Swap-Contract Graphs

by Sven Nilsen, 2020

In this paper I present a graph that combines swap and contract operations.

A swap-contract graph contains two kinds of edges:

swap contract

These two kinds are indexed by two indices `i` and `j`, operating a list of some type `T`:

swap(i : nat, j : nat) : [T] → [T]
contract(i : nat, j : nat) : [T] → [T]

Where `swap` has the standard swap operation on lists.

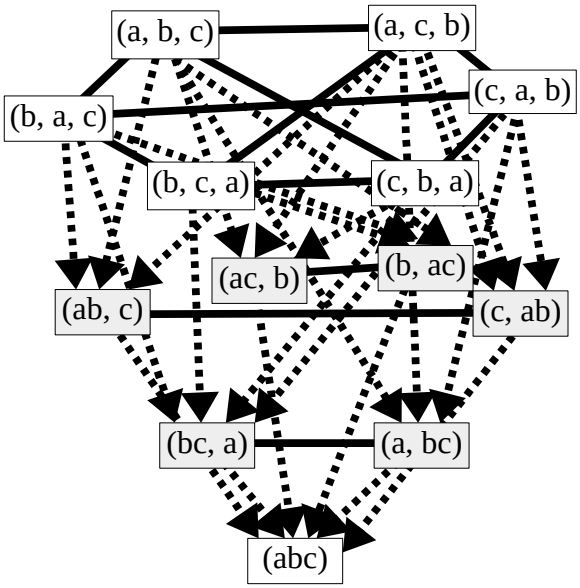
For any two non-neighbors, the `contract` function removes elements of the lists at indices `i` and `j` and inserts the result of a commutative contraction at the index of the first argument.

It follows that the `contract` function is commutative for neighbor indices:

contract(i, j) <=> contract(j, i) if |i - j| <= 1
contract(i, j) <¬=> contract(j, i) if |i - j| > 1

These graphs can be used to study transformations of operations into permutation-invariant operations.

Here is an example of a swap-contract graph:



Operations at this level might depend on the order of elements

Operations at this level might be independent of the order of elements

In the extreme case, the collection must be contracted a single element, for operations to not depend on order