A Generating Algorithm for Ribbon Tableaux and Spin Polynomials

[1]Francois Descouens

[1]*Institut Gaspard Monge, Université de Marne-la-Vallée*
77454 Marne-la-Vallée Cedex 2, France

received 13 Jan 2005, *revised 23 Apr 2005*, *accepted 25 May 2005*.

We describe a general algorithm for generating various families of ribbon tableaux and computing their spin polynomials. This algorithm is derived from a new matricial coding. An advantage of this new notation lies in the fact that it permits one to generate ribbon tableaux with skew shapes.

Keywords: Ribbon, spin, cospin, ribbon tableaux, spin and cospin polynomials.

1 Introduction

Ribbon tableaux are planar structures generalizing Young tableaux (see (5; 10) for the classical case). These are tilings of Ferrers's diagrams by *ribbons* (diagrams with special shape) labelled with integers verifying some vertical and horizontal monotonicity conditions.

Standard ribbon tableaux (all labels different) have first been introduced by Stanton and White in 1985, in order to explain some combinatorial properties of colored permutations (11). Semi-standard ribbon tableaux (repeated labels are allowed) go back to the work of Lascoux, Leclerc and Thibon (8). These authors were motivated by the introduction of q-analogues of certain combinatorial identities, and in particular of the famous Littlewood-Richardson rule describing products of Schur functions. They obtained q-analogues of decomposition coefficients for any product of Schur functions and many questions about these q-coefficients are still open.

Studying ribbon tableaux is quite a difficult subject, which mainly uses huge numerical experimentations. This is why we are interested in finding efficient algorithms for generating and computing statistics on them.

The matricial coding of ribbon tableaux used for numerical experimentations in (8; 9) is deficient because it does not give some elementary properties (shape and position of the head of the ribbons for example) without additional computations and cannot be generalized to skew shapes. The algorithm used at the time was not published and appeared only as a programming example with Maple/ACE in (6) and a distributed version is described in (13).

The aim of this paper is to present a more general algorithm for generating ribbon tableaux and computing spin polynomials, using a different and more transparent coding for ribbon tableaux. These algorithms are implemented in the combinatorial library `MuPAD-Combinat`(4) which can be downloaded at *http://mupad-combinat.sourceforge.net/*.

2 Basic definitions on ribbon tableaux

We will mainly follow (1; 5; 10) for classical notions on partitions and tableaux and (10) for notations related to partitions. Let λ and μ be two partitions such that the diagram of λ contains the diagram of μ, The skew partition of shape λ/μ can be defined as the set-theoretic difference $\lambda - \mu$ (λ is called the outer partition and μ the inner).

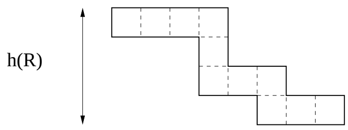


Figure 1: An 11-ribbon of height $h(R) = 4$

Definition 1 *Let k be a nonnegative integer. A k-**ribbon** R is a connected skew diagram with k cells which does not contains a 2×2 square. The first (north-west) cell is called the **head** and the last one (south-east) the **tail**. The **spin** is defined as $sp(R) = \frac{h(R)-1}{2}$.*

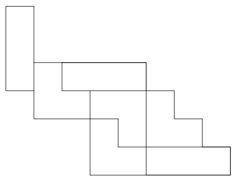


Figure 2: A tiling with 3-ribbons of the skew partition $(8, 7, 6, 5, 1, 1)/(3, 3, 1)$ of spin 3

We will denote by $Pav_k(\lambda/\mu)$ the set of k-ribbon tilings of the skew shape λ/μ. The **spin** $sp(P)$ of a tiling P is the sum of the spins of its ribbons, and the **cospin** is the associated co-statistic into $Pav_k(\lambda/\mu)$, i.e:

$$cosp(P) = \max\{sp(U),\ U \in Pav_k(\lambda/\mu)\}\ -\ sp(P).$$

Ribbon tableaux are labelled tilings verifying monotonicity conditions similar to these of Young tableaux. The spin of a k-ribbon tableau is the spin of the underlying tiling.

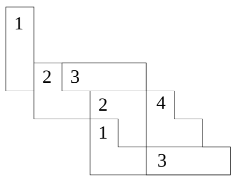


Figure 3: A 3-ribbon tableau of shape $(8, 7, 6, 5, 1, 1)/(3, 3, 1)$, weight $(2, 2, 2, 1)$, and spin 3

Definition 2 *A k-ribbon tableau of skew shape λ/μ is a tiling of the skew shape λ/μ by labelled k-ribbons such that the head of a ribbon labelled i must not be on the right of a ribbon labelled $j > i$ and its tail must not be on the top of a ribbon labelled $j \geq i$. The weight of a k-ribbon tableau is the vector ν such that ν_i is the number of k-ribbons labelled i.*

We denote by $\mathrm{Tab}_k(\lambda/\mu, \nu)$ the set of all semi-standard k-ribbon tableaux of shape λ/μ and weight ν.

Definition 3 *The* ***spin*** *and* ***cospin polynomials*** *associated to the set $Tab_k(\lambda/\mu, \nu)$ are:*

$$G^{(k)}_{\lambda/\mu,\nu}(q) = \sum_{T\ \in Tab_k(\lambda/\mu,\nu)} q^{sp(T)} \quad \textit{and} \quad \tilde{G}^{(k)}_{\lambda/\mu,\nu}(q) = \sum_{T\ \in Tab_k(\lambda/\mu,\nu)} q^{cosp(T)}.$$

If we write $sp^* = \max\{sp(T),\ T \in Tab_k(\lambda/\mu)\}$, the following property holds:

$$G^{(k)}_{\lambda/\mu,\nu}(q) = q^{sp^*} \tilde{G}^{(k)}_{\lambda/\mu,\nu}(\frac{1}{q}).$$

Example 1 *In $Tab_3((8,7,6,5,1),(3,3,2,1))$, these two polynomials are:*

$$G^{(3)}_{(8,7,6,5,1),(3,3,2,1)}(q) = 3q^2 + 17q^3 + 33q^4 + 31q^5 + 18q^6 + 5q^7,$$

3 A generating algorithm

In this section, we describe a new algorithm for generating all the ribbon tableau in $\mathrm{Tab}_k(\lambda/\mu, \delta)$ and computing their spin polynomials. The main basic idea is to apply recursively the algorithm of removing k-ribbon strips from a partition.

3.1 A new coding for ribbon tableaux

In this subsection, we extend the coding of (13) which is not well adapted to ribbon tableaux with skew shape λ/μ and does not give immediate access to the shape of the tableau. If the partition λ is $(\lambda_1, \dots, \lambda_n)$, our coding is an $n \times \lambda_1$-array $(A_{i,j})$ defined as follows:

1- $A_{i,j} = -1$ if $(i,j) \in \mu$,

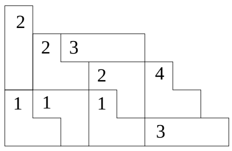
2- $A_{i,j} = p$ if there is the head of a ribbon labelled p in the cell (i,j),

3- $A_{i,j} = 0$ otherwise.

By construction, we can immediately read the shape of the corresponding ribbon tableau, and the length of the ribbons is obtained by dividing the number of non negative cells by the number of positive cells. This coding makes sense when $k = 1$ because we obtain the classical representation of a skew Young tableau (each ribbon is reduced to its head).

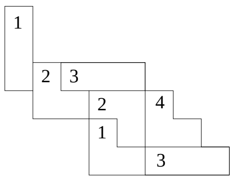
The structure used in (13) was a matrix where the entry (i,j) is equal to k if there is the tail of a ribbon labelled i in the j-th column of the shape. We show that the main difficulty in carrying out this construction is that we cannot read the shape of the partition without reconstructing the entire ribbon tableau. Furthermore, this coding become non injective when we try to generalize it on ribbon tableaux of skew shape. With the previous example we would obtain the following matrix:

$$\begin{pmatrix} 0 & 3 & 3 & 0 & 3 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \end{pmatrix}.$$



$$\longrightarrow \quad \begin{pmatrix} 2 & & & & & & & \\ 0 & 2 & 3 & 0 & 0 & & & \\ 0 & 0 & 0 & 2 & 0 & 4 & & \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \end{pmatrix}.$$

Figure 4: The new coding of a 3-ribbon tableau of shape $(8, 7, 6, 5, 1, 1)$ and weight $(3, 3, 2, 1)$



$$\longrightarrow \quad \begin{pmatrix} 1 & & & & & & & \\ 0 & & & & & & & \\ 0 & 2 & 3 & 0 & 0 & & & \\ -1 & 0 & 0 & 2 & 0 & 4 & & \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & \\ -1 & -1 & -1 & 0 & 0 & 3 & 0 & 0 \end{pmatrix}.$$

Figure 5: The new coding of a 3-ribbon tableau of skew shape $(8, 7, 6, 5, 1, 1)/(3, 3, 1)$ and weight $(3, 2, 2, 1)$

Decoding a k-ribbon tableau from an array $(A_{i,j})$ is as follows: for each label p considered in decreasing order, we fix vertically by the head a ribbon labelled p in the cell (i, j) of the shape if $A_{i,j} = p$, and we drive them to the frontier of the partition.

Algorithm 1 (From coding to ribbon tableaux)

- **Input**: *an array* $M = (m_{ij})$.
- `Initialize`:
  - $\lambda/\mu \longleftarrow$ `the shape of the tableau coded by` M,
  - $\nu \longleftarrow$ `the weight of the tableau.`
- `For each label` i `in` ν:
  - `fix ribbons labelled` i `by the head in each position corresponding to the cells labelled` i `in` M,
  - `drive these ribbons on the frontier of` λ,
  - $\lambda \longleftarrow \lambda$ `without previous ribbons labelled` i.
- **Output**: *The* k*-ribbon tableau corresponding to the array* M.

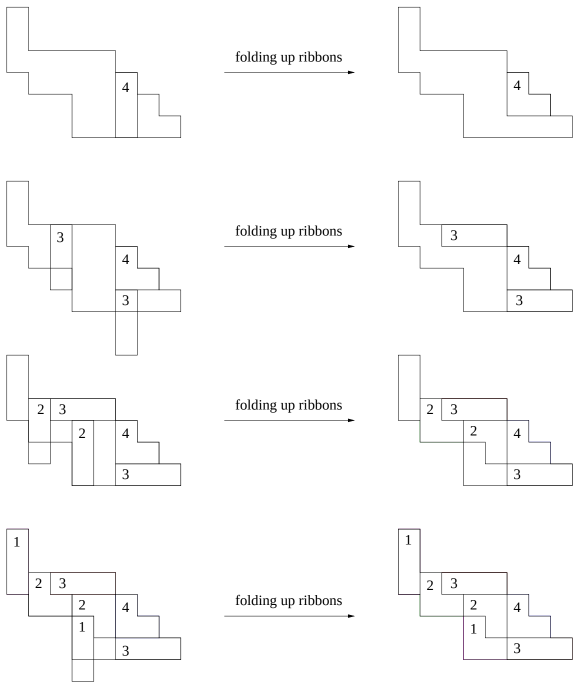


Figure 6: Running of the algorithm on the previous 3-ribbon tableau of shape (876511)/(331) and weight (2221)

3.2 Adding and removing k-ribbon strip

In order to explain our generating algorithm, we begin with a general algorithm for adding or removing a k-ribbon strip from a partition (these two operators on partitions come from the representation theory of the quantum algebra $U_q(\widehat{sl_n})$, see (2) for more details).

Definition 4 *A skew tiling by k-ribbons Θ where the tail of each ribbon is not on top of an other ribbon is called a k-**ribbon strip**. The **weight** of Θ is the number of ribbons in the tiling. Let $\Theta_\uparrow$ (resp. $\Theta_\downarrow$) be the the horizontal strip made of the top cells (resp. the bottom cells) of the columns of Θ.*

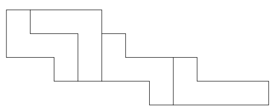


Figure 7: A horizontal 5-ribbon strip of weight 4 and spin 3/2

In a ribbon strip, the head of each ribbon lies in $\Theta_\uparrow$ and the tails in $\Theta_\downarrow$. As a k-ribbon strip has a unique tiling by k-ribbons, it is completely defined by the positions of all the ribbons's heads and the outer partition or all the tails and the inner partition. For adding a k-ribbon strip Θ to a partition, we represent this strip by the integer vector $\mathcal{P} = (p_1, \ldots, p_n)$ with $p_i = k$ if Θ has a ribbon with tail in the i-th column of $\Theta_\downarrow$. Similarly, for removing a k-ribbon strip Θ from a partition, the vector $\mathcal{P}$ is now defined by $p_i = -k$ if Θ has a ribbon with head in the i-th column of $\Theta_\uparrow$.

Algorithm 2 (Algorithm for adding a k-ribbon strip to a partition)

- ***Input****: the partition $\lambda = (\lambda_1, \ldots, \lambda_n)$ and the positions vector $\mathcal{P} = (p_1, \ldots, p_r)$.*
- `Initialize`*:*
  - $\delta \longleftarrow (max(n, r) - 1, \ldots, 0)$,
  - $\lambda \longleftarrow$ `conjugate partition of` λ.
- $\lambda \longleftarrow \lambda + \delta + \mathcal{P}$.
- `Sorting` λ:
  - $\lambda \longleftarrow \sigma(\lambda)$,
  - $I \longleftarrow$ `the inversions number of` σ.
- `if` $\lambda - \delta$ `is a partition:`
  - `then` $\lambda \longleftarrow$ `conjugate` $(\lambda - \delta)$ `and` $I \longleftarrow \frac{(k-1)(p_1+\ldots+p_r)-I}{2k}$,
  - `else` $\lambda \longleftarrow$ `FAIL`.
- ***Output****:* (λ, I) `if exists, FAIL otherwise.`

In the previous algorithm, σ corresponds to a permutation which permits to sort the vector λ. Let defined the number of inversions of the permutation σ by the cardinality of the set $\{(i, j)$ such that $i < j$ and $\sigma(i) > \sigma(j)\}$.

3.3 Generating algorithm and computation of the spin and cospin polynomials for ribbon tableaux of a given shape and weight

We will generalize the generating algorithm of (13) to the case of k-ribbon tableaux of skew shapes. A basic remark consists in the fact that the ribbons labelled i form a k ribbon strip Θ_i of weight ν_i. That's why we search recursively all the k-ribbon strip of weight ν_i contained in the shape and removable from the outer partition and we fill an array with the positions of the head in $\Theta_i\uparrow$. As the algorithm 1 also returns the spin of the added/removed k-ribbon strip, by keeping the spin in each step of the construction we obtain finally the spin of each tableau. In other words this algorithm permits one to compute the spin and cospin polynomials without additional computation.

Algorithm 3 (Generating algorithm for ribbon tableaux and spin polynomials)

- ***Input****: The shape λ/μ, the weight $\nu = (\nu_1, \ldots, \nu_p)$ and k.*
- `Initialize:` $P = 0$ `and` $L \longleftarrow \{(T, \lambda, 0)\}$ `where` T `is an array filled with -1 in cells corresponding to` μ `and 0 otherwise.`

- `For each weight` j `from` p `down to 1,`
  `For each` (T, λ_T, sp_T) `in` L `and each permutation` $\mathcal{C}$ `of the vector` $(\underbrace{0,\ldots,0}_{\lambda_1-\nu_j}, \underbrace{k,\ldots,k}_{\nu_j\ times})$ `corresponding to a valid removed` k`-ribbon strip,`
  - `fill in` T `the cells corresponding to the frontier of` λ_T `and the non-zero coordinates of` $\mathcal{C}$`,`

    $\lambda_T \longleftarrow \lambda_T$ `without the` k`-ribbon strip corresponding to` $\mathcal{C}$`,`
  - $sp_T \longleftarrow sp_T +$ `the spin of the previous` k`-ribbon strip,`
  - `if` $\mu \subset \lambda_T$ `then` $L \longleftarrow L \ \cup \ (T, \lambda_T, sp_T)$.
- `For each tableau` T `in` L`,` $P \longleftarrow P + q^{sp_T}$.
- ***Output****:*
  - *The list* L *of all the* k*-ribbon tableaux of skew shape* λ/μ *and weight* ν*,*
  - P *the spin polynomial of this set.*

By studying the progress of this algorithm with huge set of ribbon tableaux we remark that at the bottom of the tree there are a lot of nodes, but in there is only few different shape for the remaining partition. For example the number of nodes for $\lambda = (6,6,6,6,6,6)$ and $\nu = (3,1,1,1,1,1,1,1,1,1)$ is the sequence 3, 12, 48, 198, 780, 2940, 10080, 31080, 81480 and finally 43680 but the number of different remaining shape are 3, 9, 16, 27, 33, 38, 33, 27, 16, 1. Our previous algorithm search, at step i, all the possibility to retire a k ribbon strip from the remaining partition, but in fact we need to test all the possibility only on the few remaining partitions. That's why a recursive implementation of the previous algorithm with a remember option seems to be the most efficient way to generate ribbon tableaux.

Example 2 *In the case of 3-ribbons with* $\lambda = (9,9,9,9,9,9,9,9,9)$ *and standard weight* $\mu = (1^{27})$*, we have the following spin polynomial:*

$$414315330 + 8286306600q + 85027356570q^2 + 588666753870q^3 + 3062543589300q^4 + 12659483135520q^5 + 42941179272810q^6 + 121912682783970q^7 + 293410572110760q^8 + 603798294330270q^9 + 1068859924958280q^{10} + 1634693172838050q^{11} + 2166452577489720q^{12} + 2492870571244950q^{13} + 2492870571244950q^{14} + 2166452577489720q^{15} + 1634693172838050q^{16} + 1068859924958280q^{17} + 603798294330270q^{18} + 293410572110760q^{19} + 121912682783970q^{20} + 42941179272810q^{21} + 12659483135520q^{22} + 3062543589300q^{23} + 588666753870q^{24} + 85027356570q^{25} + 8286306600q^{26} + 414315330q^{27}$$

which correspond to a computation over 16 882 686 792 972 000 *ribbon tableaux.*

Acknowledgments

The author wishes to express his gratitude to the members of the algebraic combinatorics team of the University of Marne-la-Vallée for their helpful comments.

References

[1] G.E. Andrews *The theory of Partitions*, Vol. 2 of *Encyclopedia of Mathematics and Its Applications*, Addison-Wesley (1976)

[2] B. Leclerc *Symmetric functions and the Fock space*, Proceedings of the NATO Advanced Study Institute: Symmetric Functions: Survey of Developments and Perspectives (2001)

[3] J. Haglund, M. Haiman, N. Loehr, J.B. Remmel, A.Ulyanov Âš*A combinatorial formula for the character of the diagonal coinvariants*, Duke Math. J. 126 (2005), no. 2, 1995-232.

[4] F. Hivert, N. Thiery *MuPAD-Combinat, an Open Source Package for Research in Algebraic Combinatorics*, SÃl'minaire Lotharingien de Combinatoire **51** (2004)

[5] D.E. Knuth *The Art of Computer Programming*, Vol. 3:*Sorting and Searching*, Addison-Wesley (1973)

[6] D. Krob, A. Lascoux, B. Leclerc, J.-Y. Thibon, B.C.V. Ung, S. Veigneau *Algebraic Combinatorics with Maple and ACE*, Maple Technical Newsletter **4**, No. 1 (1997), 43–50.

[7] T. Lam *Ribbon Tableaux and the Heisenberg Algebra* (2003) http://arXiv.org/abs/math/0310250.

[8] A. Lascoux, B. Leclerc, J.-Y. Thibon, *Ribbon tableaux, Hall-Littlewood functions, quantum affine algebras and unipotent varieties*, Journal of Mathematical Physics **38** (1997), 1041–1068.

[9] B. Leclerc, J.-Y. Thibon, *Littlewood-Richardson coefficients and Kazhdan-Lusztig polynomials* (avec B. Leclerc), Adv. Studies in Pure Math. **28** (2000), 155–220.

[10] M. Lothaire *Algebraic Combinatorics on Words*, Vol. 90 of *Encyclopedia of Mathematics and Its Applications*, Cambridge University Press (2002)

[11] D. Stanton and D. White, *A Schensted algorithm for rim-hook tableaux*, J. Comb. Theory A **40**, (1985), 211-247.

[12] M.A.A. Van Leeuwen, *Spin-preserving Knuth correspondences for ribbon tableaux* (2003) http://arXiv.org/abs/math/0312020.

[13] S. Veigneau, *Distributed Computation of Ribbon Tableaux and Spin Polynomials*, Proceedings of the Third European PVM Users' Group Meeting, 1996.

[14] A. Schilling, M. Shimozono, D.E. White, *Branching formula for q-Littlewood-Richardson coefficients*, Advances in Applied Math. 30 (2003), 258-272.