Proposal for the course

Typed Monoids

An algebraic description of (possible non-regular) languages and its connection to logic and circuit complexity

The teaching activities we intend to participate in the PhD Program of Mathematics at the Universidad Politecnica de Valencia are framed in a field where theoretical computer science and mathematics meet. More precisely, it deals with the algebraic theory of formal languages and its connection to logic and circuit complexity. This is related to one of the research lines proposed in the PhD Program, supervised by Ana Martinez Pastor, and there are some PhD students which are involved in this area.

We will start with an overview of the topic and end with an outline of the course.

Survey

In the well established algebraic theory of automata, a language $L \subseteq \Sigma^*$ is recognized by a monoid M if there exists a morphism $h : \Sigma^* \to M$ and a subset A of M such that $L = h^{-1}(A)$.

Using this definition many classification results were discovered often yielding decidability results not known to be obtainable by other means. For instance, it can be shown:

Theorem of Myhill-Nerode, 1958: A language is regular if and only if it is recognized by a finite monoid.

Theorem of Schützenberger, 1965: A language is starfree (can be described by a regular expression constructed from the letters of the alphabet, the empty set symbol, all boolean operators and concatenation but no Kleene star) if and only if it is recognized by a finite aperiodic monoid (i.e. a finite monoid having no non-trivial groups).

As presented in the seminar *Finite monoids and recognizable languages* (held in Valencia 2005 by me) a number of important subclasses of the regular languages admit similar algebraic characterizations, and thus for these classes the membership problem is decidable.

It should be mentioned that in this setting we can assign to each language a unique smallest monoid recognizing it – the so called syntactic monoid – and that it is enough to check this monoid for decidability questions. Although we can define the syntactic monoid for non-regular languages too – here as a suitable quotient monoid of Σ^* – this is not as powerful as for regular languages (see below).

A unifying framework for characterizing classes of reguar languages via algebra was given by Eilenberg's

Variety Theorem, 1967: There is a one-to-one correspondence between varieties of regular languages and varieties of finite monoids.

(A variety of regular languages is a language class of regular languages being closed under Boolean operations, quotients and inverse morphism, a variety of finite monoids is a class of finite monoids being closed under taking submonoids, quotients and finite direct products.)

The algebraic characterization of classes of regular languages also lead to a strong connection between finite monoids and logic, there too yielding decidability results. The crucial point here is to exhibit algebraic objects that correspond to the used logic operators and then obtain the whole class by using suitable operations. For instance, the application of a modular counting quantifier is characterized by building the block product with a finite cyclic group, and in the same way, first order quantifiers are expressed by the block product with the two-element monoid U_1.

The following table gives a short overview of these connections:

Logic classes	variety of monoids	basic monoids	variety of formal languages
$FO[<]$	finite aperiodic monoids	U_1	starfree languages
$FO + MOD[<]$	finite soluble monoids	U_1, Z_p for a prime p	soluble regular languages
$FO + Group[<]$	finite monoids	U_1, G	regular languages
		G suitable finite group	

If we allow programs instead of morphism we also get a strong connection to some important low-level complexity classes, and here too this lead to some non-trivial seperating results.

Complexity classses	Variety of monoids (program)
AC^0	finite aperiodic monoids
ACC^0	finite soluble monoids
TC^0	–
NC^1	finite monoids

Unfortunately, the class TC^0 is not treatable in this setting, and so far there are no separating results known (thus whether $TC^0 \neq ACC^0$ nor $TC^0 \neq NC^1$ is known).[1]

Although these connections lead to a very fruitfull theory there remain some open problems. We just mention some concerning the connection between formal languages and algebra:

[1] A good introduction concerning the triangle logic, algebra and circuit complexity is given in the book of Straubing (Finite Automata, Formal Logic and Circuit Complexity, Birkhäuser, 1994)

Problems

- The above approach is not fine enough in the following sense: consider the two languages L_{parity}, L_{even} over the alphabet $\Sigma = \{a, b\}$ where the first one consists of all words with an even number of a's while the latter consists of all words of even length. Both languages have C_2, the cyclic group of order two, as syntactic monoid. But there are well-known results that in the model of circuit complexity classes the language L_{parity} is harder than L_{even}. (This reflects in the recognition via morphism through the fact that in L_{parity} the morphism needs to distinguish between an "a" as input and a "b" as input, whereas in L_{even} we count only the number of inputs.) Thus we need to have control over the recognizing morphism $h : \Sigma^* \to M$.
- How can we deal with non-regular languages? Consider the language L_{Maj} over the alphabet $\Sigma = \{a, b\}$ of all words with more a's than b's. This is clearly a non regular language and hence not recognized by a finite monoid. The syntactic monoid of L_{Maj} is $\mathbb{Z}$, but $\mathbb{Z}$ is also the syntactic monoid of an undecidable language (using the same morphism). Thus we also need to have control over the recognizing subset $A \subseteq M$.

If we combine these considerations i.e. limit the allowed morphisms and fix the set of acceptance subsets of the monoid, we obtain better possibilities, as seen at the following example: If we take $\mathbb{Z}$ and allow only morphisms mapping a letter to $\{-1, +1\}$, and have $\mathbb{Z}^+$ as accepting subset then we can only recognize languages that partition the alphabet into two sets A and B and test if the letters of set A occur more often than the letters of set B. All these languages are "close" to L_{Maj} in the sense that they reduce by a length preserving morphism to it.

These two observations lead us to the definition of a typed monoid. A *typed monoid* is a triple $(S, \mathfrak{S}, \mathcal{E})$, where S is a finitely generated monoid, $\mathfrak{S}$ is a finite Boolean algebra over S, and $\mathcal{E} \subseteq S$ is a finite set. The elements of $\mathfrak{S}$ are called *types* and the elements of $\mathcal{E}$ are called *units*.

A language is recognized by $(S, \mathfrak{S}, \mathcal{E})$ if there is a morphism h from $\Sigma^* \to S$, $h(\Sigma) \subseteq \mathcal{E}$, and $L = h^{-1}(\boldsymbol{\mathcal{S}})$ for a type $\boldsymbol{\mathcal{S}} \in \mathfrak{S}$. More generally, we use the units to limit the allowed morphisms while only types may be acceptance sets. For instance the typed monoid $(\mathbb{Z}, \{\mathbb{Z}^+, \mathbb{Z} \setminus \mathbb{Z}^+, \mathbb{Z}, \emptyset\}, \{-1, 1\})$ recognizes the language L_{Maj}. (Note that every finite monoid M can be regarded as typed monoid $(M, P(M), M)$ and that in this case we get the usual definition of recognition.)

Using this definition it can be shown that for every language there is a unique minimal typed monoid recognizing it (the so-called typed syntactic monoid) and that a version of Eilenbergs Theorem holds in this setting (even more: we can prove a correspondence between classes with weaker closure properties). This gives the fundament for an algebraic theory of (arbitrary) languages. By defining a suitable version of a block product it is also possible to get strong connections to logic and low-level circuit complexity,

which can be used to obtain an algebraic counterpart to the class TC^0. (A first result in this direction was presented 2008 in Valencia in the talk *An algebraic approach to the complexity class* TC^0.)

Structure of the course

In this course we will present the theory of typed monoids and its connection to logic and circuit complexity.

In the first part we will give the basic definitions and examples and present the algebraic theory of typed monoids (mainly the typed syntactic monoid, Eilenbergs theorem and a correspondence between weaker classes, and the block product of typed monoids). The second part will cover some basic concepts of logic and the connections between typed monoids and formal logic, and the third part will do the same for low-level complexity. We end this lecture with a short survey what can be done with this theory and with a presentation of the most urgent open problems concerning it.

The duration of this course is ten hours, distributed in five sessions of two hours.