Computing the $k$-binomial complexity of the Tribonacci word

Marie Lejeune$^*$, Michel Rigo$^2$, and Matthieu Rosenfeld$^{1,3}$

$^1$Département de Mathématiques [Liège] – Belgique
$^2$Département de Mathématiques [Liège] – Grande Traverse, 12 Sart-Tilman 4000 Liège, Belgique
$^3$Laboratoire d’Informatique, de Modélisation et d’Optimisation des Systèmes (LIMOS) – Université Clermont Auvergne, Université Clermont Auvergne – France

Résumé

The factor complexity of an infinite word $x$ simply counts the number of factors of length $n$ occurring in $x$. One can also consider other measures such as the abelian complexity, introduced by Paul Erdős in the sixties, or the more recent notion of $k$-binomial complexity, which counts the number of equivalence classes partitioning the set of factors of length $n$ for the so-called $k$-binomial equivalence. Two finite words $u$ and $v$ are $k$-binomially equivalent every word of length at most $k$ appears as a scattered subword in $u$ and in $v$ the same number of times.

It is already known that the $k$-binomial complexity of Sturmian words equals their factor complexity. We proved the same result for the Tribonacci word, which is the fixed-point of the morphism $0 -> 01, 1 -> 02, 2 -> 0$. However, the proof relies on completely different techniques. Surprisingly, classical combinatorial techniques seemed to be unsuccessful.

We make an important use of the concepts of templates and their ancestors, similarly to what was done by Currie and Rampersad when they considered avoidance of abelian powers in words. The proof of our result relies on an algorithm which was implemented using Mathematica. The goal of this talk will be to present in details the notions of $k$-binomial equivalence and templates, and also explain the algorithm we created.

Mots-Clés: tribonacci word, $k$ binomial complexity, binomial coefficient

$^*$Intervenant