Branch Permutation Grammar

by Sven Nilsen, 2018

Assume that you have two permutations:

abcd + cbad

Since `d` is in the same position, it can be factored out:

{abc + cba}d

Here `abc` and `cba` are called “branches”, because they branch out into multiple choices.

Since `b` is located in the same spot in both branches, it can be shared using `/`:

{a/bc + ca}d

A shared element is removed from the following branches. For any branch, the shared element can be restored by counting up the index of the shared element and inserting it.

When multiple elements are shared among branches, one can use curly braces `{}`:

cabde + dabce
{c/{ab}d + dc}e

This notation follows the similar rule for division with numbers:

a / b / c = a / (b · c)	Division with numbers
a/b/c = a/{bc}	Shared elements among branches of permutations

The reason `()` is not used, is because in generalized swap grammar, one already uses `(abc)`:

(abc) = abc + cba	`()` means forward or backward
{abc + cba} = abc + cba	`{}` means a sub-expression, often with branches
{abc} = abc	

If one wants to share `()`, one can just write like this:

a/(bcd)e + ea = abcde + adcbe + ebcda + edcba

For example, if one wants to share a sub-permutation, one can skip the curly braces:

a/{(bcd)!}e + ea = a/(bcd)!e + ea =
abcde + abdce + acbde + acdbe + adbce + adcbe +
ebcda + ebdca + ecbda + ecdba + edbca + edcba