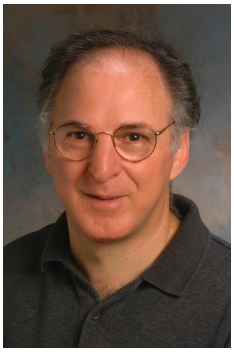
First-order logic and aperiodic languages: A revisionist history.

Howard Straubing, Boston College



A fundamental result about formal languages states:

Theorem 1 *A regular language is first-order definable if and only if its syntactic monoid contains no nontrivial groups.*

Rest assured, we will explain in the next section exactly what the various terms in the statement mean!

This beautiful theorem, the first to establish a tight link between logic and abstract algebra when applied to formal languages, has been enormously influential, spawning numerous extensions and generalizations, and serving as the source of several baffling, still-open problems. I was so taken with it that I was moved to write an entire book [Straubing 1994] devoted to its myriad consequences. As this article shows, I can't quite seem to *stop* writing about it.

There is a kind of standard, received account of the origin of Theorem 1: M. P. Schützenberger proved, in a 1965 paper [Schützenberger 1965a], that the regular languages whose syntactic monoids are aperiodic (*i.e.,* contain no nontrivial groups) are exactly the *star-free* languages–those that can be built using a restricted set of operations on sets of words. It was McNaughton and Papert, in their 1971 monograph *Counter-free Automata* [McNaughton and Papert 1971], who added logic into the mix, establishing the equivalence between star-free languages and those definable in first-order logic. I have repeated this account myself many times, and so may bear some of the responsibility for its spread.

Two years ago, I had the honor of being invited to speak at a conference in Schützenberger's honor, held on the twentieth anniversary of his death. While I struggled to come up with a subject for my talk, I turned for inspiration to Schützeberger's collected works [Berstel et al. 2009]. This wonderful resource, now available online, includes not only the definitive articles published in academic journals, but also the nearly impossible-to-find seminar proceedings and research reports (often crudely typed, with the mathematical symbols inked in by hand, bringing smiles of recognition to those of us who are old enough to remember what writing papers was like before the advent of LaTeX). Reading them, you get to see the sometimes messy birth of ideas, replete with the false starts, awkward formulations, and tentative sketches that never quite make it in to the more polished official versions.

I began to suspect that the received version of the history of Theorem 1, although not completely wrong, is not entirely right, either. Schützenberger was not one to hide his disdain for subjects in which he saw little value, and several of his former students

have told me that he was not very fond of formal logic. Nonetheless in [Schützenberger 1965a], Schützenberger makes explicit reference to the equivalence of star-free and first-order languages, and was aware of it as long ago as 1963, when he presented, in collaboration with L. Petrone [Petrone and Schützenberger 1965], something like a proof of part of this equivalence. In fact, whatever Schützenberger's views on logic might have been, it appears likely that it was a problem in logic that furnished the initial inspiration for the result in [Schützenberger 1965a], of which Eilenberg wrote, '..next to Kleene's Theorem, [it] is probably the most important result concerning recognizable sets.' [Eilenberg 1976] So while the first complete proof of Theorem 1 did indeed appear in McNaughton-Papert [McNaughton and Papert 1971], Schützenberger was very much aware of the result, and stated it explicitly in a number of articles, years before the publication of McNaughton and Papert's monograph. Most of the technical challenges involved in proving this theorem are already present in the proof of the 'star-free' formulation, which Schützenberger evidently thought of as a more congenial way to state it. [1].

The source for the problem that motivated Schützenberger's work is 'Symbolic logic and automata' , by Robert McNaughton [McNaughton 1960], yet another typewritten research report that never appeared in print, until most of it was recycled into the final chapters of *Counter-free Automata* eleven years later. In one sense this paper represents a kind of failure, since the main question McNaughton tried to tackle had already been answered in a more satisfying manner. But this error proved fortuitous: McNaughton's paper was the first step on the path that led to Theorem 1, and in the open problems and conjectures that conclude the paper, it proved extraordinarily prescient.

Mathematicians, as a rule, do not get attributions right. There's a good reason for this: the earliest manifestations of an idea are seldom tidy, elegant, or easy to follow. Often they appear in out-of-the-way places, or in languages that not everyone can read. If a theorem is influential enough to still be discussed and applied by more than a few people years after its discovery, the community of researchers converges on a cleaned-up version of its statement, along with an official origin myth, buoyed up by repeated citations of books and articles that few people actually read. [2] The present paper, then, is intended as a small corrective to this trend, a kind of revisionist history of one of my favorite theorems.

1. THEOREM 1, AS IT IS USUALLY PRESENTED

We will shortly describe the successive contributions of McNaughton, Schützenberger, and McNaughton-Papert in something close to their original form. But before that, let's give one of those 'modern' cleaned-up versions of the statement. The earliest instance of this that I can find is in Ladner, 'Application of model-theoretic games to discrete linear orders and finite automata' (1977), which attributes Theorem 1 to McNaughton-Papert (and does not even cite Schützenberger in the references!)

In the logical language presented by Ladner, and used in all subsequent accounts, sentences of first-order logic describe properties of words over a finite alphabet A. (Throughout this paper we will stick to finite words, but the same formalism can be adapted to infinite words.) Formulas are built from variable symbols $x, y, z, \ldots$ and the like. There are two kinds of atomic formulas: $x < y$, where x, y are variables, and $a(x)$,

[1] Jean-Eric Pin informs me that, in conversation, McNaughton himself attributed Theorem to Schützenberger.

[2] *Stigler's law of eponymy.* Stephen Stigler, a statistician, went so far as to propose that *no* scientific discovery is named after its original discoverer [Stigler 1980]. To drive the point home, he named this finding after himself, and then attributed it to Robert K. Merton.

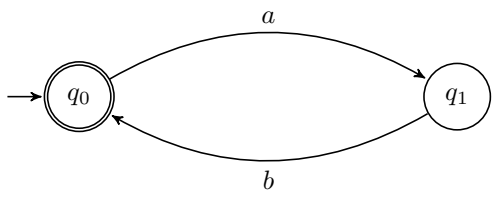


Fig. 1. An incompletely specified DFA recognizing $(ab)^*$.

where x is a variable and $a \in A$. First-order formulas are built from this base of atomic formulas in the usual manner, applying boolean operations $\wedge, \vee, \neg$ and both existential ($\exists$) and universal ($\forall$) quantification . We interpret *sentences* (first-order formulas without free variables) in words $w \in A^*$ by thinking of variables as representing positions in a word: thus $x < y$ means that position x occurs strictly to the left of position y, and $a(x)$ means that the letter in position x is a. [3] Thus every sentence ϕ defines a language $L_\phi \subseteq A^*$, consisting of all words in which ϕ is true.

We won't provide any definition of the syntax and semantics of these formulas more precise than the intuitive sketch given above; a few examples should make it clear. Let $A = \{a, b\}$. Here is a sentence that says, 'the first letter of the word is a':

$$\forall x(\forall y(x \leq y) \rightarrow a(x)).^4$$

And here is a sentence that says, 'there is no occurrence of two consecutive a's':

$$\forall x \forall y((x < y \wedge a(x) \wedge a(y)) \rightarrow \exists z(x < z \wedge z < y)),$$

(In English: for any pair of distinct positions containing a, there is some position strictly between them.)

We call this logical language FOL. It is not too much of an abuse of notation to use the same symbol to denote the class of languages definable in this logic.

We now turn to finite automata and the languages they recognize. Our automaton model is the *incompletely specified* DFA, where the next state function

$$(q, a) \mapsto \delta(q, a) = q \cdot a,$$

is in fact a partial function from $Q \times A$ into Q, where Q is the set of states. An example is shown in Figure 1: This automaton recognizes the language $(ab)^* \subseteq \{a, b\}^*$.

Figure 2 is a DFA recognizing $(aa)^* \subseteq \{a, b\}^*$. Note that since we allow incomplete specifications, the intended input alphabet must be named separately, since it is not implicit in the state-transition diagram.

The first example consists of the empty string, together with words that begin with a, end with b, have no two a's consecutive, and no two b's consecutive. We saw above how to write first-order sentences specifying each of these conditions, so the language $(ab)^*$ is in FOL, defined by the conjunction of these sentences.

---

[3] The papers discussed here use a wide assortment of notations to represent the same things. In an effort to make the presentation more coherent, I have often deviated from the precise notations of the original sources, including Ladner's, and adopted something closer to a common standard. These differences are entirely superficial.

[4] Well, not quite. What this sentence really says is that if the word *has* a first letter, then that letter is a, so the sentence is satisfied by the empty word as well. The common practice in predicate logic is to not interpret formulas in empty structures, but in this restricted context doing so works just fine, and allows us to write sentences that define languages containing the empty word. If the initial quantifier in the sentence referenced above is changed from $\forall$ to $\exists$, and the connective $\rightarrow$ is replaced by $\wedge$, then the language defined is exactly $a(a + b)^*$.

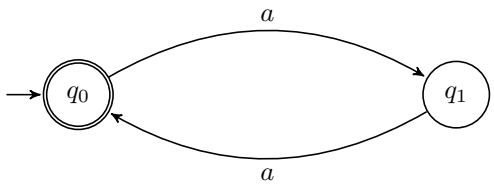


Fig. 2. An incompletely specified DFA recognizing $(aa)^*$.

What about the language $(aa)^*$? Can we define it with a logical formula? One way to do this is to broaden our logical language, and allow quantifiers over *sets* of positions as well as individual positions. We will denote the set variables by upper-case letters. Our language is thus defined by the sentence

$$\forall x a(x) \wedge \exists X(\theta_1(X) \wedge \theta_2(X)),$$

where $\theta_1(X)$ is

$$\forall x \forall y(y = x + 1 \rightarrow (x \in X \leftrightarrow y \notin X)),$$

and $\theta_2(X)$ is

$$\forall x(((\forall y(x \leq y) \rightarrow x \notin X) \wedge (\forall y(x \geq y) \rightarrow x \in X)).$$

The sentence says that every letter of the word is a. Moreover, there is a set X of positions, with the following three properties: a position belongs to X if and only if the following position does not belong to X, the first position does not belong to X, and the last position belongs to X. This is a *monadic second-order sentence*: 'second-order' because it quantifies over predicate variables as well as individual variables, 'monadic' because it only quantifies over one-place predicates; *i.e.,* sets.

The main result concerning these is simply

Theorem 2 *A language is regular if and only if it is defined by a monadic second-order sentence.*

We will have a bit to say about the history of *this* theorem as well.

Can the language $(aa)^*$ be defined in FOL, that is, without resorting to second-order quantification? Theorem 1 addresses precisely this question. Each regular language, in fact each language, if you allow infinite automata, has a unique minimal DFA. This is defined by the properties that every state can be reached from the initial state, and that if q, q' are distinct states, then there is some $w \in A^*$ leading q to an accepting state, and q' to a non-accepting state, or vice-versa. Evidently, the two automata in Figures 1 and 2 are the minimal DFAs for the languages they recognize.

In a DFA, a word w provides a partial function f_w on the set of states, where $q \cdot w = qf_w$. Obviously we have

$$f_{vw} = f_v f_w,$$

where the right-hand side denotes the left to right composition of these two functions. Thus the set $\{f_w : w \in A^*\}$ forms a monoid with composition as the operation, and the map $w \mapsto f_w$ is a homomorphism from A^* into this monoid. If the automaton is the minimal automaton of the language it recognizes, then this monoid is the *syntactic monoid* M_L, and the homomorphism from A^* is called the *syntactic morphism* of L. (More generally, a monoid M is said to recognize a language L if there is a DFA recognizing L whose transition monoid is M.)

What are the syntactic monoids of the two languages exhibited above? For $(aa)^*$ there are three partial functions: the identity map on the set of states induced by all words in A^* of even length; the transposition of the two states induced by words in A^* of odd length; and the empty partial function, induced by any word that contains a b. So here M_L is a group of order 2 with a zero tacked on.

For $(ab)^*$, there is the identity map induced by the empty word, and four partial functions induced by words in the sets $(ab)^*$, $b(ab)^*$, $(ab)^*a$ and $b(ab)^*a$. Each of these four partial functions has a domain of size 1: for example, $b(ab)^*$ maps q_1 to q_0 and is undefined at q_0. Finally, there is a sixth element, the empty partial function induced by any word that contains two consecutive a's or two consecutive b's. This monoid does not contain any group of size greater than 1. For such a group to occur, we would need an idempotent $e^2 = e$ and an element $m \neq e$ such that $me = em = m$, and $m^k = e$ for some $k > 1$, and it is easy to verify from the description of the monoid that such elements do not exist.

So Theorem 1 settles the question for both of these languages: Since the syntactic monoid of $(ab)^*$ contains no nontrivial groups, it is definable by a sentence of FOL, something we already knew by direct construction of such a sentence. Since the syntactic monoid of $(aa)^*$ contains a group of order 2, we know that there is no sentence of FOL defining it. Since we can construct the multiplication table for the syntactic monoid from the state-transition diagram, Theorem 1 provides us with a decision procedure for determining if a given regular language is definable in FOL.[5]

We now turn to Theorem 1's history. (For a reasonably self-contained proof, see Ladner's paper or Straubing and Weil [Straubing and Weil 2011]).

2. 'SYMBOLIC LOGIC AND AUTOMATA' (MCNAUGHTON, 1960)

This 30-page paper appeared as a technical report produced as part of a project on 'General Switching Theory' sponsored by the U.S. Air Force. (The abstract page includes a sign-off by an Air Force officer, approving it for publication 'to achieve an exchange and stimulation of ideas'.) It was not published elsewhere.

It is worth recalling why early theoretical computer scientists were interested in the connection between predicate logic and finite automata. It had been known for some time that any static input-output behavior—*i.e.,* n output bits whose value depended only on the values of m input bits—could be realized by a formula of propositional logic. It was Shannon, in his 1937 Master's thesis, who showed how to turn this into the design of electrical devices, what we now call *combinational circuits* built from logic gates [Shannon 1938].

Combinational circuits are memoryless devices, and propositional logic by itself is not sufficient to represent time-dependent behavior, where each output bit at time t depends on the complete history of input bits from time 0 to t. The devices that McNaughton called finite automata, but that we are more likely today to call 'finite-state sequential machines', exhibit this kind of time-dependent behavior. The machine computes a length-preserving function

$$F : A^* \to B^*,$$

which has the following sequentiality property: For each i, the i^{th} letter of the output word depends only on the first i letters of the input word. Abstractly, A and B are finite alphabets, although if we think of each input and output as consisting of m and n bits, respectively, we have $A = \{0,1\}^m, B = \{0,1\}^n$. An automaton that computes such a function has, in addition to the usual state-transition function $\delta : Q \times A \to Q$,

[5]Moreover, every proof of Theorem 1 effectively constructs a defining first-order sentence for the language, when one exists.

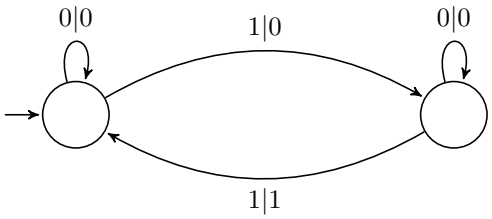


Fig. 3. Mod-2 counter finite-state machine

an output function $\lambda : Q \times A \to B$, giving the output at time t as a function of both the state at time $t-1$ and the input at time t.

An example, in which the input and output alphabets are both equal to $\{0,1\}$, is shown in Figure 3: Each edge from q to q' in the state-transition graph is labeled by a pair consisting of an input letter a and an output letter b, where $q' = \delta(q,a)$ and $b = \lambda(q,a)$. The machine outputs 1 on the second, fourth, sixth, *etc.,* occurrence of 1 on the input, and 0 otherwise. For example $F(0010011010) = 0000010010$. It thus functions as a kind of mod-2 counter.

Such finite-state machines can be explicitly built from logic gates and one-bit memories once one possesses the tables for the next-state function and the output function. The question that posed itself was whether, as was the case with propositional logic and combinational circuits, you could represent a 'natural' description of the machine's behavior in formal logic, and deduce the next-state and output functions from this description. Formulas of predicate logic, quantifying over time, looked like an obvious choice.

McNaughton's first-order language is more complicated than the logic FOL that we described above. It contains, of course, the usual variable symbols, but it also has a constant symbol 0 and two function symbols for predecessor and successor. There is a single binary relation symbol $\leq$, m one-place input predicate symbols $I_1, \ldots, I_m$ and n one-place output predicate symbol $U_1, \ldots, U_n$. To simplify things just a little, we will assume that the output alphabet is $\{0,1\}$, so that $n = 1$ and there is only a single output predicate U.

Terms of the language are built by starting with a variable or a constant and applying the function symbols. As a result, every term can be viewed as a constant number (*e.g.,* 7, obtained by applying the successor function seven times to the constant 0), or as an expression of the form $x+k$, where x is a variable and k is a constant, which could be either positive or negative. Atomic formulas are of the form $\tau \leq \tau'$, where τ and τ' are terms, or $V(\tau)$, where τ is a term, and V is either an input or an output predicate. Boolean operations are applied in the usual way. What is different how quantification works: Quantified formulas have the form

$$(\forall x)_{\tau_1}^{\tau_2}\psi, (\exists x)_{\tau_1}^{\tau_2}\psi,$$

where x is a variable and τ_1 and τ_2 are terms. Any occurrence in ψ of x, or of variables in the terms τ_1, τ_2, must be free. So, for example,

$$(\exists x)_0^t(U(x-1) \to (I(y) \vee I(t-1)))$$

is such a formula. We can treat $=$ as either built into the language, or define it as the conjunction of $\leq$ and $\geq$. The free variables in this formula are t and y. The formula above is just an abbreviation for

$$\exists x(0 \leq x \leq t \wedge (U(x-1) \to (I(y) \vee I(t-1)))).$$

McNaughton requires that *all* quantifications take this form.

McNaughton considers *behavioral specifications* of the form

$$U(t) \equiv \phi(t),$$

where the right-hand side is a formula constructed as described above, in which only the variable t appears free. His first example is the following:

$$U(t) \equiv I(t) \wedge (t \geq 1) \wedge (\exists x)_0^{t-1}(I(x) \wedge \neg U(x) \wedge (\forall z)_{x+1}^{t-1} \neg I(z)),$$

in which there is only a single input predicate. How are we to interpret this specification?

Think of $U(\tau)$ (respectively, $I(\tau)$) as meaning 'the τ^{th} output (resp. input) bit is 1', or, better yet, that the output (resp. input) at time τ is 1. The formula then says that the output at time t is 1 if and only if t is at least 1, the input at time t is 1, the input was 1 at some earlier time, and the last time the input was 1, the output was 0. If you reflect a moment, you can see that this is precisely the input-output behavior of the machine in Figure 3.

The description we have given of McNaughton's logic is not complete. If a behavioral specification defines a sequential function, then the right-hand side should not allow a term to be instantiated with a value greater than t or less than 0. Furthermore, if the term appears as the argument of the output predicate, then it cannot be instantiated with a value greater than $t - 1$. Enforcing these conditions syntactically requires McNaughton to engage in considerable contortions, and the resulting definition, which we will not give in this brief account, is a bit unwieldy. (Note that the formalism for FOL that we used in the preceding section avoids this problem: We only interpret formulas in finite words, so we can never instantiate a variable with a value that is outside the set of positions in the word.)

A sequential function F in which the output alphabet is $\{0, 1\}$ also defines a language $L_F \subseteq A^*$: We say that an input word w belongs to L_F if the last letter of $F(w)$ is 1. Seen in this light, the example we just gave is a bit jarring: Here L_F is the set of all bit strings that contain an even number of 1's, and whose last letter is 1. But Theorem 1, which, after all, is the whole point of this article, tells us that a language defined by a first-order formula has an aperiodic syntactic monoid. Yet we seem to have 'defined a language by a first-order formula' and gotten a result whose syntactic monoid contains a nontrivial group!

The additional power is conferred by the fact that McNaughton allows the formula in the right-hand side of a specification to contain occurrences of the output predicate U. If we prohibit this, then any language definable by one of McNaughton's behavioral descriptions is equivalent to one defined in FOL. To see this, observe that we can get rid of the function symbols for successor and predecessor by provisionally introducing a new binary relation symbol succ(x, y), interpreted to mean that y is the successor of x. So an atomic formula like $I(x + 2)$ turns into

$$\exists y \exists z (I(y) \wedge \text{succ}(z, y) \wedge \text{succ}(x, z)).$$

We also provisionally replace the free variable t by a constant symbol `last` meant to denote the last position of a word. We can then translate these new constants and predicates into FOL. For example succ(x, y) gets translated into

$$x < y \wedge \neg \exists z (x < z \wedge z < y).$$

The resulting sentence makes sense even if we did not purge the original specification of bounds violations. The illegal specification:

$$U(t) \equiv I(t + 1)$$

becomes $I(\texttt{last} + 1)$, which translates to

$$\exists t(\forall u(u \leq t) \wedge \exists v(t < v \wedge \neg\exists w(t < w < v) \wedge I(v))).$$

This sentence, which asserts the existence of a position following the final position of a word, is false in every word, and thus defines the empty language.

McNaughton calls his logic $\mathcal{L}$. He is aware (citing an earlier result of Elgot and Wright [Elgot and Wright 1959]) that $\mathcal{L}$ is not capable of describing the behavior of every finite-state sequential machine: For instance, it cannot define the function in which the k^{th} bit of output is 1 if and only if k is even. McNaughton therefore introduces the predicates $C_{r,s}$, where $C_{r,s}(t)$ is interpreted to mean that t is congruent to r modulo s. This strictly increases the expressive power of the logic: For example,

$$U(t) \equiv C_{0,2}(t),$$

now specifies the sequential function that outputs 1 exactly on inputs of even length. The enhanced logic is called $\mathcal{L}^{\pi}$.

The main work of the paper is the solution of the *synthesis problem*: to prove that every input-output behavior specified by the languages $\mathcal{L}$ and $\mathcal{L}^{\pi}$ can be computed by a finite-state sequential machine. McNaughton gives an algorithm for synthesizing the finite-state machine from a formula, first by converting the formula into an equivalent formula in a special normal form, and then by applying a number of standard automaton-theoretic constructions to recursively build the machine from the normalized formula.

We won't give the full definition of the normal form, but instead work with an example that highlights the crucial property possessed by normalized formulas. The following formula, which contains no occurrence of the output predicate U, is in normal form:

$$I(t) \wedge \neg I(0) \wedge (\exists x)_1^{t-1}\{I(x) \wedge (\exists y)_1^{x-1}\neg I(y) \wedge \neg(\exists w)_{x+1}^{t-1} I(w)\}.$$

The unquantified portion gives properties of the inputs at times 0 and t, namely that the input at time 0 is 0, and the input at time t is 1. The quantified subformula that follows says something about the string s of inputs between times 1 and $t-1$. That 'something' is the existence of a factorization $s = r_1 1 r_2$, and the inner quantified subformulas tell us something about the strings r_1 and r_2, namely that r_1 contains a 0 and r_2 contains no occurrence of 1. Thus the formula says that the entire string of inputs matches the regular expression

$$0(0+1)^*0(0+1)^*10^*1.$$

To be sure, we have left out many details of what constitutes normal form, and McNaughton does not argue in precisely this fashion, but this is the gist of the construction: Normal-form formulas exhibit this kind of nested structure, where each quantifier partitions an interval of input positions into two subintervals and a single position between these two subintervals. It is then possible to put the pieces together, starting from automata capturing the smallest intervals and working outwards. For this, McNaughton employs standard constructions of deterministic automata that recognize the union, intersection, complement, and concatenation of languages for which automata are known. Indeed, the hard part of the argument is the conversion of arbitrary formulas to normal form.

Matters are only slightly more complicated for formulas that contain the predicates $C_{r,s}$, and there is a neat trick for handling formulas that contain the output predicate as well.

McNaughton's article concludes with a few open problems, centered around the question of whether all finite-state sequential machines are captured by $\mathcal{L}^{\pi}$, and, if not,

whether one can go farther in the description of such machines by adding new predicates to $\mathcal{L}^\pi$. He poses the following questions:

(1) Are there finite-state machines that cannot be specified by formulas of $\mathcal{L}^\pi$? McNaughton conjectures 'yes', and asks, if this is the case, whether there is an interesting description of the finite-state machines that can be specified in this language.
(2) Is there a way to add new predicates to $\mathcal{L}^\pi$ so as to produce a proper extension (*i.e.,* so as to specify sequential functions that cannot be specified in $\mathcal{L}^\pi$) such that all the specified functions can be computed by finite-state machines? (McNaughton conjectures 'no'.)
(3) Is there an extension of $\mathcal{L}^\pi$ that can specify all the functions computed by finite-state machines? Note that if the previous conjecture is true, then any such extension will also define behaviors that cannot be computed by finite state machines. McNaughton conjectures 'no', and provides an example of a machine which, he believes, cannot be represented in any such extension. Later, in *Counter-free Automata*, he and Papert strengthened this conjecture to the claim that if a finite-state sequential function can be specified in an extension of $\mathcal{L}^\pi$, then it can already be specified in $\mathcal{L}^\pi$.

As we shall see, all of McNaughton's conjectures turned out to be true, although, surprisingly, it took advances in circuit complexity to confirm them all.

3. MEANWHILE, BACK IN THE USSR... (TRAKHTENBROT, 1958)

In *Counter-free Automata*, there is an explanation of why McNaughton's 1960 paper was never published: It turned out that two years earlier, a paper published by Boris Trakhtenbrot in the Soviet Union [Trakhtenbrot 1958], did essentially what McNaughton had set out to do, and then some. This was a familiar story in Cold War-era science: Soviet mathematicians could not easily travel abroad to spread word of their work in lectures or coffee-break conversations. Translations of works from the West could only be published after government approval (often with obligatory political commentary attacking the bourgeois tendencies of the authors), and translations of Russian papers into other languages required western mathematicians who knew Russian. The result was many instances of the same theorem being published independently, and nearly simultaneously, on both sides of the Iron Curtain. (Shannon's 1937 work on propositional logic and combinational circuits provides another instance of this phenomenon, similar results having been obtained by Shestakov [Shestakov 1941], at roughly the same time.

The journal in which Trakhtenbrot's paper appeared– *Doklady Akademia Nauk SSSR*– enforced a strict 4-page limit, so Trakhtenbrot was only able to provide an outline of the proof of the principal result, leaving it to the ambitious reader to fill in the missing pieces.

In McNaughton's notation, Trakhtenbrot's logical specifications have the form

$$U(t) \equiv \mathcal{A}(t),$$

where $\mathcal{A}$ is a formula containing the input predicates $I_1, \ldots, I_m$, but no output predicates. First-order quantifications, as in McNaughton, are bounded. Initially, these are only bounded from above, and thus take the form $(\exists x)_0^y$, or $(\forall x)_0^y$, but Trakhtenbrot also mentions allowing the quantified variable to be bounded both below and above. So far, this is identical to what McNaughton does–but Trakhtenbrot also allows quantification over unary predicate variables. By this device, the use of monadic second-order logic rather than first-order logic, he is easily able to capture all finite-state sequential functions. The principal contribution of the paper is a synthesis algorithm, very different from McNaughton's, whereby the function described by a formula in this logic is

ingeniously transformed into a finite state machine. It's no wonder McNaughton felt that he was scooped by this paper: he had solved the synthesis problem for first-order logic and plainly saw its inadequacies as a language for specifying the behavior of automata. In the meantime, Trakhtenbrot had discovered a formulation that captures all finite automata precisely, and which, when restricted to the first-order fragment, shows what McNaughton had set out to prove.

Trakhtenbrot's 1958 paper may be the first appearance of the equivalence of monadic second-order logic and finite automata. While it is stated in terms of sequential machines, rather than regular languages, it is essentially equivalent to Theorem 2 above. This was yet another result obtained independently in the U.S., at about the same time, by Büchi [Büchi 1960] and Elgot [Elgot 1961]. Interestingly, McNaughton does cite Büchi's and Elgot's papers, which at the time were only available as research reports. It may be that because the formalism of these papers is so different, McNaughton did not realize that the synthesis algorithms they contained, restricted to first-order formulas, could have been used to provide a solution to his own synthesis problem. [6]

4. PETRONE AND SCHÜTZENBERGER, 1963

'Sur un problème de McNaughton' appears as a research report of the European Community on Atomic Energy(!) with an official date of 1965. However, Schützenberger cites it in [Schützenberger 1965a], giving it a date of 1963, and it was clearly done much earlier than [Schützenberger 1965a]. As the title indicates, it was inspired by a problem in McNaughton's 1960 article, and the main result proves the first conjecture of McNaughton cited above.

This paper marks a watershed. It explicitly introduces, for the first time, what we now call the class of star-free languages over a finite alphabet, along with an extension of this class. It makes the connection between these language classes constructed using word-set operations and the logical formalism of McNaughton. Most significantly, it provides what I believe is the first nontrivial demonstration that some language does *not* belong to a class by finding an algebraic invariant of the syntactic monoids of languages in the class. [7]

Here, in outline, is the argument of the paper: Given a finite alphabet C, we define two families of subsets of C^*: $\mathcal{Q}_0$ denotes the smallest family that contains all the sets $\{c\}$, where $c \in C$, and that is closed under boolean operations and concatenation. $\mathcal{Q}_\pi$ denotes the smallest family that contains both the letters and the languages $(C^n)^*$, where $n > 0$, and is likewise closed under boolean operations and concatenation. That is, the elements of $\mathcal{Q}_0$ are represented by generalized regular expressions (regular expressions in which complementation is permitted) that do not use the star operation, hence 'star-free'. The language over $C = \{0, 1\}$ represented by the regular expression

$$0(0+1)^*0(0+1)^*10^*1.$$

that we found earlier, is star-free, although it may not look it. The reason is that

$$(0+1)^* = \bar{\emptyset}, 0^* = \overline{\bar{\emptyset}1\bar{\emptyset}},$$

where the horizontal bar denotes complementation, so we can rewrite the expression without stars.

[6] The focus of these papers was also quite different. Trakhtenbrot, like McNaughton, was concerned with the problem of synthesis, whereas Büchi and Elgot used automata as a device to solve a problem in logic, namely the decidability of monadic second-order logic. Wolfgang Thomas, writing in this same column [Thomas 2018], has provided a bit of revisionist history of this result as well.

[7] It is also—I should caution anyone who actually tries to read it—something of a mess, containing numerous typos, poor choices of notation, and, as noted later, at the core of its first part, an argument that is nearly unreadable and, I suspect, wrong.

The essence of the argument is to show that *(a)* machines specified by formulas of $\mathcal{L}$ and $\mathcal{L}^\pi$ give rise to languages in $\mathcal{Q}_0$ and $\mathcal{Q}_\pi$, respectively; *(b)* the syntactic monoids of languages in $\mathcal{Q}_\pi$ contain only abelian groups; and *(c)* there is a sequential machine that gives rise to a language whose syntactic monoid contains a nonabelian group.

This works as follows: a finite-state sequential machine $\mathcal{M}$ calculating a function F from A^* to B^*, Petrone and Schützenberger associate a language $L'_\mathcal{M}$ in C^*, where $C = A \times B$. This consists of all the strings

$$(a_1, b_1) \cdots (a_n, b_n) \in (A \times B)^*$$

such that $F(a_1 \cdots a_n) = b_1 \cdots b_n$.

Observe that this is different from the language $L_F \subseteq A^*$ that we associated with the machine in Section 2. Treating the output alphabet B as part of the input provides us with a way to understand the odd example, which we saw earlier, of the mod-2 counter machine $\mathcal{M}$ being definable in first-order logic. Petrone and Schützenberger view this example as defining a set $L'_\mathcal{M}$ of words over the alphabet

$$C = \{a = (1|0), b = (1|1), c = (0|0)\}.$$

Words in L begin with either a or c, and have the property that between any two occurrences of a there is an occurrence of b, and vice-versa. The language is thus defined by the expression

$$\overline{b\overline{\emptyset}} \cap \overline{\overline{\emptyset}ac^*a\overline{\emptyset}} \cap \overline{\overline{\emptyset}bc^*b\overline{\emptyset}},$$

and c^* itself is given by the expression

$$\overline{\overline{\emptyset}a\overline{\emptyset}} \cap \overline{\overline{\emptyset}b\overline{\emptyset}}.$$

Thus $L'_\mathcal{M}$ is star-free.

Petrone and Schützenberger claim that for every machine $\mathcal{M}$ whose behavior is definable in McNaughton's logic $\mathcal{L}$, the language $L'_\mathcal{M}$ is in $\mathcal{Q}_0$, and similarly for $\mathcal{L}^\pi$ and $\mathcal{Q}_\pi$. Three very peculiar pages are devoted to justifying this claim, beginning with a kind of disclaimer '[the sequential functions associated with the automata considered by McNaughton] satisfy very special conditions, to whose discussion McNaughton devotes far too many pages for us to be able to hope to reproduce them in their entirety', and proceeding to a cryptic and, insofar as I can understand it, erroneous argument that does not mention logic at all. (This may be evidence of Schützenberger's alleged distaste for logic.) Nonetheless, as we indicated above, this claim is already implicit in McNaughton's original argument.

The paper then takes a giant leap into unexplored territory, and turns to semigroup theory. The syntactic monoid of a language L is neither defined nor named. Instead a 1951 result of Teissier [Teissier 1951] is cited, to the effect that for every language $L \subseteq C^*$, there is a unique monoid M_L and homomorphism

$$\mu_L : C^* \to M_L$$

such that

$L = \mu_L^{-1}(\mu_L(L))$,
For any monoid homomorphism $\psi : C^* \to M$ such that $L = \psi^{-1}(\psi(L))$, M_L is a homomorphic image of $\psi(C^*)$.

This is the familiar fact that the syntactic monoid of a language L both recognizes L and is the quotient of every monoid that recognizes L.

Petrone and Schützenberger then make the following claim:

Proposition 3 If $L \in \mathcal{Q}_0$, then every group in M_L is trival (*i.e.,* M_L is aperiodic). If $L \in \mathcal{Q}_\pi$, then every group in M_L is abelian.

This is shown in three steps, in an argument that has by now become quite familiar: Given

$$\mu_{L_i} : C^* \to M_{L_i} \quad i = 1, 2$$

a new monoid, which is now universally called the *Schützenberger product* $M_{L_1} \diamond M_{L_2}$, and homomorphism

$$\phi : C^* \to M_{L_1} \diamond M_{L_2}$$

are constructed. It is shown that every group in $M_{L_1} \diamond M_{L_2}$ is a subgroup of $G_1 \times G_2$, where G_i is a group in M_i, and that $M_{L_1} \diamond M_{L_2}$ recognizes the languages $L_1 \cup L_2$, $L_1 \backslash L_2$, and $L_1 L_2$.

Further, if L_1, L_2 are regular, then M_{L_1} and M_{L_2} are finite. Finite monoids have the following property: If M' is a homomorphic image of M, then every group in M' is a homomorphic image of a group in M.

Thus, when we build the languages in $\mathcal{Q}_0$, we begin with languages $\{c\}$ whose syntactic monoids contain no nontrivial groups, and at each step, obtain new languages whose syntactic monoids still contain no nontrivial groups. Likewise, when we build the languages in $\mathcal{Q}_\pi$, we start with languages $\{c\}$ and $(C^n)^*$ whose syntactic monoids contain no nonabelian groups, and this property is preserved at each step of the way. This establishes Proposition 3.

They've almost finished. To prove McNaughton's first conjecture, Petrone and Schützenberger need to provide an example of a sequential machine $\mathcal{M}$ such that the syntactic monoid of $L'_{\mathcal{M}}$ contains a nonabelian group. Their example is shown in Figure 4. The input and output alphabets for the machine are both $\{0, 1\}$, so the alphabet of $L'_{\mathcal{M}}$ is $C = \{0, 1\} \times \{0, 1\}$.

We need to show that the syntactic monoid of $L = L'_{\mathcal{M}}$ contains a nonabelian group. If you make each of the five states in this diagram accepting, you obtain an automaton recognizing $L'_{\mathcal{M}}$. Moreover, this is the minimal automaton of $L'_{\mathcal{M}}$, which is shown by verifying that no pair of distinct states can be merged. For example, s_5 cannot be merged with s_3, as $s_5(0|0)(0|1)(0|1)$ is accepting, while $s_3(0|0)(0|1)(0|1)$ is undefined. Thus the syntactic monoid of $L'_{\mathcal{M}}$ is the transition monoid of this automaton. Observe that the transformations induced by the words $\alpha = (0|0), \beta = (1|0)(1|0)$ are both permutations of $\{s_2, s_3, s_4, s_5\}$. We have $\alpha\beta \neq \beta\alpha$, since $\alpha\beta$ is the transposition $(s_2 \quad s_5)$, while $\beta\alpha$ is the transition $(s_2 \quad s_3)$. Thus the monoid contains a nonabelian group.

Proposition 3 is the engine that makes it all work. Interestingly, Petrone and Schützenberger state the proposition as a 'Remark', and precede it with the astonishing and mysterious comment, 'The following remark is a trivial application of results of Miller and Clifford [on ideal structure in finite semigroups] to a special case of Kleene's Theorem on regular events. However we give the proof in the interests of being self-contained.' This suggests that the construction of the Schützenberger product could have been dispensed with entirely by a simple appeal to some established semigroup theory. I cannot for the life of me see what this trivial application is, and nowhere else in Schützenberger's writing, which makes many references both to the Schützenberger product and to the Miller and Clifford paper, do I find a suggestion of how such an argument might work.

5. 'SCHÜTZENBERGER'S THEOREM', AT LAST

By 1964, Schützenberger had succeeded in proving the converse to Proposition 3 for the case of the star-free languages $\mathcal{Q}_0$: If the syntactic monoid of a regular language L contains no nontrivial groups, then L is star-free. The first appearance in print was in the

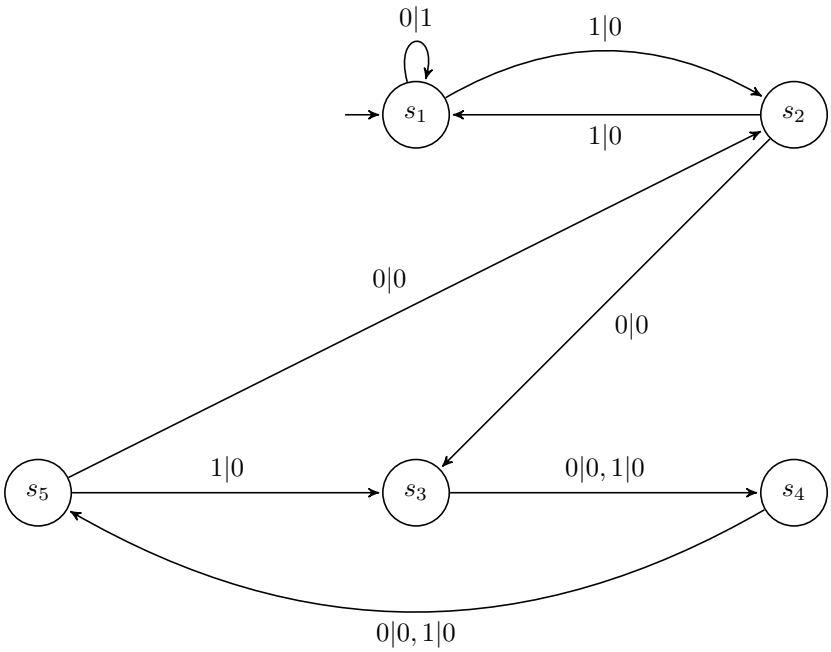


Fig. 4. A sequential machine not definable in $\mathcal{L}^{\pi}$.

proceedings of the Séminaire Dubreil in 1965 [Schützenberger 1965c], followed by the journal publication in *Information and Control* [Schützenberger 1965a] the same year. The latter has become the reference of record. Schützenberger presented the result in a book chapter [Schützenberger 1966], and in a related result on varieties of languages closed under concatenation in the same book [Schützenberger]. In each of these papers Schützenberger cited the link to McNaughton's work on logic, and in some cases to Trakhtenbrot's. Indeed, in the abstract of [Schützenberger 1965a] we read 'An alternative definition is given for a family of subsets of a free monoid that has been considered by Trahtenbrot and by McNaughton', and in a rambling screed [Schützenberger 1965b] entitled 'Algebraic Theory of Automata', presented at the IFIP Congress the same year, he writes 'the theory of Trachtenbrot and McNaughton on regular expressions with no Kleene stars, blends itself with that of finite monoids whose subgroups are degenerate'. So the equivalence between star-free languages and first-order logic was by this time firmly established in Schützenberger's mind.

The proof given here of Proposition 3 proceeds exactly as in Petrone and Schützenberger. The converse direction, which is almost always the hard direction in such proofs, requires induction on the cardinality of a monoid recognizing L, and usually an added dose of the algebraic theory of semigroups. In Schützenberger's proof, the requisite algebra is the Green-Rees theory concerning the structure of minimal and subminimal ideals I of a monoid M. This is used to show that if I is such an ideal, and if every language recognized by the quotient monoid M/I is star-free, then the same is true of every language recognized by M.

Schützenberger's early writings on this result make frequent reference to the Krohn-Rhodes theorem, which appeared around the same time [Krohn and Rhodes 1965]. This theorem shows how to construct a serial decomposition of a sequential machine, in which the components are determined by the groups present in the transition monoid

of the machine. In the case where all these groups are trivial, each component takes the form of a simple 2-state machine. It was not long before someone discovered that you could use this to prove that aperiodic monoids only recognize star-free languages. The first such published proof is due to A. R. Meyer [Meyer 1969].

6. MCNAUGHTON AND PAPERT, *COUNTER-FREE AUTOMATA,* 1971.

This research monograph in the form of a textbook is devoted to a presentation of Theorem 1 on a broad canvas. The authors consider five different ways of representing regular languages (here called 'regular events'): *(a)* by regular expressions and generalized regular expressions; *(b)* by deterministic finite automata; *(c)* by homomorphisms into finite monoids; *(d)* by *nerve nets,* a representation that holds a place of honor in the history of the theory of automata, but is now largely forgotten [8]; and *(e)* by the logical formulas from McNaughton's 1960 paper. For each of these five representations, the authors identify a natural subclass of the representable languages, and prove that all these subclasses are the same. The equivalence of star-free languages, languages with aperiodic syntactic monoids, and languages represented by 'loop-free nerve nets' is shown via the Krohn-Rhodes theorem.

The representation by logical formulas is taken up last. Since the book's focus is on languages, rather than sequential machines, it replaces McNaughton's original classes $\mathcal{L}$ and $\mathcal{L}^\pi$ by four language classes, FOL, FOL_C, FOL_U, and FOL_{CU}–the C subscript indicates that the predicates $C_{r,s}$ are allowed in formulas, and the U subscript that the output predicate is allowed. They show how to translate loop-free nerve nets directly into logical formulas, and prove that languages in FOL are star-free by application of McNaughton's original normal form. This completes their proof of Theorem 1. The result is bootstrapped to provide effective characterizations not just of FOL, but of the three other language classes as well.

As the book draws to a close, McNaughton and Papert revisit the open problems posed by McNaughton eleven years earlier. They note that the characterizations of the four language classes answer some of these questions, and they leave the others for the consideration of future researchers. And then *Counter-free Automata* passed into the semi-oblivion that was the fate of so much mathematical writing that has left an enduring legacy. Frequently cited but seldom read, its sometimes awkward notations and formalisms superseded by cleaner versions, its contributions acknowledged honorably, if not always correctly, it went out of print.

7. EPILOGUE. ENTER CIRCUIT COMPLEXITY

I will not attempt to discuss here the many generalizations and variations that followed Theorem 1, nor the stubborn still-open problems left in its wake (except to say that the problems concerning first-order descriptions of *trees* remain an enigma). But I do want to briefly follow one particular trail, since it directly concerns the open problems from McNaughton's original work.

Consider a sentence ϕ of FOL. If you want to, throw in some of the $C_{r,s}$ predicates. In fact, throw in *any* predicates you like, as long as they are only 'numerical' predicates like $C_{r,s}$ and $<$, which only depend on the values of the input positions, and not on the input letters in those positions. For example, you could have a predicate $S(x, y, z)$ meaning $z = x + y$, or $P(x)$ meaning x is prime. Let's call this class FOL_{Any}.

To make things simpler, we'll suppose that the input alphabet is $\{0, 1\}$, so that ϕ defines a language $L \subseteq \{0, 1\}^*$. Fix the length n of the input word, and then unravel

[8]The bulk of Kleene's long original work introducing regular expressions [Kleene 1956] (which was *also* written under contract to Air Force) is carried out in terms of the nerve net model, with finite automata brought in only at the very end

the formula, replacing existential quantifiers by OR gates with n inputs, and universal quantifiers by AND gates with n inputs. The result is a circuit with n input bits.[9] If we carry this out for every input length n, we obtain a family of circuits recognizing L. The depth of the circuits is bounded across the entire family by a constant d (essentially the quantifier depth of ϕ, plus a little extra for the wiring at the level of the inputs) and the number of gates by a polynomial in n (since each circuit is a tree of depth d whose nodes have n children).

The class of languages recognized by such circuit families is called AC^0. So $FOL_{Any} \subseteq AC^0$. In fact, the opposite inclusion is true as well (a beautiful insight with a fairly simple proof, shown by Immerman [Immerman 1987] , and, independently, by Gurevich and Lewis [Gurevich and Lewis 1984]), so

Theorem 4

$$AC^0 = FO_{Any}.$$

The class AC^0 occupies a special place in computational complexity, because it is one of the few areas where we possess unconditional superpolynomial lower bounds:

Theorem 5 *No constant-depth family of circuits recognizing the set of bit strings*

$$PARITY = \{w \in \{0,1\}^* : w \textit{ contains an even number of 1's } \}$$

has polynomial size. That is, $PARITY \notin AC^0$.

This famous theorem is due to Furst, Saxe and Sipser [Furst et al. 1984], and, in a very different form, to Ajtai [Ajtai 1983]. There's nothing special about counting 1's modulo 2; the same result holds if we try to count with respect to any modulus greater than 1.

So, in light of Theorem 4, we have a regular langauge that is not in FOL_{Any} (and thus certainly not in FOL_C). David Barrington, Denis Thérien and I (and, independently, Kevin Compton) tried to answer the question of what regular languages belong to AC^0, a question that had earlier been raised by Chandra, Fortune and Lipton [Chandra et al. 1985].

We found [Barrington et al. 1992] that these are exactly the languages in FOL_C. The proof of this is carried out in two steps. The first is a characterization, in the spirit of Schützenberger's theorem, of the languages in FOL_C. In terms of generalized regular expressions, these are exactly the languages in the class $\mathcal{Q}_\pi$ described by Petrone and Schützenberger. In terms of algebra, they are the regular languages L whose syntactic morphisms $\mu_L : A^* \to M_L$ have the property that for each fixed n, $\mu_L(A^n)$ contains no nontrivial groups. For example, the set of words of even length, while not definable in FOL, is in FOL_C because in this case $|\mu_L(A^n)| = 1$ for each n, while $PARITY$ is not definable because $\mu_L(A)$ is itself a group of order 2. This condition can be tested effectively, since there are only finitely many different subsets $\mu_L(A^n)$, and they can all be tabulated.[10] In the second step we invoke the circuit complexity result: If $\mu_L(A^n)$ contains a nontrivial group for some n, then an AC^0 circuit family recognizing L could be

[9]The complete circuit in this case is a tree with $2n$ leaves, one leaf for each input bit, and one leaf for the negation of each input bit. The numerical predicates we threw in only add a constant number of levels of additional 'wiring' at the between these leaves and the gates.

[10]Such an effective characterization of FOL_C answers one of McNaughton's original problems. In [Barrington et al. 1992] we noted (honorably) that McNaughton and Papert solved this problem, and remarked (incorrectly) that they used 'a different argument' for this. It was not until I was well into the writing of the present article that I actually took the time to read their characterization of FOL_C and figure out what they might have meant by 'no input violations of length k'. I realized, somewhat to my embarrassment, that their characterization is the same one that we 'discovered', and they proved it with pretty much the same argument!

turned into a family that counts 1's modulo k for some $k > 1$, contradicting Theorem 5. Thus L cannot be in AC^0.

We can write this result as

Theorem 6

$$FOL_{Any} \cap \textit{ Regular Languages } = FOL_C.$$

This solves the problem that McNaughton and Papert left unsolved: is there *any* numerical predicate that you can add to first-order logic and get a regular language that is not already definable in FOL_C? The answer, as they conjectured, is 'no'.[11]

I will end with a problem that has tantalized me for many years, and on which I have surely wasted far too much time. The only way we know how to *prove* Theorem 6 is by using results from circuit complexity. But as stated, it doesn't look like it has anything at all to do with complexity bounds; it's simply a statement about automata and logic: If you can define a regular language with a first-order sentence, then you can define it by a first-order sentence that uses only regular stuff. So, is there a 'pure automata and logic' proof (whatever that might mean) of this result? We know how to prove, independently of any considerations of circuit complexity, that $PARITY$ is not definable in FOL_C, so this would provide a new proof of Theorem 5, and a new method for obtaining circuit complexity bounds.

McNaughton, certainly, was focused on automata and logic, and probably was not thinking of anything like circuit complexity when he formulated the original versions of these conjectures. So just what was he thinking? In one of those maddening 'this margin is too small' passages, he wrote

> I regret that the reasons for my conjectures... are too complicated, and by themselves inconsequential, for presentation here.

ACKNOWLEDGMENTS

Those who think 'you can find everything on the Web', haven't spent enough time hunting for preprints and books that came out before there *was* a Web. Thanks to Alain Lascoux, Jean Berstel, and Dominique Perrin for creating and curating the collection of Schützenberger's complete works (and for putting it on the Web!), and especially to Dominique for sending me a printed version. To the intrepid librarian-detectives of the Thomas P. O'Neill Library at Boston College, who located what may be the only extant copy of McNaughton's 'Symbolic Logic and Automata' in a private collection in Kansas City, Missouri. To my recently departed colleague Jim Gips, who, while cleaning his office long ago, paused before tossing his copy of *Counter-free Automata* into the trash, and instead came across the hall to say 'I think this is the sort of thing that you like.' And to Charles Paperman and Michaël Cadilhac who helped me find a copy of the book while I was in Paris, to Jean-Eric Pin for many discussions and anecdotes, and to Mikołaj Bojańczyk for encouraging me to write this article.

REFERENCES

M. Ajtai. 1983. Σ^1_1-Formulae on finite structures. *Annals of Pure and Applied Logic* 24, 1 (1983), 1 – 48.

David A. Mix Barrington, Kevin Compton, Howard Straubing, and Denis Thérien. 1992. Regular languages in NC^1. *J. Comput. System Sci.* 44, 3 (1992), 478–499.

[11] As a corollary, we get another of McNaughton's conjectures: There is no way to properly extend FOL_C to get a class in which only regular languages can be defined. This can be proved without recourse to circuit complexity—the details are worked out in [Straubing 1994].

Our discussion has neatly pushed aside anything concerning those pesky output predicates. McNaughton and Papert conjecture that Theorem 6 also holds for $FOL_{Any,U}$ and FOL_{CU}. In [Barrington et al. 1992], we took note of this and wrote that 'our methods do not appear to apply' to this problem. Once again, we were just too lazy to try to understand what the output predicates really meant. Our methods do apply, and the conjecture is true.

Jean Berstel, Alain Lascoux, and Dominique Perrin (Eds.). 2009. *Marcel-Paul Schützenberger, Oeuvres Complètes*. http://igm.univ-mlv.fr/~berstel/Schutzenberger/Oeuvres-Completes/oeuvrescompletes.html

J. Richard Büchi. 1960. Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 6 (1960), 66–92.

Ashok K. Chandra, Stephen Fortune, and Richard J. Lipton. 1985. Unbounded Fan-In Circuits and Associative Functions. *J. Comput. Syst. Sci.* 30, 2 (1985), 222–234.

Samuel Eilenberg. 1976. *Automata, Languages, and Machines*. Vol. B. Academic Press, New York and London.

Calvin C. Elgot. 1961. Decision Problems of Finite Automata Design and Related Arithmetics. *Trans. Amer. Math. Soc.* 98, 1 (1961), 21–51.

Calvin C. Elgot and Jesse B. Wright. 1959. Quantifier elimination in a problem of logical design. *Michigan Math. J.* 6, 1 (1959), 65–69. DOI: http://dx.doi.org/10.1307/mmj/1028998141

Merrick L. Furst, James B. Saxe, and Michael Sipser. 1984. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory* 17, 1 (1984), 13–27.

Yuri Gurevich and Harry R. Lewis. 1984. A Logic for Constant-Depth Circuits. *Information and Control* 61, 1 (1984), 65–74.

Neil Immerman. 1987. Languages that Capture Complexity Classes. *SIAM J. Comput.* 16, 4 (1987), 760–778.

Steven Cole Kleene. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*, C. E. Shannon and J. McCarthy (Eds.). Number 34 in Annals of Mathematics Studies. Princeton University Press, 3–40.

Kenneth Krohn and John Rhodes. 1965. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Amer. Math. Soc.* 116 (1965), 450–464.

Robert McNaughton. 1960. Symbolic Logic and Automata. (1960). Technical report, Wright-Paterson Air Force Base.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. The MIT Press, Cambridge, Mass.

Albert R. Meyer. 1969. A Note on Star-Free Events. *J. ACM* 16, 2 (1969), 220–225.

Luigi Petrone and Marcel-Paul Schützenberger. 1965. Sur un problème de McNaughton. (1965). Euratom.

Marcel-Paul Schützenberger. Sur certaines variétés de monoïdes finis. In *Automata Theory, Ravello 1964*. Academic Press, New York, 314–319.

Marcel Paul Schützenberger. 1965a. On finite monoids having only trivial subgroups. *Information and Control* 8 (1965), 190–194.

Marcel-Paul Schützenberger. 1965b. On the algebraic theory of automata. In *Information Processing 65: Proceedings of IFIP Congress 1965*, Wayne A. Kalenich (Ed.). Spartan Books, 27–29.

Marcel-Paul Schützenberger. 1965c. Sur les monoïdes finis n'ayant que des sous-groupes triviaux. In *Séminaire Dubreil-Pisot, année 1964-65*. Inst. H. Poincaré, Paris.

Marcel-Paul Schützenberger. 1966. On a family of sets related to McNaughton's L-language. In *Automata Theory, Ravello 1964*. Academic Press, New York, 320–324.

Claude Elwood Shannon. 1938. *A Symbolic Analysis of Relay and Switching Circuits*. http://hdl.handle.net/1721.1/11173

V. I. Shestakov. 1941. The algebra of two-poled schemata constructed exclusively from two-poled networks (the algebra of A-schemes)(in Russian). *Automatics and Telemechanics* 2 (1941), 15–24.

Stephen M. Stigler. 1980. Stigler's Law of Eponymy. *Transactions of the New York Academy of Sciences* 39 (1980), 147–157. https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/j.2164-0947.1980.tb02775.x

Howard Straubing. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel and Berlin.

Howard Straubing and Pascal Weil. 2011. An introduction to finite automata and their connection to logic. In *Modern Applications of Automata Theory*, Deepak D'souza and Priti Shankar (Eds.). World Scientific Publishing Co., Inc., River Edge, NJ, USA. http://arxiv.org/abs/1011.6491

Marianne Teissier. 1951. Sur les équivalences régulières dans les dem-groupes. *Comptes Rendus de l'Académie des Sciences* 232 (1951), 1987–1989.

Wolfgang Thomas. 2018. *ACM SIGLOG News* 5, 1 (2018), 13–18.

Boris A. Trakhtenbrot. 1958. Synthesis of logical nets whose operators are given by one-place predicate calculus (in Russian). *Doklady Akademia Nauk SSR* 118, 4 (1958).