BIHAR SHODH SAMAAGAM
Multidisciplinary, Multi-Lingual, Peer Reviewed Open Access Journal
issn : 3048-6971
Vol. 3, Issue 1, January-March 2025
Available online : https://www.biharshodhsamaagam.com/

IDENTIFYING FORMAL AUTOMATA WITH SEMIGROUPS

Dr. Ravindra Kumar Dev
Ph.D. (Mathematics)
Tilka Manjhi, Bhagalpur University Bhagalpur (BIHAR)

Abstract:
In the history of the mathematics, the algebraic theory of semigroup is a relative new-comer. It is simply a set which is closed under associative binary operations. After continuous research works on the theory of semigroups, different fields have adopted it.
The theory of automata has used the concept of semigroup to study and formalise certain types of finite-state machines. Another interesting use of semigroup is in the field of sociology and anthropology. Semigroup theory is very flexible and appropriate language for the study of social relations, social networks or kinship systems. Formal languages are the other areas where the theory of semigroup is widely used. In life sciences, it is used to describe certain aspects in the crossing- over of organisms. It is also used in genetics and metabolism. Semigroup theory can be used to study problems in the field of partial differential equation. Its approach is to regard a time-dependent partial differential equation as an ordinary diffential equation on a function space.
No doubt, the study of semigroups trailed behind that of other algebraic structures with more complex axioms such as groups or rings. We will carry on an elaboration of any one of the areas.
Keywords: *Semigroup, Formal Automata, Formal Language, Translator etc.*

Introduction:

In the history of the mathematics, the algebraic theory of semigroup is a relative new-comer. But semigroup has trailed behind that of other algebraic structures with more complex axioms such as groups or rings. It is simply a set which is closed under associative binary operations. After continuous research works on the theory of semigroups, different fields have adopted it. The theory of automata has used the concept of semigroup to study and formalise certain types of finite state machines.

Here, we will discuss about the identification of formal automata with semigroups. This is possible not only for finite automata but also for all such automata that we know how to make a catenation of input data. No doubt, formal linguistic is an important part of modern computer sciences. Without mechanisms supplied by this field we would hardly know how to deal with complicated recursive structures such as programs or patterns. To analyse the input, we use the power of formal languages, grammars and formal automata. For transforming one language into another and breaking into the structure of data. Translations schemes and translators are used. There are two main reasons for using translation scheme-based mechanisms. First is a sequential automat which gives yes/no answer. This means we should construct the problem so that such an answer is informative and useful. Second is one level processing schema (Input – automata – conclusion). But it is inadequate in the case of multi-aspect problem analysed at different abstraction levels. In this situations it is better to create a processing hierarchy using consecutive translations as follows :-

Problem in L_1 language $\rightarrow$ Automat (L_1) $\rightarrow$ Problem in L_2 Languages $\rightarrow$ …. $\rightarrow$ Problem in L_{k-1} language $\rightarrow$ Automat(L_{k-1}) $\rightarrow$ Solutions in L_K language.

For better understanding, Finite automata algebraization of automata is required.

A finite automat can be written as $(Q, \sum, \Delta, \delta)$, where

Q — a set of states,

$\sum$ – a set of input symbols,

Δ — a set of output symbols,

δ — a control function $\sum$ x Q $\rightarrow$ Q x Δ^*

That is, $\delta(x, S_1) = (S_2, y)$, being in state S_1 under input x the automat switches to state S_2 and produces output y.

At first, we can expand the functions δ to take an extra argument from Δ^* what will reflect an already produced output. The output from the current step should be added to the existing one. So, we get $\delta: \Sigma \times Q \times \Delta^* \rightarrow Q \times \Delta^*$. Then we can expand the function δ to accept the input from $\sum^*$ by defining

$$\delta(x^*, a, S, y) = \delta\big(a, \delta(x^*, S, y)\big), a \in \Sigma, x \in \Sigma^*, y \in \Delta^*$$

And a function $\delta_x : Q \times \Delta^* \rightarrow Q \times \Delta^*$ such that $\delta_x(S, y) = \delta_x(x, S, y)$

If we have a non-deterministic automat we can analogously expand the function δ and get functions δ_x : $P(Q \times \Delta^*) \rightarrow P(Q \times \Delta^*)$. They are defined by taking a proper sum of results of the δ function on separate states in the input set :

$$\delta_x(\{(S_1, y_1) \cdots (S_n, y_n)\}) = \bigcup_{i=1}^{n} \delta\,(x, S_i, y_i)$$

If we define an operation of catenation for δ_x function by specifying $\delta_y o \delta_x = \delta_{xy}$

Thus, we construct a semigroup $(\{\delta_x : x \in \Sigma^*\}, o)$. The neutral element is $\delta_\in$ where $\in$ is an empty word.

Pushdown automata algebraization

In pushdown automata we have a structure of the form $(Q, \Sigma, \Gamma, \Delta, \delta)$ with the following meaning :

Q — a set of states,

Σ — a set of input symbols,

Γ — a set of stack symbols,

Δ — a set of output symbols,

Δ — a control function $\Sigma \times Q \times \Gamma \rightarrow Q \times \Gamma^* \times \Delta^*$

That is, $\delta(x, S_1, g) = (S_2, g_1 \cdots g_k, y)$, being in state S_1 with g on the top of the stack, under input x the automat switches to state S_2, substitutes g with $g_1 \cdots g_k$ and produces output y.

Similarly, we will expand the δ function. At first, we make it to accept an extra argument from Δ^* and also, we want it to take the third argument from Γ^*, not Γ. To achieve this, we can define five auxiliary functions, Let's call them $\gamma_1$and γR.

$\delta_Q : Q \times \Gamma^* \times \Delta^* \to Q, \delta_Q(S, g, y) = S$

$\delta_\Gamma : Q \times \Gamma^* \times \Delta^* \to \Gamma^*, \delta_\Gamma(S, g, y) = g$

$\delta_\Delta : Q \times \Gamma^* \times \Delta^* \to \Delta^*, \delta_\Delta(S, g, y) = y$

$\gamma_1 : \Gamma^* \to \Gamma, \gamma_1(g_1 \cdots g_k) = g_1$

$\gamma R : \Gamma^*, \gamma R((g_1 \cdots g_k) = g_1 \cdots g_k$

Thus, $\delta(x, S_1, g_1) = (S_2, h_1 \cdots h_{k2}, z_1 \cdots z_{m2})$

We perform the previous δ function on the input, state and top of the stack, substitute what δ says for the top of the stack, add what δ produces to the output and switch to the new state. Finally, we expand even more the δ function to accept not only symbols from Σ, but words from Σ^* and create δ_x functions. The catenation formula is the same as regular languages and again we have a semigroup structure $(\{\delta_x : x \in \Sigma^*\}, 0)$.

Translators algebraization

The case of translators, both finite and with pushdown ones, is nearly identical to the cases of finite automata and pushdown automata. The only difference is that we may have two outputs, the first output because of a translator definition — this is the one where words from the second language appear in the response to the input and the second output the same as the ones in previous sections — just for printing some messages of informing the user about the internal state of automata. However, this change is of a small importance and doesn't change anything in the formalism except that we have to catenate two know outputs instead of one. There are no changes in the process of defining a semigroup structure.

In general, the above procedure is described as follows.

Let us have an automat given by (X, Q, Δ, δ) where.

X — a set of input elements that can be catenated.

Q — a set of automata states,

Δ — a set of output elements,

δ — a control function $X \times Q \to Q \times \Delta^*$

That is, $\delta(x, S_1) = (S_2, y)$, being in stste S_1 under input x the automat switches to state S_2 and produces output y.

We expand the δ function to be $\delta : X^* \times Q \times \Delta^* \to Q \times \Delta^*$ by catenating the output to the already existing one and by calling the δ function repeatedly as many times as the length of the input. Then we define δ_x functions and a catenation for them in natural way. We take the neutral element and we get a semigroup structure of $(\{\delta_x : x \in X^*\}, 0)$.

Now, we focus on a non-deterministic automat to construct a semigroup as follows :-

Subautomat → An automat B = $(X^B, Q^B, \Delta^B, \delta^B)$ is a subautomat of automat

A = $(X^A, Q^A, \Delta^A, \delta^A)$

$$X^B \subset X^A, Q^B \subset Q^A, \Delta^B \subset \Delta^A, \delta^B = \delta^A \Big|_{P(X^{B*} \times Q^B \times \Delta^{B*})}$$

Theorem: - If there is an automat A, B subautomat of A, and G_A *and* G_B are the corresponding semigroups, then G_B is a homomorphic image of some subsemigroup G of a semigroup G_A.

Proof: Let G=$\{\delta_X^A : x \in X^{B*}\}$ and $\emptyset : G \to G_B$ whose result is the argument contracted to P($Q^B \times \Delta^{B*}$). G is a semigroup because it is generated by a subset of G_A generators. The result of $\emptyset$ is some function $\delta_x : P(Q^B \times \Delta^{B*}) \to P(Q^A \times \Delta^{A*})$.

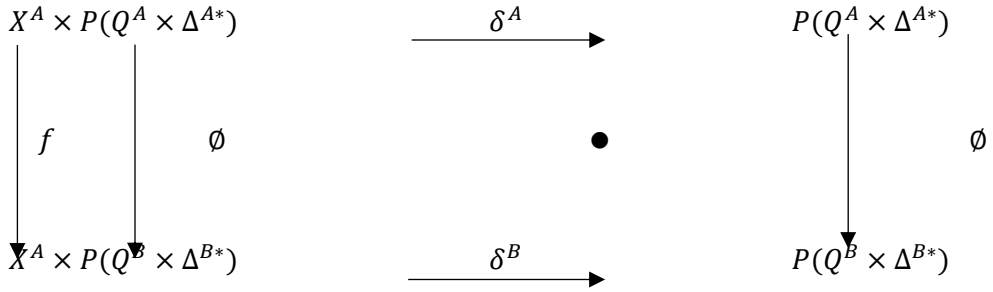
Since, B is a subautomata of A and we know that δ^B is a contraction of δ^A, δ_x leads into P($Q^B \times \Delta^{B*}$)and therefore belongs to G_B.

Thus we have $\emptyset(\delta_x^A) = \delta_x^B$

$\emptyset$ is a homomorphism: -

$$\emptyset(\delta_y^A o \delta_x^A) = \emptyset(\delta_{xy}^A) = \delta_{xy}^B = \delta_y^B o \delta_x^B = \emptyset(\delta_y^A) o \emptyset(\delta_x^A)$$

Homomorphic Image → An automat B is a homomorphic image of an automat A if there exist functions f: $X^A \to X^B$ and $\emptyset : P(Q^A \times \Delta^{A*}) \to P(Q^B \times \Delta^{B*})$ such that the following diagram is commutative :



i.e. $\emptyset \; o \; \delta^A = \delta^B \; o \; (f \times \emptyset)$

In other words, $\forall x \in X^A, \; \forall Z \in P(Q^A \times \Delta^{A*})$

$$\emptyset(\delta_x^A(Z)) = \delta_{f(x)}^B(\emptyset(Z))$$

Theorem:- If automat B is a homomorphic image of an automat A then a semigroup G_B is a homomorphic image of a semigroup G_A.

Proof. We have $\emptyset\ o\ \delta_x^A = \delta_{f(a)}^B o\ \emptyset, a \in X^A$.

Here, we extend this equation to the words from X^{A*}because

$$\emptyset\ \boldsymbol{o}\ \boldsymbol{\delta_a^A} = \emptyset\ \boldsymbol{o}\ \boldsymbol{\delta_{xn}^A}\ \boldsymbol{o}\cdots\boldsymbol{o}\boldsymbol{\delta_{x_1}^A} = \boldsymbol{\delta_{f(x_n)}^B o \emptyset o \delta_{x_{n-1}}^A o}\cdots\boldsymbol{o}\boldsymbol{\delta_{x_1}^A} = \cdots$$

$$\cdots = \boldsymbol{\delta_{f(x_n)}^B o}\cdots\boldsymbol{o}\boldsymbol{\delta_{f(x_1)}^B o}\ \emptyset = \boldsymbol{\delta_{f(x_1)}^B o}\ \emptyset$$

Where $f(x)$ is $f(x_1)\cdots f(x_n)$. Let us define a function η which we will prove homomorphism.

$$\eta(\delta_x^A) = \delta_{f(x)}^B$$

The function η is a well defined function because if two different inputs give the same δ^A function then δ^B functions for their images through the f function are identical:

$$\delta_x^A = \delta_y^A \Longrightarrow \emptyset\ o\ \delta_x^A = \emptyset\ o\ \delta_y^A \Rightarrow \delta_{f(x)}^B o\ \emptyset = \delta_{f(y)}^B o\ \emptyset$$

The function $\emptyset$ is a surjection so $\delta_{f(x)}^B = \delta_{f(y)}^B$.

η is a homomorphism: -

$$\eta(\delta_y^A\ o\ \delta_x^A) = \eta(\delta_{xy}^A) = \delta_{f(xy)}^B = \delta_{f(x)f(y)}^B = \delta_{f(y)}^B o\ \delta_{f(x)}^B = \eta(\delta_y^A)\ o\ \eta(\delta_y^A)$$

Conclusions: -

Thus, from the above discussion we see that there is no problem with identifying formal automata with Semigroups. Although there have been much effort put into research on formal languages there is still much to be done on the theoretical side of the problem.

References :

1. [Hop Ull69] Hoperoft J.E., Ullman J.D.; Formal languages and their Relation to Automata, Addison-Wesley, Reading MA, 1969.
2. [Fla93] Flasinski M.; On the parsing of deterministic graph languages for syntactic pattern recognition, Pattern Recognition, Vol. 26, Pergamon Press Ltd 1993.
3. Eilenberg, S. (1974) Automata, Languages and Machines, 2 Vols. Academic Press, New York.
4. Ginsburg, A. (1968) Algebraic Theory of Automata. Academic Press, New York.
5. Pin, Jean-Eric (2012) Mathematical Foundations of automat theory.
6. Saloman, A. (1981) Jewels of Formal Language Theory, Pitman, London.