Products of Factorial Schur Functions
Victor Kreiman
published version, The Electronic Journal of Combinatorics **15** (2008), #R84
Errata and addenda by Darij Grinberg

The following corrections cover the first 9 pages of the paper.

- **page 1:** It is worth saying that $\mathbb{N}$ means the set $\{0, 1, 2, \ldots\}$ in this paper (not $\{1, 2, 3, \ldots\}$ as in some other parts of the literature).
- **page 1:** "Biedenbarn" → "Biedenharn".
- **pages 1–8:** Both the definition of factorial Schur functions $s_\lambda(x \mid y)$ (and their products $s_{\boldsymbol{\lambda}}(x \mid \mathbf{y})$) and their properties (including Theorem 2.2) can be generalized straightforwardly to the case when λ is a skew partition instead of a partition (resp. $\boldsymbol{\lambda}$ is a sequence of skew partitions instead of a sequence of partitions). The proofs apply equally well to this generalization.
- **page 6:** "**Alorithm 1**" → "**Algorithm 1**".
- **page 6, Algorithm 2:** The description of this algorithm is slightly incorrect[1]. However, I think the algorithm can also be made somewhat clearer by rewriting its definition as follows:

  Algorithm 2 (redefined): In the following, when we write "i" or "$i+1$" without a bar over it, we will always mean an unbarred i or an unbarred $i+1$, respectively. (Barred i's and $(i+1)$'s will always be written with a bar overhead.)

  Let l be the number of i's and r the number of $i+1$'s that S contains.

  - If $l = r$: Do not change S.
  - If $l < r$: We classify the entries of S into "dead" and "alive" ones, as follows:
    * We classify all i's and $\bar{i}$'s in S as dead.
    * We classify the l rightmost $(i+1)$'s of S as dead.
    * We also classify as dead any $\overline{i+1}$ that has at most l many $(i+1)$'s to its right (in S).

[1] To wit: In the "$l < r$" case, if $l = 0$, then you define $R = S$, and later swap each $\bar{i}$ with the i immediately to its right. But this might be impossible, since there might be an $\bar{i}$ at the rightmost end of $R = S$. In order to correct for this, R should not be defined by $R = S$, but rather defined to be the part of S that ends with the rightmost unbarred $i+1$ in S. A similar correction is necessary in the "$l > r$" case.

  * The remaining entries of S (all of them $(i+1)$'s and $\overline{i+1}$'s) are classified as alive.

Here is an example: If

$$S = \boxed{i} \boxed{\bar{i}} \boxed{\bar{i}} \boxed{i} \boxed{i+1} \boxed{\overline{i+1}} \boxed{i+1} \boxed{\overline{i+1}} \boxed{i+1} \boxed{\overline{i+1}} \boxed{\overline{i+1}} \boxed{i+1}$$

(so that $l = 2$ and $r = 4$), then the leftmost four and the rightmost five entries of S are dead while the remaining three entries in the middle are alive.

We notice that the alive entries of S form a contiguous string, sandwiched between the dead i's and $\bar{i}$'s to their left and the dead $(i+1)$'s and $\overline{i+1}$'s to their right (because all i's and $\bar{i}$'s in S are dead, and because any entry of S to the right of a dead $i+1$ or a dead $\overline{i+1}$ is dead). We denote this string of alive entries by R. Note that it consists entirely of $(i+1)$'s and $\overline{i+1}$'s, and contains exactly $r-l$ many $(i+1)$'s (since there are r many $(i+1)$'s in S, and exactly l of them are dead). Moreover, this string R must end with an $i+1$ (not an $\overline{i+1}$), because our definition of "dead" ensures that no $\overline{i+1}$ can be alive unless there is an alive $i+1$ somewhere to its right. So the string R consists of a bunch of $(i+1)$'s and $\overline{i+1}$'s, ending with an $i+1$.

Now, we modify R by changing all $(i+1)$'s into i's, and changing all $\overline{i+1}$'s into $\bar{i}$'s. As a consequence, R now consists of a bunch of i's and $\bar{i}$'s, ending with an i.

Next, we rotate R cyclically to the right (by one step), so that its last entry becomes its first. Thus, R now consists of a bunch of i's and $\bar{i}$'s and begins with an i (since it used to end with an i before the cyclic rotation).

This ends the description of the algorithm in the case when $l < r$. None of the dead entries are changed.

- If $l > r$: We classify the entries of S into "dead" and "alive" ones, as follows:
  * We classify all $(i+1)$'s and $\overline{i+1}$'s in S as dead.
  * We classify the r leftmost i's of S as dead.
  * We also classify as dead any $\bar{i}$ that has at most r many i's to its left (in S).
  * The remaining entries of S (all of them i's and $\bar{i}$'s) are classified as alive.

Here is an example: If

$$S = \boxed{i} \boxed{\bar{i}} \boxed{\bar{i}} \boxed{i} \boxed{i} \boxed{i} \boxed{\bar{i}} \boxed{\overline{i+1}} \boxed{i+1} \boxed{\overline{i+1}} \boxed{\overline{i+1}} \boxed{i+1}$$

(so that $l = 4$ and $r = 2$), then the leftmost four and the rightmost five entries of S are dead while the remaining three entries in the middle are alive.

We notice that the alive entries of S form a contiguous string, sandwiched between the dead i's and $\bar{i}$'s to their left and the dead $(i+1)$'s and $\overline{i+1}$'s to their right (because all $(i+1)$'s and $\overline{i+1}$'s in S are dead, and because any entry of S to the left of a dead i or a dead $\bar{i}$ is dead). We denote this string of alive entries by R. Note that it consists entirely of i's and $\bar{i}$'s, and contains exactly $l - r$ many i's (since there are l many i's in S, and exactly r of them are dead). Moreover, this string R must begin with an i (not an $\bar{i}$), because our definition of "dead" ensures that no $\bar{i}$ can be alive unless there is an alive i somewhere to its left. So the string R consists of a bunch of i's and $\bar{i}$'s, beginning with an i.

Now, we modify R by changing all i's into $(i+1)$'s, and changing all $\bar{i}$'s into $\overline{i+1}$'s. As a consequence, R now consists of a bunch of $(i+1)$'s and $\overline{i+1}$'s, beginning with an $i+1$.

Next, we rotate R cyclically to the left (by one step), so that its first entry becomes its last entry. Thus, R now consists of a bunch of $(i+1)$'s and $\overline{i+1}$'s and ends with an $i+1$ (since it used to begin with an $i+1$ before the cyclic rotation).

This ends the description of the algorithm in the case when $l > r$. None of the dead entries are changed.

It is rather easy to see that this algorithm undoes itself when it is applied twice in succession. Here is why: First of all, when we apply the algorithm to a string S satisfying $l < r$, the resulting string has r many i's and l many $(i+1)$'s (because the $r - l$ many alive $(i+1)$'s have been replaced by $r - l$ many i's), so that the numbers l and r play the roles of r and l in the resulting string. The same holds if we apply the algorithm to a string S satisfying $l > r$. Furthermore, if we apply the algorithm to a string S satisfying $l < r$ or $l > r$, then the dead entries stay dead[2], and the alive entries stay alive (or, to be more precise, even though the alive entries themselves are changed, the new entries at their positions are again alive)[3]. Thus, if we apply the algorithm to a string S satisfying $l < r$, and

[2] This is easy to check.

[3] *Proof.* Let S be a string satisfying $l < r$. If a position is occupied by an alive entry in S, then it belongs to the substring R. After we apply the algorithm to S, this substring R consists of a bunch of i's and $\bar{i}$'s and begins with an i. Thus, after we apply the algorithm to S, any entry of R is an i or an $\bar{i}$ that has more than r many i's weakly left of it (because R begins with an i). Thus, any such entry is alive in the resulting string (keeping in mind that l and r have traded places, so the meanings of "dead" and "alive" are now defined according to the "If $l > r$" case of the algorithm). So we have shown that any position that contains an alive entry before the algorithm still contains an alive entry after the algorithm, provided that $l < r$. An

then apply the algorithm again to the the resulting string S', we recover the original string S (because the changes that the algorithm does in the "If $l > r$" case revert the changes that the algorithm does in the "If $l < r$" case). A similar argument applies to the case when $l > r$; finally, the case when $l = r$ is obvious (since the algorithm makes no changes at all in this case). Thus, the algorithm always recovers the original string when applied twice in succession.

- **page 6:** "simple transposition of S_n" $\to$ "simple transposition of $\{1, 2, \ldots, n\}$" (or "simple transposition in S_n").
- **page 7, proof of Lemma 3.1:** "σ is an involution on $\mathcal{B}_\lambda$" $\to$ "$\sigma : \mathcal{B}_\lambda \to \mathcal{B}_\lambda$ is a bijection (being a composition of involutions)".
- **page 8, proof of Lemma 3.2:** It is not "clear" that $T^* \in \mathcal{B}_\lambda$; rather, a slightly nontrivial argument is required for this. Here is how I would prove that $T^* \in \mathcal{B}_\lambda$:

  If (u, v) is any box of $\boldsymbol{\lambda}$, then $T(u, v)$ shall denote the entry of T in this box (u, v). Likewise, $T^*(u, v)$ shall denote the corresponding entry of T^*.

  Both $(T^*)_{<j}$ and $(T^*)_{\geq j}$ are barred skew tableaux (since $(T^*)_{<j} = s_i(T_{<j})$ and $(T^*)_{\geq j} = T_{\geq j}$). Thus, the entries of T^* strictly increase along any column from top to bottom, and furthermore weakly increase along any row from left to right except maybe between the $(j-1)$-st and j-th columns. It thus remains only to show that the entries of T^* also weakly increase along any row from the $(j-1)$-st to the j-th column (provided, of course, that the row does have entries in both of these columns). In other words, we must prove that $T^*(p, j-1) \leq T^*(p, j)$ for any p for which both (p, j) and $(p, j-1)$ are boxes of $\boldsymbol{\lambda}$.

  So let us prove this. Fix a p such that both (p, j) and $(p, j-1)$ are boxes of $\boldsymbol{\lambda}$. We must prove that $T^*(p, j-1) \leq T^*(p, j)$. Assume the contrary; thus, $T^*(p, j-1) > T^*(p, j)$. However, the entries of T weakly increase along any row from left to right (since $T \in \mathcal{B}_\lambda$); hence, $T(p, j-1) \leq T(p, j)$. Furthermore, $T^*(p, j) = T(p, j)$ (since $(T^*)_{\geq j} = T_{\geq j}$). Thus,

  $$T^*(p, j-1) > T^*(p, j) = T(p, j) \geq T(p, j-1)$$

  (since $T(p, j-1) \leq T(p, j)$). This means that the entry of T in box $(p, j-1)$ increases (strictly) when T is replaced by T^*. In other words, the entry of $T_{<j}$ in box $(p, j-1)$ increases (strictly) when s_i is applied to $T_{<j}$ (because T^* is obtained from T by applying s_i to $T_{<j}$). Due to the definition of s_i, this entails that this entry is an i or an $\overline{i}$ (since the only entries that can change under s_i are i's, $\overline{i}$'s, $(i+1)$'s and $\overline{i+1}$'s, and among these entries

analogous argument applies in the case $l > r$.

only i's and $\bar{i}$'s can increase), and must become an $i+1$ or an $\overline{i+1}$ when s_i is applied to $T_{<j}$ (since this is the only way it can increase). In other words, we have

$$T(p,j-1) \in \{i, \bar{i}\} \qquad \text{and} \qquad T^*(p,j-1) \in \{i+1, \overline{i+1}\}.$$

From $T(p,j) \geq T(p,j-1) \in \{i,\bar{i}\}$, we obtain $T(p,j) \geq i$. From $T(p,j) = T^*(p,j) < T^*(p,j-1) \in \{i+1, \overline{i+1}\}$, we obtain $T(p,j) \leq i$. Hence, $T(p,j)$ is either i or $\bar{i}$ (since we also have $T(p,j) \geq i$). That is, column j of T has an i or an $\bar{i}$ in box (p,j).

However, recall that column j of T must have an unbarred $i+1$. Since we already know that this column has an i or an $\bar{i}$ in box (p,j), we thus conclude that this unbarred $i+1$ must be located immediately below this box (p,j) (since the entries of T strictly increase along any column from top to bottom). In other words, this unbarred $i+1$ must be located in box $(p+1,j)$. That is, we have $T(p+1,j) = i+1$. From $(T^*)_{\geq j} = T_{\geq j}$, we obtain $T^*(p+1,j) = T(p+1,j) = i+1$.

In particular, both $(p+1,j)$ and $(p,j-1)$ are boxes of T^*. Hence, $(p+1,j-1)$ must be a box of T^* as well (since the shape of T^* is a skew diagram). The entry of T^* in this box must satisfy $T^*(p+1,j-1) > T^*(p,j-1)$ (since the entries of T^* strictly increase along any column from top to bottom). In view of $T^*(p,j-1) \in \{i+1, \overline{i+1}\}$, this becomes $T^*(p+1,j-1) > i+1$. In other words, $i+1 < T^*(p+1,j-1)$.

However, since the entries of T weakly increase along any row from left to right, we have $T(p+1,j-1) \leq T(p+1,j)$. In other words, $T(p+1,j-1) \leq i+1$ (since $T(p+1,j) = i+1$). Thus, $T(p+1,j-1) \leq i+1 < T^*(p+1,j-1)$. This means that the entry of T in box $(p+1,j-1)$ increases (strictly) when T is replaced by T^*. In other words, the entry of $T_{<j}$ in box $(p+1,j-1)$ increases (strictly) when s_i is applied to $T_{<j}$ (because T^* is obtained from T by applying s_i to $T_{<j}$). Due to the definition of s_i, this entails that this entry is an i or an $\bar{i}$ (since the only entries that can change under s_i are i's, $\bar{i}$'s, $(i+1)$'s and $\overline{i+1}$'s, and among these entries only i's and $\bar{i}$'s can increase), and must become an $i+1$ or an $\overline{i+1}$ when s_i is applied to $T_{<j}$ (since this is the only way it can increase). In other words, we have

$$T(p+1,j-1) \in \{i, \bar{i}\} \qquad \text{and} \qquad T^*(p+1,j-1) \in \{i+1, \overline{i+1}\}.$$

Of course, $T^*(p+1,j-1) \in \{i+1, \overline{i+1}\}$ contradicts $T^*(p+1,j-1) > i+1$. This contradiction shows that our assumption was false. Hence, we have proved that $T^*(p,j-1) \leq T^*(p,j)$ for any p for which both (p,j) and $(p,j-1)$ are boxes of λ; this completes our proof of the claim that $T^* \in \mathcal{B}_\lambda$.

- **page 8, proof of Lemma 3.2:** Starting with "By Lemma 3.6(ii)" and until the end of the proof of Lemma 3.2, replace every "s_i" by "σ_i".

- **page 9, first line:** "equal to λ'_1, the number of columns of λ" $\to$ "equal to λ_1, the number of columns of λ".
- **page 9, Proposition 4.1:** It should be said that $(-y)_p$ means the tuple $(-y_1, -y_2, \ldots, -y_p)$.
- **page 9, Proposition 4.1:** I think the "$(-y)_{(\mu_i+n+1-i)}$" in part (ii) should be "$(-y)_{(\mu_j+n+1-j)}$". (At least this is what your proof yields. Maybe the other version is also correct?)
- **page 9, proof of Proposition 4.1:** When you write "(i) is proven in Macdonald [Ma2]", it's worth being more precise: Your Proposition 4.1 (i) is the equality [Ma2, (6.18)], with the caveat that the "$(\lambda_j + n - j)$" in [Ma2, (6.18)] should be "$(\lambda_i + n - i)$" (the source of the error is in the computation several lines above, where both "$\beta_k - \alpha_j$"s should be "$\alpha_j - \beta_k$"s), and that the definition of $s_\lambda (x \mid a)$ in [Ma2] is not identical with the definition in your paper (but the two definitions are equivalent because of [Ma2, (6.16)]).
- **page 9, proof of Proposition 4.1:** In "Define $\mathcal{P}_{n,m} = \{\nu \in \mathcal{P}_n \mid \nu'_1 \leq m\}$", replace "$\nu'_1$" by "$\nu_1$".
- **page 9, proof of Proposition 4.1:** Replace "$c^\mu_{\lambda,n} = (\wedge^n A)_{I_\lambda, I_\mu}$" by "$c^\mu_{\lambda,n}(y) = (\wedge^n A)_{I_\lambda, I_\mu}$".