

Two-Way Finite Automata with Translucent Input Letters

Martin Kutrib¹[0000–0002–9564–2625], Andreas Malcher¹[0000–0002–9589–5833],
Carlo Mereghetti^{2,3}[0000–0002–7778–7257], and Beatrice
Palano^{2,3}[0000–0003–3948–4658]

¹ Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany

{kutrib, andreas.malcher}@informatik.uni-giessen.de

² Dipartimento di Informatica “G. Degli Antoni”, Università degli Studi di Milano
via Celoria 18, 20133 Milano, Italy

{carlo.mereghetti, beatrice.palano}@unimi.it

³ Gruppo Nazionale Calcolo Scientifico
Istituto Nazionale di Alta Matematica (GNCS-INdAM), Italy

Abstract. Finite automata with translucent input letters can be obtained by equipping classical finite automata with a translucency function which, depending on the current state, establishes the set of invisible input symbols: such symbols are skipped in the current move and dealt with subsequently, while the first visible symbol from the current input head position is processed and consumed. These models can be returning, meaning that the input head is rewinded at each move after processing a visible symbol, or not. In their original definition by Mráz and Otto [13], translucent finite automata feature a one-way input head, i.e., the head always “looks to the right” to spot the next visible symbol. Here, we extend this definition in the deterministic case by allowing a two-way input head motion, meaning that the next visible symbol can be found by looking either to the left or to the right of the head. We also consider translucent sweeping finite automata, as the restricted version of two-way devices reverting input head direction at the endmarkers only. We prove that the sweeping restriction does not affect the recognition power in the returning mode, while the opposite holds for non-returning devices. We show that our devices are strictly more powerful than their one-way version, and that they are incomparable with translucent push-down automata [9, 14, 17]. We investigate some closure properties for the families of languages recognized by translucent two-way and sweeping finite automata. In particular, while dealing with this latter topic, an open problem stated in [13] is addressed and solved.

Keywords: Translucent input letters · Finite automata · Two-way and Sweeping motion · Returning and non-returning computations · Closure Properties

1 Introduction

Language recognition devices are typically thought of as parsing input strings from left to right, one symbol after the other, in a continuous fashion; they usually sweep the whole input and finally accept or reject, unless they get stuck in some kind of loop. This one-way input head motion, which can encompass the possibility of performing stationary or λ -moves, and/or accepting in the middle of the string, is basically the default for all the most famous and studied computational models such as Turing machines, finite automata of various kinds, pushdown automata, and so on (see, e.g., [1, 4–6, 8, 12]). Naturally, alternative ways of processing input strings, such as rotating, sweeping and restarting modes, are well worth investigating; the reader is referred to, e.g., [2, 5, 7, 11, 21, 23] for an overview of these and other input processing modes. In all these cases, meaningful differences in recognition and/or descriptive power are emphasized, stating that the way of acquiring input is truly a computational resource that can be used to tune device capabilities. Among others, here we focus on two parsing forms, central in the present contribution: *two-way* motion and *translucency*.

In a two-way device, the input head can choose to move backward or forward or stay at each move. In the realm of finite automata, it is well known that two-way motion does not increase the recognition power, but it may greatly affect the descriptive power, with deep implications in structural complexity. In particular, a relevant open problem is to establish the optimal state cost of deleting nondeterminism on two-way finite automata [24]. Showing the optimality of the known exponential size blow-up would immediately separate determinism from nondeterminism in logarithmic space bounded computations [3], together with P vs. NP, one of the main open questions in theoretical computer science. A positive (unfortunately, not decisive) answer is obtained by assuming the simulating two-way deterministic finite automata to be sweeping [25], which is a restricted form of two-way dynamic where head reversals may occur at the ends of the input tape only.

The notion of translucent input letters in the realm of finite automata is introduced by Nagy and Otto in [15]. The built-in key feature for translucent devices is a translucency function which, based on the current state, establishes translucent (invisible) input letters. At each move, the device skips (by looking through) the translucent portion of the input, from the current input head position rightward, up to the first non-translucent letter. From this point of view, such devices can be understood as being *one-way*, in the sense that they always “look to the right” in order to discover the next visible input symbol to be processed. After processing the non-translucent symbol, in the *returning mode* the input head returns to the left end of the input while, in the *non-returning mode*, the input head continues to process the input according to its updated current state and the corresponding translucent symbols. Also, in this latter mode, the input head returns to the left end when the right end of the input is reached, unless acceptance/rejection takes place. Deterministic and nondeterministic finite automata with translucent letters are deeply investigated in the literature (see, e.g., [13, 16, 17, 20, 22]). However, many questions are still open.

In this paper, we equip finite automata with *both* the two-way and the translucent paradigm, thus defining and investigating *two-way deterministic finite automata with translucent input letters*, as well as their *sweeping* restriction, in both returning and non-returning mode (these devices being abbreviated as 2DFAwtl, SDFAwtl, nr2DFAwtl, and nrSDFAwtl, respectively). Very roughly speaking, we equip a classical 2DFA or SDFA (sweeping 2DFA) with a translucency function. In a 2DFAwtl M , states are partitioned into sets Q_r and Q_l . At each move, if M is in a state in Q_r (resp., Q_l), then it must look to the right (resp., left) of the current head position in order to spot the next visible input symbol to be processed. From this point of view, our devices generalize one-way automata by Nagy and Otto [15], in that they can “look both ways” to spot the next visible symbol, whence the designation *two-way*. After consuming the spotted visible symbol, in the *returning mode* the input head gets back to the left or to the right end of the input, depending on whether the reached state belongs either to Q_r or to Q_l , respectively. Instead, in the *non-returning mode*, the input head continues to process the input toward the direction implied by the reached state. In any case, the direction switches on the endmarkers. An SDFAwtl is the restricted version of a 2DFAwtl, where the direction of the input head changes at the endmarkers only. Returning and non-returning modes work as for 2DFAwtl.

It should be noted that Nagy and Otto introduced and investigated in [18, 19] *linear automata with translucent letters*, which are basically finite automata working with translucent letters that additionally have two heads starting at opposite ends and moving from left to right and from right to left, respectively. Such devices are in its deterministic variant equivalent to SDFAwtl in the returning mode. Hence, some positive and negative closure properties obtained in [18, 19] carry over to returning SDFAwtl. On the other hand, some closure properties left open in [18, 19] can now be solved in terms of returning SDFAwtl.

In Section 2, preliminary notions and notations are provided, and the formal definition of 2DFAwtl, SDFAwtl, and nr2DFAwtl, nrSDFAwtl is set forth, together with some working examples and basic relations among models.

In Section 3, we compare the recognition power of sweeping and general two-way translucent finite automata. We get that, in the returning mode, the sweeping dynamic does not lower the acceptance capability. To this aim, we show how to build an equivalent SDFAwtl from any given 2DFAwtl. Instead, this is not the case for the non-returning mode. In fact, we provide a language accepted by an nr2DFAwtl, and show that it cannot be accepted by any nrSDFAwtl.

In Section 4, we compare the power of two-way head movement vs. pushdown storage (abbreviated DPDAwtl and nrDPDAwtl [9, 10]) in the translucent setting. From a recognition power viewpoint, we show that 2DFAwtl, nrSDFAwtl, and nr2DFAwtl are incomparable with DPDAwtl and nrDPDAwtl. This is obtained by designing suitable witness languages.

In Section 5, we graphically sketch the obtained recognition power results in a hierarchy, which also encompasses the recognition power of other translucent devices investigated in the literature.

In Section 6, we investigate some closure properties for the families of languages recognized by 2DFAwtl, nrSDFAwtl, and nr2DFAwtl. By providing suitable witness languages, we prove that all these families are not closed under union, intersection, concatenation, iteration, length-preserving and inverse non-erasing homomorphisms. Instead, they are all closed for reversal and complementation. The closure under this latter operation is obtained by showing how to detect and avoid loops on our two-way translucent devices.

While designing witness languages to settle recognition limits, we define a particular language, and prove it cannot be recognized by any nrSNFAwtl. This enables us to show that one-way translucent finite automata are not closed under intersection with regular languages, thus closing an open problem proposed by Mráz and Otto [13]. In addition, we point out that the above quoted non-closures under iteration and inverse homomorphism we obtain for SDFAwtl enable us to answer analogous closure questions left open in [18, 19] for linear automata with translucent letters.

2 Preliminaries

Given a set S , we denote by 2^S its power set, and by $|S|$ its cardinality. We use $\subseteq$ for inclusions, and $\subset$ for proper inclusions. We denote by Σ^* the set of all words on the finite alphabet Σ , including the empty word λ , and let $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For any word $w \in \Sigma^*$, we let $|w|$ denote its length, w^R its reversal, and $|w|_a$ the number of occurrences of the symbol $a \in \Sigma$ in w . Maximal factors of the form a^+ in w , where $a \in \Sigma$, are called a -blocks. A language on Σ is any subset $L \subseteq \Sigma^*$. The complement of L is the language $\bar{L} = \Sigma^* \setminus L$, its reversal is $L^R = \{w^R \mid w \in L\}$. Two language families $\mathcal{L}_1$ and $\mathcal{L}_2$ are said to be incomparable whenever $\mathcal{L}_1$ is not a subset of $\mathcal{L}_2$ and vice versa.

Two-way finite automata feature a single read-only input tape whose inscription is the input word between two endmarkers. The head of the automaton can move freely on the tape but not beyond the endmarkers. Finite automata with translucent letters are extensions of classical finite automata, which do not necessarily have to read the “next” input symbol. Instead, depending on the current state of such devices, some of the input letters may be translucent (invisible). Thus, a two-way finite automaton with translucent letters reads and processes the first input letter, visible in the direction of the head movement.

Formally, a *deterministic two-way finite automaton with translucent letters* (abbreviated as 2DFAwtl) is a system $M = \langle Q, \Sigma, q_0, \triangleright, \triangleleft, \tau, \delta \rangle$, where Q is the finite set of *states* partitioned into the sets Q_l (states leading the input head leftward) and Q_r (states leading the input head rightward), Σ is the finite alphabet of *input symbols*, with $\Sigma \cap Q = \emptyset$, $q_0 \in Q_r$ is the *initial state*, $\triangleright, \triangleleft \notin \Sigma$ are, respectively, the left and right *endmarkers*, $\tau: Q \rightarrow 2^\Sigma$ is the *translucency mapping*, and $\delta: Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \rightarrow Q \cup \{\text{accept}\}$ is the *partial transition function*. The translucency mapping τ bears the following meaning: for any state $q \in Q$, the letters from the set $\tau(q)$ are *translucent (invisible) for q* . Therefore,

whenever in q , the automaton M does not see these letters (or equivalently, M can see through such letters). Note that the endmarkers are always visible.

A *configuration* of M is represented: by $(\triangleright qw\triangleleft)$ if $q \in Q_r$, by $(\triangleright wq\triangleleft)$ if $q \in Q_l$, or by **accept**, where $q \in Q$ is the current state and $w \in \Sigma^*$ is the current tape inscription. The computation of M on input $w \in \Sigma^*$ starts from the *initial configuration* $(\triangleright q_0 w \triangleleft)$, and steps through a sequence of configurations as follows. Being in some configuration $(\triangleright qw\triangleleft)$, first M determines the next symbol to scan. To this end, it processes the first visible symbol in the direction of its input head motion. Precisely, if $q \in Q_r$ and $w\triangleleft = xy\triangleleft$ with $x \in \tau(q)^*$, $y \in \Sigma^*$, and $a \notin \tau(q)$ is the first symbol of $y\triangleleft$, then M processes a . Now, M halts and rejects whenever $\delta(q, a)$ is undefined, otherwise it enters state $p = \delta(q, a)$, and the successor configuration becomes either $(\triangleright pxy\triangleleft)$ if $p \in Q_r$, or $(\triangleright xyp\triangleleft)$ if $p \in Q_l$. The behavior of M being in the state $q \in Q_l$ is symmetric. Overall, M halts and accepts if and only if δ maps the current situation to **accept**. One step from a configuration to its successor configuration is denoted by $\vdash$. We let $\vdash^*$ (resp., $\vdash^+$) denote the reflexive and transitive (resp., transitive) closure of $\vdash$.

The language accepted by the 2DFAwtl M is defined to be the set $L(M)$ of those words in Σ^* on which the computation of M eventually halts accepting, namely, $L(M) = \{w \in \Sigma^* \mid (\triangleright q_0 w \triangleleft) \vdash^+ \text{accept}\}$. In order to exemplify the dynamics of 2DFAwtl, let us propose the following

Example 1. We design a 2DFAwtl $M = \langle Q, \{a, b, c\}, q_a, \triangleright, \triangleleft, \tau, \delta \rangle$ for the language $L = \{uc^{n-1} \mid u \in \{a, b\}^*, |u|_a = n = |u|_b, n > 0\}$. We set $Q_r = \{q_a, q_b\}$, $Q_l = \{q_c\}$, and define τ and δ while elucidating the computation of M . At the beginning of the computation, M processes the first-from-the-left a (being in $q_a \in Q_r$), followed by processing the first-from-the-left b (being in $q_b \in Q_r$). This can be accomplished by setting translucency and transitions as

$$\begin{aligned} \tau(q_a) &= \{b\}, & \delta(q_a, a) &= q_b, \\ \tau(q_b) &= \{a\}, & \delta(q_b, b) &= q_c. \end{aligned}$$

After such a double-processing, M enters the state $q_c \in Q_l$ and shifts on the rightmost input symbol, if any, verifying whether this symbol is c . If M sees an a or b , it rejects; if it sees the left endmarker, it accepts. This is implemented as

$$\begin{aligned} \tau(q_c) &= \emptyset, & \delta(q_c, c) &= q_a, \\ & & \delta(q_c, \triangleright) &= \text{accept}. \end{aligned}$$

If the input was uc^{n-1} , with $u \in \{a, b\}^*$, $|u|_a = n = |u|_b$ then, after this phase, the tape inscription would reduce to vc^{n-2} , with $v \in \{a, b\}^*$, $|v|_a = n - 1 = |v|_b$, and with a new iteration starting from $q_a \in Q_r$ on the leftmost input symbol. The reader may verify that M accepts L . ■

A sweeping finite automaton with translucent letters is a restricted version of a 2DFAwtl. While the direction of the head movement can be freely switched in the two-way case, it can and must be changed *on the endmarkers only* in the sweeping case. So, a *deterministic sweeping finite automaton with translucent*

letters (abbreviated as SDFAwtl) is a 2DFAwtl $M = \langle Q, \Sigma, q_0, \triangleright, \triangleleft, \tau, \delta \rangle$, where $q_0 \in Q_r$, and the transition function obeys the following restrictions:

$$\begin{aligned} \delta: Q_r \times \Sigma &\rightarrow Q_r \cup \{\text{accept}\}, & \delta: Q_r \times \{\triangleleft\} &\rightarrow Q_l \cup \{\text{accept}\}, \\ \delta: Q_l \times \Sigma &\rightarrow Q_l \cup \{\text{accept}\}, & \delta: Q_l \times \{\triangleright\} &\rightarrow Q_r \cup \{\text{accept}\}. \end{aligned}$$

The following example shows an SDFAwtl in action.

Example 2. We design an SDFAwtl $M = \langle Q, \{a, b\}, q_0, \triangleright, \triangleleft, \tau, \delta \rangle$ for the mirror language $L_{mi} = \{ww^R \mid w \in \{a, b\}^*\}$. We set $Q_r = \{q_0, q_a, q_b\}$, $Q_l = \{p_a, p_b, p_r\}$, and explain τ and δ while describing the computation of M . At the beginning of the computation, M reads the first letter of the input:

$$\begin{aligned} \tau(q_0) &= \emptyset, & \delta(q_0, \triangleleft) &= \text{accept}, \\ & & \delta(q_0, a) &= q_a, \\ & & \delta(q_0, b) &= q_b. \end{aligned}$$

Note that, by these instructions, even the empty word is accepted. If the first input letter, i.e., the first symbol of w , is $x \in \{a, b\}$, then M enters the state q_x and remains at the left end. Then, M sweeps to the right end and enters the state p_x :

$$\begin{aligned} \tau(q_a) &= \Sigma, & \delta(q_a, \triangleleft) &= p_a, \\ \tau(q_b) &= \Sigma, & \delta(q_b, \triangleleft) &= p_b. \end{aligned}$$

Once at the right end, M tries to read the symbol x as the last letter of w^R . If the read letter does not match x , the computation gets stuck and rejects. Otherwise, it continues in the state p_r :

$$\begin{aligned} \tau(p_a) &= \emptyset, & \delta(p_a, a) &= p_r, \\ \tau(p_b) &= \emptyset, & \delta(p_b, b) &= p_r. \end{aligned}$$

If the input was $x_1x_2 \cdots x_nx_nx_{n-1} \cdots x_1$, then the tape inscription reduces to $x_2x_3 \cdots x_nx_nx_{n-1} \cdots x_2$, still to be tested. Clearly, the input has to be accepted if and only if this latter tape inscription belongs to L_{mi} . To this end, from the state p_r , M jumps back to the left end and enters state the q_0 again:

$$\tau(p_r) = \Sigma, \quad \delta(p_r, \triangleright) = q_0. \quad \blacksquare$$

In [13], a variant of (one-way) finite automata with translucent letters has been introduced, implementing the so-called non-returning mode. According to this paradigm, after processing a visible letter, the head of the automaton does not return to the left or right end of the input but it continues from the position of the letter just processed. So, we are now going to focus on *deterministic two-way finite automata with translucent letters working in the non-returning mode* (abbreviated as nr2DFAwtl). Basically, definitions and notations for 2DFAwtl apply to nr2DFAwtl as well.

Let $M = \langle Q, \Sigma, q_0, \triangleright, \triangleleft, \tau, \delta \rangle$ be an nr2DFAwtl. Now, a configuration of M is represented by $(\triangleright uqv \triangleleft)$ or **accept**, where $q \in Q$ is the current state, $uv \in \Sigma^*$

is the remaining part of the input with u to the left and v to the right of the input head. The successor configuration is now specified as follows. As before, M halts and accepts whenever δ maps the current situation to **accept**. Moreover, for $p \in Q$, $q \in Q_r$, $s \in Q_l$, $a \in \Sigma$, $x, u, v, w \in \Sigma^*$, we have

1. $(\triangleright uqxy\triangleleft) \vdash (\triangleright uxpq\triangleleft)$, if $x \in \tau(q)^*$, $a \notin \tau(q)$, and $p = \delta(q, a)$,
2. $(\triangleright uqv\triangleleft) \vdash (\triangleright uvp\triangleleft)$, if $v \in \tau(q)^*$ and $p = \delta(q, \triangleleft)$,
3. $(\triangleright xaysv\triangleleft) \vdash (\triangleright xpyv\triangleleft)$, if $y \in \tau(s)^*$, $a \notin \tau(s)$, and $p = \delta(s, a)$,
4. $(\triangleright usv\triangleleft) \vdash (\triangleright puv\triangleleft)$, if $u \in \tau(s)^*$ and $p = \delta(s, \triangleright)$.

As for the returning mode, we also consider non-returning sweeping finite automata. So, a *deterministic sweeping finite automaton with translucent letters working in the non-returning mode* (abbreviated as **nrSDFAwtl**) is an **nr2DFAwtl** with transition function enjoying the same restrictions as for **SDFAwtl**.

Throughout the rest of the paper, we will be denoting by $\mathcal{L}(X)$ the family of all languages accepted by some device X .

2.1 Basic Observations and Known Results

The computational capabilities of one-way deterministic and nondeterministic finite automata with translucent input letters working in the returning and non-returning mode (abbreviated, respectively, as **DFAwtl**, **NFAwtl**, **nrDFAwtl**, **nrNFAwtl**) are deeply investigated in [13]. For instance, all the families of languages accepted by these devices are shown to properly include the family of regular languages. Moreover, the proper inclusions $\mathcal{L}(\text{nrDFAwtl}) \subset \mathcal{L}(\text{nrNFAwtl})$, $\mathcal{L}(\text{NFAwtl}) \subset \mathcal{L}(\text{nrNFAwtl})$, $\mathcal{L}(\text{DFAwtl}) \subset \mathcal{L}(\text{NFAwtl})$, and $\mathcal{L}(\text{DFAwtl}) \subset \mathcal{L}(\text{nrDFAwtl})$ are also pointed out. So, these results witness that the resources *nondeterminism* and *non-returning mode* impact the recognition power.

The computational power of deterministic pushdown automata with translucent input letters (abbreviated as **DPDAwtl** and **nrDPDAwtl**) is studied in [9]. In particular, the following proper hierarchy is shown, where **DCFL** (resp., **DCSL**) is the family of deterministic context-free (resp., context-sensitive) languages:

$$\text{DCFL} = \mathcal{L}(\text{DPDA}) \subset \mathcal{L}(\text{DPDAwtl}) \subset \mathcal{L}(\text{nrDPDAwtl}) \subset \text{DCSL}.$$

Further comparisons among acceptance capabilities can be derived almost immediately by structural reasons and known results. For example, the deterministic context-free language $\{w\varphi(w^R) \mid w \in \{a, b\}^*, \varphi(a) = c, \varphi(b) = d\}$ is not accepted by any **nrNFAwtl** [13]. Due to the above-recalled inclusions, the proper inclusions $\mathcal{L}(\text{DFAwtl}) \subset \mathcal{L}(\text{DPDAwtl})$ and $\mathcal{L}(\text{nrDFAwtl}) \subset \mathcal{L}(\text{nrDPDAwtl})$ easily follow. Moreover, we recall from [10], that the families $\mathcal{L}(\text{DPDAwtl})$ and $\mathcal{L}(\text{nrDFAwtl})$ are incomparable.

Example 2 shows that the mirror language $\{ww^R \mid w \in \{a, b\}^*\}$ is accepted by some **SDFAwtl**. A straightforward modification of the construction yields an **SDFAwtl** for the language $\{w\varphi(w^R) \mid w \in \{a, b\}^*, \varphi(a) = c, \varphi(b) = d\}$. Since

this latter language is not accepted by any nrNFAwtl [13], we obtain the proper inclusions $\mathcal{L}(\text{DFAwtl}) \subset \mathcal{L}(\text{SDFAwtl})$ and $\mathcal{L}(\text{nrDFAwtl}) \subset \mathcal{L}(\text{nrSDFAwtl})$.

The next proposition deals with the language $L_{aba} = \{a^n b^n a^n \mid n \geq 0\}$, whose membership in $\mathcal{L}(\text{nrDFAwtl})$ is shown in [13]. On the contrary, we get

Proposition 3. $L_{aba} \notin \mathcal{L}(\text{SDFAwtl})$.

Proposition 3, the mentioned membership $L_{aba} \in \mathcal{L}(\text{nrDFAwtl})$, and the fact $\{w\varphi(w^R) \mid w \in \{a, b\}^*, \varphi(a) = c, \varphi(b) = d\} \in \mathcal{L}(\text{SDFAwtl}) \setminus \mathcal{L}(\text{nrDFAwtl})$, give

Corollary 4. *The families $\mathcal{L}(\text{SDFAwtl})$ and $\mathcal{L}(\text{nrDFAwtl})$ are incomparable.*

By the witness language L_{aba} , we are also able to obtain

Proposition 5. $\mathcal{L}(\text{SDFAwtl}) \subset \mathcal{L}(\text{nrSDFAwtl})$.

Proof. The inclusion follows by structural arguments. Moreover, since $L_{aba} \in \mathcal{L}(\text{nrDFAwtl})$, we have $L_{aba} \in \mathcal{L}(\text{nrSDFAwtl})$ as well. Finally, Proposition 3 proves $L_{aba} \notin \mathcal{L}(\text{SDFAwtl})$, whence the result follows. $\square$

These preliminary relationships between language families raise the natural question of how the resources *two-way head movement* and *sweeping* compare to each other and to the resource *pushdown store*. In what follows, we are going to explore these relationships, which are finally summarized in Figure 1, Section 5.

3 Sweeping vs. Two-Way Head Movement

We have introduced two-way and sweeping finite automata with translucent letters. Model definition directly implies the inclusions $\mathcal{L}(\text{SDFAwtl}) \subseteq \mathcal{L}(\text{2DFAwtl})$ and $\mathcal{L}(\text{nrSDFAwtl}) \subseteq \mathcal{L}(\text{nr2DFAwtl})$. A natural question arises of whether two-way strictly outperforms sweeping, i.e., whether these inclusions are proper. Here, we are going to show that the answer depends on whether the returning or non-returning mode is assumed. Let us start with the returning mode.

Lemma 6. *For any given 2DFAwtl, an equivalent SDFAwtl can effectively be constructed.*

As a direct consequence of Lemma 6, we get

Theorem 7. $\mathcal{L}(\text{SDFAwtl}) = \mathcal{L}(\text{2DFAwtl})$.

The same comparison between non-returning devices leads to a completely different result. In order to separate non-returning devices, the following witness language $L_{ta} = \{a^m \# a^{m+n} \# a^n \mid m, n \geq 0\}$ will be considered. First, we show that L_{ta} does not belong to the sweeping family.

Theorem 8. $L_{ta} \notin \text{nrSDFAwtl}$.

Finally, we are left to prove that L_{ta} can be accepted by some nr2DFAwtl .

Lemma 9. $L_{ta} \in \mathcal{L}(\text{nr2DFAwtl})$.

Theorem 8 and Lemma 9 enable us to conclude with

Theorem 10. $\mathcal{L}(\text{nrSDFAwtl}) \subset \mathcal{L}(\text{nr2DFAwtl})$.

4 Two-Way Head Movement vs. Pushdown Store

The translucency paradigm has been imposed in [9, 14, 17] to pushdown devices, leading to the definition of DPDAwtl and nrDPDAwtl . Here, we are going to compare the recognition power of these two latter devices with that of two-way and sweeping translucent finite automata. In other words, we investigate the impact of adding a pushdown store to translucent deterministic finite automata vs. adding two-way or sweeping input head movement.

To this end, we consider the language $L = \{w \in \{a, b\}^+ \mid |w|_a = |w|_b\}$, which is accepted by DFAwtl [17]. Then, we intersect L with the regular language $R = \{aa, bb\}^*$, to obtain the language $L_{dab} = L \cap R$. For L_{dab} , we can prove

Theorem 11. $L_{dab} \notin \text{nr2DFAwtl}$.

We are now ready to account for incomparability results:

Theorem 12. *Each of the families $\mathcal{L}(\text{nrDPDAwtl})$ and $\mathcal{L}(\text{DPDAwtl})$ is incomparable with each of the families $\mathcal{L}(\text{SDFawtl})$, $\mathcal{L}(\text{nrSDFawtl})$, $\mathcal{L}(\text{nr2DFAwtl})$.*

Proof. Being a deterministic context-free language, L_{dab} is accepted by some DPDAwtl and, thus, by some nrDPDAwtl as well. On the other hand, by Theorem 11, we have that L_{dab} is not accepted by any nr2DFAwtl and, thus, not by any nrSDFawtl and, thus, not by any SDFawtl .

In addition, the mirror language $\{ww^R \mid w \in \{a, b\}^*\}$ is not accepted by any DPDAwtl or nrDPDAwtl . However, by Example 2, we get that the mirror language it is accepted by some SDFawtl and, thus, by some nrSDFawtl and, thus, by some nr2DFAwtl . $\square$

5 The Final Language Scenario

The hierarchy here sketches the relationships among the recognition power of our two-way translucent automata and that of other known translucent devices.

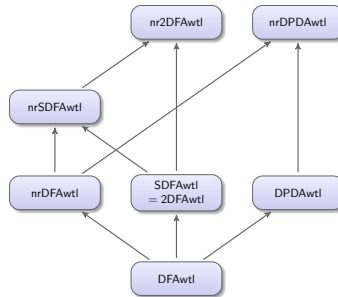


Fig. 1. Relationships between language families. An arrow between families indicates a strict inclusion. Families not connected by a path are incomparable.

6 Closure Properties

We investigate closure properties for the families $\mathcal{L}(\text{SDFAwtl}) = \mathcal{L}(\text{2DFAwtl})$, $\mathcal{L}(\text{nrSDFAwtl})$, and $\mathcal{L}(\text{nr2DFAwtl})$. Our results are collected at the end of this section in Table 1, page 13. While on this subject, we solve an open problem on nrDFAwtl and nrNFAwtl posed by Mráz and Otto in [13].

6.1 Boolean Operations

The following lemma proves that all families under consideration are closed under complementation:

Lemma 13. *From any given automaton of type $X \in \{\text{SDFAwtl}, \text{nrSDFAwtl}, \text{nr2DFAwtl}\}$, an equivalent automaton of the same type can effectively be constructed, that halts on any input.*

The following proposition gives a complete account on closure under Boolean operations. We note that the results for the family $\mathcal{L}(\text{SDFAwtl})$ have already been obtained in terms of linear automata with translucent letters [18, 19].

Proposition 14. *The language families $\mathcal{L}(\text{SDFAwtl})$, $\mathcal{L}(\text{nrSDFAwtl})$, and $\mathcal{L}(\text{nr2DFAwtl})$ are closed under complementation, but they are neither closed under union nor under intersection, even if the union or intersection are taken with a regular language.*

6.2 Solving an Open Problem

Before turning to further operations, we address an open problem posed by Mráz and Otto in [13]. The question is whether or not the language families nrDFAwtl and nrNFAwtl are closed under intersection with regular languages. We recall that an nrNFAwtl is a nondeterministic finite automaton with translucent input letters in the non-returning mode. As usual, nondeterminism is obtained by extending the transition function of an nrDFAwtl as

$$\delta: Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \rightarrow 2^Q \cup \{\text{accept}\}. \quad (1)$$

We answer the open question negatively. The witness language is again L_{dab} , used in Theorem 11 for deterministic devices, but here considered for non-determinism. So, we cannot immediately conclude $L_{dab} \notin \mathcal{L}(\text{nrNFAwtl})$ from known results, and a new proof is needed. In fact, we are going to show that $L_{dab} \notin \mathcal{L}(\text{nrSNFAwtl})$, where an nrSNFAwtl is a nondeterministic sweeping finite automaton with translucent letters in the non-returning mode (roughly speaking, an nrSDFAwtl equipped with a nondeterministic dynamic as in (1)):

Theorem 15. *The language L_{dab} cannot be accepted by any nrSNFAwtl (and hence, a fortiori, by any nrNFAwtl as well).*

Proof. As in the proof of Theorem 11, we assume by contradiction an nrSNFAwtl $M = \langle Q, \Sigma, q_0, \triangleright, \triangleleft, \tau, \delta \rangle$ for L_{dab} , and consider accepting computations on input words of the form $a^{x_1} b b a^{x_2} b^y a^{x_3} b b a^{x_4}$, where $y = x_1 + x_2 + x_3 + x_4 - 4$ and all x_i with $i \in \{1, 2, 3, 4\}$ are even and large enough. Let us focus on the computations on blocks. Even with M being an nrSNFAwtl , we get that more than $|Q|$ consecutive steps on a block without jumping would result in seeing the same input symbol in the same state twice. Thus, the input factor consumed in between could be deleted, still accepting, a contradiction. So, M must jump while processing a long block.

An accepting computation on some input $a^{x_1} b b a^{x_2} b^y a^{x_3} b b a^{x_4}$ begins in the configuration $(\triangleright q_0 a^{x_1} b b a^{x_2} b^y a^{x_3} b b a^{x_4} \triangleleft)$, from which M starts processing the input prefix a^{x_1} and eventually jumps to the right, say, in the state q_1 . For $\tau(q_1) = \{a, b\}$, the right end is reached with a jump. However, since the rightmost block of the input is also an a -block, in this case M continues to read a 's. So, jumping from the left end to the right end and back can occur only finitely often, after which M has to jump to a b -block. Without loss of generality, we may assume that this jump shows up at the left end, say, in state q_2 . So, we have $\tau(q_2) = \{a\}$, and obtain

$$\begin{aligned} (\triangleright q_0 a^{x_1} b b a^{x_2} b^y a^{x_3} b b a^{x_4} \triangleleft) &\vdash^* (\triangleright q_2 a^{x'_1} b b a^{x_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) \\ &\vdash (\triangleright a^{x'_1} q_3 b a^{x_2} b^y a^{x_3} b b a^{x'_4} \triangleleft), \end{aligned}$$

where $x_1 \geq x'_1$ and $x_4 \geq x'_4$. From now on, we distinguish between two cases:

CASE 1: M continues the computation by reading the next input symbol b , and possibly some of the following input symbols a before jumping on the a -block to the right, say, in the state q_4 . So, we have $a \in \tau(q_4)$. If $\tau(q_4) = \{a\}$, we get

$$\begin{aligned} (\triangleright a^{x'_1} q_3 b a^{x_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) &\vdash^* (\triangleright a^{x'_1} q_4 a^{x'_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) \\ &\vdash (\triangleright a^{x'_1+x'_2} q_5 b^{y-1} a^{x_3} b b a^{x'_4} \triangleleft), \end{aligned}$$

where $x_2 \geq x'_2$. If $\tau(q_4) = \{a, b\}$, we get

$$\begin{aligned} (\triangleright a^{x'_1} q_3 b a^{x_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) &\vdash^* (\triangleright a^{x'_1} q_4 a^{x'_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) \\ &\vdash (\triangleright a^{x'_1+x'_2} b^y a^{x_3} b b a^{x'_4} q_5 \triangleleft). \end{aligned}$$

Therefore, also the computation

$$\begin{aligned} (\triangleright q_0 a^{x_1+1} b b a^{x_2-1} b^y a^{x_3} b b a^{x_4} \triangleleft) &\vdash^* (\triangleright a^{x'_1+1} q_3 b a^{x_2-1} b^y a^{x_3} b b a^{x'_4} \triangleleft) \\ &\vdash^* (\triangleright a^{x'_1+1} q_4 a^{x'_2-1} b^y a^{x_3} b b a^{x'_4} \triangleleft) \\ &\vdash (\triangleright a^{x'_1+x'_2} q_5 b^{y-1} a^{x_3} b b a^{x'_4} \triangleleft) \end{aligned}$$

ends accepting. If $\tau(q_4) = \{a, b\}$, then the last configuration in the sub-computation is $(\triangleright a^{x'_1+x'_2} b^y a^{x_3} b b a^{x'_4} q_5 \triangleleft)$. In both cases for $\tau(q_4)$, we have that the input $a^{x_1+1} b b a^{x_2-1} b^y a^{x_3} b b a^{x_4}$ is accepted. However, since x_1 and x_2 are even, $x_1 + 1$ and $x_2 - 1$ are odd, and the input cannot belong to L_{dab} , a contradiction.

CASE 2: M continues by jumping over the next input symbol b in the state q_3 . If $\tau(q_3) = \{a, b\}$, then M will jump to the right end, and this case is treated below. So, we now assume $\tau(q_3) = \{b\}$, and obtain

$$(\triangleright a^{x'_1} q_3 b a^{x_2} b^y a^{x_3} b b a^{x'_4} \triangleleft) \vdash (\triangleright a^{x'_1} b q_4 a^{x_2-1} b^y a^{x_3} b b a^{x'_4} \triangleleft).$$

Next, M may read some a 's from the factor a^{x_2} , then jump to the right end or to the factor b^y from which it may read some symbols, then jump to the right end or to the factor a^{x_3} from which it may read some symbols, then jump to the right end or to the factor bb . Assume that M jumps on the factor bb in the state p_6 . Then, we have $\tau(p_6) = \{a\}$ and

$$(\triangleright a^{x'_1} b a^{x'_2} b^{y'} p_6 a^{x'_3} b b a^{x'_4} \triangleleft) \vdash (\triangleright a^{x'_1} b a^{x'_2} b^{y'} a^{x'_3} p_7 b a^{x'_4} \triangleleft),$$

where $y \geq y'$ and $x_3 \geq x'_3$. Now, assume $b \notin \tau(p_7)$. Then, M reads the second b , some a 's from the factor $a^{x'_4}$, and then jumps to the right end. This leads to a configuration $(\triangleright a^{x'_1} b a^{x'_2} b^{y'} a^{x'_3+x'_4} p_8 \triangleleft)$. But then, also the computation

$$\begin{aligned} (\triangleright q_0 a^{x_1} b b a^{x_2} b^y a^{x_3+1} b b a^{x_4-1} \triangleleft) &\vdash^* (\triangleright a^{x'_1} b a^{x'_2} b^{y'} a^{x'_3+1} p_7 b a^{x'_4-1} \triangleleft) \\ &\vdash^* (\triangleright a^{x'_1} b a^{x'_2} b^{y'} a^{x'_3+x'_4} p_8 \triangleleft) \end{aligned}$$

ends accepting, which is a contradiction since $x_3 + 1$ is odd. This contradiction implies that M cannot read the next b in state p_7 . So, it directly jumps either to the right end by $\tau(p_7) = \{a, b\}$ or to the rightmost a -block by $\tau(p_7) = \{b\}$, and then to the right end.

Finally, the situation where M has reached the right end remains to be considered. Basically, we now have to distinguish between two types of configurations. The first type has the form $(\triangleright a^\alpha b a^\beta b^\gamma a^\kappa b b a^\sigma r_1 \triangleleft)$, while the second has the form $(\triangleright a^\alpha b a^\beta b^\gamma a^\kappa b a^\sigma r_1 \triangleleft)$, with $\alpha, \beta, \gamma, \kappa, \sigma$ still being large enough. As observed, jumping from one end to the opposite end and back may occur only finitely often. So, M has to jump over the a -block to the adjacent b either from right or from left. This means, that either one b from the factor bb is read, or a single b . In the former case, we apply the same argument once more, obtaining that either the entire factor bb or the sole b is processed. Thus, we obtain the same contradiction as before, since either input $a^{x_1+1} b b a^{x_2-1} b^y a^{x_3} b b a^{x_4}$ or input $a^{x_1} b b a^{x_2} b^y a^{x_3+1} b b a^{x_4-1}$ is accepted, but none of them belongs to L_{dab} . $\square$

A direct consequence of Theorem 15 is contained in the following corollary, which negatively answers the open problem in [13]:

Corollary 16. *None of the families $\mathcal{L}(nrDFAwtl)$ and $\mathcal{L}(nrNFAwtl)$ are not closed under intersection with regular languages.*

6.3 Concatenations and Homomorphisms

To assess closure under concatenation, iteration, and length-preserving homomorphisms, we already have all the ingredients.

Proposition 17. *The language families $\mathcal{L}(\text{SDFAwtl})$, $\mathcal{L}(\text{nrSDFAwtl})$, and $\mathcal{L}(\text{nr2DFAwtl})$ are neither closed under concatenation nor under iteration nor under length-preserving homomorphisms.*

To show non-closure under inverse homomorphism, we consider the language $L_{tb} = \{b^n \# b^n \# b^n \mid n \geq 1\}$.

Theorem 18. $L_{tb} \notin \text{nr2DFAwtl}$.

This impossibility result qualifies L_{tb} as witnessing non-closure under inverse homomorphism. However, to this end, we need the following positive result on the language $L_{td} = \{x \# y \# z \mid x, y, z \in \{a, b\}^+, |x|_a = |z|_a, |y|_b = |z|_b\}$.

Lemma 19. $L_{td} \in \mathcal{L}(\text{SDFAwtl})$.

In conclusion, we can state:

Proposition 20. *The language families $\mathcal{L}(\text{SDFAwtl})$, $\mathcal{L}(\text{nrSDFAwtl})$, and $\mathcal{L}(\text{nr2DFAwtl})$ are not closed under non-erasing inverse homomorphisms.*

As a final remark, we feel it worth noticing that, for the family $\mathcal{L}(\text{SDFAwtl})$, the non-closure under concatenation and length-preserving homomorphism is shown in [18, 19] in terms of linear automata with translucent letters. However, the non-closure under iteration and inverse homomorphism is there stated as an open problem. The results in the present contribution clearly enable us to close such a problem.

The following table allows to grasp at a glance closure properties for several translucent devices.

Language Family	—	$\cup$	$\cap$	$\cap_{\text{reg}}$	$\cdot$	$*$	$h_{\text{len.pres.}}$	h^{-1}	R
$\mathcal{L}(\text{DPDAwtl})$	✓	✗	✗	✗	✗	✗	✗	✗	✗
$\mathcal{L}(\text{nrDPDAwtl})$	✓	✗	✗	✗	✗	✗	✗	✗	✗
$\mathcal{L}(\text{SDFAwtl})$	✓	✗	✗	✗	✗	✗	✗	✗	✓
$\mathcal{L}(\text{nrSDFAwtl})$	✓	✗	✗	✗	✗	✗	✗	✗	✓
$\mathcal{L}(\text{nr2DFAwtl})$	✓	✗	✗	✗	✗	✗	✗	✗	✓

Table 1. A summary of closure properties for the language families $\mathcal{L}(\text{DPDAwtl})$ and $\mathcal{L}(\text{nrDPDAwtl})$, as well as for $\mathcal{L}(\text{SDFAwtl}) = \mathcal{L}(\text{2DFAwtl})$, $\mathcal{L}(\text{nrSDFAwtl})$ and $\mathcal{L}(\text{nr2DFAwtl})$. The closure of the last three families under reversal (R) follows trivially. Non-closure properties marked ‘✗’ solve some open questions raised in [18, 19].

Acknowledgements. The authors wish to thank the anonymous referees for their valuable and helpful comments.

References

1. Bednářová, Z., Geffert, V., Mereghetti, C., Palano, B.: Removing nondeterminism in constant height pushdown automata. *Inform. Comput.* **237**, 257–267 (2014). <https://doi.org/10.1016/j.ic.2014.03.002>
2. Bensch, S., Bordihn, H., Holzer, M., Kutrib, M.: On input-revolving deterministic and nondeterministic finite automata. *Inform. Comput.* **207**, 1140–1155 (2009). <https://doi.org/10.1016/j.ic.2009.03.002>
3. Berman, P., Lingas, A.: On the complexity of regular languages in terms of finite automata. Tech. Rep. 304, Polish Academy of Sciences (1977)
4. Geffert, V., Bednářová, Z., Mereghetti, C., Palano, B.: Boolean language operations on nondeterministic automata with a pushdown of constant height. In: Bultov, A.A., Shur, A.M. (eds.) *Computer Science Symposium in Russia (CSR 2013)*. LNCS, vol. 7913, pp. 100–111. Springer (2013). https://doi.org/10.1007/978-3-642-38536-0_9
5. Hopcroft, J.E., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts (1979)
6. Jakobi, S., Meckel, K., Mereghetti, C., Palano, B.: Queue automata of constant length. In: Jürgensen, H., Reis, R. (eds.) *Descriptional Complexity of Formal Systems (DCFS 2013)*. LNCS, vol. 8031, pp. 124–135. Springer (2013). https://doi.org/10.1007/978-3-642-39310-5_13
7. Jančár, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: *Fundamentals of Computation Theory (FCT 1995)*. LNCS, vol. 965, pp. 283–292. Springer (1995). https://doi.org/10.1007/3-540-60249-6_60
8. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B.: Iterated uniform finite-state transducers on unary languages. *Theoretical Computer Science* **969**, 114049 (2023). <https://doi.org/10.1016/j.tcs.2023.114049>
9. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B., Raucci, P., Wendlandt, M.: Deterministic pushdown automata with translucent input letters. In: Day, J.D., Manea, F. (eds.) *Developments in Language Theory (DLT 2024)*. LNCS, vol. 14791, pp. 203–217. Springer (2024). https://doi.org/10.1007/978-3-031-66159-4_15
10. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B., Raucci, P., Wendlandt, M.: On properties of languages accepted by deterministic pushdown automata with translucent input letters. In: Fazekas, S.Z. (ed.) *Implementation and Application of Automata (CIAA 2024)*. LNCS, vol. 15015, pp. 208–220. Springer (2024). https://doi.org/10.1007/978-3-031-71112-1_15
11. Malcher, A., Mereghetti, C., Palano, B.: Descriptive complexity of two-way pushdown automata with restricted head reversals. *Theoretical Computer Science* **449**, 119–133 (2012). <https://doi.org/10.1016/j.tcs.2012.04.007>
12. Mereghetti, C., Palano, B.: Quantum automata for some multi-periodic languages. *Theoretical Computer Science* **387**, 177–186 (2007)
13. Mráz, F., Otto, F.: Non-returning deterministic and nondeterministic finite automata with translucent letters. *RAIRO Theor. Informatics Appl.* **57**, 8 (2023). <https://doi.org/10.1051/ita/2023009>
14. Nagy, B., Otto, F.: CD-systems of stateless deterministic R(1)-automata governed by an external pushdown store. *RAIRO Theor. Informatics Appl.* **45**, 413–448 (2011). <https://doi.org/10.1051/ITA/2011123>
15. Nagy, B., Otto, F.: Finite-state acceptors with translucent letters. In: Bel-Enguix, G., Dahl, V., De La Puente, A. (eds.) *International Workshop on AI Methods for Interdisciplinary Research in Language and Biology (BILC 2011)*. pp. 3–13. SciTePress (2011). <https://doi.org/10.5220/0003272500030013>

16. Nagy, B., Otto, F.: On CD-systems of stateless deterministic R-automata with window size one. *J. Comput. Syst. Sci.* **78**, 780–806 (2012). <https://doi.org/10.1016/J.JCSS.2011.12.009>
17. Nagy, B., Otto, F.: Deterministic pushdown-CD-systems of stateless deterministic R(1)-automata. *Acta Inform.* **50**, 229–255 (2013). <https://doi.org/10.1007/S00236-012-0175-X>
18. Nagy, B., Otto, F.: Two-head finite-state acceptors with translucent letters. In: *Theory and Practice of Computer Science (SOFSEM 2019)*. LNCS, vol. 11376, pp. 406–418. Springer (2019). https://doi.org/10.1007/978-3-030-10801-4_32
19. Nagy, B., Otto, F.: Linear automata with translucent letters and linear context-free trace languages. *RAIRO Theor. Informatics Appl.* **54**, 3 (2020). <https://doi.org/10.1051/ita/2020002>
20. Nagy, B., Otto, F.: Finite automata with sets of translucent words. In: Day, J.D., Manea, F. (eds.) *Developments in Language Theory (DLT 2024)*. LNCS, vol. 14791, pp. 236–251. Springer (2024). https://doi.org/10.1007/978-3-031-66159-4_17
21. Otto, F.: Restarting automata and their relations to the Chomsky hierarchy. In: Ésik, Z., Fülöp, Z. (eds.) *Developments in Language Theory (DLT 2003)*. LNCS, vol. 2710, pp. 55–74. Springer (2003). https://doi.org/10.1007/3-540-45007-6_5
22. Otto, F.: A survey on automata with translucent letters. In: Nagy, B. (ed.) *Implementation and Application of Automata (CIAA 2023)*. LNCS, vol. 14151, pp. 21–50. Springer (2023). https://doi.org/10.1007/978-3-031-40247-0_2
23. Otto, F.: *Restarting Automata*. Springer, Cham, Switzerland (2025)
24. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: *ACM Symposium on Theory of Computing (STOC 1978)*. pp. 275–286. Association for Computing Machinery (1978). <https://doi.org/10.1145/800133.804357>
25. Sipser, M.: Lower bounds on the size of sweeping automata. *J. Comput. System Sci.* **21**, 195–202 (1980). [https://doi.org/10.1016/0022-0000\(80\)90034-3](https://doi.org/10.1016/0022-0000(80)90034-3)