5. 2-head automata with various types of determinism

We start this section by showing that quasi-determinism is a generalisation of both determinism and state-determinism also in the case of our 2-head automata model. The following statements are consequences of the definitions of the used types of determinism.

**Proposition 5.1.** *If $A$ is a deterministic sensing $5' \to 3'$ WK automaton, then $A$ is quasi-deterministic.*

*Proof.* In deterministic sensing $5' \to 3'$ WK automata, for each possible configuration of a computation, there is at most one next configuration, and, thus, its state is the only state that can be reached from the given configuration in one computation step. □

**Proposition 5.2.** *If $A$ is a state-deterministic sensing $5' \to 3'$ WK automaton, then $A$ is quasi-deterministic.*

*Proof.* In state-deterministic sensing $5' \to 3'$ WK automata, for each possible configuration of a computation, there is at most one state that can appear in any of the next configurations, and, thus, this unique state (if any) is the one that proves the quasi-determinism. □

The previous result with Theorem 4.3 implies the following.

**Corollary 5.3.** *In the case of unary alphabet the concepts of state-determinism and quasi-determinism coincide for sensing $5' \to 3'$ WK automata.*

At this point, we can also deduct that the language class accepted by quasi-deterministic sensing $5' \to 3'$ WK automata contains 2detLIN, the language class accepted by deterministic sensing $5' \to 3'$ WK automata and also the language class accepted by state-deterministic sensing $5' \to 3'$ WK automata. Now we show that both of these inclusions are proper, and thus, in the general case, these concepts differ from each other.

**Proposition 5.4.** *The language $L_o = \{a^m b^n \mid m \leq n \leq 2m\}$ is accepted by a quasi-deterministic $5' \to 3'$ WK automaton, but is not accepted by any deterministic sensing $5' \to 3'$ WK automaton.*

*Proof.* On the one hand, $L$ is accepted by a $5' \to 3'$ WK automaton having only one state and two loop transitions with $(a, b)$ and $(a, bb)$. Obviously this automaton is quasi-deterministic, always only its sole state can appear in any next configurations.

On the other hand, we need to prove that this language is not in 2detLIN; the proof goes by contradiction. Let us assume that a deterministic sensing $5' \to 3'$ WK automaton $A$ with initial state $q$ accepts $L_o$. Let $k$ be the number of states of $A$ and let $r$ be the maximal length of the words $A$ may read in a transition (radius of $A$). Let $m = 3kr$. Consider the words $a^m b^m$ and $a^m b^{2m}$. Both are in $L_o$, however their accepting computation must start exactly in the same way in the first $2k$ steps ($A$ may read at most $2kr$ $a$-s by the first head and at most that many $b$-s by the second head during this part of the computation). Thus there is a state which appears in more than one configurations in this part of the computation, let it be $p$ (may be the same as $q$). This part of the computations on the two above words can be written as $(q, a^m b^m) \Rightarrow^* (p, a^{m-i_1} b^{m-j_1}) \Rightarrow^* (p, a^{m-i_2} b^{m-j_2}) \Rightarrow^* (f_1, \lambda)$ and $(q, a^m b^{2m}) \Rightarrow^* (p, a^{m-i_1} b^{2m-j_1}) \Rightarrow^* (p, a^{m-i_2} b^{2m-j_2}) \Rightarrow^* (f_2, \lambda)$ with accepting states $f_1, f_2$. Let us analyse the relation of $i_2 - i_1$ and $j_2 - j_1$, *i.e.*, the number of $a$-s and $b$-s read in the cycle.

- If $i_2 - i_1 > j_2 - j_1$, *i.e.*, more $a$-s are read than $b$-s, then we also have the accepting computation $(q, a^{m+i_2-i_1} b^{m+j_2-j_1}) \Rightarrow^* (p, a^{m+i_2-2i_1} b^{m+j_2-2j_1}) \Rightarrow^* (p, a^{m-i_1} b^{m-j_1}) \Rightarrow^* (p, a^{m-i_2} b^{m-j_2}) \Rightarrow^* (f_1, \lambda)$ contradicting to the fact that the word $a^{m+i_2-i_1} b^{m+j_2-j_1}$ is not in $L_o$.
- In the case $i_2 - i_1 = j_2 - j_1$, *i.e.*, the computation reads the same number of $a$-s and $b$-s in the cycle, we have the accepting computation $(q, a^{m-(i_2-i_1)} b^{2m-(i_2-i_1)}) \Rightarrow^* (p, a^{m-i_2} b^{2m-j_2}) \Rightarrow^* (f_2, \lambda)$. However, in this case, $m - (i_2 - i_1) = m - i_2 + i_1$ is less than the half of $2m - (i_2 - i_1) = 2m - i_2 + i_1$, and thus, $a^{m-(i_2-i_1)} b^{2m-(i_2-i_1)}$ is not in $L_o$.
- Finally, if $i_2 - i_1 < j_2 - j_1$, then there is an accepting computation $(q, a^{m-(i_2-i_1)} b^{m-(j_2-j_1)}) \Rightarrow^* (p, a^{m-i_2} b^{m-j_2}) \Rightarrow^* (f_1, \lambda)$, however, the word $a^{m-(i_2-i_1)} b^{m-(j_2-j_1)}$ contains more $a$-s than $b$-s showing the contradiction in this case.

□

**Proposition 5.5.** *The regular language $b^*ab^* + b^*$ is not accepted by any state-deterministic sensing $5' \to 3'$ WK automata, but it is accepted by some quasi-deterministic sensing $5' \to 3'$ WK automata.*

*Proof.* The first part of the proof goes by contradiction, thus, assume that there is a state-deterministic sensing $5' \to 3'$ WK automaton $A$ accepting $b^*ab^* + b^*$. Since the language is infinite, $A$ must have a cycle. Definitely, $A$ has a transition where letter $a$ (or a string containing letter $a$) is being read by either head and let us refer to this head by $h$. Considering this transition, it cannot be in the cycle, since it would allow to read more than one $a$ in an accepted word. However, in a state-deterministic automaton, there are only finitely many states that are not in the cycle, and these states can only be visited during a computation only before the computation enters to the cycle. Thus, the mentioned transition must be from a state which can be visited before the computation enters into the cycle, let us refer to this state by $p$. However, in the finitely many possible computations connecting the initial state $q_0$ to $p$, head $h$ can read at most a given finite number, let us say $j$, of $b$-s (depending both on the numbers of states in this path, and also on the length of the read words of the transitions of the path). But, then it would be impossible for $A$ to accept a word such that its prefix (if $h$ is the left head) or its suffix (in case $h$ is the right head) has more $b$-s (before/after the $a$, respectively) than $j$. However, as the language $b^*ab^* + b^*$ has such words, a contradiction is obtained, and this part of the proof is finished.

Now, on the other hand, the language $b^*ab^* + b^*$ is a regular language, and thus, it is accepted by some DFA. And a DFA can also be seen as a specific quasi-deterministic sensing $5' \to 3'$ WK automaton obtaining the second part of the proof. □

Based on the previous propositions, we can conclude the following result.

**Corollary 5.6.** *The class of languages accepted by quasi-deterministic sensing $5' \to 3'$ WK automata is a proper superset of both 2detLIN and the class of languages accepted by state-deterministic sensing $5' \to 3'$ WK automata. Formally,*

$$\boldsymbol{qDWK} \supsetneq \mathit{2detLIN} \qquad \mathit{and} \qquad \boldsymbol{qDWK} \supsetneq \boldsymbol{sDWK}.$$

As we already mentioned, in the case of finite automata, quasi-determinism may play only a role in complexity, as, in fact, the class REG of regular languages is characterized by DFA, NFA, NFA+$\lambda$ and also by qDFA. Contrary to this, we have seen that qD sensing $5' \to 3'$ WK automata are more powerful than deterministic sensing $5' \to 3'$ WK automata. Now, we show that qD sensing $5' \to 3'$ WK automata are still not as powerful as the general nondeterministic sensing $5' \to 3'$ WK automata, and thus the new concept, the quasi-determinism leads to new language classes.

**Theorem 5.7.** $\boldsymbol{qDWK} \subsetneq \mathit{LIN} = \boldsymbol{WK}$.

*Proof.* The inclusion is trivial from the definition; we need to show only properness. Let us consider the language $L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$. Clearly, it is a union of two linear context-free languages, and it is also linear. The rest of the proof goes by contradiction. Thus, let us assume that there is a quasi-deterministic sensing $5' \to 3'$ WK automaton $A$ that accepts $L$. (W.l.o.g., we assume that there are no states in $A$ such that the only transition is the one that reads no letters from the input.) Let the number of states of $A$ be $k$ and let the longest string read by a possible transition have length $\ell$. Now let us consider the words $u = a^{(3k+2)\ell} b^{(3k+2)\ell}$ and $v = a^{(3k+2)\ell} b^{(6k+4)\ell}$, clearly both of them are in $L$, thus $A$ must accept both of them. On the other hand, $A$ may read only the prefixes and suffixes of those words, and thus cannot distinguish them.

We shall show that by accepting both of these words by $A$ we can also accept some words that are not in $L$. Since the words are long enough, there must be a state during the computation that is repeated, more precisely, we have the following. As one step of the computation may process at most $2\ell$ letters (in fact, at most $\ell$ by the first head and at most $\ell$ by the second head), the accepting computation on $u$ (and also on $v$) must take more than $3k$ steps. By the pigeon-hole principle, there must be a state that appears at least 3 times during such accepting computation (already in the first $2k+1$ configurations, *i.e.*, during or right after the first $2k$ steps). Now, in the first $3k$ steps, both in $u$ and $v$, the first head can only read $a$-s, *i.e.*, in each step, a word from $a^*$ is read with length at most $\ell$. Similarly, the second head can read only word of $b^*$ with length at most $\ell$. Since $A$ is quasi-deterministic, in each of these first $3k$ steps, the computation goes through the same sequence of states. In this way, if $q$ is a state that is used in at least three different configurations in the first $3k$ steps (we already have shown that such a state exists, maybe $q = q_0$), then we have an accepting computation on $u$ as $(q_0, u) \Rightarrow^* (q, a^{i_1} b^{j_1}) \Rightarrow^* (q, a^{i_2} b^{j_2}) \Rightarrow^* (\lambda, q_f)$ with some accepting state $q_f \in F$. By the cycle of the computation made from state $q$ to reach again state $q$, it is clear that the word $u' = a^{(3k+2)\ell + n(i_2 - i_1)} b^{(3k+2)\ell + n(j_2 - j_1)}$ is also accepted for all positive integer $n$, which implies that $i_2 - i_1 = j_2 - j_1$. Let this number be denoted by $x$, *i.e.*, $x = i_2 - i_1$. In such a cycle, $A$ reads $x$ $a$-s and $x$ $b$-s, and $x > 0$.

Now, considering the accepting computation on $v$ which also goes through $q$ more than once during the first $3k$ steps, *i.e.*, it can be written as $(q_0, v) \Rightarrow^* (q, a^{i_3} b^{j_3}) \Rightarrow^* (q, a^{i_4} b^{j_4}) \Rightarrow^* (\lambda, q'_f)$ with an accepting state $q'_f$. Here, there is also a cycle in the computation determined by the above two configurations containing state $q$. However, if one inserts the previously studied cycle of the accepting computation on $u$ here, then we get a computation that is also accepting on the word $w = a^{(3k+2)\ell + x} b^{(6k+4)\ell + x}$ $(q_0, w) \Rightarrow^* (q, a^{i_3 + x} b^{j_3 + x}) \Rightarrow^* (q, a^{i_4 + x} b^{j_4 + x}) \Rightarrow^* (q, a^{i_4} b^{j_4}) \Rightarrow^* (\lambda, q'_f)$. However, the numbers of $a$-s and $b$-s in $w$ are neither equal (as $\ell > 0$), nor the number of $b$-s is double than the number of $a$-s (as $x > 0$). This contradicts to the fact that $A$ accepts $L$. □

We have just proven that the class of languages accepted by quasi-deterministic sensing $5' \to 3'$ WK automata is a new sublinear class of languages. From now on we also refer to this class, **qDWK**, as the class qD-LIN of quasi-deterministic linear languages.

## 6. Closure properties of the quasi-deterministic 2-head automata languages

In this section some properties of the quasi-deterministic linear languages is shown.

**Theorem 6.1.** *The class of quasi-deterministic linear languages is not closed under union, intersection, complement, concatenation and Kleene-star.*

*Proof.* Let us prove the statements one by one:

- union: We have already shown in the proof of Theorem 5.7 that the language $L = \{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$ is not accepted by any quasi-deterministic sensing $5' \to 3'$ WK automata. On the other, hand both $\{a^n b^n \mid n \geq 0\}$ and $\{a^n b^{2n} \mid n \geq 0\}$ are 2detLIN languages and thus accepted by some quasi-deterministic sensing $5' \to 3'$ WK automata, which completes the proof of this case.
- intersection: Considering the languages $\{a^n b^n a^m \mid n, m \geq 0\}$ and $\{a^n b^m a^n \mid n, m \geq 0\}$, they are both in 2detLIN. Their intersection is $\{a^n b^n a^n \mid n \geq 0\}$ which is not even context-free, thus it is definitely not accepted by any quasi-deterministic sensing $5' \to 3'$ WK automata.
- complement: Consider the language $L_o = \{a^m b^n \mid m \leq n \leq 2m\}$ of Proposition 5.4 (over $\{a, b\}$) which was proven to be in qD-LIN. Now let us take its complement $L'$. Clearly $L'$ is the union of the following five languages: $a^+$, $b^+$, $(a+b)^* ba (a+b)^*$ and $\{a^n b^m \mid m < n\}$, $\{a^n b^m \mid m > 2n\}$. Notice that the union of the first three mentioned languages is still regular, but the latter two languages are nonregular linear context-free languages and they are containing all the words of $L'$ for which the expression $a^+ b^+$ matches. The proof goes by contradiction.

  Let us assume that there is a quasi-deterministic sensing $5' \to 3'$ WK automaton $A$ that accepts $L'$. (W.l.o.g., we assume that there are no states in $A$ such that the only transition is the one that reads no letters from the input.) Let the number of states of $A$ be $k$ and let the longest string read by a possible transition have length $\ell$. Now let us consider the following words of $L'$: $u = a^{(3k+2)\ell+1} b^{(3k+2)\ell}$ and $v = a^{(3k+2)\ell} b^{(6k+4)\ell+1}$; $A$ must accept them. As $A$ can read only the prefixes and suffixes of those words, cannot distinguish them in the beginning of the computation and we shall show that by accepting both of these words, $A$ also accepts some words that are not in $L'$.

  As one step of the computation may process at most $2\ell$ letters (in fact, at most $\ell$ by the first head and at most $\ell$ by the second head), the accepting computation on $u$ (and also on $v$) must take more than $3k$ steps. By the pigeon-hole principle, there must be a state that appears at least 3 times during such accepting computation (already in the first $2k+1$ configurations, *i.e.*, during or right after the first $2k$ steps). Now, in the first $3k$ steps, both in $u$ and $v$, the first head can only read $a$-s, *i.e.*, in each step, a word from $a^*$ is read with length at most $\ell$. Similarly, the second head can read only words of $b^*$ with length at most $\ell$. Since $A$ is quasi-deterministic, in each of these first $3k$ steps, the computation goes through the same sequence of states. In this way, if $q$ is a state that is used in at least three different configurations in the first $3k$ steps, then we have an accepting computation on $u$ as $(q_0, u) \Rightarrow^* (q, a^{i_1} b^{j_1}) \Rightarrow^* (q, a^{i_2} b^{j_2}) \Rightarrow^* (\lambda, q_f)$ with some accepting state $q_f \in F$. Let $x$ and $y$ denote the number of $a$-s and $b$-s processed during the cycle, *i.e.*, $x = i_1 - i_2$ and $y = j_1 - j_2$. By repeating the cycle of the computation once more, clearly the word $u' = a^{(3k+2)\ell+1+x} b^{(3k+2)\ell+y}$ is also accepted. This ($u' \in L'$) implies that the condition $x \geq y$ should be fulfilled (as this cycle is made in at most as $k$ steps, it is not possible that $u'$ belongs to $\{a^n b^m \mid m > 2n\}$ since $y \leq k\ell$).

  Now, let us consider the accepting computation on $v$ which also goes through on the asme state $q$ more than once during the first $3k$ steps. By inserting the previously studied cycle of the accepting computation on $u$ here, we get a computation that accepts the word $w = a^{(3k+2)\ell+x} b^{(6k+4)\ell+1+y}$. However, $w \in L'$ implies that the condition $y + 1 > 2x$ met. This is equivalent to $y \geq 2x$. (Here, it is impossible that $w$ belongs to $\{a^n b^m \mid m < n\}$ as $x \leq k\ell$.)

  Now, summarizing our conditions, $u', w \in L'$ implies $x \geq y \geq 2x$ which is impossible if at least one of $x$ or $y$ is positive.

This contradicts to the fact that $A$ accepts $L'$ and thus the language $L'$ is not in qD-LIN.

- concatenation: Considering the languages $\{a^nb^n \mid n \geq 0\}$ and $\{c^md^m \mid m \geq 0\}$, they are both in 2detLIN. Their concatenation is $\{a^nb^nc^md^m \mid n \geq 0\}$ which is not a linear context-free language, thus it is not accepted by any quasi-deterministic sensing $5' \to 3'$ WK automata.
- Kleene-star: Consider the 2-detLIN language $\{a^nb^n \mid n \geq 0\}$. Its Kleene-star is not a linear context-free language, thus it is not accepted by any quasi-deterministic sensing $5' \to 3'$ WK automata.

□

## 7. Variants of sensing $5' \to 3'$ WK automata: hierarchy results

In this section, we consider some restricted variants of quasi-deterministic sensing $5' \to 3'$ WK automata and the language classes accepted by them. We concentrate on hierarchy results among these classes. The class REG of regular languages is also compared to these classes.

We start with the simplest variants, the stateless ones.

**Proposition 7.1.** *Every stateless sensing $5' \to 3'$ WK automaton is quasi-deterministic.*

*Proof.* In a computation of a stateless automaton, the sole state appears in every configuration, thus the automaton must be quasi-deterministic. □

As the quasi-determinism is a condition that does not have any influence on the other usual restrictions, the previous proposition implies the following facts.

**Corollary 7.2.** *The class of N sensing $5' \to 3'$ WK automata is the same as the class of qDN sensing $5' \to 3'$ WK automata.*

*The class of NS sensing $5' \to 3'$ WK automata is the same as the class of qDNS sensing $5' \to 3'$ WK automata.*

*The class of N1 sensing $5' \to 3'$ WK automata is the same as the class of qDN1 sensing $5' \to 3'$ WK automata.*

Based on the results proven in [16, 17], we can establish the following hierarchy results.

**Corollary 7.3.** $\boldsymbol{qDNWK \supsetneq qDNSWK \supsetneq qDN1WK}$.

Further, by observing that the language of Proposition 5.5 cannot be accepted by a quasi-deterministic stateless sensing $5' \to 3'$ WK automata), we state that to be stateless is stronger restriction than to be quasi-deterministic.

**Proposition 7.4.** $\boldsymbol{qDNWK \subsetneq qDWK} = \textit{qD-LIN}$.

Now, we analyse the relation of the class REG of regular languages to stateless variants of quasi-deterministic sensing $5' \to 3'$ WK automata.

Similarly as in case on nondeterministic and state-deterministic sensing $5' \to 3'$ WK automata (see [16, 17] and [24], respectively), on the one hand, we have:

**Proposition 7.5.** $\boldsymbol{qDNSWK} \subsetneq \textit{REG}$

*Proof.* The qDNS sensing $5' \to 3'$ WK automata can accept only special regular languages of the form $(v_1 + \cdots + v_i)^*(u_1 + \cdots + u_j)^*$, where the transitions in which the first head can read are $\{q\} = \delta(q, v_1, \lambda) = \cdots = \delta(q, v_i, \lambda)$ and the transitions with the second head are $\{q\} = \delta(q, \lambda, u_1) = \cdots = \delta(q, \lambda, u_j)$. Furthermore, we have also shown a regular language in Proposition 5.5 that cannot be accepted by any qD stateless sensing $5' \to 3'$ WK automata. □

On the other hand, there are non-regular languages that are accepted by qDN sensing $5' \to 3'$ WK automata:

**Example 7.6.** Let us consider the qDN sensing $5' \to 3'$ WK automaton with only one transition, $\{q\} = \delta(q, a, b)$. This automaton accept the non-regular language $\{a^nb^n \mid n \geq 0\}$.

**Corollary 7.7.** *The class **qDNWK** of languages accepted by qDN sensing $5' \to 3'$ WK automata and the class REG of regular languages are incomparable under set-theoretic inclusion relation.*

Now we involve to the study the all-final variants to obtain some new results.

**Proposition 7.8.** ***qDF1WK** $\supsetneq$ **qDN1WK**.*

*Proof.* The inclusion comes directly from the definition. We need to prove the properness. Let us consider the regular language $ba^* + a^*$. A qDF1 sensing $5' \to 3'$ WK automaton that accepts it can be given as $(\{q,p\},\{a,b\},q,\{q,p\},\delta)$, with $\{q\} = \delta(q,\lambda,a)$ and $\{p\} = \delta(q,\lambda,b)$.

Now, we argue that no qD stateless sensing $5' \to 3'$ WK automata can accept this language. In fact, to accept the language by a stateless variant, there must be a transition from its sole state to itself that allows to read a letter $b$ (or a string that contains it). However, then by the iterative use of this transition, words containing more than one $b$ would also be accepted. □

**Proposition 7.9.** ***qDFSWK** $\supsetneq$ **qDF1WK**.*

*Proof.* The inclusion is clear from the definition. To show its properness, let us consider the following qDFS sensing $5' \to 3'$ WK automaton: $(\{p,q\},\{a,b\},q,\{p,q\},\delta)$ with $\{p\} = \delta(q,aa,\lambda)$, $\{q\} = \delta(p,\lambda,bb)$. It accepts the language $\{a^{2n}b^{2n} \mid n \geq 0\} \cup \{a^{2(n+1)}b^{2n} \mid n \geq 0\}$.

On the other hand, it is clear that if a qDF1 sensing $5' \to 3'$ WK automaton accepts a word $w$ of length $k$, then its accepting computation contains exactly $k$ steps, and there are words in the accepted language by each positive integer length up to $k$ based on the given accepting computation in $w$ (composed by the read prefix and read suffix of the input word). Since, *e.g.*, in the previous language, there is a word with length 2, but there is no word with length 1, clearly it cannot be accepted by any qDF1 sensing $5' \to 3'$ WK automata. □

**Proposition 7.10.** *The class **qDF1WK** of languages accepted by qDF1 sensing $5' \to 3'$ WK automata is incomparable under set-theoretic inclusion relation with both **qDNWK** and **qDNSWK**.*

*Proof.* On the one hand, using longer strings $v_k$ $(1 \leq k \leq i)$ and $u_k$ $(1 \leq k \leq j)$ than one letter in the regular expression shown in the proof of Proposition 7.5 and the argument of the end of the proof of Proposition 7.9 about qDF1 sensing $5' \to 3'$ WK automata and their languages, it is easy to obtain a language that is accepted by qDNS sensing $5' \to 3'$ WK automata, but not with any qDF1 sensing $5' \to 3'$ WK automata. On the other hand, the proof of Proposition 7.8 shows a language that is accepted by qDF1 sensing $5' \to 3'$ WK automata, but not with any qDN sensing $5' \to 3'$ WK automata and this completes the proof of both statements. □

Moreover, we have:

**Proposition 7.11.** ***qDFSWK** $\supsetneq$ **qDNSWK** and **qDFWK** $\supsetneq$ **qDNWK**.*
*Further, the class **qDFSWK** of languages accepted by qDFS sensing $5' \to 3'$ WK automata is incomparable with the class **qDNWK** of languages accepted by qDN sensing $5' \to 3'$ WK automata.*

*Proof.* The inclusions come from the definitions, we need to prove only their properness. The language $ba^* + a^*$ used in the proof of Proposition 7.8 is accepted by a qDF1 sensing $5' \to 3'$ WK automaton, which is also a qDFS and a qDF sensing $5' \to 3'$ WK automaton. On the other hand, this language cannot be accepted by any qD stateless sensing $5' \to 3'$ WK automaton by the argument used in the second part of the proof of Proposition 7.8.

The above example proves also one side of the stated incomparability. To complete the proof we need to find a language that is accepted by qDN sensing $5' \to 3'$ WK automata, but not with any qDFS sensing $5' \to 3'$ WK automata. An example of such a languages is $L_3 = \{a^{3n}b^{3n} \mid n \geq 0\}$. A stateless automaton having only one loop transition by $(aaa, bbb)$ suffices. On the other hand, we need to show that we cannot accept $L_3$ by any FS automaton. The proof goes by contradiction. Let us assume that $A$ is a qDFS sensing $5' \to 3'$ WK automaton that accepts $L_3$. Let the length of the longest string $A$ can read in a transition be $r$. Consider, the word $w = a^{3r}b^{3r} \in L_3$ which must be accepted. However, in the first step of the computation $A$ may read only the prefix (by the first head) or only the suffix (by the second head) with length at $x$, where $x \leq r$. Since $A$

is all-final, it is also accepting then it would also accept $a^x$ or $b^x$ (depending on which head would start the accepting computation on $w$), but none of those words are in $L_3$. This contradiction shows that $L_3$ cannot be accepted by any qDFS sensing $5' \to 3'$ WK automata and the proof is completed. □

We can refine the hierarchy here as follows: on the one hand, it is known that state-deterministic sensing $5' \to 3'$ WK automata are more powerful that their stateless variants; on the other hand, in [24] it is also proven that every language that is accepted by state-deterministic sensing $5' \to 3'$ WK automata can also be accepted by all-final state-deterministic sensing $5' \to 3'$ WK automata. However, in fact, every all-final state-deterministic sensing $5' \to 3'$ WK automaton is an all-final quasi-deterministic sensing $5' \to 3'$ WK automaton. Further, as we have seen in, *e.g.*, Proposition 5.5, the language $b^*ab^* + b^*$ is not accepted by any state-deterministic sensing $5' \to 3'$ WK automata. But this language is accepted by the automaton $(\{a,b\}, \{q,p\}, q, \{q,p\}, \delta)$ with $\delta(q,b,\lambda) = q$, $\delta(q,a,\lambda) = p$, $\delta(p,b,\lambda) = p$. This is in fact deterministic and quasi-deterministic F1 sensing $5' \to 3'$ WK automaton. Thus, we can conclude the following:

**Proposition 7.12.** ***qDNWK*** $\subsetneq$ ***sDWK*** $\subsetneq$ ***qDFWK***.

Now, we are continuing our study with the latter class.

**Proposition 7.13.** ***qDFWK*** $\supsetneq$ ***qDFSWK***.

*Proof.* The inclusion is straightforwardly coming from the definition, we shall prove only its properness. Consider the separating language $aaa(ab)^*bbb$. On the one hand, the qDF sensing $5' \to 3'$ WK automaton $(\{p,q\}, \{a,b\}, q, \{p,q\}, \delta)$ with $\{p\} = \delta(q, aaa, bbb)$, $\{p\} = \delta(p, ab, \lambda)$ accepts it. On the other hand, in a qDFS automaton, in a transition from the initial state only one of the heads can read some input symbols. The shortest nonempty word of the language is $aaabbb$, thus either this is read by one of the heads, or if shorter word is read, then the automaton will accept that prefix or suffix of $aaabbb$ leading to a contradiction. However, if the whole $aaabbb$ is read by one of the heads, then this must be the prefix (if the left head is used) or the suffix (if the right head is used) of the other words that are accepted. Since this is not true for the language, there could not be any qDFS automaton that accepts it. □

Now we continue with relations of REG to all-final variants of our 2-head automata.

**Lemma 7.14.** *The regular language $L'$ given by $b^*ab^*$ is not accepted by any qDF sensing $5' \to 3'$ WK automata.*

*Proof.* The proof goes by contradiction. Let us assume that $A$ is a qDF sensing $5' \to 3'$ WK automaton that accepts $L'$. Let $k$ is the length of the longest string that can be read by a transition of $A$. Consider the word $w = b^{2k+1}ab^{2k+1}$ which is in $L'$. Let us consider its accepting computation. The all-final $A$ reads $k_1$ (at most $k$) $b$-s from the prefix and $k_2$ (at most $k$) $b$-s from the suffix of $w$ in the first step of the computation. However, as all states are final, this leads that the input word $b^{k_1}b^{k_2}$ would also be accepted, however it is not in $L'$. By this contradiction the lemma is proven. □

Now we are ready to state some further incomparability results.

**Proposition 7.15.** *Each of the classes* ***qDFWK***, ***qDFSWK*** *and* ***qDF1WK*** *is incomparable with the class REG of regular languages under set-theoretic inclusion relation.*

*Proof.* On the one hand, we have shown in Lemma 7.14 a regular language that is not accepted by any automata of the mentioned classes of the sensing $5' \to 3'$ WK automata.

On the other hand, let us consider the qDF1 sensing $5' \to 3'$ WK automaton defined as $(\{a,b\}, \{q,p,r\}, q, \{q,p,r\}, \delta)$ with $\delta(q,a,\lambda) = p$, $\delta(q,b,\lambda) = r$, $\delta(p,\lambda,a) = q$ and $\delta(r,\lambda,b) = q$. This automaton accepts the language of palindromes over $\{a,b\}$ which is a well-known non-regular language. □

Now, we state a relation concerning 2detLIN, the class of languages accepted by deterministic sensing $5' \to 3'$ WK automata.

**Proposition 7.16.** ***qD1WK*** $\supset$ *2detLIN*.

*Proof.* It has been shown in [22] that 2detLIN is also accepted by the class of deterministic 1-limited sensing $5' \to 3'$ WK automata. On the other hand, as an implication of their definitions, each D1 sensing $5' \to 3'$ WK automaton is a qD1 sensing $5' \to 3'$ WK automaton, which implies the statement. □

In fact, D1 sensing $5' \to 3'$ WK automata are those qD1 sensing $5' \to 3'$ WK automata in which exactly one of the heads are allowed to read in each state. As one may easily design a qD1 sensing $5' \to 3'$ WK automaton that does not satisfy this property, this latter model is more powerful than the deterministic sensing $5' \to 3'$ WK automata. We come back to this point in Theorem 7.20.

Knowing that REG $\subsetneq$ 2detLIN, based on the previous theorem and the argument in the proof of Proposition 7.9 about qDF1 sensing $5' \to 3'$ WK automata, we also infer the following inclusion:

**Corollary 7.17.** ***qD1WK*** $\supsetneq$ ***qDF1WK***.

Moreover, based on Proposition 7.15 we have also:

**Corollary 7.18.** ***qDSWK*** $\supsetneq$ ***qDFSWK*** *and* *qD-LIN* $\supsetneq$ ***qDFWK***.

**Lemma 7.19.** *The language* $\{a^n b^m \mid n \leq m \leq 2n\}$ *is accepted by the qD1 sensing* $5' \to 3'$ *WK automaton shown in Figure 3 (left).*

*Proof.* First of all, it is obvious that the automaton is 1-limited and quasi-deterministic: the only branching is at state 4, and it is a deterministic decision by the letter being read with the first head. However, the automaton is not deterministic due to the transitions from state 7 to state 2.

Now, we shall show that the automaton is accepting the language $L_o = \{a^n b^m \mid n \leq m \leq 2n\}$.

The automaton consists of a beginning part, a cycle and a tail part. Let us see which words are accepted without visiting any state more than once, *i.e.*, without completing a full cycle:

| state | 0 | 2 | 3 | 4 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| word | $\lambda$ | $ab$ | $abb$ | $aabb$ | $aaabbb$ | $aaabbbb$ | $aabbb$ | $aabbbb$ |

Clearly, all these words are in $L_o$, moreover all words of $L_o$ up to length 7 is accepted without completing a cycle in the computation. Notice that $L_o$ contains exactly one word with length 7 (and that is accepted by state 7 above).

After completing the first cycle, by the last step either the first head reads an $a$ or the second head reads a $b$, and by state 2 at this time the words $aaaabbbb$ and $aaabbbbb$ are accepted respectively, and these are exactly the two length-8 words of $L_o$.

Thus, the computation reaching state 2 in the beginning consumes exactly one $a$ from the prefix and one $b$ from the suffix of the input. One cycle consumes either 3 of the $a$-s (by the first head) and 3 of the $b$-s by the second head OR 2 of the $a$-a and 4 of the $b$-s. Now we may assume that the computation may go through on the cycle $k$ times. Then by state 2 all of the words $a^{3k+1}b^{3k+1}, a^{3k}b^{3k+2}, \dots a^{2k+1}b^{4k+1}$ are accepted, notice that they are exactly the $6k+2$ length words of $L_o$. Further, after the $k$-th cycle of the computation, the following words are accepted by the automaton with the other states:

| state | accepted words |
|---|---|
| 3 | $a^{3k+1}b^{3k+2}, a^{3k}b^{3k+3}, \dots, a^{2k+1}b^{4k+2}$ |
| 4 | $a^{3k+2}b^{3k+2}, a^{3k+1}b^{3k+3}, \dots, a^{2k+2}b^{4k+2}$ |
| 6 | $a^{3k+3}b^{3k+3}, a^{3k+2}b^{3k+4}, \dots, a^{2k+3}b^{4k+3}$ |
| 7 | $a^{3k+3}b^{3k+4}, a^{3k+2}b^{3k+5}, \dots, a^{2k+3}b^{4k+4}$ |
| 8 | $a^{3k+2}b^{3k+3}, a^{3k+1}b^{3k+4}, \dots, a^{2k+2}b^{4k+3}$ |
| 9 | $a^{3k+2}b^{3k+4}, a^{3k+1}b^{3k+5}, \dots, a^{2k+2}b^{4k+4}$ |

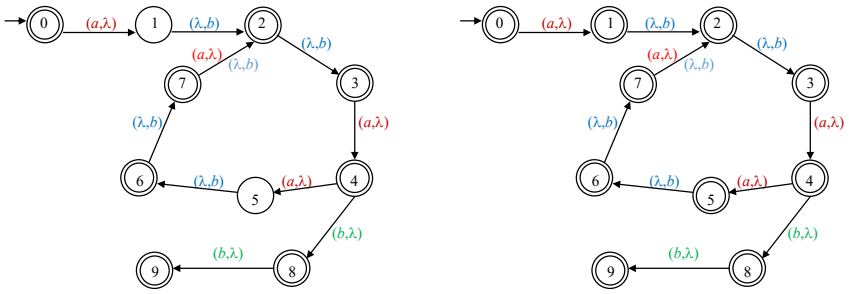


FIGURE 3. A 1 quasi-deterministic sensing $5' \to 3'$ WK automaton accepting $\{a^n b^m \mid n \leq m \leq 2n\}$ (left) and an F1 quasi-deterministic sensing $5' \to 3'$ WK automaton for a related language (right).

It is easy to see that by state 3, 4 and 8 all the words of $L_o$ are accepted with length $6k+3$, $6k+4$ and $6k+5$, respectively. (There are the same number of words in $L_o$ with each of these lengths, *i.e.*, by $6k+2$, $6k+3$, $6k+4$ and $6k+5$.) On the other hand, with states 6 and 9 all the words of $L_o$ with length $6k+6$ are accepted. Notice that their number is one more than the number of the words with length $6k+5$. The number of words with length $6k+7$ is again the same as the number of words with $6k+5$, and exactly those are accepted by state 7. By reaching state 2, and length $6k+8$, the number of words is one more than the number of words with length $6k+7$, and they are accepted by state 2, as we have already seen.

Consequently the automaton accepts exactly $L_o$. □

Now based on the above result, relations between 2detLIN and various classes of languages accepted by qD sensing $5' \to 3'$ WK automata are established.

**Theorem 7.20.** ***qD1WK*** $\supsetneq$ *2detLIN.*
*Further, each of the classes* ***qDF1WK***, ***qDFSWK***, ***qDFWK*** *and* ***qDNWK*** *is incomparable with the class 2detLIN.*

*Proof.* To show the above relations we show that the language accepted by the qDF1 sensing $5' \to 3'$ WK automaton shown in Figure 3 (right) is not in 2detLIN. On the one hand, in this way, we complement the result of Proposition 7.16 by proving properness of the inclusion, and, on the other hand, this leads to the incomparability results based also on Propositions 7.15 and 5.4.

The F1 automaton is a variant of the automaton discussed in Lemma 7.19. In fact it accepts the same language with some "extra" words by states 1 and 5: by state 1 it accepts also $a$, while with state 5, along some words of $L_o$, it also accepts one word in each cycle which does not belong to $L_o$, in the $k$-th cycle ($k \geq 1$) it is $a^{3k}b^{3k-1}$. Let us denote this language by $L'_o$.

Now, we shall prove that $L'_o$ is not in 2detLIN and our proof goes by contradiction.

Let us assume that a deterministic sensing $5' \to 3'$ WK automaton $A$ with initial state $q$ accepts $L'_o$. Let $k$ be the number of states of $A$ and let $r$ be the maximal length of the words $A$ may read in a transition (radius of $A$). Let $m = 6kr$. Consider the words $a^m b^m$ and $a^m b^{2m}$ of $L'_o$. Their accepting computation must start exactly in the same way in the first $4k$ steps ($A$ may read at most $4kr$ $a$-s by the first head and at most that many $b$-s by the second head during this part of the computation). Thus, there is a state which appears more than one configurations in this part of the computation, let it be $p$ (may be the same as $q$). This part of the computations on the two above words can be written as $(q, a^m b^m) \Rightarrow^* (p, a^{m-i_1} b^{m-j_1}) \Rightarrow^* (p, a^{m-i_2} b^{m-j_2}) \Rightarrow^* (f_1, \lambda)$ and $(q, a^m b^{2m}) \Rightarrow^* (p, a^{m-i_1} b^{2m-j_1}) \Rightarrow^* (p, a^{m-i_2} b^{2m-j_2}) \Rightarrow^* (f_2, \lambda)$ with accepting states $f_1, f_2$ (not necessarily different states). Moreover, the computation, as it is deterministic, for such long inputs must go through the cycle more than once (for us it is enough to use that at least twice). Let $x = i_2 - i_1$ and $y = j_2 - j_1$, *i.e.*, the number of $a$-s and $b$-s read in the cycle. Clearly, $A$ also accepts the

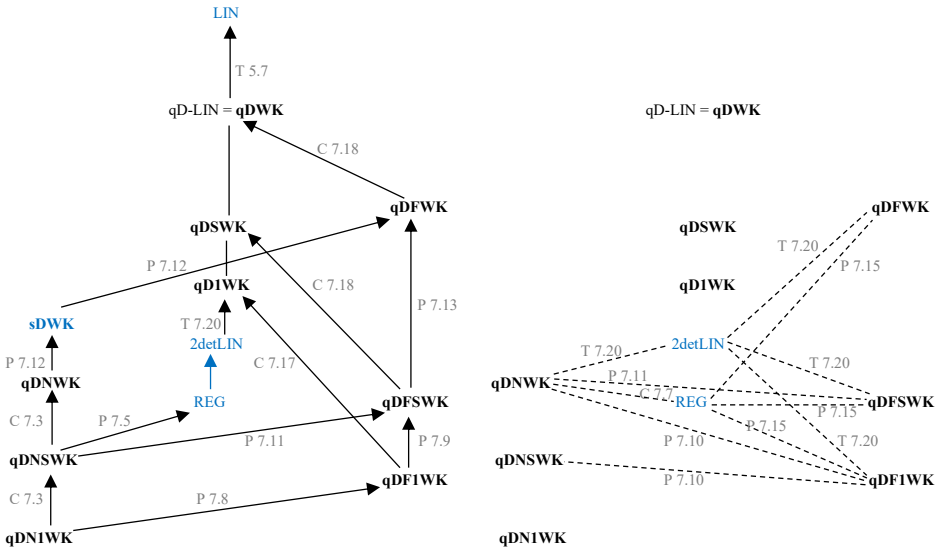


FIGURE 4. A hierarchy of the language classes of quasi-deterministic sensing $5' \to 3'$ WK automata in a Hasse diagram (left). The abbreviations **qDWK** refers to the language classes accepted by quasi-deterministic $5' \to 3'$ WK automata; S, F, N, 1 and their combinations are used to abbreviate the restricted variants of qD sensing $5' \to 3'$ WK automata. Arrows show proper inclusions, while lines without arrow head show inclusions where the properness is left open. On the right, incomparable classes are connected. For the proven inclusions and incomparability relations the number next to the arrow/line shows the number of the given Theorem, Proposition or Corollary. Blue color shows some related language classes, *e.g.*, **sDWK** refer to the language classes accepted by state-deterministic $5' \to 3'$ WK automata.

following words (by taking two less/one less/one more/two more times) of the above mentioned cycle of the computation: $a^{m-2x}b^{m-2y}$, $a^{m-2x}b^{2m-2y}$, $a^{m-x}b^{m-y}$, $a^{m-x}b^{2m-y}$, $a^{m+x}b^{m+y}$, $a^{m+x}b^{2m+y}$, $a^{m+2x}b^{m+2y}$ and $a^{m+2x}b^{2m+2y}$, respectively. Let us analyse the relation of $x$ and $y$.

- If $x > y$, then $2x > 2y + 1$ and thus $a^{m+2x}b^{m+2y}$ is definitely not in $L'_o$. (Notice that the only words of $L'_o$ that have larger number of $a$-s than $b$-s are the "extra" ones accepted by state 5 of the automaton in Figure 3 right, having exactly one more $a$-s than $b$-s.)
- In the case $x = y$, then $a^{m-x}b^{m-y} \notin L'_o$, as $m - x$ is less than the half of $2m - y = 2m - x$, and $L'_o$ does not contain any words with this property.
- Finally, if $x < y$, then $m - 2x > m - 2y + 1$, *i.e.*, the word $a^{m-2x}b^{m-2y}$ contains more $a$-s than $b$-s by at least two and thus it should not be accepted as it is not in $L'_o$.

As in each case we have proven the acceptance of a word not in $L'_o$, the proof is completed, and no deterministic sensing $5' \to 3'$ WK automaton $A$ can accept $L'_o$. Thus, the statements of the theorem are proven. □

By definition of the restricted variants, we also know the following:

**Corollary 7.21.** ***qDSWK*** $\subset$ ***qDWK*** $=$ *qD-LIN* *and* ***qDSWK*** $\supset$ ***qD1WK***.

## 8. CONCLUSIONS

We have considered a kind of generalisation of determinism in the case of finite state machines. Quasi-determinism may allow a type of nondeterminism that is based on the input being processed during a

computation step. In a quasi-deterministic automaton, the state of the next configuration in the computation is determined, but the next configuration itself may not be. This new type of determinism also generalise the recently introduced notion of state-determinism. We have shown that quasi-determinism is entirely the same as determinism in the case of $\lambda$-transition free NFA. On the other hand, for WK automata, especially, for sensing $5' \to 3'$ WK automata, because of the two heads and the string-reading feature, it is more interesting. Knowing that the family of sensing $5' \to 3'$ WK automata accept the family of linear context-free languages, we have shown that a new sublinear language class is accepted by the new model, that is a proper superclass of the class 2detLIN of languages accepted by deterministic sensing $5' \to 3'$ WK automata. Some (non)closure properties of this class are also proven.

We have also studied various restricted classes, and proved various hierarchy results among them, for their summary see Figure 4. The properness of two of the inclusions, **qDSWK** $\subset$ **qDWK** and **qDSWK** $\supset$ **qD1WK**, are left for the future research. Quasi-determinism may also be expanded to various other types of automata, including models based on $5' \to 3'$ WK automata, *e.g.*, automata with multiple runs [11], jumping $5' \to 3'$ WK automata [41], 2-head/linear automata with translucent letters [42, 43] and $5' \to 3'$ WK transducers [44, 45]. It is known that for context-free languages stateless (*i.e.*, one-state) pushdown automaton suffice. Moreover, for a class of mildly context-sensitive languages a 2-head extension of the pushdown automata can be used and stateless variants are already accepting the language class [46, 47]. In case of these models the state of each configuration is the sole state, and thus, these models are automatically fulfill the definition of both state-determinism and quasi-determinism. On the other hand by adding other type of restriction on these models, *e.g.*, having only one stack symbol by obtaining counter machines, the states become important and the new concepts make sense. Therefore, it is a task of future research to apply and establish the computational power of these models combined by the new types of "determinism" investigated here. 2-head models are also opt to accept circular words [48], it could also be interesting to see how the classes of accepted circular words change when the quasi-deterministic variants are considered.

## Acknowledgments

The preliminary version of the paper was presented in NCMA2022: 12th International Workshop on Non-Classical Models of Automata and Applications, Debrecen, Hungary, [49]. Comments of the reviewers of both the conference and journal versions and of the audience of NCMA2022 are gratefully acknowledged.

## Funding

This research does not received any funding.

## Conflicts of interest

The author reports no conflict of interest.

## Data availability statement

This is a theoretical paper, no data were used. All examples are fully described in the paper itself.

## Author contribution statement

The sole author had the idea, worked on the topic, proved the theorems and other statements and wrote the paper.

## References

[1] J.-É. Pin (ed.), *Handbook of Automata Theory*, 2 vols. EMS Press, 2021.
[2] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*. Springer (1997).
[3] L.M. Adleman, Molecular computation of solutions to combinatorial problems. *Science* **226** (1994) 1021–1024.
[4] R.J. Lipton, DNA solution of hard computational problems. *Science* **268** (1995) 542–545.
[5] G. Păun, G. Rozenberg and A. Salomaa, DNA Computing: New Computing Paradigms. Springer-Verlag (2002).

[6] E. Czeizler and E. Czeizler, A Short Survey on Watson–Crick Automata. *Bull. EATCS* **88** (2006) 104–119.

[7] R. Freund, G. Păun, G. Rozenberg and A. Salomaa, Watson–Crick finite automata, in *3rd DIMACS Symposium On DNA Based Computers*, Philadelphia (1997) 305–317.

[8] J.M. Sempere, A Representation theorem for languages accepted by Watson–Crick finite automata. *Bull. EATCS* **83** (2004) 187–191.

[9] J.M. Sempere, On the application of Watson–Crick finite automata for the resolution of bioinformatic problems, in *Tenth Workshop on Non-Classical Models of Automata and Applications*, NCMA 2018, edited by R. Freund, M. Hospodár, G. Jirásková and G. Pighizzini. Österreichische Computer Gesellschaft Invited talk (2018) 29–30.

[10] L. Hegedüs, B. Nagy and Ö. Egecioglu, Stateless multicounter $5' \to 3'$ Watson–Crick automata: the deterministic case. *Natural Comput.* **11** (2012) 361–368.

[11] P. Leupold and B. Nagy, $5' \to 3'$ Watson–Crick automata with several runs. *Fundam. Inform.* **104** (2010) 71–91.

[12] B. Nagy, On $5' \to 3'$ sensing Watson–Crick finite automata, in *DNA Computing. DNA 2007: Selected revised papers, Lecture Notes in Computer Science, LNCS*, vol. 4848, edited by M.H. Garzon and H. Yan. Springer, Berlin, Heidelberg (2008) 256–262.

[13] B. Nagy, On a hierarchy of $5' \to 3'$ sensing WK finite automata languages, in *Computaility in Europe, CiE 2009: Mathematical Theory and Computational Practice, Abstract Booklet*, Heidelberg, edited by K. Ambos-Spies, B. Löve and W. Merkle (2009) 266–275.

[14] B. Nagy, $5' \to 3'$ sensing Watson–Crick finite automata, in *Sequence and Genome Analysis II – Methods and Applications*, edited by G. Fung. iConcept Press (2010) 39–56.

[15] B. Nagy, On a hierarchy of $5' \to 3'$ sensing Watson–Crick finite automata languages. *J. Logic Computat.* **23** (2013) 855–872.

[16] B. Nagy and S. Parchami, $5' \to 3'$ Watson–Crick automata languages – without the sensing parameter. *Natural Comput.* **21** (2022) 679–691.

[17] B. Nagy, S. Parchami H.M. Mohammad Sadeghi, A new sensing $5' \to 3'$ Watson–Crick automata concept, in *Proceedings 15th International Conference on Automata and Formal Languages, AFL 2017*, *Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 252, edited by E. Csuhaj-Varjú, P. Dömösi and Gy. Vaszil (2017) 195–204.

[18] E. Czeizler, E. Czeizler, L. Kari and K. Salomaa, Watson–Crick automata: determinism and state complexity, in *10th International Workshop on Desciptional Complexity of Formal Systems, DCFS 2008*, edited by C. Campeanu and G. Pighizzini. University of Prince Edward Island (2008) 121–133.

[19] E. Czeizler, E. Czeizler, L. Kari and K. Salomaa, On the descriptional complexity of Watson–Crick automata. *Theor. Comput. Sci.* **410** (2009) 3250–3260.

[20] K. Sankar Ray, K. Chatterjee and D. Ganguly, State complexity of deterministic Watson–Crick automata and time varying Watson–Crick automata. *Natural Comput.* **14** (2015) 691–699.

[21] D. Kuske and P. Weigel, The role of the complementarity relation in Watson–Crick automata and sticker systems, in *Developments in Language Theory, DLT 2004, Lecture Notes in Computer Science, LNCS*, vol. 3340, edited by C.S. Calude, E. Calude and M.J. Dinneen. Springer, Berlin, Heidelberg (2004) 272–283.

[22] B. Nagy and S. Parchami, On deterministic sensing $5' \to 3'$ Watson–Crick finite automata: a full hierarchy in 2detLIN. *Acta Inform.* **58** (2021) 153–175.

[23] S. Parchami and B. Nagy, Deterministic sensing $5' \to 3'$ Watson–Crick automata without sensing parameter, in *Unconventional Computation and Natural Computation, UCNC 2018 LNCS*, vol. 10867, edited by S. Stepney and S. Verlan. Springer (2018) 173–187.

[24] B. Nagy, State-deterministic $5' \to 3'$ Watson–Crick automata. *Natural Comput.* **20** (2021) 725–737.

[25] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979).

[26] M. Fatima and B. Nagy, Transduced-input automata with translucent letters. *Comptes Rendus Acad. Bul. Sci.* **73** (2020) 33–39.

[27] B. Nagy and F. Otto, Finite-state acceptors with translucent letters, in *BILC 2011: AI Methods for Interdisciplinary Research in Language and Biology, Proceeding*, edited by G. Bel-Enguix, V. Dahl and A.O. De La Puente SciTePress, Portugal (2011) 3–13.

[28] B. Nagy and F. Otto, On CD-systems of stateless deterministic R-automata with window size one. *J. Comput. Syst. Sci.* **78** (2012) 780–806.

[29] F. Otto and F. Mráz, Non-returning finite automata with translucent letters. *12th International Workshop on Non-Classical Models of Automata and Applications, NCMA 2022, Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 367 (2022) 143–159.

[30] H. Chigahara, S. Zsolt Fazekas and A. Yamamura, One-way jumping finite automata. *Int. J. Found. Comput. Sci.* **27** (2016) 391–405.

[31] A. Meduna and P. Zemek, Jumping finite automata. *Int. J. Found. Comput. Sci.* **23** (2012) 1555–1578.

[32] B. Nagy, State-deterministic finite automata with translucent letters and finite automata with nondeterministically translucent letters. *16th International Conference on Automata and Formal Languages (AFL 2023)*, *EPTCS*, vol. 386 (2023) 170–184.

[33] R. Loukanova, Linear context free languages, in *Theoretical Aspects of Computing – ICTAC 2007, 4th International Colloquium, Macau, China, September 26–28, 2007, Proceedings. Lecture Notes in Computer Science*, vol. 4711, edited by C.B. Jones, Z. Liu, J. Woodcock. Springer (2007) 351–365.

[34] B. Nagy, A class of 2-head finite automata for linear languages. *Triangle*, vol. 8 (Languages. Mathematical Approaches) (2012) 89–99.

[35] B. Nagy, On language classes accepted by stateless $5' \to 3'$ Watson–Crick finite automata. *Ann. Math. Inform.* **58** (2023) 110–120.

[36] S. Dieck and S. Verwer, On bidirectional deterministic finite automata. *CIAA 2024: 28th International Conference on Implementation and Application of Automata, LNCS*, vol. 15015 (2024) 109–123.

[37] S. Dieck and S. Verwer, An active learning algorithm for bidirectional deterministic finite automata. *CIAA 2025: 29th International Conference on Implementation and Application of Automata, LNCS*, vol. 15981 (2025) 99–114.

[38] B. Nagy, A Myhill–Nerode type characterization of 2detLIN languages. *Proceedings 15th International Workshop on Non-Classical Models of Automata and Applications (NCMA 2025), EPTCS*, vol. 422 (2025) 73–88.

[39] B. Nagy and W. Yasin, On some classes of reversible 2-head automata. *Proceedings 15th International Workshop on Non-Classical Models of Automata and Applications (NCMA 2025), EPTCS*, vol. 422 (2025) 89–103.

[40] R.J. Parikh, Language generating devices. *MIT Res. Lab. Quart. Progr. Rep.* **60** (1961) 199–212.

[41] R. Kocman, Z. Krivka, A. Meduna and B. Nagy, A jumping $5' \to 3'$ Watson–Crick finite automata model. *Acta Inform.* **59** (2022) 557–584.

[42] B. Nagy and F. Otto, Two-head finite-state acceptors with translucent letters, in *SOFSEM 2019: Theory and Practice of Computer Science*, *LNCS*, vol. 11376, edited by B. Catania, R. Královic, J. Nawrocki and G. Pighizzini. Springer (2019) 406–418.

[43] B. Nagy and F. Otto, Linear automata with translucent letters and linear context-free trace languages. *RAIRO Theor. Inform. Appl.* **54** (2020) 3.

[44] B. Nagy and Z. Kovács, On simple $5' \to 3'$ sensing Watson–Crick finite-state transducers, in *Eleventh Workshop on Non-Classical Models of Automata and Applications, NCMA 2019*, edited by R. Freund, M. Holzer and J.M. Sempere. Österreichische Computer Gesellschaft (2019) 155–170.

[45] B. Nagy and Z. Kovács, On deterministic 1-limited $5' \to 3'$ sensing Watson–Crick finite-state transducers. *RAIRO Theor. Inform. Appl.* **55** (2021) 5

[46] B. Nagy, A family of 2-head pushdown automata, in *Proceddings of Seventh Workshop on Non-Classical Models of Automata and Applications, NCMA 2015*, Porto, Portugal, Österreichische Computer Gesellschaft (2015) 177–191.

[47] B. Nagy, $5' \to 3'$ Watson–Crick pushdown automata. *Inf. Sci.* **537** (2020) 452–466.

[48] B. Nagy, $5' \to 3'$ Watson–Crick automata accepting necklaces. 14th International Workshop on Non-Classical Models of Automata and Applications, NCMA 2024, Electronic Proceedings in Theoretical Computer Science, EPTCS, vol. 407 (2024) 168–185.

[49] B. Nagy, Quasi-deterministic $5' \to 3'$ Watson–Crick automata, In *12th International Workshop on Non-Classical Models of Automata and Applications, NCMA 2022, Electronic Proceedings in Theoretical Computer Science, EPTCS*, vol. 367 (2022) 160–176.