COMPACT NOTATION FOR FINITE TRANSFORMATIONS

ATTILA EGRI-NAGY[1,2] AND CHRYSTOPHER L. NEHANIV[2]

Abstract. We describe a new notation for finite transformations. This compact notation extends the orbit-cycle notation for permutations and builds upon existing notations. It gives insight into the structure of transformations and reduces the length of expressions without increasing the number of types of symbols.

Contents

1. Introduction

What is the right generalization of the orbit-cycle notation for permutations to the case of arbitrary total functions $f : X \to X$ on a set X? The discrete dynamical system that f gives on X can be visualized as digraph with nodes $x \in X$ and directed edges $(x, f(x))$. Iterating f maps a point to x to $f(x)$, to $f(f(x))$, and so on, until eventually some $f^m(x) = f^{m+k}(x)$. Taking the least $m \geq 0$ for which this happens, and the least positive k for that m, shows that x eventually must enter a (possibly degenerate) periodic orbit from which it never leaves. Points x and x' eventually entering the same periodic cycle are said to be in the same *basin of attraction* or *generalized cycle*. Drawn as digraphs, transformations may have several such disjoint basins of attraction, each consisting of a cycle of points with incoming trees (connected acyclic subgraphs). Figure 1 gives an example with four generalized cycle components in our notation, which is formally introduced in Section 3 with further examples. Unlike the permutation case, the points in a cycle can have incoming edges from outside the cycle, so a notation for transformations has to deal with these tree structures. The trees, which could also be degenerate,

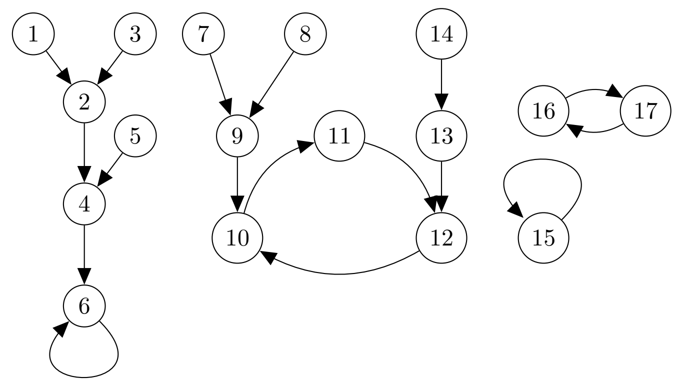


FIGURE 1. A simple discrete dynamical system. A transformation $g : X \to X$ with four components, i.e. basins of attraction (generalized cycles), is visualized as a digraph with arrows $(x, g(x))$ for each $x \in X = \{1, \ldots, 17\}$. In canonical form in our compact notation, g is denoted `[[[1,3|2],5|4],6]([[7,8|9],10],11,[14,13,12])(16,17)`. Note: Singleton components are not written, just as in orbit-cycle notation for permutations. Full, formal details for the notation are given in the text.

are directed toward the cycle, and there may be a tree of points coming into any given point of the cycle.

We call each basin of attraction associated to f, a generalized cycle or a *component*, since it a connected component of the digraph of f. We may restrict our attention to a single component only, since we can write f by concatenating our notation for what f does on each of its components.

Various previous notations have been developed for representing such transformations on a set (Section 2). The aim of these notations is to give useful information about the transformation without drawing the corresponding digraph. Readability is not an easily measurable quantity, but length and the number of distinct symbols used are influencing factors. The real importance of an efficient notation for transformations lies in the growing use of computer algebra systems in finite semigroup theory, as well as in mathematical calculations with transformations.

2. Existing Notations

Several ways for generalizing cyclic notation have been developed based on the particular purposes of each situation in which they were defined. Here we give a quick summary of previous suggestions. Our running example, whose digraph is visualized in Figure 2, with just one nontrivial component (basin of attraction) will be the transformation typically written as

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 2 & 3 & 3 \end{pmatrix},$$

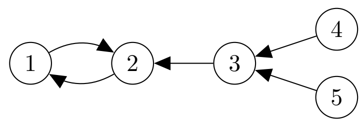


FIGURE 2. An irreversible transformation on 5 points with just one basin of attraction. Here a non-degenerate tree collapses in two iterations into a cyclic permutation $(1, 2)$.

This common notation denotes the function on the set $\{1, 2, 3, 4, 5\}$ taking an element x in the first row to the element written $f(x)$ immediately below it on the second row.
Caveat: As it usually happens in mathematics, the following notations associate different meanings to the same symbols.

2.1. **Path Notation for Partial Symmetries and Partial Transformations.** In addition to parentheses `(,)` for permutation orbits, path notation introduced by S. Lipscomb [4, 3] uses square bracket `]` following elements with no images in partial permutations. For partial transformations the paths connect into other paths and ultimately into the cycles. This is denoted by the symbol `>`, may be used as a visual indication of being funneled into a cycle. The example written in path notation is

`(1,2)(4,3,2>(5,3,2>`,

which redundantly represents the paths going into point 2.

2.2. **Factorization Notation.** In order to avoid path redundancy, G. Ayik, H. Ayik, and J.M. Howie [1] introduced a different notation. Instead of decomposing the transformation into paths and cycles, this notation decomposes it into a very particular type of generalized cycle, i.e. into a product of transformations given by the trajectory of a single point. The example written in path notation is

`[4,3,2,1|2][5,3|3]`,

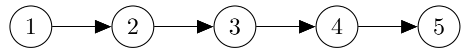
where the element after the `|` shows where the path connects to itself to form a cycle. This notation is excellent for describing factorizations in the full transformation semigroups, but it introduces maps that move things in ways are not present in the original transformation. For instance, $3 \mapsto 3$. Also, the decomposition is not unique.

2.3. **Linear Total Transformations.** In [2] a new notation was introduced by O. Ganyushkin and V. Mazorchuk, aiming to provide a natural extension of the cyclic notation of permutations. Instead of decomposing trees into paths, linear notation describes the trees explicitly. Trees with one level of branching are denoted by `[`preimages of root `;`root `]`, where preimages are separated by commas and all map to the root. If a preimage element also has incoming edges from other points, then the same square bracket structure is applied again recursively to specify the tree leading to that element. Although called 'linear' as a one-line notation, the notation describes recursive tree structure.

Parentheses are used to indicate the existence of a nontrivial permutation of the roots of the trees (which may include degenerate trees consisting of a single point). Basically the linear notation is the usual cyclic notation used for permutations but the elements in the cycle describe their incoming tree information. The running example is written as

`([[4,5;3];2],1)`.

One of the most useful features of linear notation is that when restricted to permutations it is identical to orbit-cycle form. Also, by looking for parentheses we can easily spot the existence of nontrivial permutations even in large examples. The only drawback is that describing a transformation given by following a simple path, a "line" of maps, requires many square brackets. For instance,



becomes `[[[[1;2];3];4];5]`. With the compact notation we aim to alleviate this particular problem.

3. The Compact Notation

Informally, the compact notation makes use of the following conventions and symbols. Formal semantics of the notation are in the next subsection. Here we illustrate the concepts for points:

(1) Left-to-right comma-separated enumeration indicates a "conveyor belt" of maps. For example, `1,2,3` reads as $1 \mapsto 2$, $2 \mapsto 3$.
(2) Parentheses containing a left-to-right comma-separated enumeration of points add the extra map from the last element to the first element of the enumeration, i.e. `(1,...,n)` adds the map $1 \mapsto n$.
(3) Square brackets containing a left-to-right enumeration of points leave the image of the last element undefined. Therefore they can be used to denote partial mappings, or for total trajectories terminating in a trivial cycle. If something follows the closing square bracket in a left-to-right order, then the image of the last element is defined by that following element. For example `...n],k...` defines the map $n \mapsto k$.
(4) A vertical bar `|` ("splat") appearing before the last element in a square bracket turns the preceding sequence into a set and maps all of its elements into the last point. They all 'hit the same wall'. For instance, `[1,2,3|4]` yields the maps $1 \mapsto 4$, $2 \mapsto 4$ and $3 \mapsto 4$. (Note, this is an exception to left-to-right mapping mentioned in (1).)

However, one may replace "point" by "tree" in a comma-separated list in a parenthesized cycle, to the left of a splat `|`, or in a conveyor belt, in which case the roots of the trees are mapped like a point in the corresponding position in (1)-(4) above. If a trees appear in the position of k in (3), then n is mapped to the root of the that tree. This discussion is made precise below.

3.1. **Examples.** Any permutation in the compact notation is the same as the cyclic notation. In particular, the identity is simply `()`.

A constant map to n is written as `[1,2,...,n-1|n]`, while a simple path trajectory from 1 to n is denoted by `[1,2,...,n]`. The difference is big, and that is why we changed the linear notation's `;` symbol to `|`. Otherwise, the little dot in the semicolon, which might be easily overlooked, would have the same decisive role in distinguishing these two situations. The notation for a simple trajectory makes the compact notation shorter compared to the linear notation. In other cases they are identical.

The transformation whose digraph is in Figure 3 on 9 points is written as

`([[8,9|5],6,7|1],[4,3,2])`.

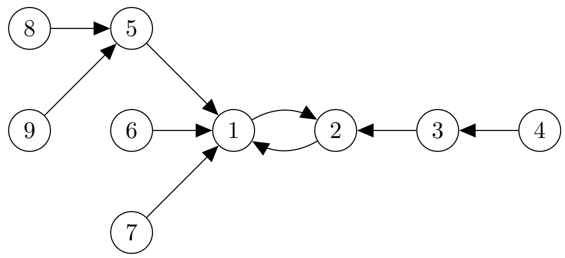


FIGURE 3. Digraph for an example transformation with a 'conveyer belt' trajectory `[4,3,2]` and a branching tree `[[8,9|5],6,7|1]` collapsing into a transposition `(1,2)`. In compact notation this is denoted `([[8,9|5],6,7|1],[4,3,2])`

3.2. **Syntax.** The following context-free grammar defines the language of compact notations. The terminal symbols are `[`,`]`,`(`,`)`,`,`,`|` and the symbols for the n points. The nonterminal symbols are C for components, N for nontrivial trees, T for trees and P for points.

$$S \to C^+ \mid \texttt{()} \tag{1}$$

$$C \to \texttt{(}(T\texttt{,})^+T\texttt{)} \mid N \tag{2}$$

$$N \to \texttt{[}(T\texttt{,})^+T \mid P\texttt{]} \mid \texttt{[}(T\texttt{,})^+T\texttt{]} \tag{3}$$

$$T \to N \mid P \tag{4}$$

$$P \to \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \ldots \mid \texttt{n} \tag{5}$$

Rule (1) states that the compact notation denotes a positive number nontrivial components (generalized cycles, i.e. basins of attractions), or we can have no nontrivial component (for the identity transformation). Rule (2) says that a component can be a nontrivial tree or a permutation cycle (with incoming trees to its members). Rule (3) describes the two cases of a nontrivial tree: either a tree with multiple branches leading to the root ("splat") or a path ("conveyor belt"'). Rule (4) allows a point to be tree. Rule (5) specifies the points. **An important additional constraint in the notation is that each point can only occur at most once.**

3.3. **Semantics.** We define the semantics by recursively interpreting the valid compact notation words as a collection of individual maps of points. We write such a map $p \mapsto q$ as a pair (p, q). We will put the total function f together from these pieces. Using the parse tree of a well-formed word w in the notation, we denote its interpretation by $\mathcal{I}(w)$. This determines a unique function f from $\{1, \ldots, n\}$ to itself, by identifying f with the set of pairs $\mathcal{I}(w)$. If $w = \texttt{()}$, then $\mathcal{I}(w)$ is the identity transformation. Otherwise w is derived using $S \to C_1 \ \ldots C_k$, and we define

$$\mathcal{I}(S) := \bigcup_{i=1}^{k} \mathcal{I}(C_i) \cup \{(p,p) \mid f(p) \text{ is not defined in any } \mathcal{I}(C_i)\}. \tag{6}$$

An auxiliary function r gives the root of a tree, so we let

$$r(p) = r(\texttt{[}T_1\texttt{,} \ldots \texttt{,}T_k \texttt{|} p\texttt{]}) = p \in \{1, \ldots, n\}.$$

$$\text{and } r(\texttt{[}T_1, \ldots, T_k\texttt{]}) = r(T_k).$$

Then the interpretation of a component derived from a nonterminal symbol C is:

$$(7)\qquad \mathcal{I}(C) := \begin{cases} \mathcal{I}(N) & \text{if } C \to N \\ \\ \bigcup_{i=1}^{k} \mathcal{I}(T_i) & \text{if } C \to \texttt{(}T_1\texttt{,}\ldots\texttt{,}T_k\texttt{)} \\ \quad \cup \{(r(T_i), r(T_{i+1})) : 1 \le i < k\} & \\ \quad \cup \{(r(T_k), r(T_1))\} & \end{cases}$$

A tree gives a nonempty set of pairs, if it is nontrivial :

$$(8)\qquad \mathcal{I}(T) := \begin{cases} \mathcal{I}(N) & \text{if } T \to N \\ \varnothing & \text{if } T \to P \end{cases}$$

A nontrivial tree gives a set of pairs as follows:

$$(9)\qquad \mathcal{I}(N) := \begin{cases} \bigcup_{i=1}^{k} \mathcal{I}(T_i) \cup \{(r(T_i), p) : 1 \le i \le k\} & \text{if } N \to \texttt{[}T_1\texttt{,}\ldots\texttt{,}T_k \texttt{|} p\texttt{]} \\ \\ \bigcup_{i=1}^{k} \mathcal{I}(T_i) & \text{if } N \to \texttt{[}T_1\texttt{,}\ldots\texttt{,}T_k\texttt{]} \\ \quad \cup \{(r(T_i), r(T_{i+1})) : 1 \le i < k\} & \end{cases}$$

The notation w thus clearly determines a unique well-defined transformation $f = \mathcal{I}(w)$ on $\{1, \ldots, n\}$ to itself, since each element p appears at most once in w. The transformation is total (i.e, not partial, but fully defined) by Equation 6. Moreover, every f can be written in this compact notation in a canonical form.

3.4. **Canonical Form.** The price to pay for the short length is the loss of uniqueness. Both `[1,2,[3,4|6]]` and `[[1,2],3,4|6]` denote the same transformation. One can simply choose between a conveyor belt (comma-separated list of elements in a trajectory) or the splat motif (using `|`). However, a simple recursive algorithm that starts from the point(s) of each component's cycle can produce a canonical form. All we need to do is to examine the cardinality of the preimage set from outside the cycle, i.e. the number of incoming arrows.

(1) If there is no incoming arrow, then we have a leaf of the tree, only the point needs to be printed.
(2) If there is only one preimage, then a conveyor belt is built by traversing the tree as long as there is only one preimage. Then recursion is done on the first element of the conveyor belt.
(3) In case there are more than one element in the preimage then splat needs to be used and recursion on all elements before `|`.

For each transformation f on n points one can now obtain a completely canonical expression in this notation by additionally requiring the C's to appear in order according to their least elements, and that the trees preceding a `|` are ordered by their least element, and the cycles start with their least point (or tree root) first.

For instance, here are the conjugacy class representatives of the full transformation semigroup T_4 on four points in canonical form:

```
[1,2,3|4]
[[1,2],3|4]
[1,2|3]
[[1,2|3],4]
```

```
[1,2,3,4]
[1,2,3]
[1,2][3,4]
[1,2]
[1,2](3,4)
()
(1,2)
([1,2],3)
(1,2,3)
([1,2|3],4)
([1,2],[3,4])
([1,2,3],4)
(1,2)(3,4)
([1,2],3,4)
(1,2,3,4)
```

Looking at the list, it is easy to spot the existence of nontrivial cycles and idempotents. Idempotents, other than the identity `()`, are exactly those transformations given by concatenating a number of conveyor belts of length 2, $[x, y]$, and single-level splats $[x_1, ..., x_k \,|\, x_{k+1}]$. Also, a common feature of all the notations discussed here is that conjugation by a permutation is just relabellng of points according to the permutation, e.g. $(1,2,3)^{-1}([1,2],3,4)(1,2,3) = ([2,3],1,4)$.

4. Conclusion

By the spreading use of computer algebra systems for investigating transformations and discrete dynamical systems, efficient notation has become a necessity. In the compact notation we tried to blend the best features of previous notations and also considered the computational experience to derive what we have found to be a useful and readable notation, giving insight into the structure of transformations.

Acknowledgment

The work reported in this article was funded in part by the EU project BIOMICS, contract number CNECT-318202. This support is gratefully acknowledged.

References

[1] Gonca Ayik, Hayrullah Ayik, and John M. Howie. On factorisations and generators in transformation semigroups. *Semigroup Forum*, 70:225–237, 2005.

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

[3] John M. Howie. Review of "*Symmetric Inverse Semigroups* (Mathematical Surveys and Monographs 46) by Stephen Lipscomb: 166 pp., US$49.00, ISBN 0 8218 0627 0 (American Mathematical Society, 1996)". *Bulletin of the London Mathematical Society*, 29(6), 1997.

[4] S. Lipscomb. *Symmetric Inverse Semigroups*, volume 46 of *Mathematical Surverys and Monographs*. American Mathematical Society, 1996.

[1]Centre for Research in Mathematics, School of Computing, Engineering and Mathematics, University of Western Sydney (Parramatta Campus), Locked Bag 1797, Penrith, NSW 2751, Australia

[2]Centre for Computer Science & Informatics Research, University of Hertfordshire, College Lane, Hatfield, Herts AL10 9AB, United Kingdom

E-mail address: A.Egri-Nagy@uws.edu.au, C.L.Nehaniv@herts.ac.uk