Monadic Spectra and Regular Sets

Mark W. Krentel
TR 86-736
February 1986

Department of Computer Science
Cornell University
Ithaca, NY 14853
1. **Introduction**

A result from the applications of finite model theory to automata theory is the equivalence of finite spectra and the set of tally languages recognizable by polynomially time bounded non deterministic turing machines. In this paper, we show another equivalence between these two fields; namely, the equivalence between a restricted type of spectra and the class of regular languages.

H. Scholz [13] first introduced the definition of spectra in 1952. If $\sigma$ is a first order sentence with equality, then the finite spectrum of $\sigma$ is the set of natural numbers $n$ where $\sigma$ has a model of cardinality $n$. This problem can be naturally related to formal languages by identifying the integer $n$ with the string $1^n$. Thus a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$ becomes a language over a one symbol alphabet, say $\{1\}^*$, called a tally language. The result of N. Jones and A. Selman [8] states that a tally language is in the class NP if and only if it is a finite spectrum. We write $(N)P$ for the set of languages recognizable in polynomial time by a (non)deterministic turing machine [2,9]. Whether or not spectra are closed under complement is an open problem posed by G. Asser [1] in 1955. Should spectra not be closed under complement, then the result of Jones and Selman would imply that $P \neq NP$.

R. Fagin [5] extended the notion of spectra to what he called generalized spectra. Let $\sigma$ be a sentence with equality whose predicates are either free or existentially quantified. Such a
is called an existential second order sentence. The generalized spectrum of \( \sigma \) is the set of finite structures \( \mathcal{O} \), consisting of a domain and interpretation for \( \sigma \)'s free predicates, such that \( \sigma \) is true in \( \mathcal{O} \). We can also encode a finite structure as a string in \( \{0,1\}^* \) in a natural way. Any reasonable encoding will work; for example, we could label the elements of the domain by \( \{1,\ldots,n\} \) and encode this set by the binary representation of \( n \). We can also write 0 for false and 1 for true and encode a monadic relation, \( U \), as a sequence of \( n \) binary digits representing \( U(1), \ldots, U(n) \). A binary relation, \( P \), would then be written as a sequence of \( n^2 \) binary digits for \( P(1,1), \ldots, P(1,n), \ldots, P(n,1), \ldots, P(n,n) \), and so on. Fagin has then shown another important correspondence between generalized spectra and NP. Specifically, a set of structures of a given type and closed under isomorphism is a generalized spectrum if and only if the set of encodings of its members is in NP. Again it is an open problem if the class of generalized spectra is closed under complement, and again if the answer is no then \( P \not\subseteq NP \). Using the Ehrenfeucht game [3], Fagin [6] has, however, shown that if we restrict \( \sigma \)'s quantified predicates to be unary, then the resulting class of monadic spectra is not closed under complement. In particular, the class of non-connected graphs is a monadic spectrum but the class of connected graphs is not.

We can get further equivalences by fixing relations in addition to equality. For example, we can consider the spectra of sentences with, say, successor, linear ordering, addition, or multiplication. Specifically, let \( x \) be a string in \( \{0,1\}^* \)
and put \( n = |x| \), the length of \( x \). Define the structure
\( \mathcal{C}_x = \langle \{1, 2, \ldots, n\}, S^n, R_x \rangle \) where the domain of \( \mathcal{C}_x \), the first \( n \) integers, has one element for each symbol in \( x \); \( S^n \) is the usual successor relation restricted to the domain of \( \mathcal{C}_x \); and \( R_x \) is a unary function with range \( \{0, 1\} \) that encodes the string \( x \). It is also convenient to enlarge the domain of \( \mathcal{C}_x \) by a function on the length of \( x \). If \( f \) is a function where \( f(n) \geq n \), then we define the augmented structure
\( \mathcal{C}_x^f = \langle \{1, 2, \ldots, f(n)\}, S_f^n, R_x^f \rangle \) where the range of \( R \) becomes \( \{0, 1, 2\} \) to designate elements of \( \mathcal{C}_x^f \) not in the string \( x \).

Lynch [10] has shown that for every \( f \)-time bounded non-deterministic turing machine, \( T \), where \( f \) is a polynomial, there is an existential second order sentence \( \sigma \) such that for all strings \( x \in \{0, 1\}^* \), \( T \) accepts \( x \) if and only if \( \mathcal{C}_x \models \sigma \). Also, there is a binary \( \sigma' \) where \( T \) accepts \( x \) if and only if \( \mathcal{C}_x^f \models \sigma' \).

Thus, the classes of languages \( \{x: \mathcal{C}_x \models \sigma\}: \sigma \) is an existential second order sentence \} and \( \{x: \mathcal{C}_x^f \models \sigma\}: f \) is a polynomial and \( \sigma \) is a binary existential sentence \} are both exactly the class NP. We also get precisely the class NP by adding the usual relation of addition to \( \mathcal{C}_x^f \) and restricting \( \sigma \) to be unary.

After Lynch showed that binary predicates with successor are sufficient to describe all of NP, he stated without proof that the class of languages \( \{x: \mathcal{C}_x \models \sigma\}: f \) is a polynomial and \( \sigma \) is unary \} is properly contained in \( P \); and he posed the problem of characterizing monadic spectra with successor. Here, we are able to answer his question. For any polynomial \( f \) and any second order monadic \( \sigma \), the language \( \{x \in \{0, 1\}^*: \mathcal{C}_x^f \models \sigma\} \) can be recognized by a deterministic multitape turing machine in time \( O(n) \) where
\( n = |x| \). In fact, for any constant \( c > 1 \), there is a turing machine that will recognize the above language in \( cn \) moves. Using the unaugmented domain we obtain a model theoretic characterization of the regular sets. That is, the class 
\[ \{x: \mathcal{L}_X \models \sigma\}: \sigma \text{ is monadic} \] 
is exactly the class of regular languages over \( \{0,1\}^* \). A similar result was obtained by C. Elgot [4] using the weak monadic second order arithmetic of the natural numbers.

The technique we use is elimination of quantifiers: a monadic \( \sigma \) can be effectively transformed to an equivalent \( \tau \) in a certain normal form. For \( \tau \) in this normal form, a finite state machine can check if \( \mathcal{L}_X \models \tau \). A further result is that while the languages describable by first order sentences with successor are a proper subset of those using linear ordering, the languages obtained from second order sentences with successor are the same as those using linear ordering. The first order languages with linear ordering are, of course, equivalent to the non-counting, star-free or group-free languages of R. McNaughton and S. Papert [11]. In fact, the closure of first order languages with successor under the boolean operations and concatenation gives precisely the first order languages with linear ordering. Thus, the difference between these two classes of languages is the ability to express concatenation.
2. Notation and Definitions

We draw on notation from two fields: automata theory and logic. Most of our definitions, conventions, notation, etc. are from Hopcroft and Ullman [7] and Monk [12], respectively.

An alphabet is any finite set of symbols and is usually denoted by $\Sigma$. Frequently, we will take $\Sigma$ to mean the binary alphabet \{0,1\}. A string is any finite sequence of elements in $\Sigma$, and $\Sigma^*$ is the set of all strings from $\Sigma$. We write $|x|$ for the length of the string $x$. A language is any subset of $\Sigma^*$. If $R$ and $S$ are languages, then $RS$, $R \cup S$, $R \cap S$, $\neg R$, and $R^*$ stand for $R$ concatenated with $S$, $R$ union $S$, $R$ intersect $S$, the complement of $R$ with respect to $\Sigma^*$, and the closure of $R$ which is the set of all finite strings in $R$. A language is regular if (i) it is a finite set, (ii) it is the union or concatenation of regular sets, or (iii) it is the closure of a regular set. $C^*$, the class of regular languages, is then the closure of the finite sets under union, concatenation, and closure. In fact, $C^*$ is also closed under complementation and intersection.

A finite state automaton (FSA) is a five tuple $M = (Q, \Sigma, \delta, q_0, F)$. $Q$ is a finite set of states, $\Sigma$ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition or next move function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ are the final or accepting states. A move consists of reading an input symbol, changing state according to $\delta$, and advancing the input head. We say that $M$ accepts a string $x$ if $M$, when started in the initial state and reading the first element of $x$, moves into a final state after reading the last symbol of $x$. The set of all strings
accepted by $M$ is called the language recognized by $M$ and is denoted by $L(M)$. A non deterministic finite state automaton (NDPFA) is also a five tuple $M = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times \Sigma \rightarrow 2^Q$. Here, for each state and input symbol, $M$ may have a choice of states to enter next. $M$ accepts $x$ if any sequence of choices leads to an accepting state. Unlike some machines, NDPFA's are no more powerful than FSA's; both recognize exactly the set of regular languages.

Theorem 7 involves a Turing machine. We don't give this definition here because we are more interested in describing an algorithm than a machine. The algorithm is not peculiar to Turing machines; the reader may substitute any of a number of equivalent models of computation.

We use the standard notation of mathematical logic. Lower case Roman letters, $x, y, \ldots$, represent first order variables, and upper case Roman letters, $R, U, \ldots$, represent second order variables. The symbols $\wedge, \lor, \neg, \rightarrow, \exists, \in, \subseteq, \subseteq, \forall, \exists$, and $|=\,$ stand for, respectively, and, or, not, implies, is equivalent to, is a subset of, is a member of, less than or equal to, for all, there exists, and satisfies. We sometimes use $Q$ to stand for either $q$ or $\nu$. In addition, we write $\bigwedge_{i=1}^n a_i$ for $a_1 \wedge \cdots \wedge a_n$ and similarly for $\bigvee$.

Formulas and sentences are denoted by lower case Greek letters, $\sigma, \tau, \phi, \psi, \ldots$, and structures are written as uppercase German letters $\mathcal{A}, \mathcal{B}, \mathcal{L}, \mathcal{U}, \ldots$. The universe of the structure $\mathcal{A}$ is written as $|\mathcal{A}|$, and $\mathcal{A}(U)$ denotes an expansion of $\mathcal{A}$.

Atomic formulas are of the form: (i) monadic predicates $R_x(y)$, $U(y)$, etc., (ii) $x = y$, (iii) $S(x,y)$, the successor relation, also
written as $x + 1 = y$. $\mathcal{L}_{\omega_1^2}$ is the class of monadic second order formulas built from atomic formulas using logical connectives and quantification. $\mathcal{L}_{\omega_1^2}^\Sigma$ (or $\mathcal{L}_{\omega_1^2}^\Pi$) is the subclass of $\mathcal{L}_{\omega_1^2}$ which only allows existential (universal) quantification of second order variables, and $\mathcal{L}_{\omega_1}^1$ is the subclass of first order formulas. $\mathcal{L}_{\nu}^2$ is defined similarly using $x \leq y$ as an atomic formula instead of $S(x,y)$; the same superscripts apply.

We establish an encoding of a string as a structure as follows. For a string $x \in \{0,1\}^*$, we define the structure $\mathcal{L}_x = \langle I_n, S_n, R_x \rangle$ where $n = |x|$, the domain of $\mathcal{L}_x$ is $|\mathcal{L}_x| = I_n = \{1,2,\ldots,n\}$, the successor relation $S_n = \{(1,2), (2,3), \ldots, (n-1,n)\}$, and $R_x = \{i \in I_n : \text{the } i^{\text{th}} \text{ element of } x \text{ is } 1\}$. The structure $\mathcal{L}_x = \langle I_n, \leq, R_x \rangle$, where $\leq$ is the usual relation of less than, is defined similarly. We could, of course, let $x \in \{a_1, \ldots, a_k\}^*$ and make $R_x$ a unary function with range $\{1, \ldots, k\}$. Our results would be the same in this case. If $f$ is a function with $f(n) \geq n$ then we define the augmented structure $\mathcal{L}_x^f = \langle I_n, S_n, R_x^f \rangle$ where $n = |x|$. $R_x^f : I_f(n) \times \{0,1,2\}$ where $R_x^f(i) = \text{the } i^{\text{th}} \text{ element of } x \text{ if } i \leq n$ and $R_x^f(i) = 2$ for $i > n$.

We also define an encoding from monadic relations to strings. Let $\mathcal{L}$ be a structure with the universe $\{1, \ldots, n\}$, i.e., $|\mathcal{L}|$ has a natural linear ordering. If $U_1, \ldots, U_q$ are monadic predicates of $\mathcal{L}$ then we define the encoding $\varepsilon(U_1, \ldots, U_q)$ as a string over $\Sigma \times \cdots \times \Sigma$ (q times) of length n where the $i^{\text{th}}$ element is $(U_1(i), \ldots, U_q(i))$. Of course, we could use any alphabet with at least $2^q$ symbols.
A basic formula is of the form: (i) $x + c = y$ for some constant $c$, (ii) $U(x + c) = 0, 1, d, n,$ or (iii) $\text{card } (T) \geq k$ for a template $T$.

We define $x + c = y$ as $\exists x_1 \cdots \exists x_c : x + 1 = x_1 \wedge x_1 + 1 = x_2 \wedge \cdots \wedge x_{c-1} + 1 = x_c \wedge x_c = y$ for $c > 0$. $U(x + c) = 0$ is defined as $\exists y : x + c = y \wedge \neg U(y)$, and similarly for $U(x + c) = 1$. $U(x + c) = d$ means $\exists y : x + c = y$, and $U(x + c) = n$ stands for $\neg U(x + c) = d$. We frequently abbreviate a conjunction of type (ii) basic formulas such as $\bigwedge_{i=1}^{n} U_j (x + c_i) = b_i$ by $T(x)$, which we call a template. Then, card $(T) \geq k \iff x_1 \cdots x_k \wedge T(x_1) \wedge \cdots \wedge T(x_k) \wedge \left( \bigwedge_{i \neq j} x_i \neq x_j \right)$ means that the template $T$ occurs at least $k$ times. We write $\mathcal{B}$ for the class of boolean combinations of basic formulas.

Finally, theorem 6 uses the following three subclasses of regular languages. $PC = \{ \alpha(\Sigma^* \beta)^k \Sigma^\gamma : k \geq 0$ and $\alpha, \beta, \gamma \in \Sigma^* \}$, which is the class of pattern counting languages. FOS and FOL, which stand for first order successor and first order linear ordering, are $\{ x : \mathcal{O}_x \models \sigma \}$: $\sigma \in \mathcal{L}_1^{\mathcal{O}}$ and $\{ x : \mathcal{O}_x \models \sigma \}$: $\sigma \in \mathcal{L}_1^{\gamma}$. Our set of languages FOL is the same set described in McNaughton and Papert [11].
3. Main Results

The main result we wish to present is the model theoretic characterization of the regular sets. We first need a technical lemma in which we perform elimination of quantifiers. This lemma enables us to characterize first order sentences with successor as boolean combinations of our basic formulas, and it has the immediate corollary that those sentences only generate regular sets. Theorems 3 and 4 then show the equivalence of regular sets, with each theorem providing inclusion in one direction.

1. **Lemma** Given any $\sigma \in \mathcal{L}^1_{\mathcal{E}}$ we can effectively find an equivalent $\psi \in \mathcal{E}$.

**Proof.** We use elimination of quantifiers, and induction on the number of quantifiers and structure of the formula $\sigma$.

The lemma is obvious for atomic formulas, and the induction steps for $\neg, \land, \lor$ are trivial. Thus it is sufficient to assume $\sigma$ has the form, $\exists x \phi$ where $\phi \in \mathcal{E}$. (Clearly, $\forall x \phi \equiv \neg \exists x \neg \phi$ and $\mathcal{E}$ is closed under negation). Suppose $\phi$ has the free variables $x, y_1, \ldots, y_k$, and put $\phi$ in disjunctive normal form. Thus,

$$\phi(x, y_1, \ldots, y_k) = \bigvee_{i} \bigwedge_{j} \alpha_{ij}$$

where each $\alpha_{ij}$ is a basic formula or its negation. The variables of each $\alpha_{ij}$ are among $x, y_1, \ldots, y_k$. Since $\exists x \bigvee_{i} \bigwedge_{j} \alpha_{ij} \equiv \bigvee_{i} \exists x \bigwedge_{j} \alpha_{ij}$, it suffices to consider the case when $\phi = \bigwedge_{j} \alpha_{j}$. Furthermore, since the following are sentences, we may assume no $\alpha_{ij}$ is of their form:

(i) $\text{card}(T) \geq k$ for a template $T$, and $k \geq 0$.

(ii) $T(y_i)$ for $1 \leq i \leq k$.

(iii) $y_i = y_j + c$ for $1 \leq i, j \leq k$. 
Thus, we may assume each $a_{ij}$ has the form:

(i) $T(x)$ for some template $T$

(ii) $x = y_i$ or $x \neq y_i$.

Thus we can write $\phi$ in the following form:

$$\phi = T(x) \wedge (\bigwedge_{i=1}^{n} x = y_{j_i} + c_i) \wedge (\bigwedge_{i=1}^{m} x \neq y_{k_i} + d_i).$$

To eliminate the variable $x$, we have two cases.

Case I. $n \geq 1$. That is, we know $x = y_{j_1} + c_1$, and can eliminate $x$ by simply substituting $y_{j_1} + c_1$ everywhere for $x$. Then,

$$\exists x \phi \equiv \exists x \ (x = y_{j_1} + c_1) \wedge T(x) \wedge (\bigwedge_{i=2}^{n} x = y_{j_i} + c_i) \wedge (\bigwedge_{i=1}^{m} x \neq y_{k_i} = d_i)$$

$$\equiv T(y_{j_1} + c_1) \wedge (\bigwedge_{i=2}^{n} y_{j_1} + c_1 = y_{j_i} + c_i) \wedge (\bigwedge_{i=1}^{m} y_{j_1} + c_1 \neq y_{k_i} + d_i).$$

Case II. $n = 0$. Here, no $a_j$ has the form $x = y_{j_i} + c_i$. We have to insure that the template $T$ occurs at least once outside the set

$$\{y_{k_1} + d_1, \ldots, y_{k_m} + d_m \}.$$ Intuitively, if $T$ occurs $k$ times inside that set, then we require that $T$ occur $k+1$ times overall which will insure that it misses the set at least once. Thus, we see that

$$\exists x \phi \equiv \exists x \ T(x) \wedge (\bigwedge_{i=1}^{m} x \neq y_{k_i} + d_i)$$

$$\begin{cases} \bigvee_{S \subseteq \{1, \ldots, m\}} \left[ (\bigwedge_{i \in S} T(y_{k_i} + d_i)) \wedge (\bigwedge_{i \notin S} \neg T(y_{k_i} + d_i)) \right] & \text{(for } m > 1 \text{)} \\
\wedge \text{card}(T) \geq |S| + 1 & \text{(for } m = 0 \text{).} 
\end{cases}$$

This completes the induction step, and we can effectively find an equivalent quantifier free $\psi \in \mathcal{E}$. 
2. **Corollary** Let $\tau \in \mathcal{L}_{\omega}$. Then the language

$$L_\tau = \{ x \in \Sigma^*: \mathcal{O}_x^\tau \vdash \tau \}$$

is regular.

**Proof.** For a template $T$ and an integer $k \geq 0$, the language

$$\{ x: \mathcal{O}_x^\tau \vdash \text{card}(T) \geq k \}$$

is regular since one can easily construct an FSA to recognize it. To show that $L_\tau$ is regular, perform elimination of quantifiers on $\tau$ as in lemma 1. Thus we obtain an equivalent $\psi \in \mathcal{B}$. Since $\tau$ is a sentence, $\psi$ can be written as

$$\bigvee_i \bigwedge_j \alpha_{ij}$$

where each $\alpha_{ij}$ is of the form $\text{card}(T) \geq k$ or its negation. Thus $L_\tau$ is a boolean combination of regular languages and must be regular itself.

3. **Theorem** Let $\sigma \in \mathcal{L}_{\omega}$. Then $\{ x \in \Sigma^*: \mathcal{O}_x^\sigma \vdash \sigma \}$ is regular.

**Proof.** We can, of course, assume $\sigma$ is in prenex normal form with all second order quantifiers outside the first order quantifiers. See Monk[12]. Thus we can write $\sigma = Q_1U_1 \cdots Q_kU_k^\tau$ where $U_1, \ldots, U_k$ are $\sigma$'s second order variables, $\tau \in \mathcal{L}_{\omega}$, and each $Q_i$ is independently either $\exists$ or $\forall$. Corollary 2 shows that $L = \{ \varepsilon(U_1, \ldots, U_k, R_x) : \mathcal{O}_x(U_1, \ldots, U_k) \vdash \tau \}$ is regular over the alphabet $\Sigma^{k+1}$. We now show $L_\Sigma = \{ \varepsilon(U_1, \ldots, U_{k-1}, R_x) : \mathcal{O}_x(U_1, \ldots, U_{k-1}) \vdash \exists U_k \tau \}$ and $L_\Pi = \{ \varepsilon(U_1, \ldots, U_{k-1}, R_x) : \mathcal{O}_x(U_1, \ldots, U_{k-1}) \vdash \forall U_k \tau \}$ are also regular over the alphabet $\Sigma^k$.

Let $M = (Q, \Sigma^{k+1}, \delta, q_0, F)$ be an FSA that recognizes $L$. Define the NDFA $M' = (Q, \Sigma^k, \delta', q_0, F)$ where

$$\delta'(q, (a_1, \ldots, a_k)) = \bigcup_{b \in \Sigma} \delta(q, (a_1, \ldots, a_k, b))$$

for $q \in Q$ and $(a_1, \ldots, a_k) \in \Sigma^k$. We then claim that $M'$ recognizes $L_\Sigma$. Essentially, $M'$ simulates $M$ while trying all possible assignments for $U_k$ and
accepts if at least one assignment causes $M$ to accept. Thus $M'$ accepts $L_{\Sigma}$ and $L_{\Xi}$ is regular. To show that $L_{\Pi}$ is also regular, we only need observe that $L_{\Pi} = \neg \{ e(U_1, \ldots, U_{k-1}, R_x) : C_x(U_1, \ldots, U_{k-1}, R_x) \rightarrow \top \}$. Regular sets are, of course, closed under complementation. Continuing this way, we see that $\{ e(U_1, \ldots, U_{k-2}, R_x) : C_x(U_1, \ldots, U_{k-2}) \rightarrow Q_{k-1} U_{k-1} Q_k U_k \rightarrow \top \}, \ldots, \{ e(R_x) : C_x \rightarrow Q_1 U_1 \ldots Q_k U_k \rightarrow \top \} = \{ \sigma : C_x \vdash \sigma \} \rightarrow$ is also regular which proves the theorem.

4. **Theorem** \{ \{ \sigma : C_x \vdash \sigma \} \} is exactly the set of regular languages.

**Proof.** Since Theorem 3 shows that \{ \{ \sigma : C_x \vdash \sigma \} \} is contained in the set of regular languages, it suffices to show that every regular set is describable by some $\sigma \in L_{\Sigma}^2$. In fact, we only need $\sigma \in L_{\Sigma}^2$.

Suppose $L_{\Sigma}^*$ is regular, and let $M = (Q, \Sigma, \delta, q_0, F)$ be an FSA that recognizes $L$. Suppose $s = |Q|$, the number of states of $M$, and let $q$ be an integer where $2^q > s$. Define $\sigma = \exists U_1 \ldots \exists U_q \top(U_1, \ldots, U_q)$. Intuitively, $U_1, \ldots, U_q$ represents a sequence of states of $M$, and $\top$ is a first order sentence which states the following:

(i) $M$ is started in the initial state, i.e., $U_1(1), \ldots, U_q(1)$ represents state $q_0$.

(ii) $M$ reaches an accepting state, i.e., $U_1(n), \ldots, U_q(n)$ represents some state $q_n \in F$ where $n = |x|$, the size of $|C_x|$.

(iii) The next move function, $\delta$, is obeyed. That is, for all $z \in |C_x|$ with $z \neq n$, if $U_1(z), \ldots, U_q(z)$ represents state $s_1$ and $U_1(z+1), \ldots, U_q(z+1)$ represents state $s_2$,
then $\delta(s_1, R_x(z)) = s_2$. $R_x(z)$ is the $z^{th}$ element in the string $x$.

Thus, $L = \{x: \forall x \forall \sigma\}$ and we have inclusion in both directions.

The following two theorems summarize the similarities and differences between sentences with successor and less than. Linear ordering is strictly more powerful in first order sentences, but the two have equal power in second order sentences.

5. **Theorem** Let $X \subseteq \{0, 1\}^*$. Then the following are equivalent:

(i) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^2\alpha}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(ii) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Sigma}}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(iii) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Pi}}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(iv) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^2\theta}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(v) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Sigma}}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(vi) There exists a $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Pi}}$ s.t. $X = \{x: \forall x \exists \sigma\}$

(vii) $X$ is regular.

**Proof.** We already know that (i), (ii), and (vii) are equivalent from Theorems 3 and 4. We can easily define successor in terms of linear ordering.

$x + 1 = y \equiv x < y \wedge \neg(\exists z \ x < z \wedge z < y)$.

Thus we see that (i) $\Rightarrow$ (iv) and (ii) $\Rightarrow$ (v). The converses are true because we can also define linear ordering using monadic relations with successor. Thus,

$x < y \equiv \exists Q \neg Q(1) \wedge \neg Q(x) \wedge Q(y) \wedge Q(n) \wedge (\forall z \ Q(z) \Rightarrow Q(z+1))$.

Therefore, (i), (ii), (iv), (v), and (vii) are equivalent.

Finally, we see that (ii) $\Rightarrow$ (iii) and (v) $\Rightarrow$ (vi) which will complete the proof. Suppose $X$ satisfies (ii). Then $X$ is regular and so is $\neg X$. From (ii) there is a $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Sigma}}$ where

$\neg X = \{x: \forall x \exists \sigma\}$. Hence, $X = \{x: \exists x \exists \sigma\}$ and $\sigma \in \mathcal{L}_{\mathcal{Q}^{2\Pi}}$. The
converse is similar, and so is the proof for (v) \( \Leftrightarrow \) (vi).

6. **Theorem**

   (i) FOS = The closure of PC under the boolean operations.

   (ii) FOL = The closure of PC under the boolean operations and concatenation.

**Proof.** (i) Clearly, FOS \( \supseteq \) PC and FOS is closed under boolean operations, so FOS \( \supseteq \) The closure of PC. The converse follows from lemma 1. A proof of (ii) is given in McNaughton and Papert[11].

**Intuitively,** we think of a language in FOS or FOL as being characterized by a finite number of patterns or islands. The added power of linear ordering is precisely the ability to tell if one island is to the left or right of another. Thus, the language \( 0^*10^*110^* \) is representative of the difference between FOL and FOS. Note that neither FOL nor FOS have much expressive power over the strings between the islands. That is, \( 0^*1(00)^*110^* \) is not in FOL.

Lastly, we solve a problem stated in Lynch[10]. Namely, monadic second order sentences with successor are insufficient in characterizing the class P. In fact, these sentences only generate languages recognizable in linear time, a very small portion of P. Binary sentences with successor, however, describe all of NP. Thus, a model theoretic representation of P is presumably somewhere between these two extremes.

7. **Theorem**

   Let \( \sigma \in \mathcal{L}_{x^2} \), and let \( f \) be a polynomial. Then

   \[ \{x : \mathcal{C}_{x}^{f} \models \sigma \} \text{ has deterministic time complexity } O(n) \text{ where } n=|x|. \]

   **Proof.** Define \( L = \{e(R_{x}^{f}) : \mathcal{C}_{x}^{f} \models \sigma(R_{x}^{f}) \} \subseteq \{0, 1, 2\}^*. \) From Corollary 2 we know L is regular. Let \( M = (Q, \Sigma, \delta, q_0, F) \) be an
FSA to recognize $L$.

It is well known that regular sets over a one symbol alphabet are eventually periodic. That is, for each regular $R \subseteq \{0\}^*$, there are constants $n$, $p \in \mathbb{N}$ where $0^x \in R \iff 0^{x+p} \in R$ for all $x \geq n$. Thus, to check if $0^x \in R$ we need only compute $x \mod p$. Therefore, for each state $s \in Q$ of $M$, the strings $2^m$ that will put $M$ in an accepting state when started in state $s$ must be of the form $m \equiv d^s_i \mod c^s_i$, where $s$ is a superscript and $1 \leq i \leq k_s$. That is, for each state $s \in Q$, we can list the pairs $(d^s_i, c^s_i)$ for $1 \leq i \leq k_s$. Then the algorithm to recognize $\{x : O^f_x \models \sigma \}$ is as follows:

Fix $x \in \mathbb{Z}^*$ and let $n = |x|$.

Step 1. Simulate $M$ when given $x$ and end in some state $s$. That is, $\delta(q_0, x) = s$. This step takes $n$ moves on a TM by incorporating $M$ in the machine's finite control.

Step 2. Write the length of $x$ as a binary integer. Thus the length of $n$ is $\lceil \log_2 n \rceil$. Well known algorithms exist for this step in time $O(n)$.

Step 3. Compute $f(n) - n$. Since $f$ is a polynomial, we only have to perform multiplications and additions. Both of these can be computed in time $O(m^2)$ where $m = \text{the length of the numbers to multiply}$. Thus, this step requires $O(\log^2 n)$ moves.

Step 4. Recall that $e_{(R^f_x)} = x2^{f(n)-n}$, and that we should accept $x$ if and only if $f(n) - n \equiv d^s_i \mod c^s_i$ for some $1 \leq i \leq k_s$. This operation requires at most $k_s$ divisions which again can be performed in time $O(\log^2 n)$. 
Finally, if \( f(n) - n \equiv d_i^S \mod c_i^S \) for some \( i \), then we accept \( x \); otherwise we reject. These steps can be performed in time \( O(n) + O(\log^2 n) \) which gives the required time bound of \( O(n) \).

Furthermore, since step 1 can be combined in step 2 and since steps 3 and 4 are asymptotically smaller, step 2 is where the time is spent. Since the time to convert the unary integer \( n \) to \( k \)-ary requires \( n(1 + 1/k + 1/k^2 + \ldots) = nk/(k-1) \) moves, we see that the above problem only requires \( cn \) moves for any \( c > 1 \).
4. Conclusion

We conclude by giving two open problems. The first is a model theoretic characterization of $P$. Theorem 7 shows that $\{x: \mathcal{O}_{\mathcal{F}}^f = \sigma\} \in \mathcal{F}_{\mathcal{M}}^2$ and $f$ is a polynomial) is too small, but generalizing either successor to addition or unary to binary gives the, presumably, too large class $NP[10]$. Thus, we are led to look for something between successor and addition or between unary and binary. This problem is likely to be very difficult because a good solution would probably also solve the $P=NP$ question. On the other hand, a good solution would greatly enhance our understanding of the power of nondeterministic processes. Another idea is to try the spectra of categorical sentences. See Fagin[5].

A second problem is to refine the characterization of $\mathcal{R}$. That is, we would like a correspondence between the complexity of a regular language (number of states in FSA, structure of a regular expression, etc.), and the complexity of a sentence (number and arrangement of quantifiers, etc.). For example, what is the relation between the number of states in the minimal FSA to recognize a given language and the number of quantifiers needed to describe it? A similar problem is to determine the first or second order hierarchical structure. Given a sequence of quantifiers $Q_1 x_1 \cdots Q_r x_r$, characterize the languages describable by a sentence of the form $Q_1 x_1 \cdots Q_r x_r \tau(x_1, \ldots, x_r)$. Also, determine if the obvious inclusion properties are proper. One suggestion is to carefully follow the elimination of quantifiers method to see if it gives a good characterization of languages.
5. References


