The Many Faces of a Translation

Pierre McKenzie[1]*, Thomas Schwentick[2],
Denis Thérien[3]*, and Heribert Vollmer[4]

[1] Informatique et recherche opérationnelle, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal (Québec), H3C 3J7 Canada
[2] Institut für Informatik, Johannes-Gutenberg-Universität Mainz, 55099 Mainz, Germany
[3] School of Computer Science, McGill University, 3480 University Street, Montréal (Québec), H3A 2A7 Canada
[4] Theoretische Informatik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

Abstract. First-order translations have recently been characterized as the maps computed by aperiodic single-valued nondeterministic finite transducers (NFTs). It is shown here that this characterization lifts to "**V**-translations" and "**V**-single-valued-NFTs", where **V** is an arbitrary monoid pseudovariety. More strikingly, 2-way **V**-machines are introduced, and the following three models are shown exactly equivalent to Eilenberg's classical notion of a bimachine when **V** is a group variety or when **V** is the variety of aperiodic monoids: **V**-translations, **V**-single-valued-NFTs and 2-way **V**-transducers.

1 Introduction

The regular languages have been characterized in many ways, in particular using DFAs, NFAs, and 2-way DFAs [RS59,She59], monadic second order logic [Büc60,Tra61] and finite semigroups [Eil76b], with an incisive algebraic parametrization arising from the latter. For instance, aperiodic DFAs recognize exactly the star-free languages [Sch65], and these are precisely captured by FO, i.e. first-order logic with order [MP71].

In a separate vein, renewed interest in FO was sparked by the development of descriptive complexity (see [Imm99]), and the old concept of a translation became the natural way to reduce one problem to another. FO-translations and projections, in particular, became the lowest level reductions used in the theory.

Experience with the regular languages would suggest characterizing FO-translations using aperiodic deterministic finite transducers. But this fails, because such transducers cannot even map $w_1 w_2 \cdots w_n$ to w_n^n. Lautemann and three of the present authors [LMSV99] showed that the appropriate automata-theoretic model required to capture FO-translations is the aperiodic *nondeterministic* transducer, restricted to output a unique string on any input.

* Supported by NSERC of Canada and by FCAR du Québec.

Here we extend this result in several directions. We first generalize from FO-translations to $\mathbf{V}$-translations, where $\mathbf{V}$ is an arbitrary monoid variety (a monoid variety is the most convincing notion imaginable of a natural set of monoids). FO-translations are obtained when $\mathbf{V}$ is the variety of aperiodic monoids, and other special cases of such $\mathbf{V}$-logical formulas were studied before, for example when $\exists$ and $\forall$ quantifiers were replaced by MOD_q quantifiers in the work of [STT88,BCST92,Str94].

Second, we define an NFA to be a $\mathbf{V}$-NFA if applying the subset construction to it yields (the familiar notion of) a $\mathbf{V}$-DFA.

Third, we consider 2-way DFAs, and define what constitutes a 2-way $\mathbf{V}$-machine. This is delicate because the appropriate definition of the extended transition function of such an automaton is not obvious, which in fact explains the absence of an algebraic treatment of 2-way DFAs in the literature.

Our main result is a striking equivalence between the above notions and the old notion of a bimachine developed by Eilenberg [Eil76a, p. 320]:

Theorem 1. *Let $f\colon \Sigma^* \to \Gamma^*$, let $\mathbf{V}$ be the variety $\mathbf{A}$ of aperiodic monoids or an arbitrary variety of groups. The following statements are equivalent:*

(i) f is a $\mathbf{V}$-translation.
(ii) There is a sequential $\mathbf{V}$-bimachine that computes f.
(iii) There is a single-valued nondeterministic $\mathbf{V}$-transducer that computes f.
(iv) There is a 2-way $\mathbf{V}$-machine that computes f.

Moreover, for the much larger class of varieties $\mathbf{V}$ closed under reversal, (i) $\Leftrightarrow$ (ii) $\Leftrightarrow$ (iii).

In the case $\mathbf{V} = \mathbf{A}$, Theorem 1 states that FO-translations, aperiodic bimachines, single-valued nondeterministic aperiodic transducers, and deterministic aperiodic 2-way transducers, all compute exactly the same functions.

Intuitively, the main ingredients of Theorem 1 are that NFAs can simulate bimachines by guessing the behavior of the second machine on the unread part v of an input string uv (and they can simulate FO by guessing the possible sets of inequivalent formulas consistent with u satisfied by uv). The link with 2-way DFAs, in the non-group case, comes from simulating NFAs in the slick way developed by Hopcroft and Ullman [HU67], yet a considerable amount of care is needed to ensure that the simulation preserves the required algebraic properties. (In this paper, we limit our claim to the statement of Theorem 1, although we suspect that the equivalence between the four models of the theorem holds for many more varieties.)

Section 2 contains preliminaries and the precise definitions of our computation models. Section 3, presenting our results, first addresses the models equivalent for any $\mathbf{V}$ closed under reversal, and then we bring in the 2-way model. Due to space restrictions, most proofs have to remain sketchy or are even omitted from this abstract. For full proofs we refer the reader to [MSTV00].

2 Definition of Models and Preliminary Results

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 [Eil76b, pp. 109ff]: it is a set of monoids closed under the finite direct product, the taking of submonoids and of homomorphic images. 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 $\mathbf{G}_{sol}$, or any set of monoids satisfying a set of identities, as we explain next.

Let $U = \{u_1, u_2, u_3, \dots\}$ be a countable alphabet of *variables*, and let $l, r \in U^*$. Then an equation of the form $l = r$ is a *monoid identity*. A monoid M satisfies the above identity if, for every homomorphism $h\colon U^* \to M$, we have $h(l) = h(r)$. Monoid identities are used to define pseudovarieties of monoids, we refer the reader to the excellent presentation in [Str94, Chapter V.6]. Precisely, a set $\mathbf{V}$ of monoids is a variety, if and only if there exists a sequence $(l_i = r_i)_i \geq 1$ of equations such that a monoid M belongs to $\mathbf{V}$ iff it satisfies all but finitely many equations $l_i = r_i$, see [Eil76b, Sect. V.2].

If M is a finite automaton (one-way or two-way) such that the transformation monoid of M satisfies the identity $l = r$, then we say that $l = r$ holds in M.

Deterministic finite automata are, as usual [HU79, p. 17], given by $M = (S, \Sigma, \delta, s, F)$, 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. 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.

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

2.1 FO-Translations and V-Translations

We consider first-order logic with linear order, denoted by $\mathbf{FO}$ (in the literature, this is often denoted $\mathbf{FO}[<]$, see, e.g., [Imm99]). We restrict our attention to *string signatures*, i. e., signatures of the form $\langle C_{a_1}, \dots, C_{a_s} \rangle$, where all the predicates C_{a_i} are unary, and in every structure $\mathcal{A}$, $\mathcal{A} \models C_{a_i}(j)$ iff the jth symbol in the input is the letter a_i. Such structures are thus words over the alphabet $\Sigma = \{a_1, \ldots, a_s\}$, and first-order variables range over positions within such a word, i.e. from 1 to the word length n. A formula from this logic is called a formula over Σ. Our basic formulas are built from variables in the usual way, using the Boolean connectives $\{\wedge, \vee, \neg\}$, the relevant predicates C_{a_i} together with $\{=, <\}$, the quantifiers $\{\exists, \forall\}$, and parentheses.

Let Σ be as above, and consider a second alphabet $\Gamma = \{b_1, \ldots, b_{t+1}\}$. Fix an order $b_1 < \cdots < b_{t+1}$ on Γ. Let $\varphi_1, \ldots, \varphi_t$ be first-order formulas over Σ, each with one free variable x. These formulas define a mapping $[\varphi_1, \ldots, \varphi_t] \colon \Sigma^* \to \Gamma^*$ as follows: Let $w = w_1 \cdots w_n \in \Sigma^n$. Then, $[\varphi_1, \ldots, \varphi_t](w) = v_1 \cdots v_n \in \Gamma^n$, where

$$v_i = \begin{cases} b_1 & \text{if } w \models \varphi_1(i), \\ b_2 & \text{if } w \models \neg\varphi_1(i) \wedge \varphi_2(i), \\ & \ldots \\ b_{t+1} & \text{if } w \models \neg\varphi_1(i) \wedge \neg\varphi_2(i) \wedge \cdots \wedge \neg\varphi_t(i), \end{cases}$$

for $1 \leq i \leq n$. In this definition, $\varphi_j(i)$ is the formula that is obtained when in φ_j variable x takes value i.

This function is called a **FO**-translation or first-order definable translation, see [LMSV99]. (In the more general case, where the formulas $\varphi_1, \ldots, \varphi_t$ are allowed to have more than one free variable, these functions are called first-order reductions or first-order interpretations, see [Imm99].)

Now let Σ and Γ be as above and $A \subseteq \Gamma^*$. Then we say that $Q_A x[\varphi_1, \ldots, \varphi_t]$ is a $Q_A^{\mathrm{un}}\mathbf{FO}$ formula (over Σ). For $w \in \Sigma^*$, we define $w \models Q_A x[\varphi_1, \ldots, \varphi_t]$ if $[\varphi_1, \ldots, \varphi_t](w) \in A$.

In this paper, the sets A defining quantifiers will most of the time be *monoid word problems*. For a monoid M and a subset $F \subseteq M$, we define the word problem $W(M, F)$ as the language $\{\, m_1 \cdots m_k \in M^* \mid k \in \mathbb{N}, m_1 \circ \cdots \circ m_k \in F \,\}$, where $\circ$ denotes multiplication in M.

Let $\mathbf{V}$ be a pseudovariety of monoids. We will now use $\mathbf{V}$ word problems to define translations as follows. First, define $\mathbf{V}$-formulas inductively by:

- Every quantifier-free formula is a $\mathbf{V}$-formula.
- If φ is quantifier-free, x a variable, $M \in \mathbf{V}$, and $F \subseteq M$, then $Q_{W(M,F)} x\, \varphi$ is a $\mathbf{V}$-formula.
- Every Boolean combination of $\mathbf{V}$-formulas is a $\mathbf{V}$-formula.

A $\mathbf{V}$*-translation* is a translation $[\varphi_1, \ldots, \varphi_t]$, where $\varphi_1, \ldots, \varphi_t$ are $\mathbf{V}$-formulas with one free variable.

Examples. The quantifier $\exists$ is the quantifier $Q_{W(OR,\{1\})}$, where OR is the aperiodic monoid defined by the binary OR. Straubing [Str94] surveys an elegant theory in which FO is supplemented with "monoidal" quantifiers such as MOD_q, where $\mathrm{MOD}_q x\, \varphi$ holds iff $\varphi(x)$ holds in a multiple of q word positions x. Such quantifiers are $Q_{W(Z_q,\{0\})}$ quantifiers. Hence $[Q_{W(Z_q,\{0\})} y (y < x \wedge C_a y)]$ is a translation, mapping $w_1 w_2 \cdots w_n \in \{a, b, c\}^*$ to $v_1 \cdots v_n \in \{0, 1\}^n$, such that $v_i = 1$ iff $w_1 \cdots w_{i-1}$ contains a multiple of q occurrences of a. Other monoidal quantifiers, e.g., $Q_{W(S_5,\{e\})}$ where S_5 is the (nonsolvable) symmetric group on 5 points (see, e.g., [BIS90]), have been used in the literature to capture the complexity class NC^1 using FO.

2.2 Bimachines and Bisequential Functions

Eilenberg [Eil76a, p. 320] defined a *bimachine* to be a triple $M = (M_1, M_2, g)$ where $M_1 = (S_1, \Sigma, \delta_1, s_1)$, $M_2 = (S_2, \Sigma, \delta_2, s_2)$ are DFAs without final states, and $g\colon S_1 \times S_2 \times \Sigma \to \Gamma$. For $i \in \{1, \ldots, |w|\}$, the output of M on input $w = w_1 \cdots w_n$ at position i is $o_M(w, i) = g(\hat{\delta}_1(s_1, w_1 \cdots w_{i-1}), \hat{\delta}_2(s_2, w_n \cdots w_{i+1}), w_i)$. The *output* of M on input w is $T_M(w) = o_M(w, 1) \cdots o_M(w, n)$. M is a $\mathbf{V}$-bimachine if the transformation monoids of both M_1 and M_2 are in $\mathbf{V}$.

We define a *bisequential function* as a pair $f = (\alpha, m)$, where α is a congruence over some alphabet Σ, and m is a function $m\colon \Sigma^*/\alpha \times \Sigma^*/\alpha \times \Sigma \to \Gamma$ for some alphabet Γ, and for all $w = w_1 \cdots w_n \in \Sigma^*$, $f(w) = v_1 \cdots v_n$ where $v_i = m\big([w_1 \cdots w_{i-1}]_\alpha, [w_{i+1} \cdots w_n]_\alpha, w_i\big)$. We say that f is a $\mathbf{V}$-bisequential function if the quotient set Σ^*/α with the usual multiplication is in $\mathbf{V}$.

Observe that if $\mathbf{V}$ is closed under reversal, a function is $\mathbf{V}$-bisequential if and only if it is computed by some $\mathbf{V}$-bimachine. Indeed, to compute the bisequential function $f = (\alpha, m)$, define $M = (M_1, M_2, m)$, where $M_1 = (S_1, \Sigma, \delta_1, s_1)$ and $M_2 = (S_2, \Sigma, \delta_2, s_2)$ are given by $S_1 = S_2 = \Sigma^*/\alpha$, $s_1 = s_2 = [\varepsilon]_\alpha$, $\delta_1\big([w]_\alpha, a\big) = [wa]_\alpha$ and $\delta_2\big([w]_\alpha, a\big) = [aw]_\alpha$ for all $a \in \Sigma$, $w \in \Sigma^*$. The reader may check that $f = T_M$ and that the transformation monoids of M_1 and M_2 are in $\mathbf{V}$.

The converse, given M to find a $\mathbf{V}$-bisequential function f such that $T_M = f$, is proved in a very similar way.

2.3 Single-valued Nondeterministic Transducers

In [LMSV99], a *nondeterministic finite transducer* is defined to be a tuple $M = (S, \Sigma, \Gamma, \delta, I, F)$, where S is the set Q of *states*, Σ is the *input alphabet*, Γ is the *output alphabet*, $\delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is the *transition relation*, $I \subseteq Q$ is the set of *initial states* and $F \subseteq Q$ is the set of *final states*. For a string $w = w_1 \cdots w_n \in \Sigma^*$ we define the *set* $T_M(w)$ *of outputs of* M *on input* w as follows. A string $v \in \Gamma^*$ of length n is in $T_M(w)$, if there is a sequence $s_0, s_1, \ldots, s_n$ of states, such that $s_0 \in I$, $s_n \in F$ and, for every i, $1 \leq i \leq n$, we have $(s_{i-1}, w_i, v_i, s_i) \in \delta$.

We say that a nondeterministic finite transducer $(S, \Sigma, \Gamma, \delta, I, F)$ is a $\mathbf{V}$-*transducer*, or that a NFA $(S, \Sigma, \delta, s_0, F)$ is a $\mathbf{V}$-*NFA*, if applying the subset construction to the pre-NFA (S, Σ, δ) yields a pre-DFA $(2^S, \Sigma, \delta')$ whose transformation monoid is in $\mathbf{V}$.

Proposition 2. *The following statements are equivalent for a transducer M having (S, Σ, δ) as pre-NFA:*
(i) M is a $\mathbf{V}$-transducer,
(ii) For each $p, q \in S$, the language $L(p, q)$ accepted by the NFA $(S, \Sigma, \delta, p, \{q\})$ is a $\mathbf{V}$-language.

We say that a nondeterministic transducer M is *single-valued*, if, for every $w = w_1 \cdots w_n \in \Sigma^*$ there is a single computation $(s_0, w_1, v_1, s_1)(s_1, w_2, v_2, s_2) \cdots (s_{n-1}, w_n, v_n, s_n)$ such that $s_0 \in I$, $s_n \in F$, and $(s_{i-1}, w_i, v_i, q_i) \in \delta$ for $1 \leq i \leq n$. Note that in this case, $|T_M(w)| = 1$ for all $w \in \Sigma^*$; if $T_M(w) = \{u\}$, then we write $T_M(w) = u$.

Remark 3. We remark that the definition above of single-valuedness is different from the one given in [LMSV99], since there it was only required that the set of values $T_M(w)$ consists of at most one word, for every $w \in \Sigma^*$. This still leaves the possibility that there may be different paths in M for a particular input w which produce the same output, which is forbidden in the present definition. [LMSV99] equate the power of aperiodic machines according to their definition with that of **FO**-translations. Since our main result in Sect. 3 gives an analogous statement for aperiodic machines according to our stricter definition, we conclude that for the aperiodic case, both definitions coincide.

2.4 Two-Way Automata

A two-way automaton with output is a 7-tuple $M = (\mathcal{L} \uplus \mathcal{R}, \Sigma, \Gamma, \delta, \lambda, l_0, F)$, 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* which, as in the case of one-way transducers, will play no role in defining the operation of a two-way automaton with output,
- Σ is the *input alphabet*, Γ is the *output alphabet*,
- $\delta\colon (S \times \Sigma) \cup (\mathcal{L} \times \{\lhd\}) \cup (\mathcal{R} \times \{\rhd\}) \to S$ is a total *transition function*, where $\rhd \notin \Sigma$ and $\lhd \notin \Sigma$ are the *leftmarker* and the *rightmarker* respectively,
- $\lambda\colon (S \times \Sigma) \to \Gamma$ is the *output function*.

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

The *initial configuration* of M on input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ 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 $w = w_1 w_2 \cdots w_n \in \Sigma^*$,

- M from its initial configuration on w eventually leaves every w_i to the right, $1 \leq i \leq n$, and eventually exits $\rhd w \lhd$; this is analogous to the (unspoken) requirement that a one-way transducer 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 transducer 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.

Each $w = w_1 w_2 \cdots w_n \in \Sigma^*$ coerces M into a behavior described by a *behavior function*

$$\delta_w : \mathcal{L} \uplus \mathcal{R} \to \mathcal{L} \uplus \mathcal{R}$$
$$s \mapsto \begin{cases} \text{state in which } M \text{ exits } w_1 w_2 \cdots w_n \text{ when started} & \text{if } s \in \mathcal{L}; \\ \text{in state } s \text{ scanning } w_1, & \\ \text{state in which } M \text{ exits } w_1 w_2 \cdots w_n \text{ when started} & \text{if } s \in \mathcal{R}. \\ \text{in state } s \text{ scanning } w_n, & \end{cases}$$

The base case is given by $\delta_w(s) = s$ if $|w| = 0$, and $\delta_w(s) = \delta(s, w)$ if $|w| = 1$. The induction step divides into two similar cases according to whether $s \in \mathcal{L}$ or $s \in \mathcal{R}$. Let $u \in \Sigma^+$ and $a \in \Sigma$. We only illustrate the case $s \in \mathcal{L}$, namely M entering ua from the left. The case breaks off into two subcases according to whether M eventually falls off ua to the left ($\delta_u(s_k) \in \mathcal{R}$) or to the right ($\delta_a(l_k) \in \mathcal{L}$):

$$\delta_{ua}(s) = \begin{cases} \delta_u(s_k) & \text{if } (\exists k \geq 0)(\exists s_0 = s)(\exists l_1, l_2, \ldots, l_k \in \mathcal{L})(\exists s_1, s_2, \ldots, s_k, \\ & \delta_u(s_k) \in \mathcal{R}) \text{ such that } l_i = \delta_u(s_{i-1}) \text{ and } s_i = \delta_a(l_i) \text{ for} \\ & 1 \leq i \leq k; \\ \delta_a(l_k) & \text{if } (\exists k \geq 0)(\exists l_0 = \delta_u(s), l_1, l_2, \ldots, l_k, \delta_a(l_k) \in \mathcal{L})(\exists r_1, r_2, \ldots, \\ & r_k \in \mathcal{R}) \text{ such that } r_i = \delta_a(l_{i-1}) \text{ and } l_i = \delta_u(r_i) \text{ for } 1 \leq i \leq k. \end{cases}$$

Finally we define the *behavior monoid* $\mathcal{B}(M)$ of M to be the monoid $\{\delta_w \mid w \in \Sigma^*\}$ under composition. M is a $\mathbf{V}$-*machine* iff $\mathcal{B}(M) \in \mathbf{V}$.

The *output of* M on input $w = w_1 w_2 \cdots w_n$ is $\lambda(s_1, w_1)\lambda(s_2, w_2) \cdots \lambda(s_n, w_n)$, where s_i is the state entered by M as it lands on w_i for the last time when started from its initial configuration on w (when $|w| = 0$ the output is the empty string).

3 Results

Here we present our different results which together prove Theorem 1. First we state that for all varieties closed under reversal, translations, bisequential functions, bimachines, and single-valued nondeterministic transducers yield the same class of functions. This proves the equivalence of the first three statements of Theorem 1, since group varieties and $\mathbf{A}$ are closed under reversal.

Theorem 4. *The following are equivalent when* $\mathbf{V}$ *is a monoid variety closed under reversal:*

(i) *f is a $\mathbf{V}$-translation.*
(ii) *f is a $\mathbf{V}$-bisequential function.*
(iii) *There is a sequential $\mathbf{V}$-bimachine that computes f.*
(iv) *There is a single-valued nondeterministic $\mathbf{V}$-transducer that computes f.*

The proof, which can be found in the full version of this paper [MSTV00], consists of a sequence of simulations among the different models that preserve monoid identities in the sense of Sect. 2. It even turns out that equivalence of (i) and (ii) holds for all varieties of monoids. Here, we only present the following implication:

Lemma 5. *Let $\varphi_1, \ldots, \varphi_t$ be* $\mathbf{V}$*-formulas. Then there is a* $\mathbf{V}$*-bisequential function f such that $[\varphi_1, \ldots, \varphi_t] = f$.*

Proof. We first assume that $t = 1$, i.e. that the image of the translation is over a two-letter alphabet, which we take to be $\{Y, N\}$. Initially, let $\varphi = \varphi_1$ be a formula over some alphabet Σ of the form $\varphi(x) = Q_A y \psi(x, y)$, where A is a $\mathbf{V}$-language and ψ is quantifier-free. Let α be the congruence relation that defines $A \subseteq \Gamma^*$, i.e., $\Gamma^*/\alpha \in \mathbf{V}$. For simplicity, we assume that $\Gamma = \{1, 0\}$; the case of non-binary alphabets is treated similarly.

For every letter $a \in \Sigma$, we define from ψ three formulas $\psi_a^<$, $\psi_a^=$ and $\psi_a^>$ as follows:

	replace $C_a(x)$ by	$C_b(x), b \neq a$ by	$y < x$ by	$y = x$ by	$x < y$ by
To get $\psi_a^<$	true,	false,	true,	false,	false.
To get $\psi_a^=$	true,	false,	false,	true,	false.
To get $\psi_a^>$	true,	false,	false,	false,	true.

Observe that all these formulas have the only free variable y, since x no longer appears. Moreover, for each $a \in \Sigma$, we write ψ_a for the *constant* formula obtained by further evaluating $\psi_a^=$ with the knowledge that $C_a(y)$. Now recall that the ith symbol in $[\varphi](w_1 \ldots w_n)$ is equal to Y iff

$$\psi(i, 1)\ \psi(i, 2)\ \cdots\ \psi(i, n) \in A, \tag{1}$$

where for convenience we wrote $\psi(i, j)$ for the zero-one truth value of $\psi(i, j)$ evaluated on $w_1 \ldots w_n$. But the $\{0, 1\}$-word appearing in (1) is precisely the $\{0, 1\}$-word

$$\psi_{w_i}^<(1)\ \ \psi_{w_i}^<(2)\ \ \cdots\ \ \psi_{w_i}^<(i-1)\ \ \psi_{w_i}\ \ \psi_{w_i}^>(i+1)\ \ \cdots\ \ \psi_{w_i}^>(n-1)\ \ \psi_{w_i}^>(n). \tag{2}$$

Fortunately, every ψ_a^R (for $a \in \Sigma$, $R \in \{<, >\}$) defines a length-preserving homomorphism, also denoted $\psi_a^R \colon \Sigma^* \to \{0, 1\}^*$. So let a congruence $\sim$ on Σ^* be defined as follows: $u \sim v$ iff $\psi_a^R(u)\ \alpha\ \psi_a^R(v)$ holds for each $a \in \Sigma$ and for each $R \in \{<, >\}$. This is a $\mathbf{V}$-congruence, being the intersection of finitely many $\mathbf{V}$-congruences (indeed each separate homomorphism ψ_a^R defines a congruence whose induced monoid is the image of a submonoid of $\{0, 1\}^*/\alpha$). The $\mathbf{V}$-bisequential function $f = (\sim, m)$ therefore computes $[\varphi]$, where

$$\begin{aligned} m \colon (\Sigma^*/{\sim}) \times (\Sigma^*/{\sim}) \times \Sigma &\to \{Y, N\} \\ ([u]_\sim, [v]_\sim, a) &\mapsto \begin{cases} Y & \text{if } [\psi_a^<(u)]_\alpha \circ [\psi_a]_\alpha \circ [\psi_a^>(v)]_\alpha \subseteq A, \\ N & \text{otherwise.} \end{cases} \end{aligned}$$

Here $\circ$ computes in $\{0,1\}^*/\alpha$, and m is well defined because $w_1 \sim w_2$ implies that $\psi_a^R(w_1)\ \alpha\ \psi_a^R(w_2)$.

Now if φ is a Boolean combination of formulas, we consider the homomorphisms defined by all subformulas of φ of the form $Q_A y \psi(x,y)$ for quantifier-free ψ. In the case of several formulas $[\varphi_1,\ldots,\varphi_t]$ instead of one formula $[\varphi]$, we again have to deal with more homomorphisms as above. In both cases, the definition of m becomes more complicated, but the congruence $\sim$ remains an intersection of constantly many $\mathbf{V}$-congruences. Hence we still obtain a $\mathbf{V}$-bisequential f. $\square$

When are 2-way machines equivalent to the models of Theorem 4? In the next lemma, we show that a 2-way $\mathbf{V}$-automaton can always be simulated by a $\mathbf{V}$-bimachine. The converse, however, requires additional assumptions about $\mathbf{V}$. Lemma 7 deals with the variety of all aperiodic monoids and the variety of all groups. The construction that is used in the proof of Lemma 7 results in particularly simple 2-way automata if $\mathbf{V}$ is an arbitrary group varity. This is stated as Corollary 8.

Lemma 6. *Let $M = (\mathcal{L} \uplus \mathcal{R}, \Sigma, \Gamma, \delta, \lambda, l_0, F)$ be a two-way $\mathbf{V}$-automaton with output. Then there exists a $\mathbf{V}$-bimachine (M_1, M_2, g) that computes T_M.*

Proof. Given M, we first show how to construct automata $M_1 = (S_1, s_1, \Sigma, \delta_1)$, $M_2 = (S_2, s_2, \Sigma, \delta_2)$ and function g. Let $\circ$ denote multiplication in $\mathcal{B}(M)$ and let e be the corresponding identity.

We define M_1 as by $S_1 = \mathcal{B}(M)$, $s_1 = e$, and for each $f \in \mathcal{B}(M), a \in \Sigma$, we set $\delta_1(f,a) = f \circ \delta_a$. Intuitively, after reading $w_1 \cdots w_{i-1}$ the automaton M_1 has computed the function $\delta_{w_1\cdots w_{i-1}}$.

The definition of M_2 is analogous: $S_2 = \mathcal{B}(M)$, $s_2 = e$, and for each $f \in \mathcal{B}(M), a \in \Sigma$, we set $\delta_2(f,a) = \delta_a \circ f$.

It is easy to show that, for each string $w = w_1 \cdots w_n$ and each position $i \in \{1,\ldots,n\}$, we obtain $\hat{\delta}_1(s_1, w_1 \cdots w_{i-1}) = \delta_{w_1\cdots w_{i-1}}$ and $\hat{\delta}_2(s_2, w_n \cdots w_{i+1}) = \delta_{w_{i+1}\cdots w_n}$. By combining this information with w_i and the fixed behavior of M on the endmarkers, the sequence of states that M takes at position i can be determined. From the very definition of the output of a two-way automaton it follows that a function g as claimed exists and can be obtained directly from the definition of M.

Next we show that all monoid identities that hold in M hold in M_1 and M_2 as well. Let $u, v \in \Sigma^*$ be such that $\delta_u = \delta_v$. We even prove the stronger claims that $\hat{\delta}_1(\cdot, u) = \hat{\delta}_1(\cdot, v)$ and $\hat{\delta}_2(\cdot, u^R) = \hat{\delta}_2(\cdot, v^R)$.

From the definition of M_1 we obtain for all $f \in \mathcal{B}(M), w \in \Sigma$, that $\hat{\delta}_1(f,w) = f \circ \delta_w$, which immediately yields the first claim. From the definition of M_2 we conclude that for each $f \in \mathcal{B}(M)$ and $w \in \Sigma^*$, we obtain $\hat{\delta}_2(f,w) = \delta_{w^R} \circ f$, where w^R denotes the reversal of the word w. This immediately implies the second claim. Hence we showed that M_1 and M_2 are $\mathbf{V}$-DFAs. $\square$

Next we turn a converse of this lemma. An important tool here is a result of Hopcroft and Ullman [HU67], showing how a two-way automaton can, for each symbol in its input, determine the states of one left-to-right and one right-to-left automaton, when they reach the corresponding input symbol. The next lemma shows that this result carries over to the case of restricted types of automata.

Lemma 7. *Let (M_1, M_2, g) be a sequential bimachine. Then there exists a 2-way automaton $M = (\mathcal{L} \uplus \mathcal{R}, \Sigma, \Gamma, \delta, \lambda, l_0, \emptyset)$ which computes the same function as (M_1, M_2, g) and has the following property: If $\hat{\delta}_1(\cdot, u) = \hat{\delta}_1(\cdot, v)$ and $\hat{\delta}_2(\cdot, u^R) = \hat{\delta}_2(\cdot, v^R)$ and $uv = vu$ then*

$$\delta_{uu} = \delta_{uv} = \delta_{vu} = \delta_{vv}.$$

Proof. We sketch the idea of the proof, and we refer the reader to [MSTV00] for the full details. Let $M' = (M_1, M_2, g)$, $M_1 = (Q_1, s_1, \Sigma, \delta_1)$, and $M_2 = (Q_2, s_2, \Sigma, \delta_2)$.

Informally, the behavior of the automaton M on input $w = w_1 \cdots w_n$ is split into two phases. Roughly, the first phase consists of a complete left-to-right scan in which M simulates M_1 and the second phase consists of n subcomputations each of which is a right-to-left movement followed by a (possibly empty) left-to-right movement. In the second phase M simulates the behavior of M_2 (which is easy) and keeps track of the states of M_1 (which is more complicated).

In more detail, each of the n subcomputations starts from a situation in which the automaton is at a position i of its input and knows $p_i = \hat{\delta}_1(s_1, w_1 \cdots w_i)$ and $q_i = \hat{\delta}_2(s_2, w_n \cdots w_{i+1})$ and ends in the corresponding situation at position $i-1$.

The obvious problem is how to compute $p_{i-1} = \hat{\delta}_1(s_1, w_1 \cdots w_{i-1})$. If there is only one state p of Q_1 with $\delta_1(p, w_i) = p_i$ then, of course $p_{i-1} = p$. Otherwise M proceeds as follows. It moves to the left and maintains, for each state p with $\delta_1(p, w_i) = p_i$, a set P_p of states. At a position $j < i$, P_p contains all states from which M_1 can reach p by reading $w_{j+1} \cdots w_{i-1}$. It is easy to see that, for each j, these sets are pairwise disjoint. This process ends in one of the following two situations. Either all but one of the sets P_p become empty or M reaches the left delimiter. In either case M can easily conclude p_{i-1} (in the latter case it is the state p for which P_p contains s_1). Now it only needs to find its way back to position i. In order to do so, it goes one step to the right and chooses one state from P_p and one state from a nonempty $P_{p'}$ at that position (these two sets were remembered by M). Then it simulates the behavior of M_1 starting from these two states until the two computations flow together into a single state. At this time M has reached position i again and now it also knows p_{i-1}, as this is given by the outcome of the simulation of M_1 which started from the correct state. It has remembered q_i during the whole subcomputation, hence it can retrieve the output symbol of M at position i. Then it starts the next subcomputation from position $i-1$. It is this last step of a subcomputation where M actually simulates the behavior of M_2.

The formal definition of M and the (technically involved) proof that it has the properties claimed in the lemma can be found in [MSTV00]. □

If $\mathbf{V}$ is a variety of groups then the construction in the proof of Lemma 7 leads to a very simple 2-way automaton. The simple reason is that, for such

$\mathbf{V}$, $\mathbf{V}$-automata have injective transition functions. Hence, for each state p of the automaton M_1 and each symbol σ there is exactly one state p' such that $\delta_1(p', \sigma) = p$. Therefore, M can always deduce p_{i-1} from p_i by one step to the left. It follows directly that all word equations that hold in M_1 and (reversed) in M_2 also hold in M and we can conclude the following corollary which considerably strengthens Lemma 7 for group varieties.

Corollary 8. *Let (M_1, M_2, g) be a sequential $\mathbf{V}$-bimachine computing a function f, where $\mathbf{V}$ an arbitrary group variety. Then f is computable by a 2-way $\mathbf{V}$-automaton.*

Theorem 9. *Let $\mathbf{V}$ be the variety $\mathbf{A}$ or an arbitrary variety of groups. A function is computed by a sequential $\mathbf{V}$-bimachine iff it is computed by a 2-way $\mathbf{V}$-machine.*

Proof. Lemma 6 shows that a sequential $\mathbf{V}$-bimachine can simulate a 2-way $\mathbf{V}$-machine. For the converse, there are two cases. The first case is when $\mathbf{V}$ is the variety $\mathbf{A}$ of all aperiodic monoids. Let a $\mathbf{V}$-bimachine (M_1, M_2, g) be simulated by the 2-way automaton M promised by Lemma 7. Since M_1 and M_2 are aperiodic, there exists $n \geq 0$ such that for all words w, $\hat{\delta}_i(\cdot, w^n) = \hat{\delta}_i(\cdot, w^{n+1})$, $i = 1, 2$. But then $\mathcal{B}(M) \in \mathbf{A}$ because Lemma 7 applies with $u = w^n$ and $v = w^{n+1}$:

$$\forall w, (\delta_w)^{2n} = \delta_{w^{2n}} = \delta_{uu} = \delta_{uv} = \delta_{w^{2n+1}} = (\delta_w)^{2n+1},$$

where $(\delta_x)^i$ is the i-fold composition of δ_x with itself.

The second case is when $\mathbf{V}$ is an arbitrary group variety. In this case the result follows from Corollary 8. □

4 Conclusion

We have characterized FO-translations in several natural ways, and we have extended the characterization to handle group varieties. The strong equivalences obtained, with the exception of the 2-way automaton characterization, extend to any monoid variety $\mathbf{V}$ closed under reversal.

We believe that Theorem 1 generalizes to many more varieties. The hurdle lies in the fine details required to ensure that a 2-way automaton simulating any one of the other models preserves enough of the algebraic structure. We feel here that we have not exhausted all the tricks. For example, crossing sequence arguments may be applicable to improve the construction and its analysis.

Our careful definition of what constitutes a 2-way $\mathbf{V}$-automaton opens the way for an algebraic treatment of 2-way automata. The translations studied here are akin to the maps underlying the wreath product and the block product constructions. This suggests approaches to handle the nesting of monoidal quantifiers in the logical framework, or equivalently to handle the series connections of 2-way automata. This emerges as a promising avenue for future research.

Acknowledgment. For helpful comments, we are grateful to Heinz Schmitz.

References

[BCST92] D. A. Mix Barrington, K. Compton, H. Straubing, and D. Thérien. Regular languages in NC^1. *Journal of Computer and System Sciences*, 44:478–499, 1992.

[BIS90] D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[Büc60] J. R. Büchi. Weak second-order arithmetic and finite automata. *Math. Logik. Grund. Math.*, 6:66–92, 1960.

[Eil76a] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, New York, 1976.

[Eil76b] S. Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York, 1976.

[HU67] J. E. Hopcroft and J. E. Ullman. An approach to a unified theory of automata. In *Proceedings 8th Symposium on Switching and Automata Theory*, pages 140–147. IEEE Computer Society Press, 1967.

[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.

[Imm99] N. Immerman. *Descriptive Complexity.* Graduate Texts in Computer Science. Springer Verlag, New York, 1999.

[LMSV99] C. Lautemann, P. McKenzie, T. Schwentick, and H. Vollmer. The descriptive complexity approach to LOGCFL. In *Proceedings 16th Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 444–454. Springer Verlag, 1999.

[MP71] R. McNaughton and S. Papert. *Counter-Free Automata.* MIT Press, 1971.

[MSTV00] P. McKenzie, T. Schwentick, D. Thérien, and H. Vollmer. The many faces of a translation. Technical Report 248, Institut für Informatik, Universität Würzburg, 2000.
URL: http://www.informatik.uni-wuerzburg.de/reports/tr.html.

[RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res.*, 3(2):115–125, 1959.

[Sch65] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information & Control*, 8:190–194, 1965.

[She59] J. C. Sheperdson. The reduction of two-way automata to one-way automata. *IBM J. Res.*, 3(2):198–200, 1959.

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

[STT88] H. Straubing, D. Thérien, and W. Thomas. Regular languages defined with generalized quantifiers. In *Proceedings 15th ICALP*, volume 317, pages 561–575, Berlin Heidelberg, 1988. Springer Verlag.

[Tra61] B. A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.