LOGIC AND ALGEBRA AS REGULAR BEHAVIOUR

KAMAL LODAYA

1. Expressions to logic

Fix a nonempty finite set A as an *alphabet*. Its elements are called *letters*. A finite sequence of letters $w : \{1, \dots, n\} \to A$, such as $aabab$, is called a *word* over the alphabet. (Some people say a *string*.) A set of words is called a *language*.

The set of all words over A is written A^*. The empty word is written ε. In general, given a language L, its iteration L^* is the language formed by concatenating words from L to form another word. For instance, given the language $\{aa, ab, b\}$, also written $aa \cup ab \cup b$, the word $aabab$ is in $\{aa, ab, b\}^*$, but the word ba is not. The null word ε is always in any L^* (by taking words from L zero times). The set of nonempty words over A is A^+.

This notation, with some ad hoc extensions, can be used to describe languages. We will not look into the details of the notation: these are called *regular expressions*, and are defined in any book on automata theory. We assume that the reader is familiar with such material. With this background, the aim of this article is to examine some subclasses of regular languages which can be characterized using logic as well as algebra. Proofs are barely sketched. The interested reader should try to follow up the material and construct more detailed proofs.

1.1. Expressions.

The *starfree expressions* are those where the iteration L^* is not allowed. The *dot depth* of a starfree expression is the maximum number of nested alternations of the boolean operations ($e_1 \cup e_2$ and $\overline{e}$) and concatenations ($e_1 e_2 \dots e_n$). Since complement is allowed, infinite languages are also describable. Thus A^* can be described as $\overline{\emptyset}$, a starfree expression of dot depth 0. The dot depth 1 expression $\overline{A^*bbbA^*}$ describes the language $NoThreeBs$ of words which do *not* have three consecutive occurrences of the letter b.

Exercise 1. *Give a starfree expression for the language $Cycle = (ab)^*$.*

Exercise 2. *Verify that the language $(ab + ba)^*$ can be described by the dot depth 2 starfree expression $\overline{(\overline{A^*a}\ b(ab)^*\ \overline{aA^*}) \cup (\overline{A^*b}\ (ab)^*a\ \overline{bA^*})}$.*

The language $(aa)^*$ is not starfree. How do we prove this?

1.2. Logic.

A more systematic notation that programmers and computer scientists have come to use is logical formulas. Here is a sentence in first-order logic describing exactly the words in the language $NoThreeBs = \overline{A^*bbbA^*}$:

$$\forall x \forall y \forall z (y = x{+}1 \wedge z = y{+}1 \wedge b(x) \wedge b(y) \supset \neg b(z))$$

Formally, we are working in a structure $\{1, \ldots, n\}$ of *positions* in the word, with pointers indicating the positions of variables (indicated below by underlines), binary predicates for the linear order and the successor ($x < y$ and $y = x{+}1$ can be thought of as binary predicates $less(x, y)$ and $successor(x, y)$), and unary predicate symbols $\{a(.) \mid a \in A\}$ for the letters. Thus the logic has a *signature* with these predicate symbols. Since successor is also definable using the order (how?) and the unary letter predicates are there in all logics on words, it is conventional to abbreviate the signature and call the logic $FO[<]$.

Exercise 3. *Define* $y = x{+}1$ *in* $FO[<]$.

At each position, the unary predicate corresponding to the letter at that position holds, and no other. Pointer functions like $s = [x \mapsto 5, y \mapsto 6]$ below are called "assignments" and written $w, s \models \alpha$ in logic textbooks. A detailed inductive definition is not given here.

$$\begin{array}{l} a\ b\ a\ \underline{b}\ \underline{a}\ b \models y = x{+}1 \wedge b(x) \supset \neg b(y) \\ a\ \underline{b}\ a\ \underline{b}\ a\ b \not\models x < y \wedge b(x) \supset \neg b(y) \\ a\ b\ a\ a\ \underline{b}\ \underline{b} \not\models y = x{+}1 \wedge b(x) \supset \neg b(y) \end{array}$$

More precisely, we put the pointers into the words, expanding the alphabet to $A \times \wp(Var_1)$, where the variables Var_1 are those which appear in the first-order formula, constrained to occur exactly once in the word model.

$$\binom{a}{\emptyset}\binom{b}{\emptyset}\binom{a}{\emptyset}\binom{b}{\{x\}}\binom{a}{\{y\}}\binom{b}{\emptyset} \models y = x{+}1 \wedge b(x) \supset \neg b(y)$$

Position i in the word having the letter $(a, \{x, y\})$ means that in addition to its having the letter a, the variables x and y are assigned the position i. Thus a word becomes a model for a sentence (a formula with no free variables). A language, a set of words, becomes a set of *finite models* for the formula.

Here is a sentence, repeatedly reusing two variables, for $AfterLastB = A^*bc^*d(a+c+d)^*$ over alphabet $A = \{a, b, c, d\}$. Since it is long, it is presented in two steps:

$$\begin{array}{lcl} AfterLastB & = & \exists x(b(x) \wedge \forall y(x < y \supset \neg b(y)) \wedge \exists y(x < y \wedge d(y) \wedge ToLastB(y)))), \\ ToLastB(y) & = & \forall x(x < y \wedge \neg c(x) \supset \exists y(x \leq y \wedge b(y))) \end{array}$$

Exercise 4. *Write a sentence for the language Cycle* $= (ab)^*$. *Use few variables.*

Robert McNaughton and Seymour Papert showed in their 1971 book that what we have seen from the examples generalizes all the way to starfree expressions.

Theorem 5 (McNaughton and Papert)**.** *The starfree languages are* $FO[<]$*-definable.*

Proof. By induction on the starfree expressions we construct a first-order logic sentence whose finite word models define the same language.

The expression $\emptyset$ maps to $false$, the expression a to the two-variable formula $(i = j) \wedge a(i)$. Boolean operations on the expressions translate into the same operations on logical formulas. We are left with concatenation e_1e_2: use the two-variable formula $\exists x \exists y(y = x{+}1 \wedge e_1^{[i,x]} \wedge e_2^{[y,j]})$, where the FO sentences for the expressions e_1, e_2 obtained from the induction hypothesis are *relativized* to intervals. When the empty word is in e_1 or e_2, add disjuncts for e_1e_2 being

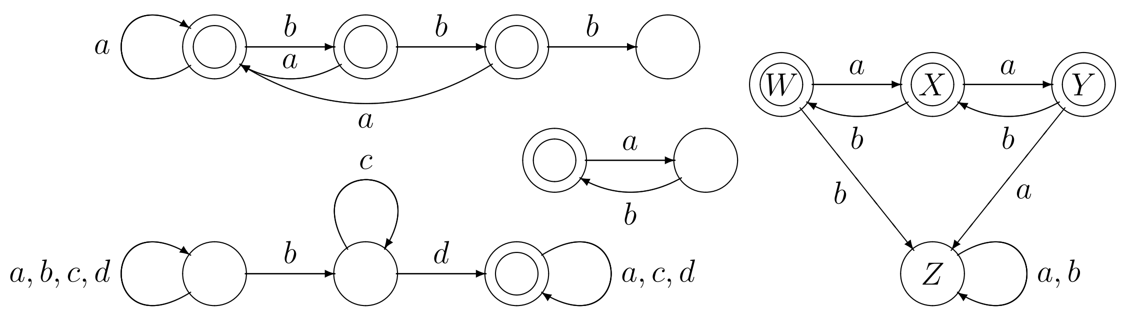

FIGURE 1. Automata for the languages *NoThreeBs*, *AfterLastB*, *Cycle* and *Bicycle*

e_2 or e_1 respectively. Relativization can be defined by induction on the logical formula: for boolean combinations we push the relativization down onto the subformulas. For the quantifier, we have $(\exists x \phi(x))^{[i,j]} = \exists x (i \le x \le j \wedge (\phi(x)^{[i,j]}))$. Finally the endpoints i and j have to be quantified out existentially, when the empty word is in the language an exception is made. □

2. Logic as automata behaviour

Cognitive scientists, like Warren McCulloch and Walter Pitts in the 1940s, invented a graphical notation (called *finite automata*) to represent regular languages.

Formally, a *transition system* is a directed graph (Q, δ), with the edges (called *transitions*) labelled by letters of the alphabet ($\delta \subseteq Q \times A \times Q$) which can be alternately seen as the function $\delta : A \to \wp(Q \times Q)$. The vertices are called *states*. A finite automaton is a finite transition system with a set of *initial states* and a set of *final states* $I, F \subseteq Q$. Figure 1 shows you automata for the languages we have been talking about. The leftmost state of each automaton is its only initial state, and final states are marked by double circles.

Here is how an automaton operates. It begins in an initial state. On each letter of the word, it takes the corresponding transition from the current state into a (possibly) new state. At the end of the word, if the automaton is in a final state, it *accepts* the word. The *language accepted* by the automaton is all words for which there is a sequence of moves from an initial state to a final state.

Exercise 6. *Design an automaton accepting the language Even of all even-length words.*

Does looking at languages using automata help in our attempt to describe them in logic? Yes, here is a sentence for the transition system of the automaton for *Bicycle*:

$$\begin{array}{l}
\exists W \exists X \exists Y \exists Z \\
(\ \forall w \forall x (W(w) \wedge z = w + 1 \supset ((a(w) \supset X(z)) \wedge (b(w) \supset Z(z)))) \\
\wedge \forall x \forall z (X(x) \wedge z = x + 1 \supset ((a(x) \supset Y(z)) \wedge (b(x) \supset W(z)))) \\
\wedge \forall y \forall z (Y(y) \wedge z = x + 1 \supset ((a(y) \supset Z(z)) \wedge (b(y) \supset X(z)))) \\
\wedge \forall z \forall x (Z(z) \wedge x = z + 1 \supset Z(x)) \\
)
\end{array}$$

This is a sentence of *monadic second-order logic*, with the variables $W, X, Y, Z \in Var_2$ ranging over sets of positions. Each such *set variable* stands for a state of the transition system, and is interpreted as being true for the positions of the word at which the automaton, operating on the word, is in that state. A little more is required to fully describe an automaton —for instance, the initial and final states must be used.

Exercise 7. *Given an arbitrary finite automaton, make up a sentence describing the language accepted by it.*

Formally, our word models are now also equipped with an interpretation for the second-order variables. Think of the automaton alphabet as being expanded with a set of first- and second-order variables, specifying at each position the variables which are true there. It is conventional to write this as the signature $MSO[+1]$ because in monadic second-order logic the order relation is definable using successor. Here is a hint. Think of how you would represent the set obtained by taking the image of the transitive closure of a relation.

Exercise 8. *Construct automata on the alphabet $A \times \wp(\{x, y\}) \times \wp(\{X\})$ which check if $a(x)$, $y = x + 1$ and $X(y)$ are true.*

Using these techniques, independently BORIS TRAKHTENBROT in 1958, RICHARD BÜCHI in 1960 and CALVIN ELGOT in 1960, were able to prove a fundamental result:

Theorem 9. *The languages accepted by finite automata are exactly those which can be defined in the monadic second-order theory of the successor relation ($MSO[+1]$) over words.*

Proof. One direction of the proof was sketched above.

For the converse we use induction on the structure of formulas. Finite automata can be constructed for the atomic formulas. From automata theory, we know that regular languages are closed under the boolean operations as well as under projection (how?), which we use for the cases dealing with both the first- and second-order quantifiers. $\square$

A little more is true. Checking satisfiability or validity of monadic second-order logic is reduced to checking the automaton constructed above for nonemptiness or universality.

Corollary 10 (BÜCHI, ELGOT AND TRAKHTENBROT). *The monadic second-order theory of successor $MSO[+1]$ over finite words is decidable.*

3. ALGEBRAIC REPRESENTATION OF AUTOMATA

MICHAEL RABIN AND DANA SCOTT in a 1965 paper showed, using a powerset construction which they derived from JOHN MYHILL, that for finite automata, it is sufficient to consider *deterministic* transition systems where there is a unique a-labelled edge leaving a state q, for every a and q.

First observe that the set of relations $\wp(Q \times Q)$ over the finite set Q is the *transition monoid* of the system, with relation composition being the monoid operation and the identity relation as the unit 1 of the monoid. Recall that a *monoid* is a set with a binary associative operation $\circ$ which has a unit, that is, $m \circ 1 = m = 1 \circ m$.

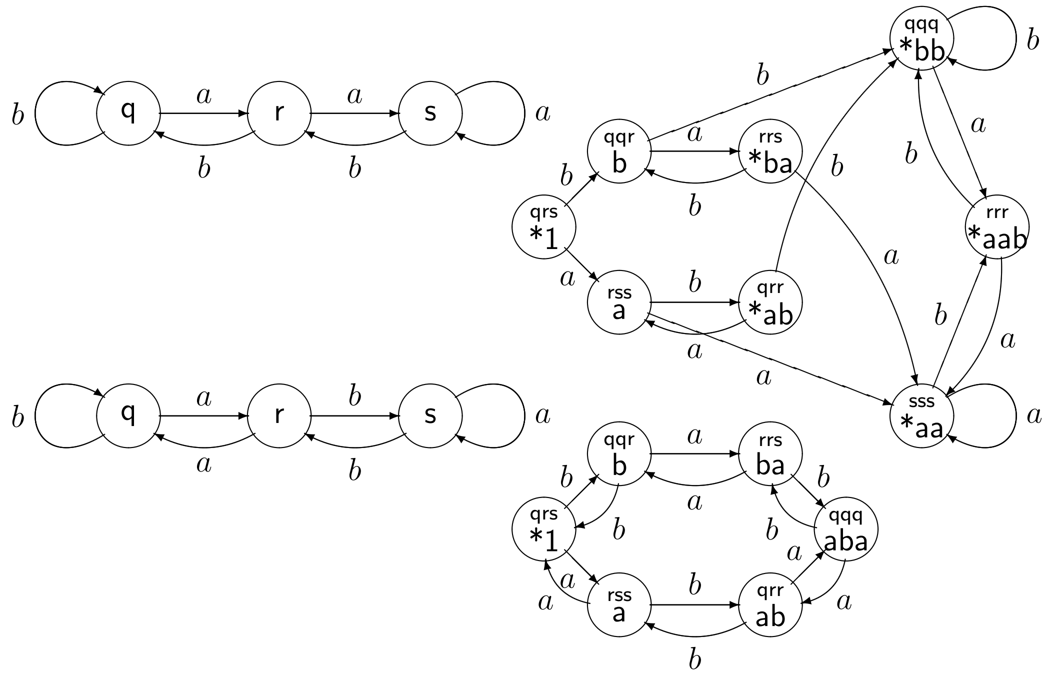

Figure 2. Two transition systems and their transition functions as monoids

Since the set of all words A^* is the free monoid generated by A, the function δ can be extended to a monoid homomorphism $h : A^* \to \wp(Q \times Q)$ using $h(xy) = h(x) \circ h(y)$ and $h(\varepsilon) = 1$. Conventionally $\widehat{\delta}$ is used as a name for the extended morphism h.

In this sense finite monoids can be derived from finite transition systems. Figure 2 (which appeared in McNaughton and Papert's book) shows, for a deterministic transition system with states $Q = \{\mathsf{q,r,s}\}$, its set of transition functions $Q \to Q$. A triple like mnp is the image of $\mathsf{q,r,s}$ respectively. The "transition" on letter a is seen as the right-composition of the "state" (function in the transition monoid) by the function corresponding to a. *Idempotent* elements (functions e such that $e = e \circ e$) are starred.

Exercise 11. *Construct the minimal dfa and transition monoid of the automaton for Cycle. It has eight elements, five of which are idempotent.*

3.1. Congruences of finite index.

Two words x and y are equivalent if they define the same relation $h(x) = h(y)$ on Q. This is a finite-index congruence (called the *kernel* of h) studied by Myhill in the 1950s. Anil Nerode studied the more compact "right" congruences. The relations which map initial states I to final states F in an automaton are a subset of its transition monoid, and the language accepted is $h^{-1}(I \times F)$. The result is:

Theorem 12 (Myhill and Nerode). *The regular languages are exactly those which are inverse images of morphisms into (designated subsets of) finite monoids.*

Proof. The explanation above gives the argument in the forward direction. For the converse, the monoid itself can be used as the set of states and each multiplication $ma = m'$, $a \in A$, gives a deterministic a-labelled transition. 1 is the unique initial state and the designated subset gives the final states. □

The finite monoid is said to *recognize* the corresponding regular language. MYHILL AND NERODE showed that the congruences corresponding to finite automata form a lattice. Therefore, given a regular language L, there is a maximal saturating finite-index congruence for it, and by a quotient construction, a minimal deterministic finite automaton accepting L. Its transition monoid is called the *syntactic* monoid of the language.

3.2. **Inside regular languages.** In this article we will consider a subclass of regular languages, also of automata and of monoids.

Given a finite automaton, the nonempty word $w \in A^+$ is a *counter* if the transition function $\widehat{\delta}(w)$ induces a nontrivial permutation on the states Q. In Figure 1, the word a is a counter in the bottom automaton on the states X, Y, Z, while the word b is a counter on X, Y. An automaton without any counter is *counter-free*, as studied by MCNAUGHTON AND PAPERT. For example, the top automaton in Figure 2 is counter-free. It has cycles, but the cycles are not formed from counters. Recently FABRICE CHEVALIER, VOLKER DIEKERT, DEEPAK D'SOUZA, PAUL GASTIN, RAJ MOHAN AND PAVITHRA PRABHAKAR observed that counter-freeness is preserved across automata.

Lemma 13. *Given a counter-free nondeterministic automaton, there is a counter-free deterministic automaton accepting the same language.*

Proof. Use the Rabin-Scott construction from a counter-free automaton M to a powerset automaton N accepting the same language. Suppose N has a counter $X \overset{w^n}{\Longrightarrow} X$ for some state X, some $w \in A^+$, $n \geq 1$. From the subset construction, for each $q \in X$ from M, we find $p \xrightarrow{w^n} q$ in the automaton M for some $p \in X$. Iterating backward and using the pigeonhole principle on the finite automaton M, we can find some $p \in X$ and positive j, k such that $p \xrightarrow{w^{jn}} p \xrightarrow{w^{kn}} q$ in M. Since M was counter-free, w could not be a nontrivial counter, so $p \xrightarrow{w} p$. Hence p is in $\widehat{\delta}(w^{1+kn})(X)$, which is the same as $\widehat{\delta}(w)(X)$. That is, $X \subseteq \widehat{\delta}(w)(X)$. By induction $\widehat{\delta}(w)(X) \subseteq \widehat{\delta}(w^n)(X) = X$. So $X \overset{w}{\Longrightarrow} X$ in N and N is counter-free. $\square$

Theorem 14 (SCHÜTZENBERGER 1966, MCNAUGHTON-PAPERT 1971)**.** *Given an $FO[<]$ sentence (formula), the (pointed) language defined by it is accepted by a finite counter-free automaton.*

Proof. By induction on $FO[<]$ formulas with free variables V_1, we construct a counter-free automaton over the pointers alphabet $A \times \wp(V_1)$: for $a(x)$, $x = y$ and $x < y$ we directly construct the automata; for $\phi \wedge \psi$ we have a product construction; for $\neg\, \phi$ we assume a deterministic automaton and exchange final and non-final states; for $\exists x \phi$ we project the automaton to the alphabet $A \times \wp(V_1 \setminus \{x\})$ by nondeterministically guessing the position interpreting x. Now use Lemma 13. $\square$

The subclass of monoids required to represent counter-free automata is easy to define. A *group* is a monoid in which every element has an inverse. The bottom automaton in Figure 2 shows that (minimal) transition systems which are groups have a very symmetric structure.

Exercise 15. *Given a finite monoid, how can one find its maximal subgroups?*

A monoid which is not a group can have submonoids which turn out to be groups. For example, any idempotent e defines a trivial subgroup $\{e\}$ since e is its own inverse. This gives us the right definition, of a monoid *which has only trivial subgroups.*

Theorem 16 (Schützenberger 1965)**.** *The transition monoid of a reduced counter-free automaton does not contain any nontrivial subgroup.*

Proof. Let G be a subgroup in the syntactic monoid of the language accepted by the automaton, containing powers of an element $g \in G$. One of those powers must be an idempotent $g^n = (g^n)^2 = g^{2n}$ for $n \geq 1$. Towards a contradiction assume nontriviality of the group, so $g^n \neq g^{n+1}$. By minimality of the automaton there is a word w mapping to g. So for some state q, we find states $\widehat{\delta}(w^n)(q) = r = \widehat{\delta}(w^{2n})(q) = \widehat{\delta}(w^n)(r)$ and by supposition, $r \neq \widehat{\delta}(w)(r)$. Hence w is a counter which permutes the states $\{r, \widehat{\delta}(w)(r), \ldots, \widehat{\delta}(w^{n-1})(r)\}$. □

3.3. **Starfree languages.** Using a theory of monoid ideals from a paper by James Green in 1951, Schützenberger in 1965 proved the remaining result which characterizes regular languages corresponding to first-order logic. The theory of monoid ideals will not be given here. A second proof of this theorem, based on the theory of implementing an automaton by a product of simpler automata (developed by Kenneth Krohn and John Rhodes in 1965), was first published by Albert Meyer in 1969. This theory is not given here.

Thomas Wilke came up with a third proof, published in 1999. His proof is described by Howard Straubing and Pascal Weil in an introductory article, where there is more material on automata, logic and algebra.

Theorem 17 (Schützenberger)**.** *If the syntactic monoid of L has only trivial subgroups, then the language recognized is starfree.*

Exercise 18. *Prove that there are no sentences of $FO[<]$ defining languages $(aa)^*$ and Even.*

Corollary 19. *There is an algorithm to check, given a regular language, whether it is definable using a sentence of first-order logic $FO[<]$.*

Proof. The converse of Theorem 17 is by using the implications in Theorems 5,14,16. All the implications therefore become equivalences. Now use Exercise 15. □

The Institute of Mathematical Sciences, C.I.T. Campus, Chennai 600 113, India.