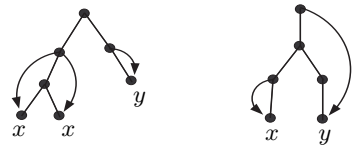
which are studied in combinatory logic (see [10, 9, 11]). We will derive the asymptotic order of the number of $BCI(p)$-terms. Computing the asymptotic number of $BCK(p)$-terms appears to be much more involved. Thus we only derive the functional equation for the generating function corresponding to $BCK(p)$-terms (as well as that of closed lambda-term without any further restriction) and briefly discuss this case.

The enumeration of $BCI(1)$-terms was carried out by Bodini et al. [2] by constructing a nice bijection to certain diagrams. They also determined the asymptotic number of $BCK(1)$-terms.

2 Notation and basic facts

A lambda-term can be regarded as a so-called *enriched tree* which is a particular directed acyclic graph. In fact, consider a Motzkin tree (i.e., a rooted unary-binary tree) and add directed edges connecting a unary node and a leaf such that each leaf is "bound" by a directed edge from exactly one of the unary nodes that are its ancestors in the tree. The correspondence is obvious (see Figure 1): leaves correspond to variables, unary nodes to abstractions, binary nodes to applications and the additional directed edges to the binding relations between abstractions and variables. Clearly, since all leaves are bound, the lambda-term is closed. Of course, open lambda-terms can be represented in an analogous manner by a directed acyclic graph where some leaves have in-degree zero.



$(\lambda x.(x * x) * \lambda y.y)$ $\quad$ $\lambda y.(\lambda x.x * \lambda z.y)$

Figure 1: Two enriched trees and the closed lambda-terms corresponding to them. Note that the node labels can be omitted, since $(\lambda x.(x * x) * \lambda y.y)$ and $(\lambda a.(a * a) * \lambda b.b)$ are the same term.

We will not distinguish between a lambda-term and its enriched tree representation. In addition, when speaking of lambda-terms, we will utilize the following abuse of the wording: A *unary node* of a lambda-term is a unary node (i.e. node of out-degree one) of the underlying Motzkin tree (i.e. a node becoming unary if all directed edges are removed). These are precisely the nodes corresponding to abstractions. Analogously, we call the nodes corresponding to applications *binary nodes* and nodes corresponding to variables *leaves* of the lambda-term. In a strict sense, leaves have always degree one and in-degree one as well.

Moreover, we distinguish between *edges*, i.e. edges of the underlying Motzkin tree, and *pointers*, i.e. directed edges from a unary node to a leaf.

Definition 1.
- $BCI(p)$ is the set of (non-empty) closed lambda-terms where each unary node has *exactly* p pointers, i.e. binds exactly p occurrences of its variable.
- $BCK(p)$ is the set of closed lambda-terms where each unary node binds *at most* p leaves.

A lambda-term from $BCI(p)$ has three types of nodes: unary nodes (which are actually of arity $p+1$, as there are p pointers going from this node to leaves), binary nodes, and leaves. The size of such a lambda-term is the total number of its nodes. We start with some obvious observations:

Fact 1. The smallest terms of $BCI(p)$ have one unary node at the root and p leaves. There are p pointers from the root to all the leaves. Obviously, if we remove the root and all its pointers, we are left with a binary tree. Cleary, their size is $2p$.

The number of such terms is therefore equal to the number of binary trees with $p-1$ binary nodes and p leaves. This is precisely the Catalan number $C_{p-1} = \binom{2p-2}{p-1}/p$.