UNIVERSITÀ DEGLI STUDI DI UDINE

Scuola Superiore

End year thesis

An algebraic approach to formal languages

SUPERADVISOR:
Prof. Fabio Alessi
TUTOR:
Prof. Pietro Corvaja

STUDENT:
Mignani Michele

Accademic year 2022/2023

Contents

Introduction

The main goal of this work is to present a connection between two different worlds. The first one is one very familiar to mathematics and regards algebraic structures; the second one is, instead, one of the most foundational topics in theoretical informatics and deals with formal languages.

Moved by the concept, inspired by the Latin etymology, that intelligence means to build links between things, we found the connection we will talk about really interesting, and maybe it can suggest to us that how we face a mathematical problem, regardless of the specific features of the question itself, is almost the same.

In particular, after giving a very short mathematical introduction to the topics that in general are not studied in a mathematical background, we try to establish this aforementioned bridge in two very special cases: regular languages (Chapter 2) and star-free languages (Chapter 3). In doing this, we let a little digression to extend these results to logic, too. Namely, in these examples (and in many others that we will not quote in the following) the good interplaying between formal languages, logics, and algebraic structures allows in many cases to solve a problem regarding one of these fields with tools that belong to another one. In other cases, this is also the hope (see Descriptive Complexity Theory).

Motivated by these examples, we move towards a generalization of the previous results. To do so, in Chapter 4, we will develop some categorical tools that will be useful to handle the very generical objects that we will build and to define in a more precise sense what is and what means the connection we talked about before.

1 Very short mathematical introduction

The automata are one of the first objects studied in theoretical informatics. They can be seen as prototypes of Turing machines, in which only a few operations are allowed.

Definition 1.1. A (non-deterministic) *automaton* $M = (\Sigma, Q, q_0, F, \delta)$ is the given of:

- a set (called *alphabet*) of symbols (called *letters*) Σ;
- a set of *states* Q;
- a *initial state* $q_0 \in Q$;
- a set of *final states* $F \subseteq Q$;
- a *transition function* $\delta : Q \times \Sigma \to Q$.

The automaton is called *finite* if Q is finite.

Informally we can think of an automaton like a labeled graph in which:

- the nodes are the states in Q;
- every edge between two nodes is labeled with a letter of the alphabet Σ and between two nodes $q, q' \in Q$ there is the edge labeled with $a \in \Sigma$ if and only if $\delta(q, a) = q'$.

This should explain why δ is called the *transition* function: in the case of the example above we will say that from q we can go to q' via a.

One of the approaches used to study the power of an automaton deals with linking an automaton $M = (\Sigma, Q, q_0, F, \delta)$ to a *formal language* of Σ, i.e. a $L \subseteq \Sigma^*$[1]. Namely, given an automaton M like above, we say that it *reads a word* $w = a_1 \dots a_n \in \Sigma^*$ if and only if there is a path through the nodes $q_0, \dots, q_n$ such that:

- q_0 is an initial state;
- q_n is a final state;
- for every $i = 1, \dots, n$, there is an edge labelled with a_i from q_{i-1} to q_i, i.e. $\delta(q_{i-1}, a_i) = q_i$. In this case, we say that L is *recognized by the automaton* M.

[1]From now on, given a generic set A, with A^* we mean the free monoid built with the elements of A

Definition 1.2. An automaton $M = (\Sigma, Q, q_0, F, \delta)$ *recognizes* a language $L \subseteq \Sigma^*$ if and only if

$$L = \{w \in \Sigma^* \mid M \text{ reads } w\}.$$

From now on, we will assume that the alphabet Σ is finite.

Actually, even if we give the definition only in the case in which M is an automaton, the relation between a computing machine and a formal language is more generic: for instance, the definition is still valid if we modify the power of M (allowing it to make more\less or different operations).

Historically, the automata were the first way to understand languages: over time, other different strategies were born. We are interested in two of them, the ones that deal with logic and algebra.

For what we need[2], a *vocabolary* σ can be seen as a set of relation symbols and a *logic over* σ as a set of instructions to build well-defined formulas (syntax) and a way to understand what "this formula is true"means (semantics). We hope that for the non-expert reader, two examples are sufficient to get used to it. From now on, we deal only with logics without individual parameters and constants.

Example 1.1.

- In the first-order logic (FO) we have a countable number of first-order variables, usually marked with $x, y, \dots$. These and only these are well-defined formulas:
  - the one of the form $R(x_1, \dots, x_n)$, where R is a n-ary relation symbol in σ;
  - the one of the form $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, $\neg\phi$, $\exists x\phi$, $\forall x\phi$, with ϕ and ψ are well-defined formulas.
- In the second-order logic (SO) we have a countable number of first-order variables $(x, y, \dots)$ and a countable number of second-order variables $(X, Y, \dots)$. These and only these are well-defined formulas:
  - the one of the form $R(x_1, \dots, x_n, X_1, \dots, X_m)$, where R is a $n+m$-ary relation symbol in σ that take in input n first order variables and m second-order variables;
  - the one of the form $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$, $\neg\phi$, $\exists x\phi$, $\forall x\phi$, with ϕ and ψ are well-defined formulas. In addition to these, also formulas like $\exists X\phi$ and $\forall X\phi$ are well-defined formulas if ϕ is.

[2]This is far away from being an exhaustive definition of a logic and of the concept of capture. To not be too long-winded with this stuff, we decide to give a definition complete for the goals of this work.

The world in which logical formulas will be true or false will be the words over an alphabet Σ[3]. Without being meticulous, we give the relevant examples for this work regarding the concept of truth of formulas. The reader should keep in mind that a word can be seen as an ordered set with positions labeled with letters.

Example 1.2. • We consider the first order logic over the vocabulary $\sigma = \{\leq\} \cup \{\bar{a}\}_{a\in\Sigma}$, where $\leq$ is a binary relation and a is unary for all $\bar{a} \in \Sigma$. Informally, the variables refer to the positions, $x \leq y$ means that the position x is before the position y, $\bar{a}(x)$ means that in the position x we have a. More formally, given a word w long m, a formula $\phi(x_1, \ldots, x_n)$ and n positions in the word w, marked with number $i_1, \ldots, i_n$ less or equal to m, we say that $w, (i_1, \ldots, i_n)$ satisfies $\phi(x_1, \ldots, x_n)$ if and only if:

  - if $\phi(x_1, \ldots, x_n)$ is $x_j \leq x_k$ and $i_j \leq i_k$;
  - if $\phi(x_1, \ldots, x_n)$ is $\bar{a}(x_j)$ and in the position i_j there is an a;
  - if $\phi(x_1, \ldots, x_n)$ is conjunction [disjunction] of two formulas and w satisfies both [at least one of] the formulas;
  - if $\phi(x_1, \ldots, x_n)$ is the negation of a formula and w doesn't satisfies the latter;
  - if $\phi(x_1, \ldots, x_n)$ is the implication of two formulas and w satisfies the consequence or doesn't satisfy the premise;
  - if $\phi(x_1, \ldots, x_n)$ is $\exists x\psi(x_1, \ldots, x_n, x)$ $[\forall x\psi(x_1, \ldots, x_n, x)]$ and there is a position x such that [for every position x] $w, (i_1, \ldots, i_n, x)$ satisfies $\psi(x_1, \ldots, x_n, x)$.

  When is the case that $w, (i_1, \ldots, i_n)$ satisfies $\phi(x_1, \ldots, x_n)$ we say also that $\phi(x_1, \ldots, x_n)$ holds, is valid or is true in w; in symbol we write $w, (i_1, \ldots, i_n) \vDash \phi(x_1, \ldots, x_n)$.

- Dealing with the monadic second-order logic (MSO), whose syntax is the same as the second-order logic, we only have to add some consideration to what is already explained above:

  - the vocabulary has in addition the binary membership symbol $\in$, with $x \in X$ means that x belongs to X. In MSO X is informally interpreted as a subset of the set of positions;

[3]In logical terms, words will be the class of models that we will consider.

– if $\phi(x_1, \ldots, x_n, X_1, \ldots, X_m)$ is a formula, now the concept of satisfiability is referred to a word w, n of its positions $i_1, \ldots, i_n$ and m subsets of its positions $I_1, \ldots, I_m$. The reader should not find it difficult to extend the considerations made for the first-order logic to the new relation symbol $\in$ and the quantifiers over second-order variables.

We have now all to understand the following definitions that will be central in the next section, where they will assume a more specific meaning (after having specified the logic for Definition 1.3 and the algebraic structure for Definition 1.4).

Definition 1.3. Let Σ be an alphabet and $L \subseteq \Sigma^*$ one of its languages. Given a statement (= formula without free variables) ϕ of a suitable[4] logic, we say that L is *recognised* or *definable* by ϕ if and only if

$$L = \{w \in \Sigma^* \mid w \vDash \phi\}.$$

The language L is *definable in a logic* if there is a formula ϕ of this logic that defines it.

We end this section, as promised, with another way to go deep in the formal languages theory. This time the main character will be the algebraic structure of Σ^*.

Definition 1.4. Let Σ be an alphabet and $L \subseteq \Sigma^*$ one of its languages. Given a suitable[5] algebraic structure M (monoid, semigroup, etc.), a homomorphism $h : \Sigma^* \to M$ that preserves this structure and a set $F \subset M$ of accepting elements, we say that L is *recognised* by h if and only if

$$L = \{w \in \Sigma^* \mid h(w) \in F\}.$$

The language L is *recognized by a given algebraic structure* if it is recognized by some homomorphism into one set with this algebraic structure and some accepting subset of it.

[4]We must give a semantic meaning to all the formulas of this logic.

[5]As the reader can see later in the definition, also Σ^* has to possess this algebraic structure.

2 Example 1: Regular languages

We begin our investigation of the role of logic and algebra in the formal theory of language with the example of regular languages.

Definition 2.1. A language $L \subseteq \Sigma^*$ is called *regular* if it's recognized by a finite automaton M.

This "computational concept" has an algebraic analogous of the form of Definition 1.4, where the algebraic structure that we consider is the monoid. Indeed,

Theorem 2.1. *Given a language $L \subseteq \Sigma^*$, the following are equivalent:*

1) L is regular;

2) L is recognized by a finite monoid.

Proof.

1) $\to$ 2) Let $M = (\Sigma, Q, q_0, F, \delta)$ one of the finite automaton that recognizes L. The set $\mathscr{P}(Q^2)$ (that is finite) can be equipped with this monoid structure:

- If $A, B \subseteq Q^2$, then

$$A \circ B = \{(p, q) \mid \text{exists } r \in Q \text{ s.t. } (p, r) \in A \text{ and } (r, q) \in B\};$$

- The neutral element is the diagonal subset of Q^2.

Consider the function $h : \Sigma^* \to \mathscr{P}(Q^2)$ that maps a word w into all the pairs (p, q) for which there is a path from p to q in which can be read the word w. Equipping Σ^* with the structure of free monoid, we see that h is a homomorphism. The language L is composed of the word w for which $h(w)$ is a set that contains at least one pair of the form (q_0, q), where q_0 is the initial state and q is an accepting one.

2) $\to$ 1) Let N be a finite monoid that recognizes, via the homomorphism h, the language L. Then we can construct the following automaton:

- the set Q of the states is the set of all the elements of N;
- the initial state is the identity;
- the set of final state is F;

– from the state n, reading the letter a, I can reach the state $nh(a)$, i.e. the transition function δ is such that $\delta(n, a) = nh(a)$.

For being the initial state the identity, after having read a letter $a \in \Sigma$, the state a is reached and after having read a word w, the state $h(w)$ is the one reached. Hence, a word $w \in L$ is read by this automaton, for being $h(w)$ a final state. Conversely, if a word $w \in \Sigma^*$ is read, it means that $h(w)$ is a final state and then $h(w) \in F$ and $w \in L$.

□

Now we move towards the logical characterization of regular languages. The theorem that we will show is due to Trakhtenbrot, Büchi and Elgot and it's the first proof of the existence of a link between logic and computational models.[6].

Theorem 2.2. *A language L is regular if and only if it is MSO-definable.*

Let M be a finite automaton that recognizes L and suppose that the number of states is $n + 1$. This means that for every word $w \in L$ (and only for these) there is a path in the automaton that goes from the initial state q_0 to a final one and that reads w. Rephrasing a bit more, we can label each position $x_0, \ldots, x_n$ of w by the name of the state in which the cursor is when the letter in that position is read; then if the position m is labeled with q_i and the position $m + 1$ can be labeled with q_j if and only if the m-th letter is $a \in \Sigma$ such that $\delta(q_i, a) = q_j$.

Due to this remark, it's easy now to understand what MSO-formula a word w in L must satisfy: there must be n subsets X_{q_i} of the set of positions, one for each state q_i, $[\exists X_{q_0} \ldots \exists X_{q_n}]$ such that:

- the initial position state is the initial state q_0 $[0 \in X_{q_0}]$;
- for every q_i and for every non-terminal position x in which the cursor is in the state q_i, the transaction between that position and the next is allowed by δ $[\forall x(x \in X_{q_i}) \rightarrow \bigvee_{(j,a)\in R_i}(a(x) \wedge S(x) \in X_{q_j})$, where $R_i = \{(j, a) : \delta(i, a) = j\}$ and $S(x)$ denote the successive position of x[7]];

[6]The interested reader can find many similar results, either for formal languages or for the computational complexity class. This field of study is generally called "Descriptive Complexity Theory".

[7]This function is definable by the following formula (that belongs to FO, too)

$$y = S(x) \equiv x < y \wedge \forall z(x < z \rightarrow y \leq z),$$

where $x < y \equiv x \leq y \wedge \neg(y \leq x)$.

- the last position state is a final state $[\bigvee_{q_f \in F} x_n \in X_{q_f}]$.

Conversely, to show that a language that is MSO-definable, is also regular, we will proceed inductively. To do so, as it's very common in this kind of proof, we have to deal not only with statements (i.e. formulas without free variables) but also with generic formulas: we have, hence, to provide an interpretation for them. In MSO there are two kinds of variables (first-order ones and second-order ones) and it will be easier to handle only the second-order ones. To this aim, we introduce in our vocabulary σ the symbols explained in the following:

- $X \subseteq Y$: informally, the position in X are also position in Y;
- $X \leq Y$: informally, every position x in X comes before every position y in Y;
- $X \subseteq a$: informally, for every position x in X, $a(x)$ is true.

With this extension, we can delete all the occurrences of first-order variables.[8] Then, as we have already told in the Example 1.2, if we have a MSO-formula $\varphi(X_1, \ldots, X_n)$ with only second-order free variables, we can, however, define a notion of satisfiability, equipping the word w with some subsets of position $I_1, \ldots, I_n$. A more suitable way, in what follows, to deal with it, is to think to words $\bar{w}$ not in Σ^*, but in $(\Sigma \times \{0,1\}^n)^*$; this description is analogous to the other one if we think I_j as the set of position which have 1 in the j-th component. In this way, the language of $\varphi(X)$ is

$$L = \{w \in (\Sigma \times \{0,1\}^n)^* \ : \ (w, I_1, \ldots, I_n) \vDash \varphi(X_1, \ldots, X_n)\},$$

Now we are ready to show the remaining implication of Theorem 2.2. For the Theorem 2.1, we need only to show that a language L described by a MSO-formula, is recognizable by a monoid. We proceed by induction on the height of the formula:

- (*Atomic formulas:*) We should find, for each atomic formula built up with the relational symbols of σ, a suitable monoid, and homomorphism. We exhibit how it works with an example.

[8] We won't go further in details. The only delicate thing to deal with is when occurs "$\exists x$ ". In that case, we can rephrase it in the following way

$$\exists X \ X \neq \emptyset \wedge \forall Y \ Y \subseteq X \rightarrow (Y = \emptyset \vee Y = X)$$

.

Consider $\varphi(X) = X \subseteq a$. The related language is

$$L = \{w \in (\Sigma \times \{0,1\})^* : \text{in every letter in } \Sigma \times \{0,1\} \text{ in which the second component is 1, the first one is } a\}.$$

We can define, hence, $M = (\{0,1\}, \min)$ and $h : (\Sigma \times \{0,1\})^* \to M$ the homomorphism that maps the words with the letter a in the first component of each pair in which the second component is 1. This is an homomorphism because the concatenation of two words is mapped in 1 if and only if every word is mapped in 1.

- (*height* > 0*:*)
  - $\wedge$) Let $\varphi(X_1, \ldots, X_n) = \varphi_1(X_1, \ldots, X_n) \wedge \varphi_2(X_1, \ldots, X_n)$. Then we have

    $$h_1 : (\Sigma \times \{0,1\}^n)^* \to M_1$$
    $$h_2(\Sigma \times \{0,1\}^n)^* \to M_2$$

    that recognize the language of φ_1 and the one of φ_2. Then we have

    $$h = (h_1, h_2) : (\Sigma \times \{0,1\}^n)^* \times (\Sigma \times \{0,1\}^n)^* \to M_1 \times M_2,$$

    the morphism product. Then, if F_1 and F_2 are the accepting sets, respectively of M_1 and M_2, then $F_1 \times F_2$ is the accepting set for the language of φ;
  - $\neg$) The language of $\neg\varphi(X_1, \ldots, X_n)$ is recognizable by the same monoid and homomorphism that recognize the one of $\varphi(X_1, \ldots, X_n)$; but the accepting set now is the complementary;
  - $\vee$) Follows by the De Morgan Laws and the previous points. [9];
  - $\exists$) Assume that the language L of $\varphi(X_1, \ldots, X_n, X)$ is recognisible by a monoid M and an homomorphism $h : (\Sigma \times \{0,1\}^{n+1})^* \to M$. Then, the language $L_\exists$ of $\exists X \varphi(X_1, \ldots, X_n, X)$ can be characterised as follows:

    $$L_\exists = \{w \in (\Sigma \times \{0,1\}^n)^* \mid \text{ exists } b \in \{0,1\} \text{ tale che } wb \in L\}.$$

    Then we can consider the projection map that forgets the last letter

    $$\pi : (\Sigma \times \{0,1\}^{n+1})^* \to (\Sigma \times \{0,1\}^n)^*$$

[9] For a more explicit proof, reason as for $\wedge$, but the accepting set now is $F_1 \times M_2 \cup M_1 \times F_2$

and the map

$$H : (\Sigma \times \{0,1\}^n)^* \to P(M)$$
$$w \mapsto \{h(v) \ : \ \pi(v) = w\}.$$

It's easy to see now that if $w \in L_\exists$, then $H(w)$ must be a subset of M with at least one accepting element of M. We can endow $P(M)$ with the monoid structure, defining the product of two subsets A, B of M as the set of the elements of the form $a \cdot b$, where $a \in A$ and $b \in B$. With this structure, H is a homomorphism.

– $\forall$) Follows by the De Morgan Laws and the previous points.

3 Example 2: Star-free languages

In this section, we present another example where we can see the correspondence already introduced between these different mathematical fields. Namely, we can think that if regular languages, monoid, and MSO are closely related to each other, it can be also the case that if we consider a nice subclass of regular languages, then it can be described by monoid with some particular properties and with sublogic of MSO. Surprisingly, the substructures that come up are really easy to study.[10]

Let's start with the formal languages. To explain why the subclass of regular languages that we will characterize is "natural", we define regular language that is equivalent to the one with automaton: in contrast to the latter, this approach is bottom-up.

Definition 3.1. Given an alphabet Σ,

- the empty language is regular;
- for every $a \in \Sigma$, the language composed by only the word a is a regular language;
- the union, intersection of two regular languages is regular;
- the complement of a regular language is regular;
- given two regular languages L_1, L_2, the concatenation language, i.e. the language whose words are the concatenation of a first word in L_1 and a second one in L_2 is regular;
- given a language L, the language L^* whose words are the concatenation of words in L is regular; [11]
- a regular language can be built only with the rules of these previous points.

As usual in these recursive definitions of structures, we are interested also in omitting one rule and seeing what is the outcome. This is the case also in

[10] It's truly an unexpected fact! In every field (logic, algebra, formal language theory) only some classes of structure are studied and these are also related to each other and, in a certain sense, analogous.

[11] Note that $\emptyset^* = \{\epsilon\}$.

formal language theory and a very studied class of languages are the *star-free* ones, i.e. the ones without the second last rule in the Definition 3.1.[12]

The reader surely found an analogy between this bottom-up construction of regular language and the one that we gave about the MSO logic. Even in this case, we have some rules, and by omitting some of them we can obtain a lot of however important logics; the most natural omission in this context is the reference to the second-order world, which leads us to the first-order logic.

Theorem 3.1. *Given an alphabet Σ and a language $L \in \Sigma^*$, then it is recognizable by an FO-formula if and only if it is a star-free language.*

Before proving this theorem we introduce some considerations about FO that will be necessary to understand the proof.

We note that for a FO-formula φ there can be a lot of words that satisfy it, say e.g. w_1, w_2 are two of them. From another point of view, this means that φ has not enough information to distinguish between w_1 and w_2. Maybe, w_1 and w_2 cannot be distinguished by any FO-formula and if this is the case, we call them *FO-equivalent* and we write $w_1 \equiv w_2$. We can refine this equivalence relation by introducing the notion of *rank* for a formula: intuitively this is the maximum number of nested quantifiers. Hence, if two words are indistinguishable with every formula of rank up to k, we say that they are *k-equivalent*, and we denote this fact by $w_1 \equiv_k w_2$.

There is a connection between k-equivalent words and their structure. In particular it can be proved that

Lemma 3.2. *Given an alphabet Σ and $w_1, w_2 \in \Sigma^*$, then the following are equivalent:*

- $w_1 \equiv_{k+1} w_2$*;*
- *for every $a \in \Sigma$ and for every decomposition of the form $w_1 = m_1 a n_1$ there exists a decomposition of the form $w_2 = m_2 a n_2$ where $m_2 \equiv_k m_1$ and $n_2 \equiv_k n_1$ and, conversely, for every decomposition of the form $w_2 = m_2 a n_2$ there exists a decomposition of the form $w_2 = m_1 a n_1$ where $m_1 \equiv_k m_2$ and $n_1 \equiv_k n_2$.*

[12]Pay attention to the fact that even if the Definition 3.1 seems redundant (because for example, if we can take union and complementation of languages we have also the intersection), a different definition of regular language (like one that omits the possibility of taking intersection between two regular languages to the remark above) leads us to a different definition of star-free languages (for example, without allowing complementation we can't show that Σ^* is star-free, but allowing complementation this is true, since $\Sigma^* = (\emptyset)^c$). This is the reason why we chose a less economic definition.

We won't prove this lemma since it requires some deeper knowledge about logic[13] and it leads us far away from our high-level approach to the topic; the interested reader can be seen the proof in [1].

Proof. (Theorem 3.1)

- $\Leftarrow$): We can prove the thesis by induction following the construction of the star-free language L that we are considering:
  - if L is the empty set, then it can be described by the formula $\exists x \neg(x = x)$;
  - if L is the set $\{a\}$, where a is a letter in Σ, then it is described by the formula
    $$\varphi := \forall x \forall y (x = y \wedge a(x));$$
  - if L is a union of the languages L_1 and L_2, respectively described by φ_1 and φ_2, then it's described by $\varphi = \varphi_1 \vee \varphi_2$;
  - assume that L is the concatenation language of L_1 and L_2, respectively described by φ_1 and φ_2. The formula φ we are looking for should say that there exists a position x (the last of the word in L_1) such that the word truncated at x satisfies φ and the remaining word (the piece right to x) satisfies φ_2. So
    $$\varphi = \exists x (\tilde{\varphi}_1(x) \wedge \bar{\varphi}_2(x)),$$
    where $\tilde{\varphi}_1(x)$ is the relativization of the formula φ_1 to the positions less or equal than x[14] and $\bar{\varphi}_2(x)$ is the relativization of the formula $\varphi_2(x)$ to the positions strictly greater than x.
- $\Rightarrow$): Let L be recognised by a formula φ and assume that its rank is k. Since Σ is a finite alphabet, the vocabulary that we use for FO-formulas are finite and then there are only finitely many atomic formulas (i.e. formulas of rank 0). For the Lemma 3.2, there are only finitely many formulas (up to equivalence) of rank up to k. Then L is a finite union of k-equivalence classes and then we have only to show that every k-equivalence class (that is a set of words) is a star-free language. This can be done by induction on k:

[13] It's a direct consequence of the Ehrenfreucht-Fraïssé games.

[14] i.e., when a quantified formula occurs as a subformula of φ_1, for instance $\exists y \psi(y)$, we change it in $\exists y(y < x \vee y = x) \wedge \psi(y)$. With regard to the universal quantification, $\forall y \psi(y)$ becomes $\forall y(y < x \vee y = x) \rightarrow \psi(y)$.

- $k = 0$: since there are no atomic formulas without free variables, there is only one equivalence class that is the whole Σ^*. This is a star-free language because $\Sigma^* = (\emptyset)^c$.
- $k+1 > 0$: let X be a $k+1$-equivalence class. Then if a word w in X has the form man, all the other words in X have a similar form, where the part before a is k-equivalent to the word m, and the one after a is k-equivalent to n. Then, if Y and Z are k- equivalence classes, $m \in Y$ and $n \in Z$ we will write YaZ for the set of all words with the aforementioned decomposition. For Lemma 3.2, hence, $YaZ \subseteq X$. This means that, calling

  $$\Omega = \{(Y, a, Z) \in P(\Sigma^*) \times \Sigma \times (\Sigma^*) \ : \ Y, Z \text{ are } k\text{-equivalence class}\},$$

  we have

  $$X = \bigcup_{(Y,a,Z)\in\Omega; X\subseteq YaZ} YaZ \cap \bigcap_{(Y,a,Z)\in\Omega; X\cap YaZ=\emptyset} (YaZ)^c.$$

  The thesis follows for the inductive hypothesis and for the fact that a boolean combination of star-free languages is still a star-free language

□

We move now to the algebraic characterization, finding a feature of monoids such that this subclass that we will create, will be the suitable one to "catch" the star-free languages.

First of all, we can notice that the k-equivalence respects the monoid structure on Σ^*. Formally, if v_1, v_2, w_1, w_2 are words in Σ^*, then

$$v_1 \equiv_k v_2 \text{ and } w_1 \equiv_k w_2 \Rightarrow v_1 w_1 \equiv_k v_2 w_2.$$

This is a consequence of Lemma 3.2; we can prove it by induction:

- if $k = 0$, then, since there are no atomic formulas with free variables, all the words are 0-equivalent and the thesis is trivially satisfied;
- assume that the implication holds for $k > 0$ and that $v_1 \equiv_{k+1} v_2$ and $w_1 \equiv_{k+1} w_2$. Then, by Lemma 3.2, we simply have to prove that $v_1 w_1 \in LaK$ if and only if $v_2 w_2 \in LaK$ for $a \in \Sigma$ and L and K, $\equiv_k$-classes. Hence, suppose that $v_1 w_1 = l_1 a k_1$, with $l_1 \in L$ and $k_1 \in K$, for some equivalence classes. Without loss of generality, we can also assume that the word a is a letter of w_1, and for an easier notation,

we have the decomposition $v_1 w_1 = v_1 l_1' ak$. Then $w_1 = l_1' ak$ and since $w_1 \equiv_k w_2$, we have also a decomposition $w_2 = l_2' ak_2$ with $l_2 \in L$ and $k_2 \in K$. By induction hypothesis so, $l_1 = v_1 l_1' \equiv_k v_2 l_2$, because $v_1 \equiv_k v_2$ and $l_1' \equiv_k l_2'$.

The consideration above tells us that the projection map $\pi_k : \Sigma^* \to \Sigma^*/\equiv_k$ is a homomorphism for every k into a finite monoid (because the k-equivalence classes are finitely many). Furthermore, this homomorphism recognises all the languages that are definable by a rank k FO-formula, because such a language will be the union of k-equivalence classes.

If we want to point out an algebraic characterization of star-free languages, we have to find some interesting features of these monoids $\Sigma^*/\equiv_k$. One of the first things that comes to mind to study a finite monoid, is to establish if there is any "torsion". Recall that in a monoid an element doesn't have to have the inverse, so a more suitable concept for a monoid is the periodicity.

Definition 3.2. A monoid M is called *periodic* with *period* T if for every element $a \in M$, $a^T = a$. M is called *aperiodic* if for every element $a \in M$, the succession of the power of a, i.e. $a, a^2, a^3, \ldots$, has no a periodic behaviour definitely.

In a finite aperiodic monoid M, since the elements are finitely many, it must happen that for every element $a \in M$, exists $n \in \mathbb{N}$ such that $a^n = a^{n+1}$, i.e. the succession of the power is definitely constant. Indeed, for M being finite, if such succession doesn't stabilize, exist $m, n \in \mathbb{N}$ such that $m < n$ and $a^m = a^n$. This means that $a^{n+k} = a^n a^k = a^m a^k = a^{m+k}$ for every k and so the power succession of a is periodic.

In $\Sigma^*/\equiv_k$, given a word w we are interested to understand if there is a n for which $w^n \equiv_k w^n w$. We won't give the detail of the proof of this fact, because it requires more knowledge of FO properties[15] and a degression on this would lead us far from the intent of this work. However, we can show that the following proposition holds.

Theorem 3.3. *Given an alphabet Σ and a language $L \subseteq \Sigma^*$ that is recognized by an FO-formula, then it is also recognized by a finite aperiodic monoid.*

Actually also the converse holds and, by Theorem 3.1, we can state that

Theorem 3.4. *Given an alphabet Σ and a language $L \subseteq \Sigma^*$, the following are equivalent:*

[15]Namely, the fact that FO-expressiveness is tied only to local things, see [3] for more details.

- *L is recognizable by a FO-formula;*
- *L is star-free;*
- *L is recognizable by a finite aperiodic monoid.*

For the interested reader, we note also that, from a logic perspective, the previous theorem can be generalised, involving the linear temporal logic LTL. We won't give the details, but it can be shown that each of the three properties of L in the Theorem 3.4, is actually equivalent to the LTL-definability.[16]

[16] See [1]

4 Eilenberg-Moore Theorem

The previous chapter provided a link between some classes of languages (regular, star-free) and some classes of algebraic structures (monoid, aperiodic monoid). In literature can be found other correspondences of this kind: rise, then, up the question of whether this is a mere coincidence or there is some structural "connection" between these worlds. The main goal of this chapter is to give the reader all the essential tools to understand the Eilenborg-Moore Theorem (Theorem 4.2) and its role in linking these two apparently different branches of mathematics and theoretical informatics. As in other questions of this kind, the topic is presented in a categorical fashion: to deeply understand what is written from now on, the reader should be familiar with the basic notions of functor and natural transformations.

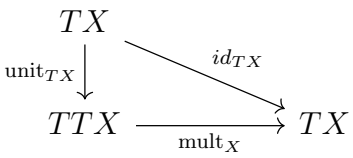
Let's start by introducing one of the most useful object in dealing with this topic.

Definition 4.1. A *monad* in **Set**[17] is composed by:

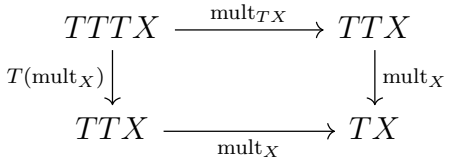
- a functor $T : \mathbf{Set} \to \mathbf{Set}$;
- a natural transformation unit : $id_{\mathbf{Set}} \to T$;
- a natural transformation mult : $TT \to T$;

such that the following diagrams (Associative Axioms (AA in short)) commute hold for every $X \in \mathbf{Set}$:

- AA1

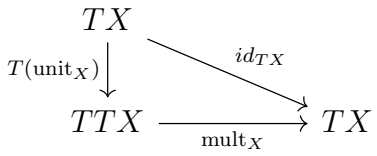


- AA2



[17]The Definition 4.1 extends in an obvious way in other categories: for what we need, is sufficient to work in **Set**.

- AA3



Let's give some examples to understand better this definition.

Example 4.1 (Monad of finite words)**.** We can take $^* : \mathbf{Set} \to \mathbf{Set}$ as the map that to every set X associates the set of finite words on it Σ^*. This is actually a functor. In addition to this, we have:

- for every X a map $\text{unit}_X : X \to TX$ that maps a letter in X into the word composed only by the letter itself;
- for every X a map $\text{mult}_X : TTX \to TX$ that maps a word composed by words on X into the concatenation of this, that is a word on X.

The definition of these maps doesn't depend on the particular set X and this can be shown by proving that, indeed, these are natural transformations.

We have now to verify that the associative axioms hold. In the following is convenient to use $a_1, \dots, a_n$ to name letters in X, $w_1, \dots, w_m$ as words of letter in X (i.e. in X^*) and $\alpha_1, \dots, \alpha_s$ as words of words of letter in X (i.e. in $((X^*)^*)^*$):

- AA1) says that if a word in X^* is regarded as a "word of one letter" in $(X^*)^*$ and then we concatenate its component and we see it as a word in X^*, we end up with the initial word;
- Suppose to have a word $\alpha_1 \dots \alpha_s$ in $((X^*)^*)^*$, where $\alpha_i = w_{1i} \dots w_{n_i i}$ and $w_{jt} = x_{1jt} \dots x_{n_{jt}jt}$. Then AA2) is equivalent to say that we have the same outcome in both of the following cases:

$$\begin{aligned} \alpha_1 \dots \alpha_s &\mapsto w_{11} \dots w_{n_1 1} \dots w_{1s} \dots w_{m_s s} \\ &\mapsto a_{11} \dots a_{n_{11}1} a_{12} \dots a_{n_{12}2} \dots a_{1m_s} \dots a_{n_{m_s s} m_s} \end{aligned}$$

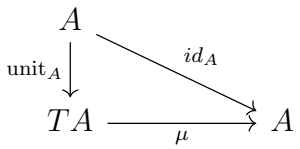
$$\begin{aligned} \alpha_1 \dots \alpha_s &\mapsto (a_{11} \dots a_{n_{11}1})(a_{12} \dots a_{n_{12}2}) \dots (a_{1m_s} \dots a_{n_{m_s s} m_s}) \\ &\mapsto a_{11} \dots a_{n_{11}1} a_{12} \dots a_{n_{12}2} \dots a_{1m_s} \dots a_{n_{m_s s} m_s} \end{aligned}$$

- AA3) says that if we start with a word in X^*, regard it as a word in $(X^*)^*$ (composed by only one letter), and then regard it as a word in X^*, then we end up with the original word.

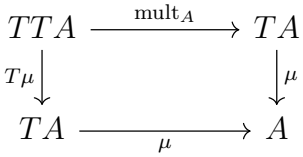
Example 4.2 (Group Monad)**.** A more interesting example of monad is given by the free group functor. For a set X we can construct the "minimal" group that "contains"[18] X. We can define $\overline{X}$ as the set of all elements of X but with a line above and we can take $(X + \overline{X})^*/\sim$, where $w_1 \simeq w_2$ if and only if after some simplification of the type $x\overline{x} = \epsilon$ or $\overline{x}x = \epsilon$ we end up with the same word.

Definition 4.2. An *Eilenberg-Moore algebra* in a monad T (or a *T-algebra*) is a set A endowed with a *multiplication operation* $\mu : TA \to A$ such that the following associative axioms hold, i.e. the following diagrams commute:

- AA4



- AA5



Even in this case, we hope that an example will be useful to the reader.

Example 4.3 (Group)**.** We can show that a group is a T-algebra, where T is the free-group-functor defined in the Example 4.2.

If A is an algebra, we have $\mu : TA \to A$ and we can define $1 = \mu(\epsilon)$, $x \cdot y = \mu(\text{mult}_A(\text{unit}_A(x)\text{unit}_A(y)))$ and $x^{-1} = \mu(\overline{x})$). Let's show that 1 is the neutral element and we leave the other thing to check to the reader. We have to show that for all $x \in A$, $x \cdot 1 = 1 \cdot x = x$, i.e. $\mu(\text{mult}_A(\text{unit}_A(1)\text{unit}_A(x))) = x$ and $\mu(\text{mult}_A(\text{unit}_A(x)\text{unit}_A(1))) = x$. We note that morally $\epsilon x = x = x\epsilon$, but stated in this way this statement has no sense. Let's try to explain what it means, recalling that $\epsilon \not\in nA$, but $\epsilon \in TA$: what we want to say is that the concatenation between the words ϵ and $\text{unit}_A(x)$ is equal to $\text{unit}_A(x)$. But also this is not true: indeed, the aforementioned concatenation is in TTA, while the $\text{unit}_A(x) \in TA$. What we want to show is that in TTA

$$\epsilon\text{unit}_A(x) = \text{unit}_{TA}(\text{unit}_A(x))$$

[18]This can be stated more precisely through the notion of universal property.

and this is true because if we apply mult_A in both the sides we have for AA1, that

$$\text{mult}_A(\epsilon\text{unit}_A(x)) \stackrel{?}{=} \text{mult}_A(\text{unit}_{TA}(\text{unit}_A(x))) = id_{TA}(\text{unit}_A(x)) = \text{unit}_A(x)$$

and this is true for the property of the empty word ϵ. Therefore, applying μ on the previous equation,

$$\mu(\text{mult}_A(\epsilon\text{unit}_A(x))) = \mu(\text{unit}_A(x)) \stackrel{AA4}{=} x$$

and noting that for AA4 $\text{unit}_A(1) = \epsilon$, we have the thesis. Analogously can be shown that $x\dot{1} = x$.

Conversely, if we have a group $(G, \cdot)$, then we can define μ in an obvious way in TG through the multiplication $\cdot$, identifying in it $\overline{x}$ with x^{-1}. The axiom AA4 is truly satisfied and the axiom AA5 is the categorical analog of associativity.

Even if we won't deal with the details, there are a lot of other algebraic structures that are Eilenberg-Moore algebras for the aforementioned functors. For instance, in the monad of the finite words (Example 4.1) an algebra is a monoid; in the monad of nonempty finite words, an algebra is a semigroup.

If we want to confront two different algebras we should have a concept of homomorphism and this is what we are about to define.

Definition 4.3. Let T be a monad and $(A, \mu_A), (B, \mu_B)$ two T-algebras. A T-homomorphism is a function $h : A \to B$ such that the following diagramm commutes:

$$\begin{array}{ccc} TA & \xrightarrow{Th} & TB \\ \downarrow{\scriptstyle \mu_A} & & \downarrow{\scriptstyle \mu_B} \\ A & \xrightarrow{h} & B \end{array}$$

Going on with the special case of groups (or of monoids/semigroups), we notice that a map $h : A \to B$ between groups is a homomorphism in the sense of the Definition 4.3 if and only if given $a_1 \dots a_n$ words in TA, the two following maps have the same outcome:

right-down) $a_1 \dots a_n \mapsto h(a_1) \dots h(a_n) \mapsto h(a_1) \cdot \dots \cdot h(a_n)$;

down-right) $a_1 \dots a_n \mapsto a_1 \cdot \dots \cdot a_n \mapsto h(a_1 \cdot \dots \cdot a_n)$.

Therefore, h is a homomorphism if and only if it is a group homomorphism. The same holds for monoids and semigroups.

For what we are interested in, we want to study formal languages by linking them to the algebraic structure now presented. We need, to do so, to catch the notion of "recognizable by a monad-algebra".

Definition 4.4. Given a monad T, a T-algebra A and a map $L : A \to U$, we call L a *T-recognizable language* if it factors through a homomorphism into a finite algebra (i.e. a T-algebra whose underlying set is finite); in other words, there exists a finite T-algebra B, a homomorphism $h : A \to B$ and a function $g : B \to U$ such that the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{L} & U \\ & \searrow^{h} & \uparrow{\scriptstyle g} \\ & & B \end{array}$$

We can notice that this definition is exactly the generalization of the concept of recognizable by a finite monoid (or by a finite aperiodic semigroup). Indeed, suppose that we have a finite monoid M, a subset F of accepting elements, and a monoid homomorphism $f : \Sigma^* \to M$ that recognizes L as in the Definition 1.4. Then recalling that a monoid is an algebra for the monad T of finite words, Σ^* is a T-algebra and the indicator function is a map $\chi_L : \Sigma^* \to \{0, 1\}$. It factors through the finite monoid M by the homomorphism h and g is simply the indicator function χ_F of $F \subseteq M$.

To state a theorem to answer the question at the beginning of the chapter, we have to note that not all the classes of algebraic structures or formal languages can be involved in this connection between these two worlds. As a paradigm, we can take the proof of the Theorem 2.2. In this latter, we can see that some features of monoids are essential for proving some features of the formal languages: for instance, the fact that the product of two monoids is a monoid, is necessary to prove that the class of languages recognized by them are closed under boolean operations. In this sense, some features that we have on the algebraic structures (like closure under product) reflect in some features that we have in the class of formal languages that we are studying (like closure under boolean operations).

As we will see in the last part of the chapter, the notion of variety rises for this reason: it collects all the features (for classes of algebraic structures and classes of formal languages) that are essential to establish a link as we want. For now, we introduce these concepts, giving some reasons suggested by the example encountered before.

Definition 4.5. Given a monad T, an *algebra variety* $\mathscr{A}$ is a class of finite T-algebras such that:

- if $A \in \mathscr{A}$, then all the quotients of A are in $\mathscr{A}$;
- if $A \in \mathscr{A}$, then all the sub-algebras of A (i.e. all the algebras whose underlying set is a subset of A and for which the inclusion is a T-algebra homomorphism) are in $\mathscr{A}$;

- if $A, B \in \mathscr{A}$, then the product $A \times B$ is in $\mathscr{A}$.

We can note that a lot of the classes already encountered of algebraic structures are a variety. Consider T the monad of finite words. The algebras are monoids and we have a lot of interesting classes in them: finite groups, finite aperiodic monoids, and finite monoids. They all are algebra varieties since groups/semigroups/monoids are closed under products/quotients/substructures and since finiteness and aperiodicity are preserved by products/quotients/substructures.

But some classes are not varieties: in the monad of nonempty finite words, where an algebra is a semigroup, the class of monoids is not a variety. Indeed, a sub-algebra of a monoid M is (in this monad!) a semigroup contained in M and it can be without the neutral element, so it can be not a monoid. These examples show then that to be a variety is a feature that depends not only on the property of the class but also on the monad setting in which we work.

An important notion, especially in these concrete categories, is the one of term. This is essential to talk about elements like $a \cdot b$, for instance.

Given a monad T, a *(polynomial) term* t over variables X is any element of TX. If we have a T-algebra A and a way to interpret the set variables X in A (usually with a $\eta \in A^X$), we can *interpret* this term t in A obtaining

$$t^A = T\eta(t).$$

Usually when the variables involved in a term t are $x_1, \ldots, x_n$ we will write $t(x_1, \ldots, x_n)$ and, if $\eta(x_i) = a_i$ for $i = 1, \ldots, n$, then we will write $t^A(a_1, \ldots, a_n)$ for the intepretation of t in A under $\eta \in A^X$; if this is the case, $a_1, \ldots, a_n$ will be called also parameters. This procedure and the notation reflect exactly what we do in logic with logical terms and in algebra with polynomials.

Let's investigate better this definition to understand why with this new tool we can easily talk about elements like $a \cdot b$ as said previously. Indeed, a monoid M is, with the words developed in this section, an algebra over the monad of finite words. Given two elemnts a, b, formally the multiplication $a \cdot b$ is the term over $X = \{x, y\}$ $t(x, y) = xy \in TX$ evaluated in M under $\eta \in M^X$ such that $\eta(x) = a$ and $\eta(y) = b$. In the following, we denote briefly $a \cdot b$ for the aforementioned interpretation. As is shown in algebra, all the polynomial terms can be studied regarding them as a composition of very simple polynomial terms: unary polynomials.

Definition 4.6. Given an algebra A, a *unary polynomial* is any function $A \to A$ of the form $a \mapsto t^A(c_1, \ldots, c_n, a)$, for some term $t(x_1, \ldots, x_n, y)$ and for some parameters $c_1, \ldots, c_n$.

Unary polynomials are really useful because they collect in themselves the main features of all the polynomial terms in the sense of the following lemma.

Lemma 4.1. *Given a finite algebra A and a equivalence relation $\sim$, the following are equivalent and define the concept of* congruence *in A:*

a) the projection function $a \mapsto [a]_\sim$ is compositional, i.e. $[a \cdot b]_\sim$ depends only on $[a]_\sim$ and $[b]_\sim$;

b) $\sim$ is the kernel of some homomorphism from A to some algebra B;

d) for all terms t with variables in X, if $\eta_1(x) \sim \eta_2(x)$ for all $x \in X$, then the interpretations are equivalent, i.e. $t^A(\eta_1) \sim t^A(\eta_2)$;

d) for all unary polynomial f and for all $a, b \in A$, $a \sim b$ implies $f(a) \sim f(b)$.

We can note that for b) of the previous lemma, a congruence induces a quotient algebra.

We are now ready to define the analog of Definition 4.5 but for formal languages.

Definition 4.7. Let T a monad, a class $\mathscr{L}$ is a language variety if:

- the elements of the class are recognizable subsets of free algebras over a finite alphabet, i.e. $L \subseteq \Sigma^*$ such that Σ is finite and the indicator function $\chi_L : \Sigma^* \to \{0, 1\}$ is T-recognizable;
- $\mathscr{L}$ is closed under finite union, finite intersection, and complementation;
- $\mathscr{L}$ is closed under inverses of homomorphisms, i.e. if $h : T\Sigma \to T\Gamma$ is a homomorphism, then for every term $t \in T\Gamma$, $h^{-1}(t) \in \mathscr{L}$;
- $\mathscr{L}$ is closed under inverses of unary polynomials, i.e. if $f : T\Sigma \to T\Sigma$ is a homomorphism, then for every term $t \in T\Sigma$, $f^{-1}(t) \in \mathscr{L}$.

Like in the case of algebra varieties, the languages that we have already met form a variety.

Example 4.4. The first example that comes to mind is the regular languages over a finite alphabet. Let's show that they are a variety in the monas of finite words:

- they are recognizable for what we have already shown in Theorem 2.2;

- finite union, intersection and complementation of regular languages is still regular;
- let $h : \Sigma^* \to \Gamma^*$ a homomorphism of free algebras and let $L \subseteq \Gamma^*$ be a regular language. Then we have a finite automaton that recognizes this language, say $M = (\Gamma, Q, q_0, F, \delta)$. Then, we can construct the automaton $M' = (\Sigma, Q, q_0, F, \delta')$ such that $\delta'(q, a) = \delta(q, h(a))$. Then inductively is easy to show that if we read a word $w \in \Sigma^*$ in M' ending up in a final state, then in M, we can read (following the "analogous" path) the word $h(w)$. Then the automaton M' accepts all and only the elements in $h^{-1}(L)$;
- for unary polynomials the proof is similar to the previous one. If we have a unary polynomial f, is easy to concatenate automata for the languages $\{a\}_{a\in\Sigma}$ and for the original language L to obtain an automaton for the language $f^{-1}(L)$. Obviously, in doing this, the finiteness of the automata is preserved.

Example 4.5. Let's discuss the case of FO-definable languages in the monad of finite words:

- all of these languages are T-recognizable because they are regular and so recognisable by a finite monoid: this means that for the remarks after Definition 4.4, the indicator function of a language in this class, factors through a finite T-algebra;
- the class is closed under boolean combinations because so the FO-formulas are;
- given $h : \Sigma^* \to \Gamma^*$ and a language $L \in \mathscr{L}$ that also belongs to Γ^*, we have to show that $h^{-1}(L) \in \mathscr{L}$. If we recall Theorem 3.1, we can say that L is a union of $\equiv_k$-classes, for some k, and then we have only to show that also $h^{-1}(L)$ is a union of $\equiv_{k'}$-classes, for some k'. This can be shown by proving that if $w \equiv w'$, then $h(w) \equiv h(w')$ and this fact is related to Ehrenfeucht-Fraïssé games. Informally, the idea behind this approach is that if we can distinguish by a formula $h(w)$ and $h(w')$ then we can "copy the instructions that lead us to distinguish the words and adapt them to w and w'"[19];
- for the closure under unary polynomials, all the thing works as in the previous point and a proof can be performed through Ehrenfeucht-Fraïssé games.

[19]The reader that knows how the Ehrenfeucht-Fraïssé games work should see in this sentence the proof of the statement

We have now all the ingredients to state the main theorem of the work; it tries to answer the beginning question that was the file rouge in the whole paper. The theorem is due to Eilenberg and provides a very precise description of the link between formal languages (and automaton) and algebra.

Theorem 4.2. *Let T be a monad and let $\mathbb{A}$ the class of algebra varieties and $\mathbb{L}$ the class of language varieties. Then, we have two maps:*

- $L : \mathbb{A} \to \mathbb{L}$ *that maps an algebra variety $\mathscr{A}$ to the set of languages recognized by at least one algebra from $\mathscr{A}$;*
- $A : \mathbb{L} \to \mathbb{A}$ *that maps a language variety $\mathscr{L}$ to the set of algebras that recognise only languages of $\mathscr{L}$.*

In addition to the fact that these are well-defined, L and A are also mutually inverse and hence bijection.

The proof involves something that we have not discussed in this work, which is a syntactic homomorphism between languages. However, there are some easy considerations that we can make about the statement.

Assume to have a variety $\mathscr{A}$ and $L(\mathscr{A})$ is hence a set of languages. If $\mathscr{L}'$ is another set of languages that contains $L(\mathscr{A})$, then what is the relation between $A(\mathscr{L}')$ and $\mathscr{A}$? $A(\mathscr{L}')$ will be the set of all the algebras that recognize only languages in $\mathscr{L}'$; but if an algebra in $\mathscr{A}$ recognizes a language L, then $A \in L(\mathscr{A})$, and so $A \in L'$. Then, $\mathscr{L}' \subseteq A(\mathscr{L}')$. Also, the converse is true and eventually, we have that

$$\mathscr{A} \subseteq A(\mathscr{L}') \text{ if and only if } L(\mathscr{A}) \subseteq \mathscr{L}'.$$

Categorically, if we regard the poset structure of $\mathbb{A}$ and of $\mathbb{L}$, L and A are two functors between these categories and they are an adjoint pair.

We note that the previous statement is still valid if we don't talk about varieties. Indeed, it can be easily shown that L, A can be defined also between the classes of finite algebras closed under finite products ($\mathbf{A}$) and the classes of languages closed under finite unions and intersections ($\mathbf{L}$). Notice that in the proof of Theorem 2.2 we have already shown the role of the product of monoids in recognizing union/intersection of regular languages. Also in these posets L and A form an adjoint pair.

We can see this thing as a Galois connection. In Galois theory closed object ($\mathscr{A}$ such that $AL(\mathscr{A}) = \mathscr{A}$ or $\mathscr{L}$ such that $LA(\mathscr{L}) = \mathscr{L}$) are really important because for them the connection is a one-to-one correspondence. In this context, what the Theorem 4.2 says is that the algebra varieties and the language varieties are the closed elements of the aforementioned posets $\mathbf{A}, \mathbf{L}$ with the Galois connection induced by L and A.

Bibliografy

References

[1] M. Bojańczyk, *Langauges recognized by finite semigroups and their generalizations to objects such as trees and graphs with an emphasis on definability in monadic second-order logic*. University of Warsaw, 1st edition, 2020.

[2] S. Eilenberg, *Automata, languages, and machines, Vol. B*. New York Accademic Press, 1976.

[3] H.-D. Ebbinghaus, J. Flum, *Finite Model Theory*. Springer, 2nd edition, 1995.