String Rewriting Rules for Cartesian Combinatorics

by Sven Nilsen, 2020

In this paper I describe two simple rules that generates every pure Cartesian product.

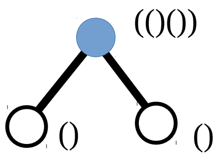
The following two rules for string rewriting generates every pure Cartesian product:

```
() → (())
(()) → ()()
```

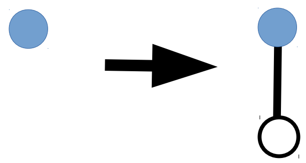
To construct a “proof” of a pure Cartesian product, one can apply these rules in any reverse order. For example, starting with `(()())`, one can trace back the construction in reverse:

(()())		this pure Cartesian product has a unique construction
((()))	(()) → ()()	can only apply the second rule in reverse
(())	() → (())	can only apply the first rule in reverse
()	() → (())	can only apply the first rule in reverse

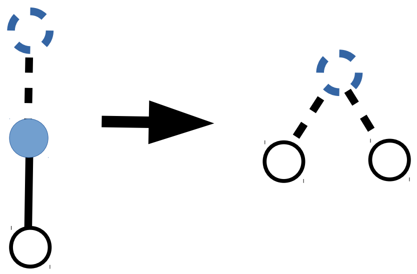
Pure Cartesian products can be represented as a tree:



The two string rewriting rules can only be applied to a leaf or singleton branch of the tree:



() → (()) Applied to leaf



(()) → ()() Applied to singleton branch

All trees can be constructed using these two rules. Since the string rewriting rules are acyclic, trees form a partial ordered set, or a kind of “power tree” under these two rules. Since there are two rules, the power tree can be thought of as an alternating binary tree where children are the choices for each rule.