Partially-ordered Two-way Automata: A New Characterization of DA

Thomas Schwentick[1], Denis Thérien[2*], and Heribert Vollmer[3]

[1] Fachbereich Mathematik und Informatik, Philipps-Universität Marburg, Hans-Meerwein-Straße, 35032 Marburg, Germany
[2] School of Computer Science, McGill University, 3480 University Street, Montréal (Québec), H3A 2A7 Canada
[3] Theoretische Informatik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

Abstract. In this paper, we consider finite automata with the restriction that whenever the automaton leaves a state it never returns to it. Equivalently we may assume that the states set is partially ordered and the automaton may never move "backwards" to a smaller state.
We show that different types of partially-ordered automata characterize different language classes between level 1 and $\frac{3}{2}$ of the Straubing-Thérien-Hierarchy.
In particular, we prove that partially-ordered 2-way DFAs recognize exactly the class **UL** of *unambiguous languages* introduced by Schützenberger in 1976. As shown by Schützenberger, this class coincides with the class of those languages whose syntactic monoid is in the variety **DA**, a specific subclass of all "group-free" (or "aperiodic") semigroups. **DA** has turned out to possess a lot of appealing characterizations. Our result adds one more to these: partially-ordered two-way automata recognize exactly those languages whose syntactic monoid is in **DA**.

1 Introduction

Finite automata that may move their input head back and forth in both directions are well-known to recognize exactly the regular languages, see, e.g., [HU79], i.e., they are of the same power as usual one-way finite automata.

In spite of the naturalness of the model, there has been comparatively little work done on two-way automata (2DFAs). Some papers compare the conciseness of two-way automata with other models (e.g., [Mic81, Bir92, GH96, HS99]). Other work considered them as a means to compute translations on strings [EH99, MSTV00]. The present authors together with Pierre McKenzie developed the groundwork for an algebraic study of 2DFAs in [MSTV00] by presenting a sensible definition of the so called behavior monoid of such a machine, and by showing how restricted classes of 2-way automata with output correspond in a

* Supported by NSERC of Canada, by FCAR du Québec, and by the Alexander-von-Humboldt-Gesellschaft. Work done while on leave at the Universität Tübingen, Germany.

nice way to algebraically restricted NFAs and first-order logic with monoidal quantifiers. (A concept similar to the behaviour functions from [MSTV00] was examined from a different perspective much previously in [Bir89].)

Here, we take up the study of subclasses of 2DFAs. In particular, we look at the following natural restriction: Say that an automaton, one-way or two-way, deterministic or not, is partially-ordered (p.-o., for short) if, whenever M leaves a state, it will never return to it. This means that there is a partial order on the states of M such that, whenever M moves from state p to state q, then p is less than or equal to q in that order.

Partially-ordered one-way automata are well studied. It is known that, in the deterministic case, they recognize exactly those languages whose syntactic monoid is R-trivial, where we say that a monoid M is R-trivial if, whenever two monoid elements s, t satisfy $sM = tM$, then $s = t$. (Analogously one can define L-trivial as above but using $Ms = Mt$. Then clearly, p.-o. reverse automata, i.e., FAs moving their head from right to left in a one-way manner, recognize exactly those languages with an L-trivial syntactic monoid.) The intersection of both is known to equal level 1 of the Straubing-Thérien-Hierarchy (STH) of star-free regular languages, known to correspond to so called J-trivial monoids. For background on these classes, we refer the reader to [Pin97], see also [Pin86].

These classes have interesting combinatorial characterizations: Consider an alphabet A. A *left-deterministic product* over A is a concatenation of the form $A_0^* a_1 A_1^* \dots a_k A_k^*$, where $a_i \in A$, $A_i \subseteq A$, and $a_i \notin A_{i-1}$. Now a language is recognized by a p.-o. FA iff it is a finite disjoint union of left-deterministic products. (Equivalently, a language is recognized by a p.-o. reverse FA iff it is a finite disjoint union of right-deterministic products, where *right-deterministic* is defined replacing in the above the condition $a_i \notin A_{i-1}$ by $a_i \notin A_i$.)

Schützenberger in 1976 [Sch76] studied the following superclass of these: Say that an *unambiguous product* over alphabet A is a concatenation of the form $A_0^* a_1 A_1^* \dots a_k A_k^*$, where $a_i \in A$, $A_i \subseteq A$, if every word w has at most one factorization $w = w_0 a_1 w_1 \dots a_k w_k$ with $w_i \in A_i^*$. A language is *unambiguous* if it is a finite disjoint union of unambiguous products. The class **UL** is the class of all unambiguous languages. Schützenberger also obtained an algebraic characterization of **UL**. A monoid M belongs to the variety **DA** if it satisfies for every idempotent element $s \in M$ the condition that if $MsM = MtM$ then t is also idempotent. Schützenberger proved that a language is in **UL** iff its syntactic monoid belongs to **DA**.

Variety **DA** is known to have an impressive number of combinatorial, algebraic and logical characterizations, some of which will be used later in the present paper. It can be shown that **UL** is the intersection of level $\frac{3}{2}$ of the STH and its complement, i.e. languages in **UL** are exactly those that can be defined both by a first-order sentence in prenex normal-form with a Σ_2-prefix and by a sentence with a Π_2-prefix [PW97]. In this paper we develop a new machine characterization of **DA**: We prove that partially-ordered two-way automata recognize exactly those languages in **UL**. Hence a language L has a syntactic monoid in **DA** iff there is a p.-o. 2DFA recognizing L.

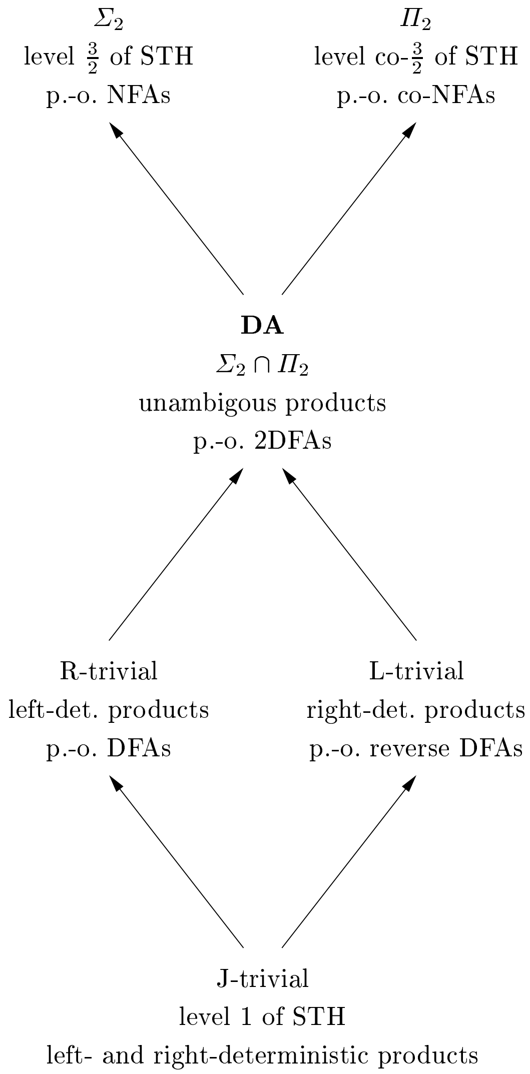

Fig. 1. Inclusions among considered language classes

An interesting intermediate step in our proof of equivalence of p.-o. 2DFAs and **DA** are so called *turtle languages*. Given an input word w and a position of w, a turtle move is an instruction of the form $(\leftarrow, a)$ or $(\rightarrow, a)$, meaning that the next position is determined by moving to the right (or left), until a letter a occurs. If no a is found, the move fails. A turtle language is a language that can be recognized by a sequence of such turtle moves, in the sense that a word w belongs to the language if and only if none of the moves in the sequence fails. We will show that a language has its syntactic monoid in **DA** iff it is a Boolean combination of turtle languages. Particular types of turtle languages can be related to further monoid varieties as we point out in Sect. 4.

Concerning partially-ordered NFAs, we will see that both in the one-way and in the two-way case these characterize level $\frac{3}{2}$ of the STH. Hence, a language can be accepted by a p.-o. 2DFA if and only if it and its complement can be recognized by a p.-o. NFA, either one-way or two-way.

Our results are summarized in Fig. 1. We see that partially-ordered finite automata can be used to characterize a number of classes of current focal interest in the study of concatenation hierarchies.

2 Preliminaries

2.1 Monoids and Congruences

An associative binary operation on a set containing an identity for this operation defines a *monoid*. By a monoid *variety*, we mean a pseudovariety in the sense of Straubing [Str94, pp. 72ff] or Eilenberg [Eil76, pp. 109ff]: it is a set of monoids closed under finite direct product, the taking of submonoids and of homomorphic images. Well-known examples of varieties include the commutative monoids, the groups $\mathbf{G}$, the aperiodics $\mathbf{A}$ (i.e., the monoids containing only trivial groups), the solvable groups $\mathbf{G}_{sol}$.

It is known that any set of monoids satisfying (or ultimately satisfying) a set of identities (in the sense described, e.g., in [Str94, Chapter V.6]) forms a variety. Of particular interest for this paper will be the following case: The variety **DA** consists of all finite semigroups satisfying $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$, where for any semigroup element s, s^ω denotes the least power of s that is idempotent. This variety was originally introduced by Schützenberger [Sch76] using the definition presented in Sect. 1, and is known to recognize the unambiguous languages, i.e., the finite unions of products $A_0^* a_1 A_1^* a_2 \cdots a_k A_k^*$ where every string is unambiguously factorizable w.r.t. this product; here, every $a_i \in A$ and every $A_i \subseteq A$ where A denotes the input alphabet.

Deterministic finite automata are, as usual [HU79, p. 17], given by $M = (S, A, \delta, s_0, F)$, where the different components denote the state set, the input alphabet, the transition function, the initial state, and the set of final states, respectively. The extended transition function [HU79, p. 17] of M is denoted by $\hat{\delta}$. The *transformation monoid* of M is the set $\left\{ \hat{\delta}(\cdot, w) \colon S \to S \;\middle|\; w \in \Sigma^* \right\}$ with the operation of composition of functions. Certainly, this set of transformations will always be finite. For a variety $\mathbf{V}$, we say that M is a $\mathbf{V}$-DFA if its transformation monoid is in $\mathbf{V}$. A $\mathbf{V}$-language is a language accepted by a $\mathbf{V}$-DFA. Non-deterministic automata are defined similarly, except that δ is now a function from $Q \times A \to 2^Q$ and we allow a set of possible initial states.

An equivalence relation $\sim$ on A^* is a *congruence* if $u \sim v$ implies $(\forall x, y \in A^*)[xuy \sim xvy]$. A congruence $\sim$ induces a monoid $A^*/\!\sim$ isomorphic to the transformation monoid of the pre-automaton $(A^*/\!\sim, A, ([u]_\sim, a) \mapsto [ua]_\sim)$. An example of a congruence is the syntactic congruence of a language L: $u \sim_L v$ iff $(\forall x, y \in A^*)[xuy \in L$ iff $xvy \in L]$. Let a DFA $M = (S, A, \delta, s, F)$ be given. Another example of a congruence is: $u \sim_M v$ iff $\hat{\delta}(\cdot, u) = \hat{\delta}(\cdot, v)$. Then $A^*/\!\sim_M$ is isomorphic to the transformation monoid of M. Writing $\sim_{pq}$ for the syntactic congruence of the language $L(p, q) = \{w \mid \hat{\delta}(p, w) = q\}$, there is an injective morphism $A^*/\!\sim_M \longrightarrow \Pi_{p,q\in S}(A^*/\!\sim_{pq})$ and a surjective morphism $A^*/\!\sim_M \longrightarrow A^*/\!\sim_{pq}$. These facts can be shown to imply that M is a $\mathbf{V}$-DFA iff $L(p, q)$ is a $\mathbf{V}$-language for each $p, q \in S$.

2.2 Two-way Automata

As mentioned in the introduction, the ground for an algebraic study of two-way automata was laid in [MSTV00]. The following definitions are mainly taken from that paper. We remark that our results also remain valid for the standard definition, see [HU79].

A two-way automaton is a tuple $M = (\mathcal{L} \uplus \mathcal{R}, A, \delta, l_0, F)$, where the set S of states is the disjoint union $\mathcal{L} \uplus \mathcal{R}$ of a set $\mathcal{L}$ (the states "entered from the left") and a set $\mathcal{R}$ (the states "entered from the right"), $l_0 \in \mathcal{L}$ is the *initial state*, $F \subseteq S$ is the set of *final states*, A is the *input alphabet*, $\delta\colon (S \times A) \cup (\mathcal{L} \times \{\lhd\}) \cup (\mathcal{R} \times \{\rhd\}) \to S$ is a total *transition function*, where $\rhd \notin A$ and $\lhd \notin A$ are the *leftmarker* and the *rightmarker* respectively,

The meaning of $\delta(s, a) \in \mathcal{L}$ is that M in state s scanning a moves its head to the *right* upon entering state $\delta(s, a)$; M moves its head to the *left* when $\delta(s, a) \in \mathcal{R}$.

The *initial configuration* of M on input $w = w_1 w_2 \cdots w_n \in A^*$ is the situation in which the state of M is l_0 and M scans w_1 within the string $\rhd w_1 w_2 \cdots w_n \lhd$ (M scans $\lhd$ when $|w| = 0$). We say that M *eventually exits* $\rhd w \lhd$ if it eventually encounters a transition $\delta(r, \rhd) \in \mathcal{R}$ or a transition $\delta(l, \lhd) \in \mathcal{L}$ (of course M will generally bounce off the end markers several times before exiting). We require that, for any $w = w_1 w_2 \cdots w_n \in A^*$,

- M from its initial configuration on w eventually leaves every w_i to the right, $1 \le i \le n$, and eventually exits $\rhd w \lhd$; this is analogous to the (unspoken) requirement that a one-way automaton must traverse its input and halt,
- from *any* state $l \in \mathcal{L}$ scanning w_1, M eventually exits $\rhd w \lhd$; this requirement is analogous to the (unspoken) fact that a one-way automaton eventually runs out of input regardless of its initial configuration,
- from *any* state $r \in \mathcal{R}$ scanning w_n, M also eventually exits $\rhd w \lhd$; this is justified by the natural desire to maintain symmetry between left and right

in a two-way machine. The input word is *accepted* by M if M exits $\rhd w \lhd$ in a final state.

Each $w = w_1 w_2 \cdots w_n \in A^*$ coerces M into a behavior described by a *behavior function* $\delta_w : \mathcal{L} \uplus \mathcal{R} \to \mathcal{L} \uplus \mathcal{R}$, where $\delta_w(s)$ is the state in which M exits $w_1 w_2 \cdots w_n$ when started in state s scanning w_1, if $s \in \mathcal{L}$; and $\delta_w(s)$ is the state in which M exits $w_1 w_2 \cdots w_n$ when started in state s scanning w_n, if $s \in \mathcal{R}$. Finally we define the *behavior monoid* $\mathcal{B}(M)$ of M to be the monoid $\{\delta_w \mid w \in A^*\}$ under the operation $\delta_u \circ \delta_v =_{\text{def}} \delta_{uv}$. (Note that by this definition, $\circ$ is *not* composition of functions.) M is a $\mathbf{V}$-*machine* iff $\mathcal{B}(M) \in \mathbf{V}$.

As in the one-way case, the non-deterministic version of the model is obtained by allowing the transition relation to be multiple-valued and allowing several initial states.

2.3 Partially-ordered Automata

Central for this paper will be the following definition: An automaton M (one-way or two-way, deterministic or not) is *partially-ordered* (p.-o., for short) if there is a partial order $\prec$ on the state set S of M such that whenever $\delta(s, a) = s'$ then $s = s'$ or $s \prec s'$. This means that whenever M leaves a particular state, it can never return to it.

Let M be a partially ordered automaton with input alphabet A. With the computation of M on an input w we can associate a *characteristic sequence* $S = s_1, \ldots, s_l$, consisting of tuples (p_i, A_i, a_i) where

- $p_1, \ldots, p_l$ is the sequence of distinct states that M assumes on input w in that order (and, of course, p_1 is the initial state of M),
- M stays in p_i while reading symbols from $A_i \subseteq A$, and
- M goes from state p_i into state $p_{i+1} \neq p_i$ by reading the symbol a_i (in w).

Note that each state of M occurs at most once in S, thus, the number of different state sequences is finite and depends only on M. If M is a 2DFA, recall that each state of M is entered either only from the left or only from the right. Therefore, all movements of M that let it stay in a certain state are in the same direction (for that state).

For a characteristic sequence S, we denote by $L(S)$ the set of those strings accepted by M when obeying sequence S.

The following is known about the power of deterministic p.-o. (one-way) automata. Say that a monoid M is R-trivial (L-trivial, resp.) if, whenever two monoid elements s, t satisfy $sM = tM$ ($Ms = Mt$, resp.) then $s = t$. A language has a syntactic monoid that is R-trivial iff it is recognized by a p.-o. DFA. A language has a syntactic monoid that is L-trivial iff it is recognized by a p.-o. DFA that moves its head from right to left in a one-way manner.

To our best knowledge, nondeterministic p.-o. FAs have not been looked at so far. However, it can be seen that both in the one-way and two-way case they recognize exactly level 3/2 of the Straubing-Thérien-Hierarchy.

Proposition 1. *A language L can be recognized by a non-deterministic one-way p.-o. automaton iff it can be recognized by a non-deterministic two-way p.-o. automaton iff it belongs to level $\frac{3}{2}$ of the Straubing-Thérien-Hierarchy (i.e. Σ_2).*

Proof. It is known [Arf91] that a language is in Σ_2 iff it is a union of languages of the form $A_0^* a_1 A_1^* \dots a_s A_s^*$, where a_i are alphabet symbols and A_i are subsets of the alphabet. The direct construction of a one-way non-deterministic automaton from the expression will yield a p.-o. machine. Of course, every one-way automaton is also a two-way automaton. Finally, consider a non-determinsitic two-way p.-o. automaton M. Take any word w that is accepted by M. Let S be a characteristic sequence of M encountered during the computation, that ends in an accepting state. Let $(q_0, q_1, \dots, q_s)$ be the sequence of states in S. Consider the induced decomposition of w as $w = w_0 a_1 w_1 \dots a_s w_s$, where the a_i's correspond to positions of w where the computation of M has changed its state (and possibly its direction). Consider the i^{th} segment of this decomposition. It may have been traversed from left to right a number of times, e.g., while in the states $q_{i_1}, \dots, q_{i_p}$, and from right to left, a number of times, e.g. while in states $q_{j_1}, \dots, q_{j_r}$. Let A_i be the set of letters a with the property that for each t between 1 and p, q_{i_t} is in $\delta(q_{i_t}, a)$ and for each t between 1 and r, q_{j_t} is in $\delta(q_{j_t}, a)$. Then all words that are in $A_0^* a_1 A_1^* \dots a_s A_s^*$ will also be accepted by M, via a computation that traverses exactly the same states as the one on w. Taking the union over the finitely many possible monotone sequences of states that end up in an accepting state will yield a Σ_2 expression for the language recognized by M. $\square$

3 Main Result

Before we state and prove the main result of this paper we need some further definitions. Let A be an alphabet. A *turtle instruction* is a pair $I = (d, \sigma)$ consisting of a *direction* $d \in \{\leftarrow, \rightarrow\}$ and a *target symbol* $\sigma \in A$. If $d = \leftarrow$ we say that I is a $\leftarrow$-instruction, otherwise a $\rightarrow$-instruction. A *turtle program* P is a non-empty sequence $I_1 \cdots I_k$ of turtle instructions.

The semantics of turtle programs is defined as follows. Let $w = w_1 \cdots w_n$ be a string of some length n and let $i \in \{0, \dots, n+1\}$, i.e., either i is a position in w or $i = 0$ or $i = n+1$. Let $I = (\leftarrow, \sigma)$ be a $\leftarrow$-instruction. Let $j_0 = \max\{j \in \{1, \dots, i-1\} \mid w_j = \sigma\}$. We define $I(w,i) = (w, j_0)$ if j_0 exists, i.e., if $\{j \in \{1, \dots, i-1\} \mid w_j = \sigma\}$ is non-empty. Otherwise, $I(w,i)$ is undefined, denoted as $I(w,i) = \perp$. Correspondingly, we define $I(w,i)$ for $\rightarrow$-instructions but in this case j_0 is defined as $\min\{j \in \{i+1, \dots, n\} \mid w_j = \sigma\}$. For a turtle program $P = I_1 \cdots I_k$ and a string w of length n we define the *final position of P on w* as follows.

- If I_1 is a $\rightarrow$-instruction and $(w,j) = I_k(\cdots(I_1(w,0))\cdots)$ is defined then we set $P(w) = j$;

– If I_1 is a $\leftarrow$-instruction and $(w,j) = I_k(\cdots(I_1(w,n+1))\cdots)$ is defined then, again, we set $P(w) = j$;
– Otherwise, we say that $P(w)$ is undefined and write $P(w) = \perp$.

Note that the interpretation of a turtle program starts at the left end of a string if its first instruction is a $\rightarrow$-instruction and at the right end otherwise. It should also be noted that if $P(w)$ is not undefined it is always a position of w, i.e., $P(w) \in \{1,\ldots,n\}$.

We write $L(P)$ for the set of strings w for which $P(w)$ is defined. Languages of the form $L(P)$ are called *turtle languages.*

It is not hard to see that turtle languages can be accepted by partially ordered two-way automata. On the other hand, we will show that, if a language has a syntactic monoid in **DA**, it can be expressed as a Boolean combination of turtle languages. Therefore, turtle languages are, in a sense, a link between **DA** and partially ordered two-way automata. The above relations and their convereses are the contents of our main result, stated formally as follows:

Theorem 2. *For a language L, the following are equivalent.*

(a) The syntactic monoid of L is in **DA**.
(b) L is a Boolean combination of turtle languages.
(c) L is accepted by a partially ordered two-way automaton.

The proof is given by a series of lemmas. The first one is of a technical nature. Given two turtle programs, P_1, P_2, we write $L(P_1,P_2)$ for the set of strings $w \in L(P_1) \cap L(P_2)$ for which $P_1(w) < P_2(w)$. In the following, we do not distinguish in notation between a turtle instruction I and program with the single instruction I.

Lemma 3. *For every turtle program P and every turtle instruction I, the languages $L(I,P)$ and $L(P,I)$ can be expressed as Boolean combinations of turtle languages.*

Proof. Let $I = (d,\sigma)$. We write I^{-1} for the *inverse* of I, i.e., the instruction $(\rightarrow,\sigma)$ if $d = \leftarrow$, and $(\leftarrow,\sigma)$ if $d = \rightarrow$. If $d = \rightarrow$ then the following hold, for every string w.

$$\begin{aligned} w \in L(I,P) &\iff w \in L(PI^{-1}). \\ w \in L(P,I) &\iff w \in \overline{L(PII^{-1})} \cap L(PI). \end{aligned}$$

In the first equivalence, the term $L(PI^{-1})$ intuitively says that there is a σ in w to the left of $P(w)$, hence $I(w) < P(w)$. In the second equivalence, $L(PI)$ contains all w for which there is a σ to the right of $P(w)$. The term $\overline{L(PII^{-1})}$ rules out all strings for which $I(w) \leq P(w)$.

If $d = \leftarrow$ then, similarly, the following hold, for every string w.

$$\begin{aligned} w \in L(I,P) &\iff w \in \overline{L(PII^{-1})} \cap L(PI), \\ w \in L(P,I) &\iff w \in L(PI^{-1}). \end{aligned}$$

□

Lemma 4. *If the syntactic monoid of a language L is in* **DA** *then L is a Boolean combination of turtle languages.*

Proof. Let the syntactic monoid of L be in **DA**.

Recall the following characterization of **DA** from [TW98]: For a word w, let $\alpha(w)$ denote the set of letters occurring in w. If $a \in \alpha(w)$ we say that $w = uav$ is an a-left decomposition of w if $a \notin \alpha(u)$. Analogously, $w = uav$ is an a-right decomposition of w if $a \notin \alpha(v)$. We define a relation $\sim_n$ on words (shown to be a congruence in [TW98]) by induction on n:

First, $x \sim_0 y$ for all strings x, y. Next, for $n > 0$, $x \sim_n y$ if $\alpha(x) = \alpha(y)$ and

(1) for all $a \in \alpha(x)$, if $x = x_0ax_1$ and $y = y_0ay_1$ are the a-left decompositions of x and y, then $x_0 \sim_n y_0$ and $x_1 \sim_{n-1} y_1$, and
(2) for all $a \in \alpha(x)$, if $x = x_0ax_1$ and $y = y_0ay_1$ are the a-right decompositions of x and y, then $x_0 \sim_{n-1} y_0$ and $x_1 \sim_n y_1$.

[TW98] prove that a language has its syntactic monoid in **DA** iff it is a finite union of equivalence classes of $\sim_n$ for some n. As in [TW98] it should be noted that this characterization is well-defined because $|\alpha(x_0)| < |\alpha(x)|$ in (1) and $|\alpha(x_1)| < |\alpha(x)|$ in (2).

We show in the following that each class C of $\sim_n$ is a Boolean combination of turtle languages. The proof is by induction on n and the size of the alphabet of C. Note that by the above definition, all strings in C use the same letters, if $n > 0$.

For $n = 0$ the proof is straightforward, as A^* is $L(I) \cup \overline{L(I)}$ for some turtle move I.

Now let $n > 0$, C be an equivalence class of $\sim_n$ with alphabet $B \subseteq A$, y a representant of C and $a \in B$. Let y_0ay_1 be the a-left-decomposition of y. We show next how the language of strings $w = w_0aw_1$ with $w_0 \sim_n y_0$ and $w_1 \sim_{n-1} y_1$ can be expressed as a Boolean combination of turtle languages. Given the above characterization of **DA**-languages, and as the case of right-decompositions can be handled completely analogously, the statement of the lemma follows directly. L is simply the intersection of all the Boolean combination obtained.

First, we show how the language L_0 of strings $w = w_0aw_1$ with $w_0 \sim_n y_0$ can be expressed. By induction, as $|\alpha(y_0)| < |B|$, the equivalence class C_0 of y_0 (over the alphabet $B - \{a\}$) can be expressed as a Boolean combination of turtle languages. Let $L(P)$ be a language occuring in this characterization. Let $I = (\rightarrow, a)$. If the first instruction of P is a $\rightarrow$-instruction then

$$w_0 \in L(P) \quad \text{if and only if} \quad w \in L(P) \cap \bigcap_{P' \sqsubseteq P} L(P', I).$$

Here, $P' \sqsubseteq P$ denotes that P' is a non-empty prefix of P. If the first instruction of P is a $\leftarrow$-instruction then

$$w_0 \in L(P) \quad \text{if and only if} \quad w \in L(IP) \cap \bigcap_{P' \sqsubseteq P} L(IP', I).$$

It is straightforward to check that these equivalences indeed hold. Note that the big intersections on the right ensure that the application of P on w only uses w_0. By Lemma 3 it follows that the languages of the form $L(P', I)$ and $L(IP', I)$ can be expressed as Boolean combinations of turtle languages.

By replacing each $L(P)$ in the characterization of C_0 with the corresponding right hand expression we obtain a characterization for L_0.

Finally, we show how the language L_1 of strings $w = w_0 a w_1$ with $w_1 \sim_{n-1} y_1$ can be expressed. Again, by induction, the equivalence class C_1 of y_1 can be expressed as a Boolean combination of turtle languages. Let $L(P)$ be a language occuring in this characterization. If P starts with a $\rightarrow$-instruction then

$$w_1 \in L(P) \quad \text{if and only if} \quad w \in L(IP) \cap \bigcap_{P' \sqsubseteq P} L(I, IP').$$

If P starts with a $\leftarrow$-instruction then

$$w_1 \in L(P) \quad \text{if and only if} \quad w \in L(P) \cap \bigcap_{P' \sqsubseteq P} L(I, P').$$

Again, the languages on the right hand side can be expressed as Boolean combinations of turtle languages and we obtain the desired characterization of L_1. This completes the proof of the lemma. $\square$

Lemma 5. *If a language L is accepted by a p.-o. 2DFA, then its syntactic monoid is in* **DA**.

Proof. Let M be a p.-o. 2DFA accepting L. Let S be a characteristic sequence of M (see Sect. 2.3), $S = (p_1, A_1, a_1)(p_2, A_2, a_2) \cdots (p_n, A_n, a_n)$, where the p_i are states of M, A_i are alphabet subsets, and a_i are alphabet symbols. We will show that the syntactic monoid of $L(S)$ is in **DA** by proving that it satisfies the equation $(xyz)^n y (xyz)^n = (xyz)^n$. For this, it is sufficient to show that, for all words u, v, x, y, z, we have $u(xyz)^n y (xyz)^n v \in L(S)$ iff $u(xyz)^n v \in L(S)$. The statement of the lemma then follows as L is a union of languages $L(S)$ and **UL** is closed under Boolean combinations.

Let w_l be a tape content of M of the form $\triangleright u(xyz)^n y (xyz)^n v \triangleleft$, and let w_r be of the form $\triangleright u(xyz)^n v \triangleleft$. Suppose that M switches from p_1 to p_2 while reading the prefix $\triangleright uxyz$ of w_l. Then, certainly, the same switch will occur at the same position when reading w_r. If M, reading w_l, does not switch from p_1 to p_2 while within $\triangleright uxyz$ then this switch can only occur in the suffix $v \triangleleft$. Again, the same switch must occur at exactly the same position within $v \triangleleft$ when M reads w_r. Note that this means that in the latter case, M in state p_1 skips every letter in u, x, y, and z.

In general, consider the switch from p_i to p_{i+1} in M working on w_l. Then one of the following cases holds:

- The switch occurs in the prefix $\triangleright u(xyz)^i$ of w_l, and the switch occurs at exactly the same position in w_r.

– The switch occurs in the suffix $(xyz)^i v\lhd$ of w_l, and the switch will occur at the same position of the suffix of w_r.

This means that the middle y of w_l is *always* skipped entirely, hence we conclude $u(xyz)^n y(xyz)^n v \in L(S) \iff u(xyz)^n v \in L(S)$, what we had to show. □

Proof. (*of Theorem 2.*) (a) $\Longrightarrow$ (b) is the contents of Lemma 4.

(b) $\Longrightarrow$ (c) follows from the trivial fact that every turtle language is accepted by some partially ordered two-way automaton, and the fact that the class of languages accepted by partially-ordered two-way automata is closed under Boolean operations. For complement this is trivial, since we are dealing with deterministic machines. An automaton M recognizing the union of the languages accepted by the partially-ordered two-way automata M_1 and M_2, given an input word w, first simulates M_1 on w. If M_1 accepts then M exits the input accepting. If not, M is transferred into a new state s; then it uses s to move its head back to the left end-marker and then, moving into the initial state of M_2, simulates M_2 on w. The order of states of M is the ordinal addition of the state set of M_1, the set $\{s\}$, and the state set of M_2.

Finally, (c) $\Longrightarrow$ (a) is the contents of Lemma 5. □

4 Conclusion

In the present paper we gave a characterization of the class **UL**, or, alternatively, the class of languages whose syntactic monoid is in **DA**, using finite two-way automata. A question that arises immediately is if our characterization leads to new decidability results. A first step would be to determine the complexity of deciding if a 2DFA is p.-o. It is relatively easy to see that this problem is in coNP; we conjecture that it is actually complete for that class.

Coming back to Fig. 1, our results suggest a new proof of one of the results of [PW97], namely that languages with a syntactic monoid in **DA** are exactly those definable in $\Sigma_2 \cap \Pi_2$: For this, one would have to show independently, what follows from our paper, namely that a language and its complement can be recognized by p.-o. NFAs if and only if it can be recognized by a p.-o. 2DFA.

Turtle languages, which were introduced in the present paper as a technical vehicle, also turn out to capture other nice classes of varieties: It can be shown that a language has an R-trivial syntactic monoid iff it is a Boolean combination of turtle languages of the form $(\rightarrow, a_1)(\rightarrow, a_2)\dots(\rightarrow, a_s)(\leftarrow, b_1)\dots(\leftarrow, b_t)$, i.e., they start from the left and are allowed only one turn in direction. Starting from right, one obtains L-trivial; allowing one turn, starting from either direction gives the join of the varieties. No turn turtle programs gives J-trivial. These results hint in the direction that turtle languages might turn out to be a helpful tool for further studies in algebraic language theory. It would be nice to add more evidence to this thesis.

References

[Arf91] M. Arfi. Opération polynomiales et hiérarchies de concaténation. *Theoretical Computer Science*, 91:71–84, 1991.

[Bir89] J.-C. Birget. Concatenation of inputs in a two-way automaton. *Theoretical Computer Science*, 63:141–156, 1989.

[Bir92] J.-C. Birget. Positional simulation of two-way automata: Proof of a conjecture of R. Kannan and generalizations. *Journal of Computer and System Sciences*, 45(2):154–179, 1992.

[EH99] J. Engelfriet and H. Hoogeboom. Two-way finite state transducers and monadic second-order logic. In *Proceedings 26th International Colloqium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 311–320, Berlin Heidelberg, 1999. Springer Verlag.

[Eil76] S. Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York, 1976.

[GH96] N. Globerman and D. Harel. Complexity results for two-way and multipebble automata and their logics. *Theoretical Computer Science*, 169:161–184, 1996.

[HS99] J. Hromkovic and G. Schnitger. On the power of Las Vegas II. Two-way finite automata. In *Proceedings 26th International Colloqium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 433–442, Berlin Heidelberg, 1999. Springer Verlag.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Series in Computer Science. Addison-Wesley, Reading, MA, 1979.

[Mic81] S. Micali. Two-way deterministic finite automata are exponentially more succinct than sweeping automata. *Information Processing Letters*, 12:103–105, 1981.

[MSTV00] P. McKenzie, T. Schwentick, D. Thérien, and H. Vollmer. The many faces of a translation. In *Proceedings 27th International Colloqium on Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 890–901, Berlin Heidelberg, 2000. Springer Verlag. By a mistake of the organizers while reformatting this paper, alas, all equality signs disappeared in the proceedings version. Refer to ftp://ftp-info4.informatik.uni-wuerzburg.de/pub/TRs/mc-sc-th-vo00.ps.gz for a full version *with* equality signs.

[Pin86] J. E. Pin. *Varieties of Formal Languages.* Plenum Press, New York, 1986.

[Pin97] J. E. Pin. Syntactic semigroups. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume I, pages 679–746. Springer Verlag, Berlin Heidelberg, 1997.

[PW97] J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30:383–422, 1997.

[Sch76] M. Schützenberger. Sur le produit de concatenation non ambigu. *Semigroup Forum*, 13:47–75, 1976.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, 1994.

[TW98] D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation: $\mathrm{FO}^2 = \Sigma_2 \cap \Pi_2$. In *Proceedings 30th Symposium on Theory of Computing*, pages 234–240, New York, 1998. ACM Press.