BACK TO BASICS: DECOMPOSING FULL TRANSFORMATION SEMIGROUPS

ATTILA EGRI-NAGY

ABSTRACT. Transformation semigroups are models of discrete dynamical systems. We can use their hierarchical decompositions as cognitive tools to get insights about the underlying structure and dynamics. The Krohn-Rhodes theory gives mathematical guarantee for the existence of these decompositions. However, the details of obtaining decompositions in practice, and the ways of using them for understanding are still work in progress. Here, we look at the decompositions of familiar algebraic objects, the full transformation semigroups, to provide an elementary approach to this kind of algebraic modeling.

Here we use the Covering Lemma method [6, 2], the easiest algorithm for the Krohn-Rhodes theory [5], to decompose the nicest possible example of full transformation semigroups. The purpose here is educational (bordering entertainment), thus very little new findings are presented. We put emphasis on the explanation and accessibility. The typeface is chosen accordingly, following the classical example-driven textbook *Concrete Mathematics* [4].

First we give a minimal set of definitions, then describe a general algorithm for two-level decompositions, and finally we investigate how the algorithm works on the full transformation semigroups.

1. PRELIMINARIES

Here, we define the objects we are interested in, the ways we can combine them into more complex structures, and how to establish relationships between them. For more details, see [1, 2].

1.1. The algebraic objects.

In essence, we talk about *semigroups*, sets closed under an associative binary operation. However, we are interested in one particular representation, the one closest to automata theory.

Definition 1.1 (Transformation Semigroup). A *transformation semigroup* (X, S) is a finite nonempty set of *states* (points) X and a set S of total transformations of X, i.e., functions of type $X \to X$, closed under composition.

We also say that S *acts* on X, expressed as a functions of type $X \times S \to X$. For this action on the right we write $x \cdot s$, or simply xs.

The action is *faithful* if $\forall x \in X, x \cdot s_1 = x \cdot s_2 \implies s_1 = s_2$ for all $s_1, s_2 \in S$. When the action is not faithful, we can identify transformations if they act the same way on X, thus defining an equivalence relation $\equiv$. Then the quotient semigroup $S/\equiv$ is faithful on X.

The *image* of a transformation s is $\text{Im}(s) = \{x \cdot s \mid x \in X\}$. The stabilizer of a set of states $A \subseteq X$ is the subsemigroup $\text{Stab}_S(A) = \{s \in S \mid A \cdot S \subseteq A\}$, i.e., stabilizing A in the weaker sense of not leaving A. We write $S|_A^{\equiv}$ for $\text{Stab}_S(A)$ made faithful. It is the image of a surjective homomorphism from a subsemigroup of S. Such an image is often called a *divisor* of S.

1.2. **Decompositions.** We combine transformation semigroups into bigger ones, where the components have asymmetric roles: the top level just performs its computation, while the bottom level listens to the top level state in order to decide what to do. The control information flows one way, thus we have a *hierarchical* system.

Definition 1.2 (Wreath Product). The *wreath product* $(X,S) \wr (Y,T)$ of transformation semigroups is the *cascade transformation semigroup* $(X \times Y, W)$ where

$$W = \{(s,d) \mid s \in S, d \in T^X\},$$

whose elements map $X \times Y$ to itself as follows

$$(x,y) \cdot (s,d) = (x \cdot s, y \cdot d(x))$$

for $x \in X, y \in Y$. We call $d : X \to T$ a dependency function.

Definition 1.3. We call a (proper) subsemigroup of the wreath product a *cascade product*. We denote such a product operation by $\wr_{\supset}$, indicating that it is a substructure and also hinting the direction of the control flow.

1.3. **Relationships.** For expressing relationships between transformation semigroups we use the usual idea of structure preserving maps, the so-called homomorphisms. However, we use relations instead of functions, and we need a pair of them for the semigroup action.

Definition 1.4 (Relational Morphism). A *relational morphism* of transformation semigroups $(X,S) \xrightarrow{\theta,\varphi} (Y,T)$ is a pair of relations $(\theta : X \to Y, \varphi : S \to T)$ that are fully defined, i.e., $\theta(x) \neq \emptyset$ and $\varphi(s) \neq \emptyset$, and satisfy the condition of compatible actions for all $x \in X$ and $s \in S$:

$$y \in \theta(x), t \in \varphi(s) \implies y \cdot t \in \theta(x \cdot s),$$

or more succinctly: $\theta(x) \cdot \varphi(s) \subseteq \theta(x \cdot s)$.

2. The Covering Lemma Decomposition Method

To investigate the structure of a transformation semigroup (X,S), we construct a surjective relational morphism:

$$(X,S) \xrightarrow{R(\theta,\varphi)} \!\!\!\!\!\!\twoheadrightarrow (Y,T).$$

The target semigroup (Y,T) serves as an *approximation* of (X,S). There are two extreme cases of approximations. When (Y,T) is the trivial transformation monoid, all structural information is lost through R. When $(Y,T) \cong (X,S)$, information is completely transferred through the morphism. For a successful study of (X,S) we want something in-between. We want to control what is preserved and omitted in the process of coarse-graining.

Given an approximation, we can build an *emulation*:

$$(X,S) \xhookrightarrow{E(\psi,\mu)} (Y,T) \wr_{\supset} (Z,U).$$

The central idea of the method is that the bottom level component (Z,U) recovers all the information in (X,S) missing from (Y,T).

Using a physical metaphor, we can say that elements of Y are *macro* states. The macro dynamics is defined by the action of T. For each $y \in Y$, we have a subset of Z representing the corresponding *micro* states. T only moves the macro states, and the transformations in U only make sense with respect to a top level macro state. We connect the micro states in Z (corresponding to a y) to the original states in X by a partially defined invertible labelling function $w_y : X \to Z$. The labelling needs to be defined only for $\theta^{-1}(y)$. It is arbitrary but fixed for a macro state.

The statistical mechanics interpretation keeps appearing in these decompositions.

Now we have all the ingredients to define the relational morphism for the emulation.

$$\psi(x) = \{(y, x \cdot w_y) \mid y \in \theta(x)\},$$
$$\mu(s) = \{(t, w_y^{-1} s w_{yt}) \mid t \in \varphi(s), y \in \operatorname{Im}\theta\}.$$

3. Decomposing Full Transformation Semigroups

The *full transformation semigroup of degree* n, $\mathcal{T}_n$, is the set of all transformations of an n-element set, canonically denoted by positive integers $\{1, \dots, n\}$. It is a monoid, and it is the semigroup analogue of the *symmetric group*, $\mathcal{S}_n$, the group of all permutations of degree n. Also, $\mathcal{S}_n \leq \mathcal{T}_n$. Both are examples of what we call classical transformation semigroups [3].

When working with $\mathcal{T}_n$, we do not need to worry about the existence of a particular transformation. They are all there.

For transformations of small degree we can use a condensed notation. For instance, $\left(\begin{smallmatrix}1&2&3&4\\1&3&2&2\end{smallmatrix}\right)$ can be written as 1322.

3.1. R for $\mathcal{T}_n$.

When we build a surjective relational morphism for a transformation semigroup, the most straightforward method is to identify some states, get the smallest congruence (equivalence relation on the states compatible with the action), and see what action it induces. However, for the minimal-degree transformation representation of $\mathcal{T}_n$, there is no nontrivial equivalence relation on $\{1, \dots, n\}$ compatible with the action of $\mathcal{T}_n$. If we try to identify any two points and find the coarsest compatible partition, we end up with a single class containing all points. Thus we need to use the peculiar feature of relational morphism: the image sets can overlap. In the physical metaphor this means that a micro state can be part of several macro states.

3.1.1. *Mapping the states: θ for $\mathcal{T}_n$.*

We map each state $X = \{1, 2, \dots, n\}$ to a set of states by $\theta(x) = X \setminus \{x\}$. Each state goes to the set of all states but itself. For instance, if $n = 4$,

It will become clear later why this weird 'backwards' relation works.

$$\theta(1) = \{2,3,4\},\ \theta(2) = \{1,3,4\},\ \theta(3) = \{1,2,4\},\ \theta(4) = \{1,2,3\}.$$

When computing the preimages, we see that $\theta = \theta^{-1}$, since a state appears in all image sets, except in the image of itself.

3.1.2. *Mapping the transformations: φ for $\mathcal{T}_n$.*

For a transformation $s \in S$, we distinguish between two possibilities: s is a permutation, or it is not, i.e., it collapses some states. We map a permutation to itself, or to be more precise, to the singleton set containing the permutation. If s is not a permutation, then its image has missing states. We form *constant* maps to those states. Again, this is backwards, matching θ. Formally,

$$\varphi(s) = \begin{cases} \{s\} & \text{if } X \cdot s = X, \\ \{c_j \mid j \notin \operatorname{Im} s\} & \text{if } X \cdot s \subsetneq X. \end{cases}$$

For instance, in $\mathcal{T}_5$, $\varphi(23451) = \{23451\}$, $\varphi(22333) = \{c_1, c_4, c_5\}$, and $\varphi(c_1) = \{c_2, c_3, c_4, c_5\}$.

3.1.3. *Why does $R(\theta,\varphi)$ work as a relational morphism?* The 'backwards' relations θ, φ have a simple intuitive explanation. Composing transformations can only reduce the size of the image. To make the morphism work, we need a bigger $\varphi(s)$ set for smaller $\mathrm{Im}(s)$.

Let's see how the cycle $p = 23451$ acts on $x = 2$. It is just a permutation action: $2 \cdot p = 3$. This result in Y is $\theta(3) = \{1,2,4,5\}$, X with a 'hole' at 3. Now, let's do the action in (Y,T): $\theta(2) = \{1,3,4,5\}$ and $\varphi(p) = \{p\}$. We have the same permutation in T by φ, thus $\{1,3,4,5\} \cdot p = \{2,4,5,1\} = \{1,2,4,5\}$. The hole moves exactly the same way as 2 would do. Therefore, $\theta(x) \cdot \varphi(p) = \theta(xp)$, the action is compatible for permutations.

Moving a point is the same as moving a corresponding hole.

As a non-permutation example, 22333 moves 4 to 3, and $\theta(3) = \{1,2,4,5\}$. On the other hand, $\theta(4) = \{1,2,3,5\}$ and $\varphi(s) = \{c_1, c_4, c_5\}$. Those constant maps produce $\{1,4,5\}$, no matter what set we apply them to. Therefore, $\theta(4) \cdot \varphi(s) \subset \theta(4 \cdot s)$; the relational morphism works in this case. It works in general too: $\theta(xs)$ will have a single hole for some $x' \in \mathrm{Im}(s)$, but it will have all other states. By definition, $\varphi(s)$ will produce the set $X \setminus \mathrm{Im}(s)$, thus this set will not contain x', the only state that could disprove the subset relation.

Similar argument can show that φ is a relational morphism for semigroups as well, i.e., $\varphi(s_1) \cdot \varphi(s_2) \subseteq \varphi(s_1 \cdot s_2)$. For a non-permutation s_2, $\varphi(s_1) \cdot \varphi(s_2) = \varphi(s_2)$, since constant transformations are right zeroes.

3.2. Emulation.

At this stage, we have $R(\theta,\varphi)$ defining a surjective morphism. To build an emulation by a cascade product, we need to construct (Z,U), the bottom level components. To be more precise, we will have a transformation semigroup (Z, U_y) for each top level state y. Here is the main intuition. If we fix a macro state $y \in Y$, so nothing happens on the top level, there can still be action on the preimages $\theta^{-1}(y)$, back in the original transformation semigroup (X,S). We want to copy that action to this bottom level component. The preimage states $\theta^{-1}(y)$ will be the micro states Z (after some technical relabelling). The micro dynamics semigroup U_y will be then $S|^{\equiv}_{\theta^{-1}(y)}$.

Since θ misses only a single point, we will have $n-1$ states on the second level. It is true in general that $|Z| = \max_y |\theta^{-1}(y)|$. For a top level state y, we have a hole in the state set, so we need a labelling function to take care of that. We define $w_y : \theta^{-1}(y) \to Z$ by

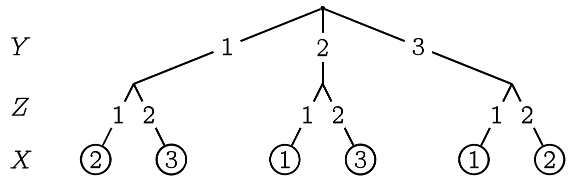
We can, of course, use more states in Z and avoid reusing them in different contexts.

$$w_y(x) = \begin{cases} x & \text{if } x < y, \\ x - 1 & \text{if } x > y. \end{cases}$$

Anything above the hole moves down. Note that w_y is partial since it is not defined for y itself. It is also invertible. This gives a $n(n-1)$ cascade transformation representation of a degree n transformation semigroup. What are those coordinatized states?

3.2.1. *Coordinatized States.* We lift states x to pairs of states (y,z). There are $|\theta(x)|$ coordinate pairs for each state.

For $\mathcal{T}_3$, we have $1 \mapsto \{(2,1),(3,1)\}$, $2 \mapsto \{(1,1),(3,2)\}$, $3 \mapsto \{(1,2),(2,2)\}$, which does not seem to make too much sense. However, drawing a tree reveals the pattern.



The coordinates directly encode θ, or rather θ^{-1}, which is the same in this case. In general, the Z-coordinates under a top level state y encode the states in $\theta^{-1}(y)$.

3.2.2. *Cascade Transformations.* For each $s \in S$ we will have as many cascade transformations as many lifts s has under φ, i.e., $|\varphi(s)|$. The independent, top level action is defined by $t \in \varphi(s)$. For each such $t \in \varphi(s)$, we need to go through all top level states $y \in Y$ and define the value of the dependency function. In other words, we need to give a transformation in U for a state y. This is defined by $w_y^{-1} s w_{yt}$. To spell it out, w_y^{-1} takes states in Z to the original states in X. The image of the mapping is $\theta^{-1}(y)$, missing a single state. Since we are in X, s acts on these points, so we do that. Then we come back to Z by a different mapping, by w_{yt}, since we did action xs and its compatible counterpart yt, so the top level state is (potentially) changed.

What action is transferred from S? On the top level we have all permutations and all the constant maps. The inclusion of the constant maps is denoted by a bar over the component. On the bottom level, everything on $n-1$ points, which is $\mathcal{T}_{n-1}$. Thus we have the emulation:

Permutation group augmented with constant transformations is a *permutation-reset* component.

$$\mathcal{T}_n \hookrightarrow \overline{\mathcal{S}_n} \wr_{\circ} \mathcal{T}_{n-1}.$$

The construction can be iterated, therefore we have the standard result

$$\mathcal{T}_n \hookrightarrow \overline{\mathcal{S}_n} \wr_{\circ} \overline{\mathcal{S}_{n-1}} \wr_{\circ} \cdots \wr_{\circ} \overline{\mathcal{S}_2}.$$

3.3. **Investigating the cascade product solution.** What is $\overline{\mathcal{S}_n} \wr_{\circ} \mathcal{T}_{n-1}$ exactly? It is a cascade transformation representation of $\mathcal{T}_n$ on $n(n-1)$ coordinate pairs.

How big is this cascade product? How many lifts does φ produce? The number of transformation in $\mathcal{T}_n$ with image size k is $\left\{ {n \atop k} \right\} \frac{n!}{(n-k)!}$. The Stirling number of the second kind $\left\{ {n \atop k} \right\}$ gives the number of partitions of the n-element state set with exactly k classes. The equivalence is defined by having the same image. Then we need to choose k states, the image values. Thus, we have $\binom{n}{k}$, but the order of the classes matter, so we multiply by $k!$. To calculate the number of lifts, we need to sum over all possible image sizes with the multiplicity by the number of lifts and add the number of permutations:

$$|\varphi(\mathcal{T}_n)| = n! + \sum_{k=1}^{n-1} (n-k) \left\{ {n \atop k} \right\} \frac{n!}{(n-k)!}.$$

We can see how the difference is growing for the first few values of n. The difference is growing steadily.

n	2	3	4	5	6	7
$\lvert\mathcal{T}_n\rvert = n^n$	4	27	256	3125	46656	823543
$\lvert\varphi(\mathcal{T}_n)\rvert$	4	30	348	52407	94470	1964592

After all, the nice example of full transformation semigroups does not produce the most 'economical' decompositions, due to the need of overlapping image sets in the relations of the morphism.

References

[1] Attila Egri-Nagy. "Finite Computational Structures and Implementations: Semigroups and Morphic Relations". In: *International Journal of Networking and Computing* 7.2 (2017), pp. 318–335. DOI: `10.15803/ijnc.7.2_318`.

[2] Attila Egri-Nagy and Chrystopher L. Nehaniv. *From Relation to Emulation and Interpretation: Implementing the Covering Lemma for Finite Transformation Semigroups*. 2024. DOI: `10.48550/arXiv.2404.11923`. arXiv: `2404.11923 [math.GR]`.

[3] Olexandr Ganyushkin and Volodymyr Mazorchuk. *Classical Transformation Semigroups*. Algebra and Applications. Springer, 2009.

[4] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Second Edition. Reading, MA: Addison-Wesley, 1994.

[5] Kenneth Krohn and John Rhodes. "Algebraic Theory of Machines. I. Prime Decomposition Theorem for Finite Semigroups and Machines". In: *Transactions of the American Mathematical Society* 116 (Apr. 1965), pp. 450–464.

[6] Chrystopher L. Nehaniv. "From relation to emulation: The Covering Lemma for transformation semigroups". In: *Journal of Pure and Applied Algebra* 107.1 (1996), pp. 75–87. DOI: `10.1016/0022-4049(95)00030-5`.

AKITA INTERNATIONAL UNIVERSITY, JAPAN
Email address: `egri-nagy@aiu.ac.jp`