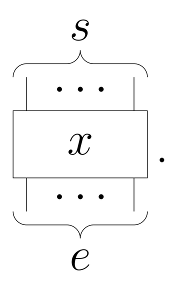
Enumerative Combinatorics of Prographs

Christophe Cordero

Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est Marne-la-Vallée, France

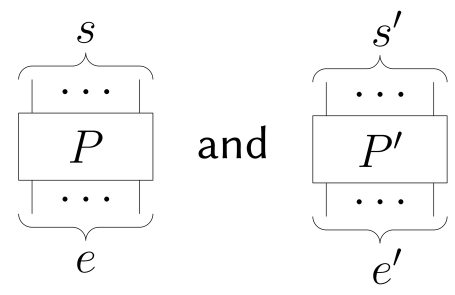
Prographs

A **generator** is an operator with a fixed number of inputs and outputs. We represent a generator x with e inputs and s outputs by

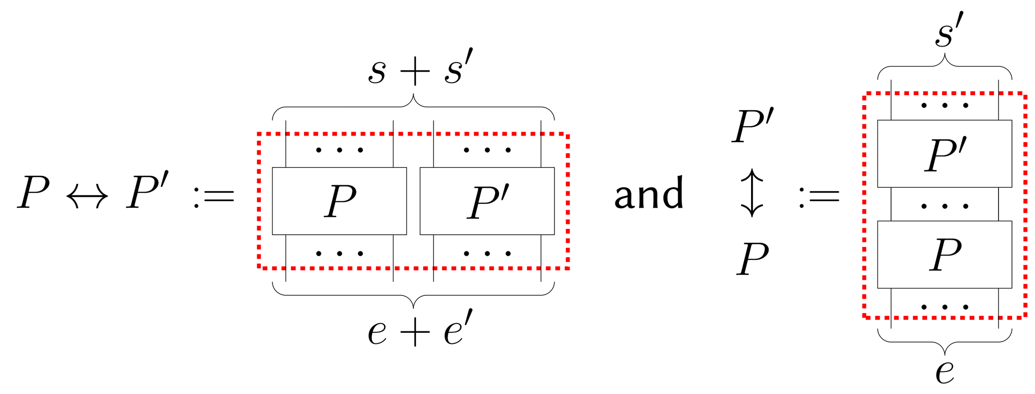


We can combine generators to build **prographs**. Formally, we define prographs by the following recursive grammar:

- A generator with e inputs and s outputs is a prograph with e inputs and s outputs;
- The **wire** $|$ is a prograph with 1 input and 1 output;
- Given two prographs

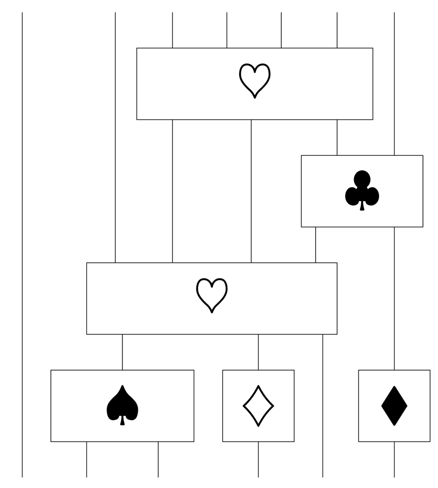


the assemblies



are prographs with respectively $e+e'$ inputs and $s+s'$ outputs and e inputs and s outputs. The second assembly is well defined if and only if $s=e'$.

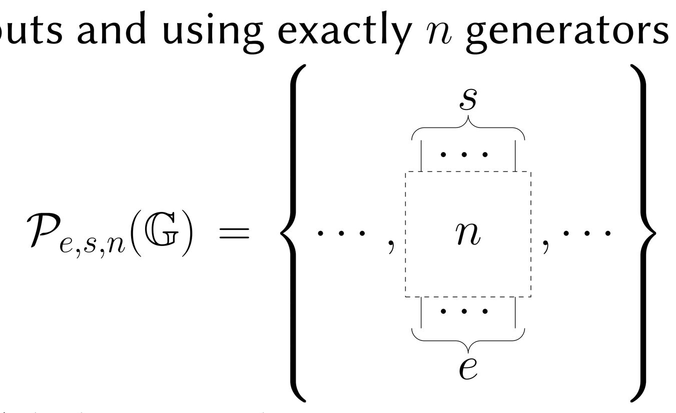
Here is an example of prograph with 6 generators, 6 inputs and 7 outputs:



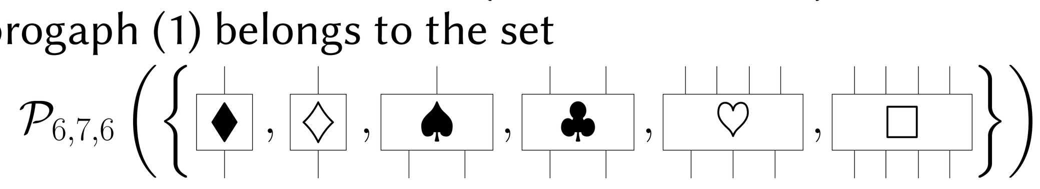
(1)

Problem

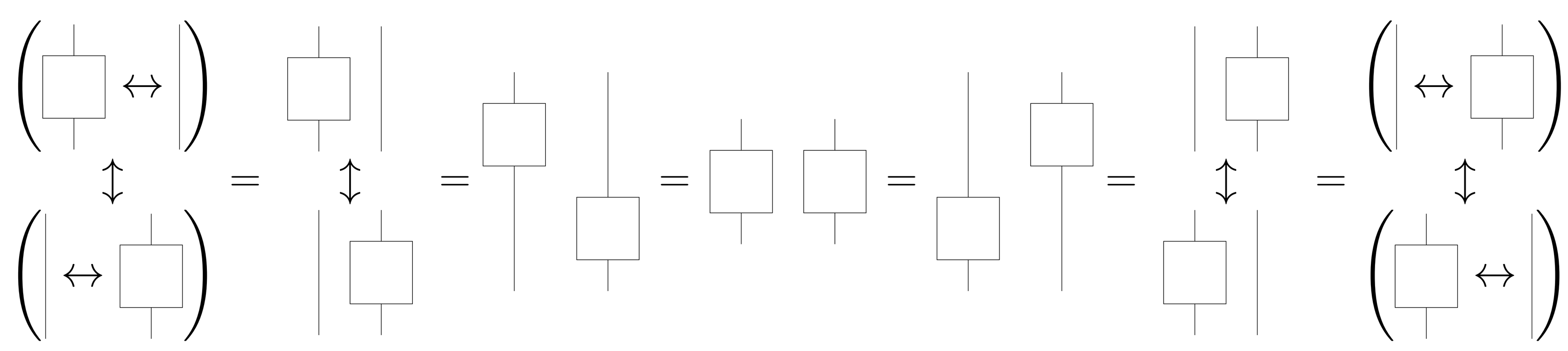
For a set of generators $\mathbb{G}$ and a triple $(e,s,n)\in\mathbb{N}^3$, we denote by $\mathcal{P}_{e,s,n}(\mathbb{G})$ the set of prographs with e inputs, s outputs and using exactly n generators from $\mathbb{G}$:



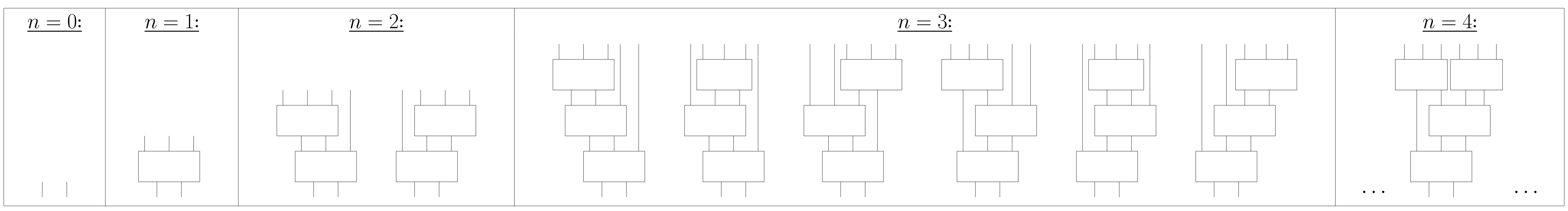
For example, the progaph (1) belongs to the set



Given a set of generators $\mathbb{G}$ and a triple $(e,s,n)\in\mathbb{N}^3$, our goal is to count the prographs of $\mathcal{P}_{e,s,n}(\mathbb{G})$. The main difficulty in counting prographs is that the grammar provided by their definition is ambiguous:



Example $\mathcal{P}_{2,\,n+2,\,n}$

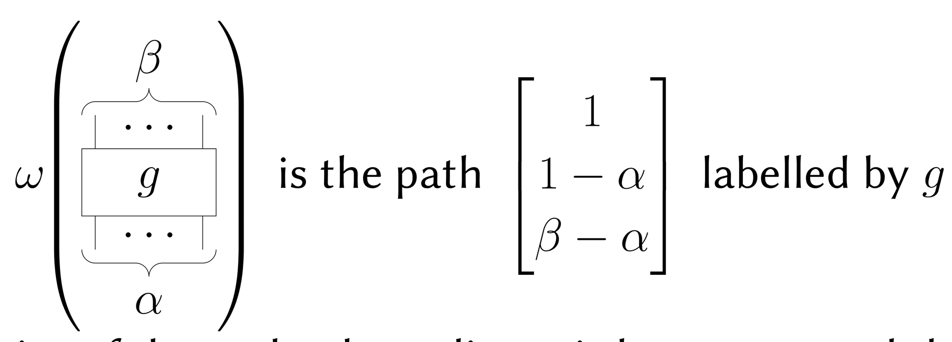


We obtain the sequence $1,1,2,6,22,92,420,2042,\ldots$ which is the sequence of *rooted tandem duplication trees on* n *gene segments* [OEIS: A264868].

Bijection between prographs and some lattice paths

We denote by $\mathcal{L}_{e,n,k,s}(\mathbb{G})$ the set of **lattices paths**:

- from $(0,1,e)$ to (n,k,s)
- using paths U and the paths from the set $\{\omega(g),\ g\in\mathbb{G}\}$, where U is the path $(0,1,0)$ and



- such that in any point of the paths the ordinate is between 1 and the applicate $(1\leq "y" \leq "z")$.

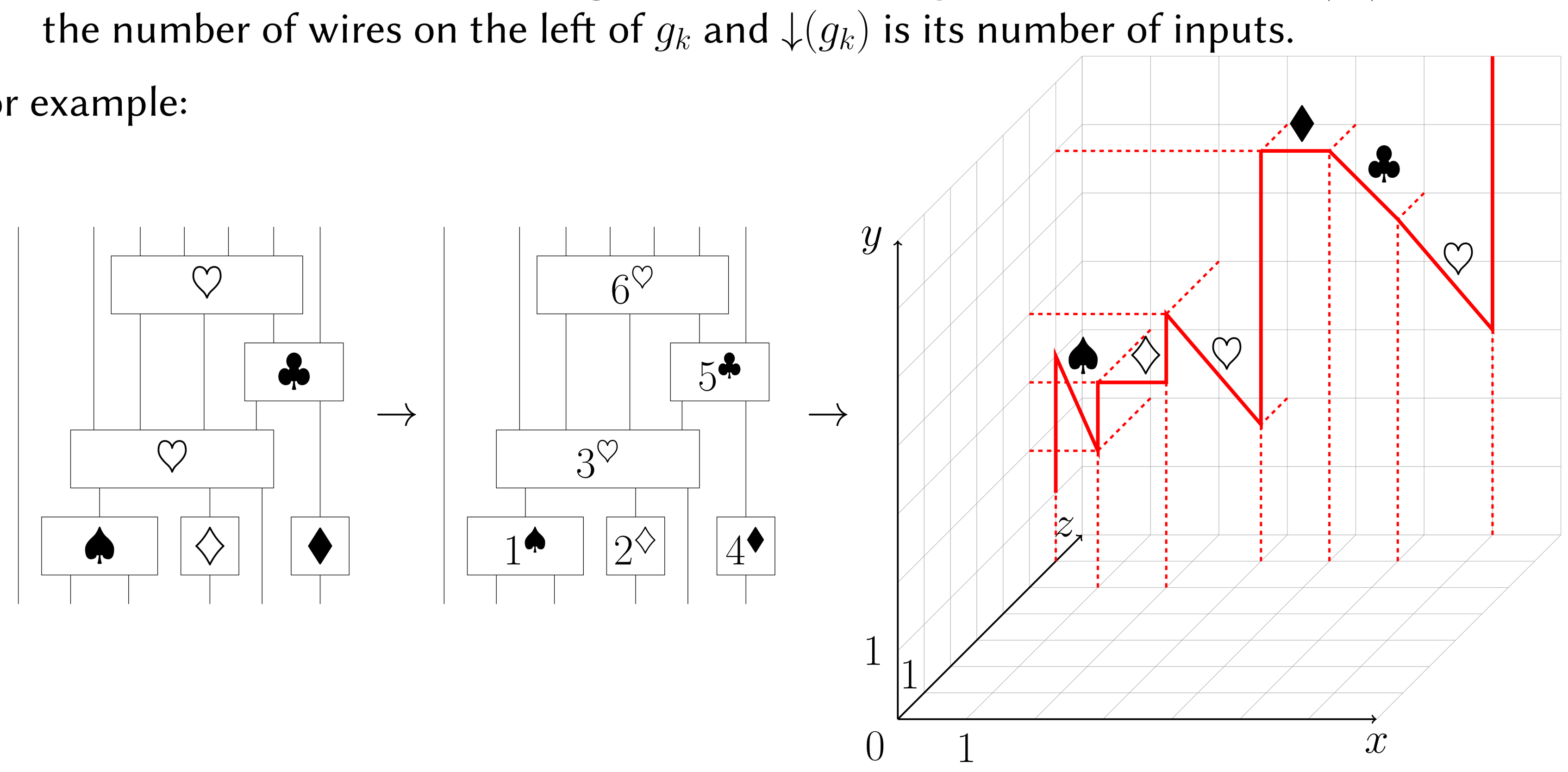
Theorem I

For $(e,s,n)\in\mathbb{N}^3$, we have $|\mathcal{L}_{e,n,s,s}(\mathbb{G})| = |\mathcal{P}_{e,s,n}(\mathbb{G})|$.

The bijection works as follows:

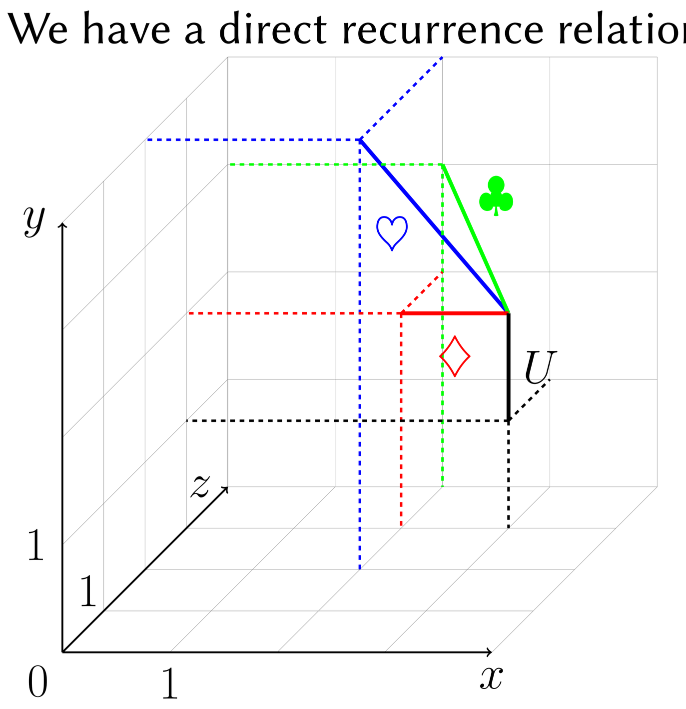
- We numbered generators by a **depth-left first numbering** with the additional condition that a generator can be numbered only if all the generators connected to its inputs are already numbered;
- Then we match in the order, a generator g_k to the path $U^{i_k+\downarrow(g_k)-1-i_{k-1}}\omega(g_k)$, where i_k is the number of wires on the left of g_k and $\downarrow(g_k)$ is its number of inputs.

For example:



Recurrence formulas

We have a direct recurrence relation on these lattices paths:



Proposition

The sequence $|\mathcal{L}_{e,n,k,s}(\mathbb{G})|$ satisfies the following recurrence relation:

$$\begin{cases} 1 \text{ if } n=0,\ k=1 \text{ and } s=e; \\ |\mathcal{L}_{e,n,k-1,s}(\mathbb{G})| + \sum_{i=1}^{d} m_i\, |\mathcal{L}_{e,n-1,k-1+\alpha_i,s-\beta_i+\alpha_i}(\mathbb{G})| \\ \qquad\qquad \text{if } n\geq 0 \text{ and } 1\leq k\leq s; \\ 0 \text{ otherwise.} \end{cases}$$

According to Theorem I, it is enough to specialize k to s in order to obtain a recurrence relation satisfied by prographs. The following theorem gets rid of the refinement parameter k, so it provides a recurrence relation directly on the prographs.

Theorem II

Let $a_{n,s} := |\mathcal{L}_{e,n,s,s}(\mathbb{G})| = |\mathcal{P}_{e,s,n}(\mathbb{G})|$. It satisfies the recurrence relation:

$$a_{n,s} = \begin{cases} 1 \text{ if } n=0 \text{ and } s=e; \\ \sum_{\ell=1}^{n}(-1)^{\ell+1}\sum_{c_1+\cdots+c_d=\ell}\binom{\ell}{c_1,\ldots,c_d}\binom{s+\ell-\sum_{i=1}^{d}c_i\beta_i}{\ell} m_1^{c_1}\ldots m_d^{c_d}\, a_{n-\ell,\,s-\sum_{i=1}^{d}c_i(\beta_i-\alpha_i)} & \text{if } n,s\geq 1; \\ 0 \text{ otherwise.} \end{cases}$$

The 30th international conference on *Formal Power Series and Algebraic Combinatorics*. Dartmouth College, Hanover (USA). July 16–20 2018.