Relating logic to formal languages

Kamal Lodaya

The Institute of Mathematical Sciences, Chennai

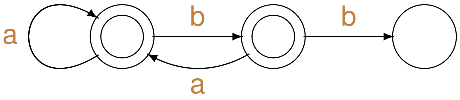
October 2018

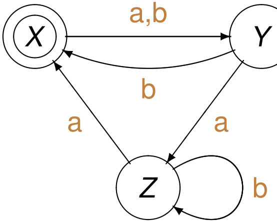
Reading

1. Howard Straubing: *Formal languages, finite automata and circuit complexity*, BIRKHÄUSER.
2. Wolfgang Thomas: Languages, automata and logic, in *Handbook of formal language theory III*, SPRINGER.
3. Pascal Tesson and Denis Thérien: Logic meets algebra: the case of regular languages, LMCS.

Finite automata (McCulloch and Pitts 1943)

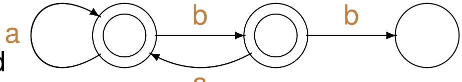
- Fix finite alphabet A
- $M = (Q, I, F, \delta)$
- Finite set of states Q
  Initial states $I \subseteq Q$
  Final states $F \subseteq Q$
- "Nondeterministic" transition relation $\delta(a) \subseteq Q \times Q,\ \ a \in A$
- M accepts a word in A^*
- The set of words accepted is its language



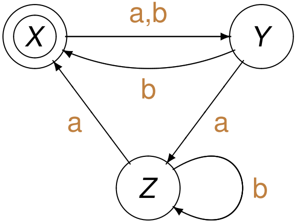


Algebra (Myhill 1957)

- Binary relations $\wp(Q \times Q)$ form a finite monoid $(M, \circ)$ under composition, generated by the A-labelled transitions using $\delta(w) \circ \delta(x) = \delta(wx)$
- Morphism $\delta : A^* \to \wp(Q \times Q)$ from A^* (the free monoid on A) to a finite monoid
- M accepts a word w when $\delta(w) \cap (I \times F) \neq \emptyset$ (language is inverse image of a finite subset of the finite monoid)
- Congruence $\equiv \subseteq A^* \times A^*$
  $w \equiv x \iff \delta(w) = \delta(x)$



$A \circ a = a, \; b \circ b = 0$



$a \circ a \circ a = 1, \; b \circ b = 1$

(symmetric group S_3)

The automata-algebra connection (Myhill-Nerode 1950s)

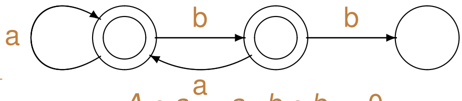
Theorem (Myhill 1957, Nerode 1958, Rabin-Scott 1959)
Nondeterministic finite automata, finite monoids and deterministic finite automata are equivalent.

1. *Given a finite automaton (possibly nondeterministic), the language accepted by it is an inverse image of a subset of a finite monoid (those relations which take an initial state to a final state), called its transition monoid*
2. *Given a finite monoid M with distinguished set of elements D, $(M, 1, \{\_ \circ a \mid a \in A\}, D)$ is a deterministic finite automaton (transition function instead of relation)*

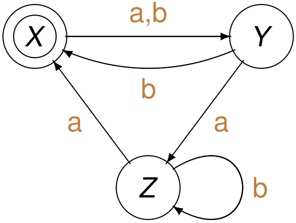
For a language, the syntactic monoid is the transition monoid of the minimal deterministic finite automaton for that language

Counter-free automata (McNaughton-Papert 1971)

- Given a finite automaton, the nonempty word $w \in A^+$ is a counter if $\delta(w)$ induces a nontrivial permutation on Q
- The word a is a counter in the bottom automaton on the states X, Y, Z, the word b is a counter on X, Y
- An automaton without any counter is counter-free, for example, the top automaton



$A \circ a = a, \ b \circ b = 0$



$a \circ a \circ a = 1, \ b \circ b = 1$
(symmetric group S_3)

The transition monoid of a counter-free automaton does not contain any nontrivial subgroup.

Partially ordered automata (Meyer-Thompson 1969)

- In a partially ordered automaton, the states $(Q, \geq)$ are partially ordered
- A transition from state q can only go to states r such that $q \geq r$
- Hence the only cycles allowed are self-loops where $q = r$, falling transitions where $q > r$ cannot climb back
- (Schwentick-Thérien-Vollmer 2002) A partially ordered two-way deterministic automaton can accept more languages (e.g. checking the k'th last letter of the word)
- Partially ordered two-way automata are counter-free

The transition monoid of a partially ordered two-way deterministic automaton is in DA, defined as: if there is an idempotent element in a D-class, then the entire D-class is idempotents (this separates the self-loops from the falling transitions)

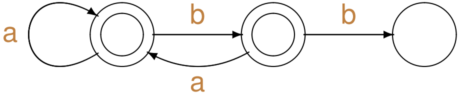
Logic on words (Büchi 1960)

$$FO \quad ::= \quad a(x) \mid x = y \mid x < y \mid Suc(x,y) \mid \neg\phi \mid \phi \vee \psi \mid \exists x \phi, \;\; x, y \in Var_1$$

- Formulas are interpreted over words with pointers indicating the positions of variables
  $ababab \models Suc(x,y) \wedge b(x) \supset \neg b(y)$
  $ababab \not\models x < y \wedge b(x) \supset \neg b(y)$
  $abaabb \not\models Suc(x,y) \wedge b(x) \supset \neg b(y)$
- Pointer functions like $s = [x \mapsto 5, y \mapsto 6]$ above are called "assignments" and written $w, s \models \phi$ in logic textbooks
- Formally one can use a "pointers" alphabet $A \times \wp(V_1)$ where each variable is constrained to occur exactly once in the word model. For the first formula above:
  $\binom{a}{\emptyset}\binom{b}{\emptyset}\binom{a}{\emptyset}\binom{b}{\{x\}}\binom{a}{\{y\}}\binom{b}{\emptyset} \models Suc(x,y) \wedge b(x) \supset \neg b(y)$

Sentences

$$FO \quad ::= \quad a(x) \mid x = y \mid x < y \mid Suc(x, y) \mid$$
$$\neg\phi \mid \phi \vee \psi \mid$$
$$\exists x \phi, \;\; x, y \in Var_1$$



- A sentence is a formula with no free variables, all variables are bound to quantifiers
- The language $\{w \mid w \models \forall x \forall y (Suc(x, y) \wedge b(x) \supset \neg b(y))\}$ of words where the sentence holds is that accepted by the top automaton; the sentence defines the language $\overline{A^*bbA^*}$
- Sentence $\exists x \exists y (Suc(x, y) \wedge b(x) \wedge b(y))$ defines A^*bbA^*

Sentences define languages

- Let $max(x) = \neg \exists y Suc(x, y)$ be an abbreviation, similarly define $min(x)$, etc.
- $\forall x[a(x) \supset \exists y(Suc(x, y) \wedge b(y)) \wedge$ $(b(x) \wedge \neg max(x) \supset \exists y(Suc(x, y) \wedge a(y)))]$ defines $(A \setminus \{a, b\})^*((ab)^* \cup b(ab)^*)$
- Adding conjuncts $min(x) \supset a(x)$ and $max(x) \supset b(x)$ to the previous sentence defines $(ab)^*$
- $\forall x \forall y[(min(x) \supset a(x)) \wedge (max(x) \supset b(x)) \wedge$ $Suc(x, y) \supset (b(x) \supset \neg b(y)) \wedge (a(x) \supset \neg a(y))]$ defines:
  Over the alphabet {a,b}, the language $(ab)^*$
  Over the alphabet {a,b,c}, the language $c^*(ac^*bc^*)^*$

Formulas define pointed languages

Let A be {a,b,c}; a truth checking procedure is outlined below:

1. We have $uav \models a(x)$
2. Since u,v are arbitrary, the formula $a(x)$ defines the pointed language $A^*\mathrm{a}A^*$
3. Similarly $uav(\mathrm{b} \cup \mathrm{c})w \models \alpha(x,y) \stackrel{\mathrm{def}}{=} x < y \supset \neg a(y)$
4. So $\forall y\, \alpha(x,y)$ defines $A^*\mathrm{a}(b \cup c)^*$
5. Again $t\mathrm{a}u\mathrm{b}v\mathrm{c}w \models \beta(x,y,z) \stackrel{\mathrm{def}}{=} x < y < z \supset b(y)$
6. So $\forall y\, \beta(x,y,z)$ defines $A^*\mathrm{a}b^*\mathrm{c}A^*$
7. Hence $\exists z(c(z) \wedge \forall y\, \beta(x,y,z)))$ defines $A^*\mathrm{a}b^*cA^*$
8. Since b^*c is included in $(b \cup c)^*$ (and not conversely), the intersection $A^*ab^*c(b \cup c)^* = A^*a(b \cup c)^* \cap A^*a(b^*cA^*)$ of 4 and 7 is definable in $\Sigma_2[<]$ by $\exists x(a(x) \wedge \forall y(x < y \supset \neg a(y)) \wedge \exists z(c(z) \wedge \forall y(x < y < z \supset b(y))))$

FO^2 logic to partially ordered two-way automata

Theorem (Schwentick-Thérien-Vollmer 2002)

Given an FO^2 sentence (formula), the (pointed) language defined by it is accepted by a finite partially ordered two-way automaton.

Proof.

For FO^2 formulas with free variables V_1, we construct an automaton over the alphabet $A \times \wp(V_1)$: for $a(x)$, we have an edge; for $\neg\, \phi$ we exchange final and non-final states; for $\phi \wedge \psi$ we have a product construction. All done using partially ordered one-way automata.

There are only two variables, so one can have $\exists y > x(a(y) \wedge \phi)$ or $\exists y < x(a(y) \wedge \phi)$. These determine instructions to go forward and/or backward on the word looking for letters of the alphabet on a self-loop. Finding the position y one falls down the partial order. This can be done by partially ordered two-way automata. Boolean operations now done by satisfying each formula in sequence. $\square$

FO logic to counter-free automata

Theorem (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. □

Corollary
$\{w \mid |w| \equiv 0 \mod q,\ q \geq 2\}$ *and* $(aaa)^*$ *are not FO-definable.*

Because their syntactic monoids contain subgroups Z_q and Z_3.

More logics on words

$MSO ::= (FO \text{ and })\ x \in Y \mid \exists Y\phi,\ x, y \in Var_1,\ \ Y \in Var_2$
$FO ::= a(x) \mid x = y \mid Suc(x, y) \mid x < y \mid \neg\, \phi \mid \phi\ \vee\ \psi \mid \exists x\phi,\ x, y \in Var_1$

- The MSO sentence
  $\exists O \exists E \forall x[a(x) \wedge (min(x) \supset x \in O) \wedge (max(x) \supset x \in E) \wedge$
  $\forall y((x \in O \wedge Suc(x, y) \supset y \in E) \wedge (x \in E \wedge Suc(x, y) \supset y \in O))]$
  defines the language $(aa)^*$ which is not FO-definable
- An FO sentence is $\Sigma_r[<]/\Pi_r[<]$ if it has r alternating blocks of quantifiers, with first block existential/universal
- $\Delta_r[<]$ is the class of languages which are definable by both $\Sigma_r[<]$ and $\Pi_r[<]$ sentences ($\Sigma_r[<] \cap \Pi_r[<]$ languages)
- An MSO sentence is $MQ^1_s\text{-}q^0_r$ if it has s alternating blocks of set quantifiers, followed by r alternating blocks of first-order quantifiers (sentence above is $M\Sigma^1_1\text{-}\Pi^0_1[<]$)

MSO logic to finite automata

Theorem (Büchi 1960, Elgot 1961, Trakhtenbrot 1962)

Given an MSO sentence (formula), the (pointed) language defined by it is accepted by a finite automaton.

Proof.

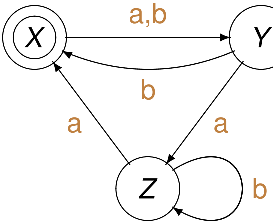
Extending the proof for *FO* formulas, with free first-order variables V_1 and free set variables V_2, we construct an automaton over the extended pointers alphabet $A \times \wp(V_1) \times \wp(V_2)$: for the atomic formula $x \in Y$, we have a direct construction and for the set quantifier $\exists Y \phi$ we again do a projection by nondeterministically guessing the positions which are labelled Y. □

- As there can be many such positions labelled Y, there is no guarantee that the construction is counter-free.
- For the “even-length words” *MSO* sentence, a nontrivial cycle is introduced around an E-state (and an O-state).

The automata-logic connection (Büchi-Elgot-Trakhtenbrot)

Theorem

Given a finite automaton, the language accepted by it is defined by an $M\Sigma^1_1$-$\Pi^0_1[<]$ sentence.



$\exists X \exists Y \exists Z$ “state positions”

$[\forall x \forall y (x \in X \wedge Suc(x, y) \supset y \in Y)$

$\wedge \forall y \forall z (y \in Y \wedge Suc(y, z) \supset$

$\quad ((b(y) \supset z \in X) \wedge (a(y) \supset z \in Z)))$

$\wedge \forall z \forall x (z \in Z \wedge Suc(z, x) \supset$

$\quad ((b(z) \supset x \in Z) \wedge (a(z) \supset x \in X)))$

$\wedge \forall y \forall z (y \in Y \wedge \neg Suc(y, z) \supset b(y))$ “goes to final state X”

$\wedge \forall z \forall x (z \in Z \wedge \neg Suc(z, x) \supset a(z))$ “goes to final state X”

$\wedge \forall x ((X(x) \oplus Y(x) \oplus Z(x)) \wedge (a(x) \oplus b(x)))$ “unique state/letter”

$]$

By closure under complement, on finite words $MSO = M\Delta^1_1$.

Starfree expressions

Starfree expressions $e ::= \emptyset \mid a \in A \mid e_1 e_2 \mid e_1 \cup e_2 \mid \overline{e_1}$
Starfree expressions are defined as $\emptyset, \{a \mid a \in A\}$, corresponding to the empty and singleton languages, and taking the closure under the operations $e_1 e_2$ (concatenation), $e_1 \cup e_2$ (union) and $\overline{e_1}$ (complement). (Avoiding the empty word.)

Regular expressions $e ::= \emptyset \mid a \in A \mid e_1 e_2 \mid e_1 \cup e_2 \mid e_1^+$
The regular expressions are obtained by closing the starfree expressions under the operation star (iteration of concatenation). Here we use plus. The corresponding language is $\{w_1 \dots w_n \mid w_i \in L(e_1),\ 1 \leq i \leq n\}$.

Theorem (Kleene 1956)

Regular expressions define exactly the languages accepted by finite automata.

Dot-depth hierarchy (Brzozowksi-Knast-Thomas)

Starfree expressions $e ::= \emptyset \mid a \in A \mid e_1 e_2 \mid e_1 \cup e_2 \mid \overline{e_1}$

- The empty language $\emptyset$ and its complement $\overline{\emptyset}$ (which is A^+) are dot-depth 0 expressions
- Closing dot-depth r expressions under concatenation and then boolean operations gives dot-depth $r + 1$ expressions

Theorem (McNaughton-Papert 1971)

The dot-depth r languages are $\mathcal{B}_r[<, min, max, Suc]$-definable. Hence the starfree languages are FO-definable.

- $\emptyset \mapsto false$, $\quad a \mapsto (min = max) \wedge a(min)$
- $e_1 e_2 \mapsto \exists x(e_1^{[min,x]} \wedge e_2^{[Suc(x),max]})$. Here an FO sentence is relativized to an interval, e.g. $a(x)^{[i,j]} = i \leq x \leq j \supset a(x)$ and $(\exists x \phi(x))^{[i,j]} = \exists x(i \leq x \leq j \wedge (\phi(x)^{[i,j]}))$
- The positions min and max at the beginning and end of a word, the successor function $Suc(x)$ can be defined in FO

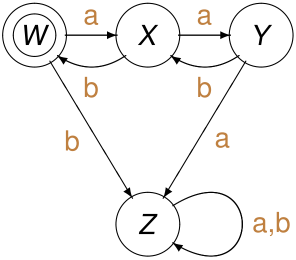
Dot-depth hierarchy (Brzozowksi-Knast-Thomas)

Suppose the alphabet A has at least two letters.

Theorem (Brzozowski-Knast 1978)

The dot-depth hierarchy is infinite: $\mathcal{B}_0[<] \subset \mathcal{B}_1[<] \subset \mathcal{B}_2[<] \subset \dots$

Let *Cycle*$_r$ be the language of all words w having an equal number of a's and b's, such that for all prefixes v of w, the number of b's is at most the number of a's, the number of a's is greater than the number of b's by at most r.



Automaton for *Cycle*$_2$

There is a $\mathcal{B}_{r+1}[<]$-sentence defining the language *Cycle*$_r$. There is no $\mathcal{B}_r[<]$-sentence which defines *Cycle*$_r$.

Algebra-expression connection for *FO*(Schützenberger)

Theorem (Schützenberger 1965)

The language recognized by a finite group-free monoid is starfree. Hence counter-free automata can only accept starfree languages.

- The two-sided ideals $MmM = \{nmp \mid n, p \in M\}$ in the finite monoid M (with $h : A^* \to M$), are partially ordered by inclusion
- The absence of a nontrivial subgroup means that the intersection of a right ideal $mM = \{mp \mid p \in M\}$ and a left ideal $Mm = \{nm \mid n \in M\}$ of the monoid M is at most a singleton
- Using this one can build a starfree expression for the inverse $h^{-1}(m)$ of every singleton by an induction on the ideal order
- For the automaton for the language $(ab \cup ba)^*$—not obviously starfree— Schützenberger's proof yields the starfree expression $\overline{(\overline{A^*a}\ b(ab)^*\ \overline{aA^*}) \cup (\overline{A^*b}\ (ab)^*a\ \overline{bA^*})}$, where the language $(ab)^*$ is described by the expression $aA^* \cap A^*b \cap \overline{A^*(aa \cup bb)A^*}$

Algebra-logic connection for FO^2

Theorem (Schütz.1976, Schwentick-Thérien-Vollmer 2002)

The language recognized by a finite monoid in DA is unambiguous starfree. Hence partially ordered two-way automata can only accept unambiguous languages, which are definable in FO^2.

- The left and right ideals Mm, mM in the finite monoid with $h : A^* \to M$, are partially ordered by inclusion
- A word u can be factorized $u_0 a_1 \dots a_t u_t$ where each u_i stays in an R-class, and each a_i moves to a new R-class for $a_i u_i$
- Each u_i maps to an idempotent and each a_{i+1} takes one to the next D-class, a_{i+1} has to use a letter not in u_i
- A word $v = v_0 a_1 \dots a_t v_t$, where every pair $h(u_i) = h(v_i)$ maps to the same element, must map to the same $h(u) = h(v)$
- A symmetric result holds for right-to-left factorizations, giving an unambiguous expression $A_0^* a_1 \dots a_t A_t^*$, $A_i \subseteq A$
- Boolean combination of expressions can be written in FO^2