A JUMPING $5' \to 3'$ WATSON-CRICK FINITE AUTOMATA MODEL

Radim Kocman[(A)] Benedek Nagy[(B)]
Zbyněk Křivka[(A)] Alexander Meduna[(A)]

[(A)] Centre of Excellence IT4Innovations, Faculty of Information Technology,
Brno University of Technology, Božetěchova 2, Brno Czech Republic
{ikocman,krivka,meduna}@fit.vutbr.cz

[(B)] Department of Mathematics, Faculty of Arts and Sciences,
Eastern Mediterranean University, Famagusta, North Cyprus, Mersin-10, Turkey
nbenedek.inf@gmail.com

Abstract
This paper introduces and studies a combined model of jumping finite automata and sensing $5' \to 3'$ Watson-Crick finite automata. The accepting power of the new model is compared with the original models and also with some well-known language families. Furthermore, the paper investigates changes in the accepting power when restrictions are applied on the model.

1. Introduction

In recent years, several papers studied finite automata models with multiple heads that process the input string in non-conventional ways (see [2, 3, 7, 8, 9, 10]). Traditionally, when the model utilizes several heads, either each head works on its own tape, or all heads read the same input string in a symbol-by-symbol left-to-right way. In contrast, there are also well-established formal grammars that generate strings in a parallel way, but this process is usually very different than the reading with several heads. In the grammars, the sentential form is repeatedly rewritten on several places at once until the process creates the final string. The presented finite automata models with the non-conventional processing have their behavior set somewhere between the mentioned models. They utilize several heads, but these heads cooperate on a single tape to process the single input string. Therefore, every symbol in the input string is read only once, and the heads do not work in the traditional symbol-by-symbol left-to-right way.

The first group of these models is based on *jumping finite automata* (see [5, 6, 2, 3]). This concept is in its core focused on discontinuous information processing. In essence, a jumping finite automaton works just like a classical finite automaton except it does not read the input string in a symbol-by-symbol left-to-right way. After the automaton reads a symbol, the head can jump over (skip) a portion of the tape in either direction. Once an occurrence of a symbol is read on the tape, it cannot be re-read again later. Generally, this model can very easily define even non-context-free languages if the order of symbols is unimportant for the language. On the other hand, the resulting language families of these models are usually incomparable with the classical families of regular, linear, and context-free languages. When this concept utilizes multiple heads, the heads can naturally jump on specific positions in the tape, and thus they can easily work on different places at once in parallel.

The second group is represented by *sensing* $5' \to 3'$ *Watson-Crick (WK) finite automata* (see [7, 8, 9, 10, 11]). This is a biology-inspired concept. In essence, a WK automaton also works just like a classical finite automaton except it uses a WK tape (i.e., double-stranded tape), and it has a separate head for each of the two strands in the tape. This is therefore a concept that always naturally uses two heads. In a $5' \to 3'$ WK automaton, both heads read their specific strand in the biochemical $5'$ to $3'$ direction. In a computing point of view, however, this means that they read the double strand sequence in opposite directions. Finally, a $5' \to 3'$ WK automaton is *sensing* if the heads sense that they are meeting each other, and the processing of the input ends if for all pairs of the sequence one of the letters is read. Sensing $5' \to 3'$ WK automata generally accept the family of linear languages.

Even though that these concepts are significantly different, their models sometimes work in a very similar way. Both concepts are also not mutually exclusive in a single formal model. This paper defines *jumping* $5' \to 3'$ *WK automata*—a combined model of jumping finite automata and sensing $5' \to 3'$ WK automata—and studies their characteristics. We primarily investigate the accepting power of the model and also the effects of restrictions on the model.

2. Preliminaries

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [4, 13]). This section recalls only the crucial notions used in this paper.

For a set Q, $\mathrm{card}(Q)$ denotes the cardinality of Q, and 2^Q denotes the power set of Q. For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Members of V^* are called *strings*. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For $x \in V^*$, $|x|$ denotes the length of x, and $\mathrm{alph}(x)$ denotes the set of all symbols occurring in x; for instance, $\mathrm{alph}(0010) = \{0, 1\}$. For $a \in V$, $|x|_a$ denotes the number of occurrences of a in x. Let X and Y be sets; we call X and Y to be *incomparable* if $X \not\subseteq Y$, $Y \not\subseteq X$, and $X \cap Y \neq \emptyset$.

A *general grammar* or, more simply, a *grammar* is quadruple $G = (N, T, S, P)$, where N and T are alphabets such that $N \cap T = \emptyset$, $S \in N$, and P is a finite set of rules of the form $x \to y$, where $x, y \in (N \cup T)^*$ and $\mathrm{alph}(x) \cap N \neq \emptyset$. If $x \to y \in P$ and $u, v \in (N \cup T)^*$, then $uxv \Rightarrow uyv\ [x \to y]$, or simply $uxv \Rightarrow uyv$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language generated by G, $L(G)$, is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$. We recognize several special cases of grammars: G is a *context-sensitive grammar* if every $x \to y \in P$ satisfies $x = \alpha A \beta$ and $y = \alpha y \beta$ such that

$A \in N$, $\alpha, \beta \in (N \cup T)^*$, and $y \in (N \cup T)^+$. G is a *context-free grammar* if every $x \to y \in P$ satisfies $x \in N$. G is a *linear grammar* if every $x \to y \in P$ satisfies $x \in N$ and $y \in T^*NT^* \cup T^*$. G is a *regular grammar* if every $x \to y \in P$ satisfies $x \in N$ and $y \in TN \cup T$.

A *finite automaton* is a quintuple $A = (V, Q, q_0, F, \delta)$, where V is an input alphabet, Q is a finite set of states, $V \cap Q = \emptyset$, $q_0 \in Q$ is the initial (or start) state, and $F \subseteq Q$ is a set of final (or accepting) states. The mapping δ is a transition function. If $\delta \colon Q \times (V \cup \{\varepsilon\}) \to 2^Q$, then the device is non-deterministic; if $\delta \colon Q \times V \to Q$, then the automaton is deterministic. A string w is accepted by a finite automaton if there is a sequence of transitions starting from q_0, ending in a state in F, and the symbols of the sequence yield w. A language is regular if and only if it can be recognized by a finite automaton.

Let **REG**, **LIN**, **CF**, and **CS** denote the families of regular, linear, context-free, and context-sensitive languages, respectively.

2.1. Jumping Finite Automata

A *general jumping finite automaton* (see [5, 6]), a *GJFA* for short, is a quintuple $M = (Q, \Sigma, R, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $s \in Q$ is the start state, and $F \subseteq Q$ is a set of final states. Members of R are referred to as rules of M. If $(p, y, q) \in R$ implies that $|y| \leq 1$, then M is a *jumping finite automaton*, a *JFA* for short. A configuration of M is any string in $\Sigma^* Q \Sigma^*$. The binary jumping relation, symbolically denoted by $\curvearrowright$, over $\Sigma^* Q \Sigma^*$, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $(p, y, q) \in R$; then, M makes a *jump* from $xpyz$ to $x'qz'$, symbolically written as $xpyz \curvearrowright x'qz'$. In the standard manner, extend $\curvearrowright$ to $\curvearrowright^n$, where $n \geq 0$; then, based on $\curvearrowright^n$, define $\curvearrowright^+$ and $\curvearrowright^*$. The language accepted by M, denoted by $L(M)$, is defined as $L(M) = \{uv : u, v \in \Sigma^*,\ usv \curvearrowright^* f,\ f \in F\}$. We say that M accepts w if and only if $w \in L(M)$. M rejects w if and only if $w \in \Sigma^* - L(M)$.

More recently, *double-jumping modes* for GJFAs were introduced (see [2]), which perform two single jumps simultaneously. Both jumps always follow the same rule, however, they are performed on two different positions on the tape and thus handle different parts of the input string. Additionally, these jumps cannot ever cross each other (i.e., the initial mutual order of reading positions is preserved during the whole accepting process). The specific double-jumping modes then assign one of the three jumping directions to each of the two jumps—(1) to the left, (2) to the right, and (3) in either direction. We omit the precise formal definition.

2.2. Watson-Crick Finite Automata

In this part we recall some well-known concepts of DNA computing and related formal language theory. Readers who are not familiar in these topics should read [11].

Let V be an alphabet and $\rho \subseteq V \times V$ be its complementary relation. For instance, $V = \{A, C, G, T\}$ is usually used in DNA computing with the Watson-Crick complementary relation

$\{(T,A),(A,T),(C,G),(G,C)\}$. The strings built up by complementary pairs of letters are double strands (of DNA).

A *Watson-Crick finite automaton* (or shortly, a *WK automaton*) is a finite automaton working on a Watson-Crick tape, that is, a double-stranded sequence (or molecule) in which the lengths of the strands are equal and the elements of the strands are pairwise complements of each other: $\left[\begin{smallmatrix}a_1\\b_1\end{smallmatrix}\right]\left[\begin{smallmatrix}a_2\\b_2\end{smallmatrix}\right]\dots\left[\begin{smallmatrix}a_n\\b_n\end{smallmatrix}\right] = \left[\begin{smallmatrix}a_1a_2\dots a_n\\b_1b_2\dots b_n\end{smallmatrix}\right]$ with $a_i, b_i \in V$ and $(a_i,b_i) \in \rho$ $(i = 1,\dots,n)$. The notation $\left[\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right]$ is used only for strings w_1, w_2 with equal length and satisfying the complementary relation ρ. The set of all double-stranded strings with this property is denoted by $\mathrm{WK}_\rho(V)$. For double-stranded strings for which these conditions are not necessarily satisfied, the notation $\left(\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right)$ is used throughout the paper. Formally, a WK automaton is $M = (V,\rho,Q,q_0,F,\delta)$, where V, Q, q_0, and F are the same as in finite automata, $\rho \subseteq V \times V$ is a symmetric relation, and the transition mapping $\delta\colon (Q \times \left(\begin{smallmatrix}V^*\\V^*\end{smallmatrix}\right)) \to 2^Q$ in such a way that $\delta(q,\left(\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right))$ $(q \in Q,\ w_1,w_2 \in V^*)$ is non-empty only for finitely many values of $(q,\left(\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right))$.

The elementary difference between finite automata and WK automata, besides the doubled tape, is the number of heads. WK automata scan each of the two strands separately with a unique head. In classical WK automata, the processing of the input sequence ends if all pairs of the sequence are read with both heads. There are also some restricted variations of WK automata which are widely used in the literature (see, e.g., [11]):

- **N** : stateless, i.e., with only one state: if $Q = F = \{q_0\}$;
- **F** : all-final, i.e., with only final states: if $Q = F$;
- **S** : simple (at most one head moves in a step) $\delta\colon (Q \times (\left(\begin{smallmatrix}V^*\\\{\varepsilon\}\end{smallmatrix}\right) \cup \left(\begin{smallmatrix}\{\varepsilon\}\\V^*\end{smallmatrix}\right))) \to 2^Q$;
- **1** : 1-limited (exactly one letter is being read in a step) $\delta\colon (Q \times (\left(\begin{smallmatrix}V\\\{\varepsilon\}\end{smallmatrix}\right) \cup \left(\begin{smallmatrix}\{\varepsilon\}\\V\end{smallmatrix}\right))) \to 2^Q$.

Further variations such as **NS**, **FS**, **N1**, and **F1** WK automata can be identified in a straightforward way by using multiple constraints.

In $5' \to 3'$ *WK automata* (see [7, 8, 9, 10]), both heads start from the $5'$ end of the appropriate strand. Physically/mathematically and from a computing point of view they read the double-stranded sequence in opposite directions, while biochemically they go to the same direction. A $5' \to 3'$ WK automaton is *sensing* if the heads sense that they are meeting (i.e., they are close enough to meet in the next step or there is a possibility to read strings at overlapping positions). In sensing $5' \to 3'$ WK automata, the processing of the input sequence ends if for all pairs of the sequence one of the letters is read. Due to the complementary relation, the sequence is fully processed; thus, the automaton makes a decision on the acceptance.

In the usual WK automata, the state transition is a mapping of the form $(Q \times \left(\begin{smallmatrix}V^*\\V^*\end{smallmatrix}\right)) \to 2^Q$. In a transition $q' \in \delta(q,\left(\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right))$, we call $r_l = |w_1|$ and $r_r = |w_2|$ the left and right *radius* of the transition (they are the lengths of the strings that the heads read from *left to right* and from *right to left* in this step, respectively). The value $r = r_l + r_r$ is the radius of the transition. Since $\delta(q,\left(\begin{smallmatrix}w_1\\w_2\end{smallmatrix}\right))$ is non-empty only for finitely many triplets of (q,w_1,w_2), there is a transition (maybe more) with the maximal radius for a given automaton. Let δ be extended by the sensing condition in the following way: Let r be the maximum of the values $r_l + r_r$ for the values given in the transition function of the original WK automaton. Then, let $\delta'\colon (Q \times \left(\begin{smallmatrix}V^*\\V^*\end{smallmatrix}\right) \times D) \to 2^Q$, where D is the *sensing distance set* $\{-\infty, 0, 1, \ldots, r, +\infty\}$. This set gives the distance of the two heads between 0 and r, $+\infty$ when the heads are further than r, or $-\infty$ when the heads are after their meeting point. Trivially, this automaton is finite, and D can be used only to control the sensing (i.e., the appropriate meeting of the heads). To describe the work of the automata, we use the concept of configuration. A configuration $\left(\begin{smallmatrix} w_1 \\ w_2 \end{smallmatrix}\right)(q, s)\left(\begin{smallmatrix} w_1' \\ w_2' \end{smallmatrix}\right)$ consists of the state q, the actual sensing distance s, and the input $\left[\begin{smallmatrix} w_1 w_1' \\ w_2 w_2' \end{smallmatrix}\right] \in \mathrm{WK}_\rho(V)$ in such a way that the first head (upper strand) has already processed the part w_1, while the second head (lower strand) has already processed w_2'. A step of the automaton, according to the state transition function, can be of the following two types:

(1) Normal steps : $\left(\begin{smallmatrix} w_1 \\ w_2 y \end{smallmatrix}\right)(q, +\infty)\left(\begin{smallmatrix} x w_1' \\ w_2' \end{smallmatrix}\right) \Rightarrow \left(\begin{smallmatrix} w_1 x \\ w_2 \end{smallmatrix}\right)(q', s)\left(\begin{smallmatrix} w_1' \\ y w_2' \end{smallmatrix}\right)$, for $w_1, w_2, w_1', w_2', x, y \in V^*$ with $|w_2 y| - |w_1| > r$, $q, q' \in Q$, if and only if $\left[\begin{smallmatrix} w_1 x w_1' \\ w_2 y w_2' \end{smallmatrix}\right] \in \mathrm{WK}_\rho(V)$ and $q' \in \delta(q, \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right), +\infty)$, and $s = \begin{cases} |w_2| - |w_1 x| & \text{if } |w_2| - |w_1 x| \le r; \\ +\infty & \text{in other cases.} \end{cases}$

(2) Sensing steps : $\left(\begin{smallmatrix} w_1 \\ w_2 y \end{smallmatrix}\right)(q, s)\left(\begin{smallmatrix} x w_1' \\ w_2' \end{smallmatrix}\right) \Rightarrow \left(\begin{smallmatrix} w_1 x \\ w_2 \end{smallmatrix}\right)(q', s')\left(\begin{smallmatrix} w_1' \\ y w_2' \end{smallmatrix}\right)$, for $w_1, w_2, w_1', w_2', x, y \in V^*$, if and only if $\left[\begin{smallmatrix} w_1 x w_1' \\ w_2 y w_2' \end{smallmatrix}\right] \in \mathrm{WK}_\rho(V)$ and $q' \in \delta(q, \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right), s)$, and $s' = \begin{cases} s - |x| - |y| & \text{if } s - |x| - |y| \ge 0; \\ -\infty & \text{in other cases.} \end{cases}$

In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \ge 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The accepted language, denoted by $L(M)$, can be defined by the final accepting configurations that can be reached from the initial one: A double strand $\left[\begin{smallmatrix} w_1 \\ w_2 \end{smallmatrix}\right]$ is accepted by a sensing $5' \to 3'$ WK automaton M if and only if $\left(\begin{smallmatrix} \varepsilon \\ w_2 \end{smallmatrix}\right)(q_0, s_0)\left(\begin{smallmatrix} w_1 \\ \varepsilon \end{smallmatrix}\right) \Rightarrow^* \left[\begin{smallmatrix} w_1' \\ w_2' \end{smallmatrix}\right](q_f, 0)\left[\begin{smallmatrix} w_1'' \\ w_2'' \end{smallmatrix}\right]$, for $q_f \in F$, where $\left[\begin{smallmatrix} w_1' \\ w_2' \end{smallmatrix}\right]\left[\begin{smallmatrix} w_1'' \\ w_2'' \end{smallmatrix}\right] = \left[\begin{smallmatrix} w_1 \\ w_2 \end{smallmatrix}\right]$ with the proper value of s_0 (it is $+\infty$ if $|w_1| > r$, elsewhere it is $|w_1|$); since the full input is processed by the time the heads meet.

From a biochemical point of view, a double-stranded sequence has no distinguishable start and end. Consequently, each word that is accepted by a WK automaton has a complement-symmetric pair which is also in the language. This fact does not cause any problem in connection to formal language theory. For instance, double strands having only A and C in a strand (and thus having T and G in the other) can represent languages over a binary alphabet: considering the pair $\left[\begin{smallmatrix} A \\ T \end{smallmatrix}\right]$ as letter a and $\left[\begin{smallmatrix} C \\ G \end{smallmatrix}\right]$ as letter b in the new alphabet V'.

At the end, we briefly mention other closely related $5' \to 3'$ WK automata models. Besides the sensing version, the papers [7, 8, 9] also define the *full-reading sensing version.* The formal definition remains practically identical, however, the automaton continues with the reading after the meeting point, and both heads have to read the whole strand from the $5'$ end to the $3'$ end. The resulting behavior therefore combines some properties of classical WK automata and sensing $5' \to 3'$ WK automata. It can be easily seen that the full-reading sensing version is generally stronger than the sensing version. Lastly, the paper [10] introduces a version of sensing $5' \to 3'$ WK automata without the sensing distance. It shows that it is not strictly necessary to know the precise sensing distance and that we can obtain the same power even if we are able to recognize only the actual meeting event. Nonetheless, this result does not hold in general if we consider the restricted variations of these models.

3. Definitions

Considering sensing $5' \to 3'$ WK automata and full-reading sensing $5' \to 3'$ WK automata, there is quite a large gap between their behaviors. The definition of sensing $5' \to 3'$ WK automata states that we need to read only one of the letters from all pairs of the input sequence before it is fully processed. However, this also limits the positioning of the heads because they can read letters only until they meet. On the other hand, the definition of full-reading sensing $5' \to 3'$ WK automata allows the heads to traverse the whole input. Nonetheless, this also means that all pairs of the input sequence will be read twice. Considering other models, jumping finite automata offer a mechanism that allows heads to skip (jump over) some symbols. Moreover, in some of the recently introduced double-jumping modes, these automata behave very similarly to $5' \to 3'$ WK automata. It is therefore our goal to fill the gap by introducing the jumping mechanism into sensing $5' \to 3'$ WK automata. We want the heads to be able to traverse the whole input, but we also want to read all pairs of the input sequence only once.

It is possible to fit the jumping mechanism straightforwardly into the original definition of sensing $5' \to 3'$ WK automata. Observe that we are also newly tracking only the meeting event of the heads and not the precise sensing distance.

Definition 3.1 *A* sensing $5' \to 3'$ WK automaton with jumping feature *is a 6-tuple* $M = (V, \rho, Q, q_0, F, \delta)$*, where* V*,* ρ*,* Q*,* q_0*, and* F *are the same as in WK automata,* $V \cap \{\#\} = \emptyset$*,* $\delta\colon (Q \times \binom{V^*}{V^*} \times D) \to 2^Q$*, where* $D = \{\oplus, \ominus\}$ *indicates the mutual position of heads, and the transition function assigns a nonempty set only for finitely many triplets of* $(Q \times \binom{V^*}{V^*} \times D)$*. We denote the head as ▶-head or ◀-head if it reads from left to right or from right to left, respectively. We use symbol* $\oplus$ *if the ▶-head is on the input tape positioned before the ◀-head; otherwise, we use symbol* $\ominus$*. A configuration* $\binom{w_1}{w_2}(q, s)\binom{w_1'}{w_2'}$ *has the same structure as in sensing* $5' \to 3'$ *WK automata; however,* s *indicates only the mutual position of heads, and a partially processed input* $\binom{w_1 w_1'}{w_2 w_2'}$ *may not satisfy the complementary relation* ρ*. A step of the automaton can be of the following two types: Let* $w_1', w_2, x, y \in V^*$ *and* $w_1, w_2' \in (V \cup \{\#\})^*$*.*

(1) Reading steps: $\binom{w_1}{w_2 y}(q, s)\binom{x w_1'}{w_2'} \curvearrowright \binom{w_1 \{\#\}^{|x|}}{w_2}(q', s')\binom{w_1'}{\{\#\}^{|y|} w_2'}$*, where* $q' \in \delta(q, \binom{x}{y}, s)$*, and* s' *is either* $\oplus$ *if* $|w_2| > |w_1 x|$ *or* $\ominus$ *in other cases.*

(2) Jumping steps: $\binom{w_1}{w_2 v}(q, s)\binom{u w_1'}{w_2'} \curvearrowright \binom{w_1 u}{w_2}(q, s')\binom{w_1'}{v w_2'}$*, where* s' *is either* $\oplus$ *if* $|w_2| > |w_1 u|$ *or* $\ominus$ *in other cases.*

Note that the jumping steps are an integral and inseparable part of the behavior of the automaton, and thus they are not affected by the state transition function. In the standard manner, extend $\curvearrowright$ *to* $\curvearrowright^n$*, where* $n \geq 0$*; then, based on* $\curvearrowright^n$*, define* $\curvearrowright^+$ *and* $\curvearrowright^*$*. The accepted language, denoted by* $L(M)$*, can be defined by the final accepting configurations that can be reached from the initial one: A double strand* $\left[\substack{w_1 \\ w_2}\right]$ *is accepted by a sensing* $5' \to 3'$ *WK automaton with jumping feature* M *if and only if* $\binom{\varepsilon}{w_2}(q_0, \oplus)\binom{w_1}{\varepsilon} \curvearrowright^* \binom{w_1'}{\varepsilon}(q_f, \ominus)\binom{\varepsilon}{w_2'}$*, for* $q_f \in F$*, where* $w_1' = a_1 a_2 \ldots a_n$*,* $w_2' = b_1 b_2 \ldots b_n$*,* $a_i, b_i \in (V \cup \{\#\})$*, and either* $a_i = \#$ *or* $b_i = \#$*, for all* $i = 1, \ldots, n$*, for some* $n \geq 0$*.*

From a practical point of view, however, this definition is not ideal. The automaton can easily end up in a configuration that cannot yield accepting results, and the correct positions of auxiliary symbols # need to be checked separately at the end of the process. Therefore, we present a modified definition that has the jumping mechanism integrated more into its structure. We are also using a simplification for complementary pairs and treat them as single letters. Such a change has no effect on the accepting power, and this form of input is more natural for formal language theory.

Definition 3.2 *A* jumping $5' \to 3'$ WK automaton *is a quintuple $M = (V, Q, q_0, F, \delta)$, where V, Q, q_0, and F are the same as in WK automata, $V \cap \{\#\} = \emptyset$, the state transition function $\delta\colon (Q \times V^* \times V^* \times D) \to 2^Q$, where $D = \{\oplus, \ominus\}$ indicates the mutual position of heads, and δ assigns a nonempty set only for finitely many quadruples of $(Q \times V^* \times V^* \times D)$. A configuration (q, s, w_1, w_2, w_3) consists of the state q, the position of heads $s \in D$, and the three unprocessed portions of the input tape: (a) before the first head (w_1), (b) between the heads (w_2), and (c) after the second head (w_3). A step of the automaton can be of the following four types: Let $x, y, u, v, w_2 \in V^*$ and $w_1, w_3 \in (V \cup \{\#\})^*$.*

(1) $\oplus$-reading: $(q, \oplus, w_1, xw_2y, w_3) \curvearrowright (q', s, w_1\{\#\}^{|x|}, w_2, \{\#\}^{|y|}w_3)$, where $q' \in \delta(q, x, y, \oplus)$, and s is either $\oplus$ if $|w_2| > 0$ or $\ominus$ in other cases.

(2) $\ominus$-reading: $(q, \ominus, w_1y, \varepsilon, xw_3) \curvearrowright (q', \ominus, w_1, \varepsilon, w_3)$, where $q' \in \delta(q, x, y, \ominus)$.

(3) $\oplus$-jumping: $(q, \oplus, w_1, uw_2v, w_3) \curvearrowright (q, s, w_1u, w_2, vw_3)$, where s is either $\oplus$ if $|w_2| > 0$ or $\ominus$ in other cases.

(4) $\ominus$-jumping: $(q, \ominus, w_1\{\#\}^*, \varepsilon, \{\#\}^*w_3) \curvearrowright (q, \ominus, w_1, \varepsilon, w_3)$.

In the standard manner, extend $\curvearrowright$ to $\curvearrowright^n$, where $n \geq 0$; then, based on $\curvearrowright^n$, define $\curvearrowright^+$ and $\curvearrowright^*$. The accepted language, denoted by $L(M)$, can be defined by the final accepting configurations that can be reached from the initial one: A string w is accepted by a jumping $5' \to 3'$ WK automaton M if and only if $(q_0, \oplus, \varepsilon, w, \varepsilon) \curvearrowright^* (q_f, \ominus, \varepsilon, \varepsilon, \varepsilon)$, for $q_f \in F$.

Even though the structure of this definition is considerably different from Definition 3.1, it is not hard to show that both models accept the same family of languages.

Proposition 3.3 *The models of Definitions 3.1 and 3.2 accept the same family of languages.*

Proof. (sketch). This proposition can be proven by construction (from both sides). Let $M_1 = (V_1, \rho, Q, q_0, F, \delta_1)$ from Definition 3.1 and $M_2 = (V_2, Q, q_0, F, \delta_2)$ from Definition 3.2. The states can clearly remain identical. We can define bijection $\varphi\colon \rho \to V_2$. Let $\varphi(a_i, a'_i) = x_i$ and $\varphi(b_i, b'_i) = y_i$, where $a_i, a'_i, b_i, b'_i \in V_1$, $(a_i, a'_i), (b_i, b'_i) \in \rho$, $x_i, y_i \in V_2$, for all $i = 1, \ldots, n$, n is a positive integer. Any $\delta_1(q, \binom{a_1 \ldots a_n}{b'_1 \ldots b'_m}, s)$ can be converted into $\delta_2(q, x_1 \ldots x_n, y_1 \ldots y_m, s)$, for some $n, m \geq 0$, and vice versa. With this transformation, we reason that both models accept the same inputs. Observe that the reading in the first model marks processed positions in the configuration with the auxiliary symbol #, and, at the end, the model checks whether each pair of symbols was read precisely once. On the other hand, the second model allows only correct transitions that do not violate this reading condition. Furthermore, it keeps only unprocessed parts of the input in the configuration. Consequently, the second model requires more types of steps to handle the different stages of the process. Before the heads meet, either the $\oplus$-reading reads some symbols and marks them (with #) for the other head, or the $\oplus$-jumping skips some symbols and leaves them for the other head. After the meeting point, either the $\ominus$-reading reads remaining symbols that were previously skipped, or the $\ominus$-jumping erases marked symbols from the configuration. Besides that, the reading and jumping behave analogically in both models and thus give the same resulting accepting power. A rigorous version of this proof is left to the reader. □

Hereafter, we primarily use Definition 3.2.

4. Examples

To demonstrate the behavior of the automata, we present a few simple examples.

Example 4.1 *Let us recall that $L = \{w \in \{a,b\}^* : |w|_a = |w|_b\}$ is a well-known non-linear context-free language. We show that, even though the jumping directions in the model are quite restricted, we are able to accept such a language. Consider the following jumping $5' \to 3'$ WK automaton*

$$M = (\{a,b\}, \{s\}, s, \{s\}, \delta)$$

with the state transition function δ: $\delta(s,a,b,\oplus) = \{s\}$ and $\delta(s,a,b,\ominus) = \{s\}$. Starting from s, M can either utilize the jumping or read simultaneously with both heads (the ▶-head reads a and the ◀-head reads b), and it always stays in the sole state s. Now, consider the inputs $aaabbb$ and $baabba$. The former can be accepted by using three $\oplus$-readings and one $\ominus$-jumping:

$$(s,\oplus,\varepsilon,aaabbb,\varepsilon) \curvearrowright (s,\oplus,\#,aabb,\#) \curvearrowright (s,\oplus,\#\#,ab,\#\#) \curvearrowright \\ (s,\ominus,\#\#\#,\varepsilon,\#\#\#) \curvearrowright (s,\ominus,\varepsilon,\varepsilon,\varepsilon).$$

The latter input is more complex and can be accepted by using one $\oplus$-jumping, two $\oplus$-readings, one $\ominus$-jumping, and one $\ominus$-reading:

$$(s,\oplus,\varepsilon,baabba,\varepsilon) \curvearrowright (s,\oplus,b,aabb,a) \curvearrowright (s,\oplus,b\#,ab,\#a) \curvearrowright \\ (s,\ominus,b\#\#,\varepsilon,\#\#a) \curvearrowright (s,\ominus,b,\varepsilon,a) \curvearrowright (s,\ominus,\varepsilon,\varepsilon,\varepsilon).$$

It is not hard to see that, by combining different types of steps, we can accept any input containing the same number of a's and b's, and thus $L(M) = L$.

Example 4.2 *Consider the following jumping $5' \to 3'$ WK automaton*

$$M = (\{a,b\}, \{s\}, s, \{s\}, \delta)$$

with the state transition function δ: $\delta(s,a,b,\oplus) = \{s\}$. Observe that this is almost identical to Example 4.1, however, we cannot use the $\ominus$-reading anymore. Consequently, we also cannot effectively use the $\oplus$-jumping because there is no way how to process remaining symbols afterwards. As a result, the accepted language changes to $L(M) = \{a^n b^n : n \geq 0\}$.

Lastly, we give a more complex example that uses all parts of the model.

Example 4.3 *Consider the following jumping $5' \to 3'$ WK automaton*

$$M = (\{a, b, c\}, \{s_0, s_1, s_2\}, s_0, \{s_0\}, \delta)$$

with δ: $\delta(s_0, a, b, \oplus) = \{s_1\}$, $\delta(s_1, \varepsilon, b, \oplus) = \{s_0\}$, $\delta(s_0, c, c, \ominus) = \{s_2\}$, and $\delta(s_2, \varepsilon, c, \ominus) = \{s_0\}$. We can divide the accepting process of M into two stages. First, before the heads meet, the automaton ensures that for every a on the left side there are two b's on the right side; other symbols are skipped with the jumps. Second, after the heads meet, the automaton checks if the part before the meeting point has double the number of c's as the part after the meeting point. Thus, $L(M) = \{w_1w_2 : w_1 \in \{a, c\}^*,\ w_2 \in \{b, c\}^*,\ 2 \cdot |w_1|_a = |w_2|_b,\ |w_1|_c = 2 \cdot |w_2|_c\}$.

5. General results

These results cover the general behavior of jumping $5' \to 3'$ WK automata without any further restrictions. Let **SWK**, **JWK**, **GJFA**, and **JFA** denote the language families accepted by sensing $5' \to 3'$ WK automata, jumping $5' \to 3'$ WK automata, general jumping finite automata, and jumping finite automata, respectively.

Due to space constraints, some of our proofs are only sketched.

Considering the previous results on other models that use the jumping mechanism (see [5, 6, 1, 2]), it is a common characteristic that they define language families that are incomparable with the classical families of regular, linear, and context-free languages. On the other hand, sensing $5' \to 3'$ WK automata (see [7, 8, 9, 10]) are closely related to the family of linear languages. First, we show that the new model is able to accept all regular and linear languages. Furthermore, the accepting power goes beyond the family of linear languages.

Lemma 5.1 *For every regular language L, there is a jumping $5' \to 3'$ WK automaton M such that $L = L(M)$.*

Proof. Consider a finite automaton $N = (V, Q, q_0, F, \delta_1)$ such that $L(N) = L$. We can construct the jumping $5' \to 3'$ WK automaton $M = (V, Q, q_0, F, \delta_2)$ where $\delta_2(q, a, \varepsilon, \oplus) = \delta_1(q, a)$ for all $q \in Q$, $a \in (V \cup \{\varepsilon\})$. Observe that with such a state transition function the $\oplus$-reading steps always look like this: $(q, \oplus, w_1, aw_2, w_3) \curvearrowright (q', s, w_1\{\#\}^{|a|}, w_2, w_3)$, where $q' \in \delta_2(q, a, \varepsilon, \oplus)$, $w_2 \in V^*$, $w_1, w_3 \in (V \cup \{\#\})^*$, and s is either $\oplus$ if $|w_2| > 0$ or $\ominus$ in other cases. There are no possible $\ominus$-reading steps. The $\oplus$-jumping can be potentially used to skip some symbols before the heads meet; nonetheless, the resulting configuration will be in the form (q, s, w_1, w_2, w_3) where $\text{alph}(w_1w_3) \cap V \neq \emptyset$. Since there is no way how to read such symbols in w_1 and w_3, the configuration cannot yield an accepting result. Consequently, any input string will be read in M the same way as in N (the remaining #'s will be erased with the $\ominus$-jumping afterwards). Thus, $L(M) = L(N) = L$. □

Lemma 5.2 *For every sensing $5' \to 3'$ WK automaton M_1, there is a jumping $5' \to 3'$ WK automaton M_2 such that $L(M_1) = L(M_2)$.*

Proof. This can be proven by construction. Consider any sensing $5' \to 3'$ WK automaton M_1. A direct conversion would be complicated, however, let us recall that **LIN** = **SWK** (see Theorem 2 in [9]). Consider a linear grammar $G = (N, T, S, P)$ such that $L(G) = L(M_1)$. We can construct the jumping $5' \to 3'$ WK automaton M_2 such that $L(M_2) = L(G)$. Assume that $q_f \notin (N \cup T)$. Define $M_2 = (T, N \cup \{q_f\}, S, \{q_f\}, \delta)$, where $B \in \delta(A, u, v, \oplus)$ if $A \to uBv \in P$ and $q_f \in \delta(A, u, \varepsilon, \oplus)$ if $A \to u \in P$ $(A, B \in N,\ u, v \in T^*)$. By the same reasoning as in the proof of Lemma 5.1, only the $\oplus$-reading can be effectively used before the heads meet. Consequently, it can be easily seen that M_2 reads all symbols in the same fashion as G generates them. Moreover, the heads of M_2 can meet each other with the accepting state q_f if and only if G can finish the generation process with a rule $A \to u$. Thus, $L(M_2) = L(G) = L(M_1)$. □

Theorem 5.3 ***LIN*** = ***SWK*** $\subset$ ***JWK***.

Proof. **SWK** $\subseteq$ **JWK** follows from Lemma 5.2. **LIN** = **SWK** was proven in [9]. **JWK** $\not\subseteq$ **LIN** follows from Example 4.1. □

The next two characteristics follow from the previous results.

Theorem 5.4 *Jumping $5' \to 3'$ WK automata without $\ominus$-reading steps accept linear languages.*

Proof. Consider any jumping $5' \to 3'$ WK automaton $M = (V, Q, q_0, F, \delta)$ that has no possible $\ominus$-reading steps. Expanding the reasoning in the proof of Lemma 5.2, if there are no possible $\ominus$-reading steps, the $\oplus$-jumping cannot be effectively used, and we can construct a linear grammar that generates strings in the same fashion as M reads them. Define the linear grammar $G = (Q, V, q_0, R)$, where R is constructed in the following way: (1) For each $p \in \delta(q, u, v, \oplus)$, add $q \to upv$ to R. (2) For each $f \in F$, add $f \to \varepsilon$ to R. Clearly, $L(G) = L(M)$. □

Proposition 5.5 *The language family accepted by double-jumping finite automata that perform right-left and left-right jumps (see [2]) is strictly included in **JWK**.*

Proof. First, Theorem 3.18 in [2] shows that jumping finite automata that perform right-left and left-right jumps accept the same family of languages. Second, Theorem 3.7 in [2] shows that this family is strictly included in **LIN**. Finally, Theorem 5.3 shows that **LIN** is strictly included in **JWK**. □

Even though the model is able to accept some non-linear languages, the jumping directions of the heads are quite restricted compared to general jumping finite automata. Consequently, there are some languages accepted by jumping $5' \to 3'$ WK automata and general jumping finite automata that cannot be accepted with the other model.

Lemma 5.6 *There is no jumping $5' \to 3'$ WK automaton M such that $L(M) = \{a^n b^n c^n : n \geq 0\}$.*

Proof. (sketch). It is clear that with a finite memory the automaton can remember only a finite amount of symbols that were already processed. Intuitively, therefore, for a long enough input the automaton must be able to repeatedly perform some sequence of steps in some phase that reads a correlated number of a's, b's, and c's. Nonetheless, in any such a sequence, some head has to jump over a portion of the unprocessed input to read b's, and this head can no longer read the symbols it skipped. The skipped portion has to contain either all remaining a's or c's. These a's or c's can still be read with the other head, but, in order to reach them, this head would have to skip the other remaining portion of the input. This portion would have to contain all remaining b's and also either all remaining c's or a's. Consequently, there cannot be a repeatable sequence of steps that reads distinct symbols from three different places. $\square$

Lemma 5.7 *There is no jumping* $5' \to 3'$ *WK automaton* M *such that* $L(M) = \{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w|_c\}$.

Proof. (sketch). Assume that there is a jumping $5' \to 3'$ WK automaton M such that $L(M) = \{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w_c|\}$. Intuitively, such an automaton must be able to properly check the number of symbols in any input $w = a^n b^n c^n$, where n is a positive integer. However, the argument in the sketch of the proof of Lemma 5.6 shows that it is not possible. $\square$

Proposition 5.8 ***JWK*** *is incomparable with* ***GJFA*** *and* ***JFA***.

Proof. The language $\{w \in \{a,b\}^* : |w|_a = |w|_b\}$ from Example 4.1 and the language $\{w \in \{a,b,c\}^* : |w|_a = |w|_b = |w|_c\}$ from Lemma 5.7 are accepted with (general) jumping finite automata (see Example 5 in [5]). The language $\{a^n b^n : n \geq 0\}$ from Example 4.2 is not accepted with (general) jumping finite automata (see Lemma 19 in [5]). $\square$

The last group of results compares the accepting power of the model with the families of context-sensitive and context-free languages.

Theorem 5.9 ***JWK*** $\subset$ ***CS***.

Proof. Clearly, the use of two heads and the jumping behavior can be simulated by linear bounded automata, so **JWK** $\subseteq$ **CS**. From Lemma 5.6, **CS** $-$ **JWK** $\neq \emptyset$. $\square$

Lemma 5.10 *There are some non-context-free languages accepted by jumping* $5' \to 3'$ *WK automata.*

Proof. Consider the following jumping $5' \to 3'$ WK automaton

$$M = (\{a,b,c,d\}, \{s\}, s, \{s\}, \delta)$$

with the state transition function δ: $\delta(s,a,c,\oplus) = \{s\}$ and $\delta(s,d,b,\ominus) = \{s\}$. The accepting process has two stages. First, before the heads meet, the automaton reads the same number of a's and c's. Second, after the heads meet, the automaton reads the same number of d's and b's. Thus, $L(M) = \{w_1 w_2 : w_1 \in \{a,b\}^*,\ w_2 \in \{c,d\}^*,\ |w_1|_a = |w_2|_c,\ |w_1|_b = |w_2|_d\}$.
Proof by contradiction. Assume that $L(M)$ is a context-free language. The family of context-free languages is closed under intersection with regular sets. Let $K = L(M) \cap \{a\}^*\{b\}^*\{c\}^*\{d\}^*$. Clearly, there are some strings in $L(M)$ that satisfy this forced order of symbols. Furthermore, they all have the proper correlated numbers of these symbols. Consequently, $K = \{a^n b^m c^n d^m : n, m \geq 0\}$. However, K is a well-known non-context-free language (see Chapter 3.1 in [12]). That is a contradiction with the assumption that $L(M)$ is a context-free language. Therefore, $L(M)$ is a non-context-free language. $\square$

Lemma 5.11 *There is no jumping $5' \to 3'$ WK automaton M such that $L(M) = \{a^n b^n c^m d^m : n, m \geq 0\}$.*

Proof. (sketch). It is clear that with a finite memory the automaton can remember only a finite amount of symbols that were already processed. As it is known from finite automata, a single head alone cannot recognize strings $a^n b^n$ or $c^m d^m$. Intuitively, therefore, the automaton has to use both heads to process such a string. But in order to do so, some head has to jump over the other part ($a^n b^n$ or $c^m d^m$). However, since the heads cannot travel back on the tape, there is no way how to use both heads to process both parts. □

Theorem 5.12 ***JWK*** *and* ***CF*** *are incomparable.*

Proof. **JWK** $\nsubseteq$ **CF** follows from Lemma 5.10. **CF** $\nsubseteq$ **JWK** follows from Lemma 5.11. Lastly, **LIN** $\subset$ **JWK** and **LIN** $\subset$ **CF**. □

6. Results on restricted variations

In this section, we compare the accepting power of unrestricted and restricted variations of jumping $5' \to 3'$ WK automata. This paper considers the same standard restrictions as they are defined for Watson-Crick finite automata. Since these restrictions regulate only the state control and reading steps of the automaton, the jumping is not affected in any way. Let **JWK** denote the language family accepted by jumping $5' \to 3'$ WK automata. We are using prefixes **N**, **F**, **S**, **1**, **NS**, **FS**, **N1**, and **F1** to specify the restricted variations of jumping $5' \to 3'$ WK automata and appropriate language families.

In the field of DNA computing, the empty string/empty sequence usually does not belong to any language because it does not refer to a molecule. This paper is not so strict and thus considers the empty string as a possible valid input. Nonetheless, the following proofs are deliberately based on more complex inputs to mitigate the impact of the empty string on the results.

Note that there are some inherent inclusions between language families based on the application of restrictions on the model. Additionally, several other basic relations can be established directly from the restriction definitions:

Lemma 6.1 *The following relations hold: (i)* ***N JWK*** $\subseteq$ ***F JWK****; (ii)* ***1 JWK*** $\subseteq$ ***S JWK****; (iii)* ***F1 JWK*** $\subseteq$ ***FS JWK****; (iv)* ***N1 JWK*** $\subseteq$ ***NS JWK****; (v)* ***NS JWK*** $\subseteq$ ***FS JWK****; (vi)* ***N1 JWK*** $\subseteq$ ***F1 JWK****.*

Proof. These results follow directly from the definitions since the stateless restriction (**N**) is a special case of the all-final restriction (**F**) and the 1-limited restriction (**1**) is a special case of the simple restriction (**S**). □

Due to space constraints, we present only a quick overview of the results.

Theorem 6.2 ***S JWK*** $=$ ***JWK***.

Proof. (idea). Any general reading step can be replaced with two simple reading steps and a new auxiliary state that together accomplish the same action. □

Example 6.3 *Consider the following jumping $5' \to 3'$ WK automaton $M = (\{a,b,c\}, \{s,f\}, s, \{f\}, \delta)$ with the state transition function δ:*

$$\delta(s,a,b,\oplus) = \{s\}, \quad \delta(f,a,b,\oplus) = \{f\}, \quad \delta(f,a,b,\ominus) = \{f\},$$
$$\delta(s,cc,\varepsilon,\oplus) = \{f\}, \quad \delta(s,\varepsilon,cc,\oplus) = \{f\}.$$

It is clear that the first three transitions mimic the behavior of Example 4.1. The other two transitions ensure that the input is accepted only if it also contains precisely one substring cc. Therefore, $L(M) = \{w_1ccw_2 : w_1, w_2 \in \{a,b\}^*,\ |w_1w_2|_a = |w_1w_2|_b\}$.

Theorem 6.4 ***1 JWK*** $\subset$ ***JWK***.

Proof. (idea). There is no **1** jumping $5' \to 3'$ WK automaton M such that $L(M) = \{w_1ccw_2 : w_1, w_2 \in \{a,b\}^*,\ |w_1w_2|_a = |w_1w_2|_b\}$. □

Example 6.5 *Consider the following **1** jumping $5' \to 3'$ WK automaton $M = (\{a,b\}, \{s,p\}, s, \{s\}, \delta)$ with the state transition function δ:*

$$\delta(s,a,\varepsilon,\oplus) = \{p\}, \quad \delta(p,\varepsilon,b,\oplus) = \{s\},$$
$$\delta(s,a,\varepsilon,\ominus) = \{p\}, \quad \delta(p,\varepsilon,b,\ominus) = \{s\}.$$

It is not hard to see that the resulting behavior is similar to Example 4.1. The automaton now reads a's and b's with separate steps and uses one auxiliary state that is not final. Consequently, $L(M) = \{w \in \{a,b\}^* : |w|_a = |w|_b\}$.

Theorem 6.6 ***LIN*** $\subset$ ***1 JWK***.

Proof. (idea). For every linear grammar G, there is a **1** jumping $5' \to 3'$ WK automaton M such that $L(G) = L(M)$. Example 6.5 shows that **1 JWK** $\not\subseteq$ **LIN**. □

Theorem 6.7 ***F JWK*** $\subset$ ***JWK***.

Proof. (idea). There is no **F** jumping $5' \to 3'$ WK automaton M such that $L(M) = \{ca^ncb^nc : n \geq 0\} \cup \{\varepsilon\}$. □

Example 6.8 *Consider the following **F** (in fact, even **N**) jumping $5' \to 3'$ WK automaton $M = (\{a,b,c\}, \{s\}, s, \{s\}, \delta)$ with the state transition function δ:*

$$\delta(s,a,b,\oplus) = \{s\}, \quad \delta(s,a,b,\ominus) = \{s\},$$
$$\delta(s,cc,\varepsilon,\oplus) = \{s\}, \quad \delta(s,\varepsilon,cc,\oplus) = \{s\}.$$

*This is a slightly modified version of Example 6.3 where the substring cc can occur arbitrarily many times. Therefore, $L(M) = \{w \in \{a,b,cc\}^* : |w|_a = |w|_b\}$. $L(M) \notin$ **1 JWK**.*

Theorem 6.9 ***N JWK*** $\subset$ ***F JWK***.

Proof. (idea). From Lemma 6.1, **N JWK** $\subseteq$ **F JWK**. There is no **N** jumping $5' \to 3'$ WK automaton M such that $L(M) = \{\varepsilon, a\}$. □

Proposition 6.10 ***FS JWK*** $\subset$ ***F JWK***.

Proof. (idea). There is no **FS** jumping $5' \to 3'$ WK automaton M such that that $L(M) = \{cca^n cc : n \geq 0\} \cup \{\varepsilon\}$. □

Example 6.11 *Consider the following* ***FS*** *jumping* $5' \to 3'$ *WK automaton* $M = (\{a, b, c\}, \{s, p\}, s, \{s, p\}, \delta)$ *with the state transition function* δ*:*

$$\begin{aligned} \delta(s, a, \varepsilon, \oplus) &= \{p\}, & \delta(p, \varepsilon, b, \oplus) &= \{s\}, \\ \delta(s, a, \varepsilon, \ominus) &= \{p\}, & \delta(p, \varepsilon, b, \ominus) &= \{s\}, \\ \delta(s, cc, \varepsilon, \oplus) &= \{s\}, & \delta(s, \varepsilon, cc, \oplus) &= \{s\}, \\ \delta(p, cc, \varepsilon, \oplus) &= \{p\}, & \delta(p, \varepsilon, cc, \oplus) &= \{p\}. \end{aligned}$$

As a result, $L(M) = \{w \in \{a, b, cc\}^* : |w|_a = |w|_b$ *or* $|w|_a = |w|_b + 1\}$.
This automaton is just a combination of previous approaches from Examples 6.5 and 6.8. Note that $L(M)$ *resembles the resulting language of Example 6.8.*

Proposition 6.12 ***F1 JWK*** $\subset$ ***FS JWK***.

Proof. (idea). There is no **F1** jumping $5' \to 3'$ WK automaton that accepts $\{aa\}^*$. □

Corollary 6.13 ***F1 JWK*** $\subset$ ***1 JWK***. □

Theorem 6.14 ***NS JWK*** $\subset$ ***REG***.

Proof. (idea). For any **NS** jumping $5' \to 3'$ WK automaton we can construct a finite automaton that accepts the same language. □

Proposition 6.15 ***N1 JWK*** $\subset$ ***NS JWK***.

Proof. This proof is analogous to that of Proposition 6.12. □

Corollary 6.16 *The following relations hold: (i)* ***NS JWK*** $\subset$ ***N JWK****; (ii)* ***NS JWK*** $\subset$ ***FS JWK****; (iii)* ***N1 JWK*** $\subset$ ***F1 JWK***. □

All the obtained results comparing the accepting power of unrestricted and restricted variations of jumping $5' \to 3'$ WK automata are summarized in Figure 1.

$$\begin{tikzcd}
 & & \mathbf{S\ JWK = JWK} & & \\
\mathbf{1\ JWK} \arrow[urr] & & & \mathbf{F\ JWK} \arrow[ul] & \\
\mathbf{LIN} \arrow[u] & \mathbf{FS\ JWK} \arrow[urr] & & & \mathbf{N\ JWK} \arrow[ul] \\
\mathbf{REG} \arrow[u] & \mathbf{F1\ JWK} \arrow[u] \arrow[uul] & & & \mathbf{NS\ JWK} \arrow[u] \arrow[ulll] \arrow[llll] \\
 & \mathbf{N1\ JWK} \arrow[u] \arrow[urrr] & & &
\end{tikzcd}$$

Figure 1: A hierarchy of language families closely related to the unrestricted and restricted variations of jumping $5' \to 3'$ WK automata is shown. If there is an arrow from family X to family Y in the figure, then $X \subset Y$. Furthermore, if there is no path (following the arrows) between families X and Y, then X and Y are incomparable.

7. Conclusion

The results clearly show that, with the addition of the jumping mechanism into the model, the accepting power was increased above sensing $5' \to 3'$ WK automata. The model is now able to accept some non-linear and even some non-context-free languages. On the other hand, the jumping movement of the heads is restricted compared to jumping finite automata, and this limits its capabilities to accept languages that require discontinuous information processing. Considering the comparison with full-reading sensing $5' \to 3'$ WK automata, the results are not yet clear. However, we know that there are some languages, like $\{a^n b^n c^n : n \geq 0\}$, that cannot be accepted by jumping $5' \to 3'$ WK automata and that are accepted by full-reading sensing $5' \to 3'$ WK automata (see [7, 8, 9]).

If we compare the hierarchies of language families related to the restricted variations of jumping $5' \to 3'$ WK automata and sensing $5' \to 3'$ WK automata (see [8, 9, 10]), there are several noticeable remarks. Most importantly, the 1-limited restriction (**1**) has a negative impact on the accepting power, which is usually not the case in sensing $5' \to 3'$ WK automata. In parts where several restrictions are combined together, the hierarchy structure resembles sensing $5' \to 3'$ WK automata without the sensing distance. Nonetheless, almost all restricted variations, with the exception of **NS** and **N1**, are still able to accept some non-linear languages.

Lastly, the reader may notice that the $\ominus$-jumping can be used only in situations where it is forced by the current configuration. Furthermore, jumping finite automata usually immediately erase symbols from the configuration and do not use the auxiliary symbol #. It is therefore a question whether this part could be safely removed from the model. Without it, the conversion from Definition 3.1 cannot be straightforward, and it is not clear whether the accepting power remains identical. Observe that, if we remove #'s, the configuration can create new connected strings that were not in the original input and for which there can be a possible $\ominus$-reading step.

Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602; the TAČR grant TE01020415; and the BUT grant FIT-S-17-3964.

The authors thank all the anonymous referees for their useful comments and suggestions.

References

[1] Z. KŘIVKA, A. MEDUNA, Jumping Grammars. *International Journal of Foundations of Computer Science* 26 (2015) 6, 709–731.

[2] R. KOCMAN, Z. KŘIVKA, A. MEDUNA, On Double-Jumping Finite Automata. In: *Eighth Workshop on Non-Classical Models of Automata and Applications (NCMA 2016)*. books@ocg.at 321, Osterreichische Computer Gesellschaft, 2016, 195–210.

[3] R. KOCMAN, A. MEDUNA, On Parallel Versions of Jumping Finite Automata. In: *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*. Advances in Intelligent Systems and Computing 511, Springer International Publishing, 2016, 142–149.

[4] A. MEDUNA, *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[5] A. MEDUNA, P. ZEMEK, Jumping Finite Automata. *International Journal of Foundations of Computer Science* 23 (2012) 7, 1555–1578.

[6] A. MEDUNA, P. ZEMEK, *Regulated Grammars and Automata*. Springer, 2014.

[7] B. NAGY, On $5' \rightarrow 3'$ Sensing Watson-Crick Finite Automata. In: *DNA Computing: 13th International Meeting on DNA Computing, DNA13*. LNCS 4848, Springer, 2008, 256–262.

[8] B. NAGY, $5' \rightarrow 3'$ Sensing Watson-Crick Finite Automata. In: G. FUNG (ed.), *Sequence and Genome Analysis II – Methods and Applications*. iConcept Press, 2010, 39–56.

[9] B. NAGY, On a hierarchy of $5' \rightarrow 3'$ sensing Watson-Crick finite automata languages. *Journal of Logic and Computation* 23 (2013) 4, 855–872.

[10] B. NAGY, S. PARCHAMI, H. MIR-MOHAMMAD-SADEGHI, A New Sensing $5' \rightarrow 3'$ Watson-Crick Automata Concept. In: *Proceedings 15th International Conference on Automata and Formal Languages (AFL 2017)*. EPTCS 252, Open Publishing Association, 2017, 195–204.

[11] G. PAUN, G. ROZENBERG, A. SALOMAA, *DNA Computing: New Computing Paradigms*. Springer-Verlag Berlin Heidelberg, 1998.

[12] G. ROZENBERG, A. SALOMAA, *Handbook of Formal Languages, Vol. 2: Linear Modeling: Background and Application*. Springer-Verlag, 1997.

[13] D. WOOD, *Theory of Computation: A Primer*. Addison-Wesley, Boston, 1987.