Hook Length Formula - History and Proofs by Example

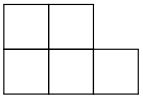
Conager Mrozek and Alissa Romero

May 6, 2023

1 Introduction to the hook length formula

A standard young tableau (SYT) diagram is a young tableau such that the rows and columns are both strictly increasing from bottom to top and left to right. Here is an example:

Example 1.1. For a partition $\lambda = (3, 2)$, then the tableaux' shape is:



for this tableau to be standard, it must be filled with five distinct, orderable numbers because if there were repeated numbers then the tableaux' rows and columns could not both be strictly increasing. Suppose we fill this tableau with a 1, 2, 3, 4, and 5.

Then by case distinction, we can show that there are five possible standard young tableau of shape $\lambda = (3, 2)$ because there is only one position to put the 1 (in the bottom left corner). Then, the 2 could go on top of the 1 or to the right of it. If the 2 is above the 1, then the 3 must be to the right of the 1 because if it was to the right of the 2, then a 4 or 5 would have to go below the 3, which would break the strictly increasing columns condition. So with the 1 in the bottom left corner, the 2 on top of the 1, and the 3 to the right of the 1; the remaining two locations for the 4 and 5 are in position $(1, 3)$ or $(2, 2)$. In the pair (i, j), i is the row and j is the column, with indexing starting at 1 from bottom to top and left to right. Therefore, we have addressed the two possible cases:

2	4	
1	3	5

2	5	
1	3	4

.

However, if the 2 is to the right of the 1, then the 3 could either go to the right of the 2 or on top of the 1. If the 3 is to the right of the 2, then the 4 and 5 must both be in row two in increasing order:

4	5	
1	2	3

.

Finally, if the 2 is to the right of the 1 and the 3 is above the 1, then the 4 and 5 can swap between positions $(1, 3)$ and positions $(2, 2)$:

3	4	
1	2	5

3	5	
1	2	4

.

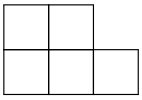
And, by case distinction, we found that there are five standard young tableau of shape $(3, 2)$. But as tableau shapes increase in size and/or become more complex, it can become difficult to enumerate the number of standard young tableau by case distinction alone. For example, there are 42 SYT of shape $\lambda = (3, 3, 3)$. The hook length formula is a way to directly compute the number of standard young tableau of any tableau of shape λ.

1.1 Hook Length Formula Statement

To understand the hook length formula, we need to define a tableau diagram and a hook length.

Definition 1.2. A young diagram is a finite collection of cells such that they are left and bottom justified. A young diagram has an associated partition λ such that $\lambda = (\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_k)$. The partition is weakly decreasing because each row in the tableau must have less than or equal to the number of cells in the row before it. Otherwise, the cells in the next row would not have a cell to sit on from the prior row. For example:

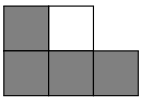
Example 1.3. We saw that the tableau



has the partition $\lambda = (3, 2)$. The partition $\lambda = (2, 3)$ would not have an associated young diagram.

Definition 1.4. The hook length of a box in a tableau is the total number of boxes above and to the right of that box in the tableau, including the box itself. For example:

Example 1.5. The hook length of the box in position $(1, 1)$ of the standard young tableaux shaped $\lambda = (3, 2)$, is 4 because there are 4 boxes in box $(1, 1)$'s "hook":



The young diagram below shows the shape $\lambda = (3, 2)$, where each box is filled with its hooks' length:

2	1	
4	3	1

Theorem 1.6. *The hook length formula*

Let f^λ be the number of standard young tableau of a tableau shape λ and $h(i, j)$ be the hook length of cell $(i, j) \in \lambda$. Then,

$$f^\lambda = \frac{n!}{\prod_{(i,j)\in\lambda} h_\lambda(i, j)},$$

where n is the total number of cells in the tableau, or, $n = \lambda_1 + \lambda_2 + \ldots \lambda_k$.

Example 1.7. We saw that the total number of standard young tableaux of shape $\lambda = (3, 2)$ is five by case distinction. And, by the hook length formula, the number of SYT of shape $\lambda = (3, 2)$ is 5! divided by the product of the hook lengths of shape λ:

$$\frac{5!}{4 * 3 * 2 * 1 * 1} = 5$$

The hook length formula was first discovered in 1953 by Robinson, Frame, and Thrall.

1.2 History of the hook length formula

Sagan [1] described the story of how Robinson, Frame, and Thrall discovered the hooklength formula, with specific details. According to Sagan, in 1953, Robinson visited Frame at Michigan State University and, together, the two mathematicians were discussing the work of Robinsons' student, Staal. Their discussion led Frame to conjecture the hooklength formula. Both mathematicians were shocked to think that such a simple formula worked, but were convinced after several examples; and Robinson and Frame worked together to prove the result. One Saturday, Robinson and Frame presented the hook length formula at a conference. Thrall was sitting in the audience and had planned to present the same result at the same conference on the same day.

Therefore, together, Robinson, Frame, and Thrall [2] published the first proof of the hook length formula in September of 1953. Their proof made use of a formula called the Frobenius formula. As exciting as Robinson, Frame, and Thrall's achievement was, their proof did not provide motivation for the importance of the hook lengths in determining the number of SYT of a shape λ. Since then, many mathematicians have searched for a more combinatorial or bijective proof to help them understand the significance of the hooks in the hook length formula. The problem of finding a simple, intuitive, combinatorial proof of the hook length formula is a challenging one.

Many proofs of the hook length formula have been discovered. Sagan [3] outlined several ways in which the hook length formula has been proven: (1) inductively, (2) probabilistically, and (3) combinatorially. Sagan explained that, while the hook length formula is easy to prove inductively, the inductive proof of the formula provides absolutely no indication of why hook lengths are an important part of the result. According to Sagan, Greene, Nijenhuis, and Wilf's [4] 1979 probabilistic proof of the hook length formula does help provide clarity about the significance of the hook lengths. And Sagan explained that the existing bijective proofs of the hook length formula are difficult to understand.

In 1982, Franzblau and Zeilberger [5] were the first to come up with a combinatorial proof of the hooklength formula based on the probabilistic proof. Franzblau and Zeilberger's bijective proof of the hook length formula, according to Sagan [3], has challenging details. The hook length formula is interesting because despite the simplicity of the formula statement itself, the known proofs of the formula lack the same level of simplicity. Sagan [3] wrote that "it was not until the 1990s that a truly straightforward one-to-one correspondence was found (even though the proof that it is correct is still somewhat involved)," which is the first proof (by example) that we will highlight.

2 Novelli, Pak, and Stoyanovskii's 1997 direct bijective proof

In 1997, Novelli, Pak, and Stoyanovskii [6] published a direct bijective proof of the hook length formula. Their proof makes use of a modified forward and reverse *jeu de taquin*, hook functions, a function called a successor function, tableau cells called "maximum candidate cells," and ordering cells in a lexicographic way. Novelli, Pak, and Stoyanovskii's proof describes a forward and backwards algorithm between two sets, and then shows that the two algorithms are inverses to prove

a bijection. We will walk through one example of the algorithm in both the forward and reverse directions. We will use the tableau shape $\lambda = (3, 2)$ for our example. The example will paint a picture of why the two algorithms are inverses; but we do not show that the two algorithms are invereses in general, which is how the original paper proved a bijection. To create a bijection, Novelli, Pak, and Stoyanovskii re-wrote the hook length formula as:

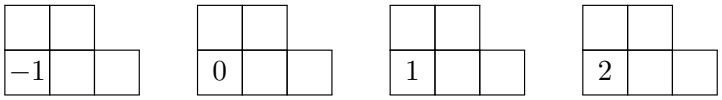
$$f^\lambda * \prod h_\lambda(i, j) = n!$$

and formed a bijection between the right hand side and left hand side of the equals sign. An important definition in Novelli, Pak, and Stoyanovskii's bijective proof is a function called a *hook function*, which sends each cell in a tableau with shape λ to an integer that satisfies certain conditions. The set of hook functions for a tableau of shape λ has cardinality equal to $\prod h_\lambda(i, j)$.

2.1 The hook function

Definition 2.1. A hook function is a filling of each cell of a tableau shape λ with an integer between minus the number of cells above it and plus the number of cells to the right of it. For example:

Example 2.2. The cell $(1, 1)$ in $\lambda = (3, 2)$ could be filled with an integer i such that $-1 \leq i \leq 2$ because there is one cell above of cell $(1, 1)$ and there are two cells to the right of cell $(1, 1)$. So the fillings of cell $(1, 1)$ in the shape $\lambda = (3, 2)$ that could be included in a hook function are:



One hook function is one way of filling an entire tableau of shape λ in this way. For example, the tableau:

1	0	
2	1	0

describes a single hook function. And the tableau:

0	0	
−1	−1	0

describes another hook function.

The set of hook functions for a young tableau of shape λ has cardinality equal to $\prod h_\lambda(i, j)$. The reason is because the set of all hook functions is generated by taking a tableau and filling each box simultaneously with a number of integers that is exactly equal to the hook length of that box. For example:

Example 2.3. For the shape $\lambda = (3, 2)$ there are four integers that can fill box $(1, 1)$ to make a hook function and the hook length of box $(1, 1)$ is four as well. Similarly, there are three integers that could go in position $(1, 2)$ to construct a hook function, one integer could go in position $(1, 3)$, two integers could go in position $(2, 1)$, and one integer could go in position $(2, 2)$. The number of integers that could fill a given box to define a hook function is equal to exactly the length of that boxs' hook.

Specifically, the box $(1,2)$ could be filled with an integer between -1 and 1, the box $(1,3)$ could only be filled with 0, the box $(2,1)$ could be filled with an integer between 0 and 1, and the box $(2,2)$ could only be filled with a 0. Then, if all of these fillings of boxes could occur simultaneously for the tableau shape $\lambda = (3,2)$, then the number of possible hook functions is $4*3*2*1*1 = 24$, which is exactly $\prod h_\lambda(i,j)$. And the set of hook functions for the tableau of shape $\lambda = (3,2)$ are:

0 0 2 −1 0	0 0 2 0 0	0 0 2 1 0	1 0 2 −1 0	1 0 2 0 0	1 0 2 1 0
0 0 1 −1 0	0 0 1 0 0	0 0 1 1 0	1 0 1 −1 0	1 0 1 0 0	1 0 1 1 0
0 0 0 −1 0	0 0 0 0 0	0 0 0 1 0	1 0 0 −1 0	1 0 0 0 0	1 0 0 1 0
0 0 −1 −1 0	0 0 −1 0 0	0 0 −1 1 0	1 0 −1 −1 0	1 0 −1 0 0	1 0 −1 1 0

2.2 Outline of Novelli, Pak, and Stoyanovskii's bijection

Novelli, Pak, and Stoyanovskii's proof of the hook length formula relies on the observation that the number of SYT of shape λ times the number of hook functions of shape λ is equal to the total possible fillings of the tableau shape λ with $n = \lambda_1 + \lambda_2 + ... + \lambda_k$ distinct, orderable entries. In other words:

$$\left(\begin{array}{c}\text{Number of}\\ \text{SYT shape } \lambda\end{array}\right) * \left(\begin{array}{c}\text{Number of hook}\\ \text{functions of a}\\ \text{tableau shape } \lambda\end{array}\right) = \left(\begin{array}{c}\text{Total ways to fill a tableau of}\\ \text{shape } \lambda\text{, which has } n \text{ boxes, with}\\ n \text{ distinct, orderable entries}\end{array}\right)$$

Example 2.4. For example, we saw by case distinction that for $\lambda = (3,2)$ the the number of total fillings of the young tableau to make the shape $\lambda = (3,2)$ standard was 5. Additionally, we saw that there are 24 possible hook functions on the tableau shape $\lambda = (3,2)$. And the ways of filling a tableau with 5 boxes with 5 distinct characters is 5!. We can check that $5 * 24 = 120 = 5!$.

The way that Novelli, Pak, and Stoyanovskii proved the hook length formula was by pairing each SYT of shape λ with every hook function of shape λ. Then, they wrote an algorithm such that each SYT/hook function pair mapped onto one of the $n!$ fillings of a young tableau with n distinct, orderable characters.

Example 2.5. For example, each of the five SYT of shape $\lambda = (3,2)$ gets paired with the 24 hook functions on the shape $\lambda = (3,2)$ and there is an algorithm from all 120 SYT/hook function pairs onto the $n!$ fillings of the tableau shape $\lambda = (3,2)$.

Next, we describe the two algorithms: (1) $n! \to f^\lambda * \prod h_\lambda(i,j)$ and (2) $f^\lambda * \prod h_\lambda(i,j) \to n!$, and continue to use the tableau shape $\lambda = (3,2)$ as an example. The algorithm in both directions

requires the position of a cell in a young tableau shape. So the correct bijection is really between sets of triples which include a young tableau, hook function, and cell's position. But this bijection includes the set that we are looking at, and is still enough to prove the hook length formula.

2.3 The algorithm $n! \to f^\lambda * \prod h_\lambda(i,j)$

The algorithm in this direction relies on three input values: (1) a young tableau (one of $n!$, not necessarily standard), (2) a hook function, f, where the cells are all filled with a 0, and (3) a cell's position in the tableau: (i_0, j_0). To choose the position of (i_0, j_0), we always start with the first position in a totally ordered tableau.

Definition 2.6. A totally ordered tableau is filled with a 1 in the rightmost and topmost cell. Then, moving down and to the right, the numbers increase by 1.

Example 2.7. For the tableau shape $\lambda = (3, 2)$ the total order looks like this:

4	2	
5	3	1

So cell (i_0, j_0) is where the 1 is, which is in the cell $(1, 3)$ in this case.

Example 2.8. An example of an input for our algorithm is:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 1 & \\ \hline 5 & 3 & 4 \\ \hline \end{array} \;,\; \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & 0 & 0 \\ \hline \end{array} \;,\; (1,3) \right).$$

Because the tableau is one of the $n!$ possible tableau, we have a hook function of all zeros, and $(1, 3)$ is the first cell in our total order.

Then, we iterate steps of an algorithm on the young tableau, hook function, and cell position triple to make a new triple.

Algorithm 2.9.

1. Pick (i_0, j_0) to be the first cell in the total order of the tableau shape λ.
2. Let (i_1, j_1) be the cell after (i_0, j_0) in total order. And let a be the number in the position of (i_1, j_1) in the tableau.
3. If a is greater than the cell above it or to the right of it, then switch a with the smaller of those two cells. Repeat this step for a until it cannot be repeated again because no cell above or to the right of a is smaller than a.
4. After step 3, let (i_2, j_2) be the position of a in the new tableau created by step 2.
5. And adjust the hook function according to the following conditions:
   - for all $i_1 \leq i < i_2$, put $f'(i, j_1) = f(i + 1, j_1) - 1$
   - $f'(i_2, j_1) = j_2 - j_1$
   - $f'(i, j) = f(i, j)$ otherwise

Where f is the original input hook function and f' is the hook function that goes through the next iteration of the algorithm.

6. Now, we have a new tableau from step 2 and a new hook function, f', from step 5. Repeat steps 2-6, replacing (i_0, j_0) with the cell following it in total order.

Example 2.10. For example, we know that our triple in example 2.8 has $(i_0, j_0) = (1, 3)$ so $(i_1, j_1) = (2, 2)$ because $(2, 2)$ is the next in total order. And $a = 2$ because 2 is the number in the position $(2, 2)$ of the young tableau. Then, in step 2, we have that:

$$\begin{array}{|c|c|c|}\hline 4 & 2 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 1 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|c|}\hline 4 & 2 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 1 \\ \hline \end{array}$$

because 2 does not move positions, since there is no cell above or to the right of 2 that is smaller than it. So by step 3 $(i_2, j_2) = (2, 2)$ And, by step 4, $(2, 2) = f(3, 2) - 1$, but $f(3, 2)$ is not a cell that is defined, $(2, 2) = 0$ and the rest of the cells in the hook function f' are the same as the last entry to the algrorithm. so our new triple is:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 1 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \multicolumn{1}{c}{} \\ \hline 0 & 0 & 0 \\ \hline \end{array} \ , \ (2,2) \right).$$

Then, this triple iterates through the algorithm and so do the following triples until we have a SYT. The first iteration was not as interesting, but over time the iterations eventually change the tableau into a SYT and the hook function into one of the 24 possible hook functions. This specific triple put through the algorithm iterates through the triples in the order below:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 1 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \multicolumn{1}{c}{} \\ \hline 0 & 0 & 0 \\ \hline \end{array} \ , \ (3,1) \right) \rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 1 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \multicolumn{1}{c}{} \\ \hline 0 & 0 & 0 \\ \hline \end{array} \ , \ (2,2) \right)$$

$$\rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 3 & \multicolumn{1}{c}{} \\ \hline 5 & 1 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \multicolumn{1}{c}{} \\ \hline 0 & -1 & 0 \\ \hline \end{array} \ , \ (1,2) \right) \rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 3 & \multicolumn{1}{c}{} \\ \hline 5 & 1 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \multicolumn{1}{c}{} \\ \hline 0 & -1 & 0 \\ \hline \end{array} \ , \ (2,1) \right)$$

$$\rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 5 & \multicolumn{1}{c}{} \\ \hline 1 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 1 & 0 & \multicolumn{1}{c}{} \\ \hline -1 & -1 & 0 \\ \hline \end{array} \ , \ (1,1) \right)$$

Therefore, we know that the triple we started with, which contains a tableau from the set of 5! tableaux of shape $\lambda = (3, 2)$ is in bijection with a triple at the end of this algorithm. Even though the triples are what define the bijection, by looking at the first entries in the triples (the young tableau alone), we will see that the algorithm includes a mapping from a tableau onto a standard young tableau. And the SYT that is mapped to is just paired with a specific hook function as and cell, which are required for the algorithm:

$$\begin{array}{|c|c|c|}\hline 2 & 1 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 4 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|c|}\hline 2 & 3 & \multicolumn{1}{c}{} \\ \hline 5 & 1 & 4 \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|c|}\hline 2 & 5 & \multicolumn{1}{c}{} \\ \hline 1 & 3 & 4 \\ \hline \end{array}$$

2.4 The algorithm $f^\lambda * \prod h_\lambda(i,j) \to n!$

The algorithm in this direction relies on three input values, which are similar to those in the other direction: (1) a SYT of shape λ, (2) any hook function f on the shape λ, and (3) the greatest cell in the totally ordered tableau of shape λ.

Example 2.11. An example of a triple in this direction for the tableau shape $\lambda = (3,2)$ is:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 5 & \multicolumn{1}{c}{} \\ \hline 1 & 3 & 4 \\ \hline \end{array} \quad , \quad \begin{array}{|c|c|c|}\hline 1 & 0 & \multicolumn{1}{c}{} \\ \hline -1 & -1 & 0 \\ \hline \end{array} \quad , \quad (1,1) \right)$$

The algorithm iterates in an order that first uses the greatest cell in total order and then works backwards to the smallest cell in total order.

Example 2.12. Notice that $(1,1)$ is the greatest cell in the total order of $\lambda = (3,2)$ because the total order is:

$$\begin{array}{|c|c|c|}\hline 4 & 2 & \multicolumn{1}{c}{} \\ \hline 5 & 3 & 1 \\ \hline \end{array}$$

and $(1,1)$ is the position of the 5.

Now, we need to define a cell called a 'candidate cell' in our SYT, which depends on a cell's position (i_0, j_0).

Definition 2.13. A candidate cell of (i_0, j_0) is one that meets the following conditions:

- for all $i \geq i_0'$
- $f(i, j_0') \geq 0$ and $j = j_0' + f(i, j_0')$

Example 2.14. For example, if $(i_0', j_0') = (1,1)$ and the tableau and hook function are the ones given by example 2.11, then the candidate cell must have $i \geq 1$, so i could be 1 or 2. But additionally, the condition $f(1,1) \geq 0$ and $f(2,1) \geq 0$ must be satisfied. Since this is only true for $i = 2$, then we have $j = 1 + f(2,1) = 2$. With $i = 2$ and $j = 2$, we know that the candidate cell is the one in position $(2,2)$ of our SYT. Therefore, the candidate cell is 5 in this case.

But in general, it is possible to have more than one candidate cell. If there is more than one candidate cell, then we can define a cell called a 'maximum candidate cell.'

Definition 2.15. A maximum candidate cell is a candidate cell with a largest lexicographic coding, where the lexicographic coding is defined by looking at how a cell moves get to position (i_0', j_0'). To move to position (i_0', j_0'), a cell switches with the larger of the cell below it and to the left of it until it is at position (i_0', j_0'). And when a cell moves left, it is coded with a W (for a west move); but when a cell moves down, it is coded with an S (for a south move). Then, reading from right to left, the maximum candidate cell is the one with the largest lexicographic coding, where $S < \emptyset < N$.

Example 2.16. For example, if the tableau and hook function are

6	11	
5	9	10
3	4	8
1	2	7

1	0	
−1	1	0
2	−1	0
1	1	−2

and the cell's position that we want to find the maximum candidate cell of is $(i_0, j_0) = (1, 1)$, then the possible candidate cells are 2, 8, and 11 because our conditions are that $i \geq 1$ and for each i besides $i = 3$, $f(i, j'(0) \geq 0$. So we can have $j = 1 + f(1, 1) = 2$ for $i = 1$, $j = 1 + f(2, 1) = 3$ for $i = 2$, or $j = 1 + f(3, 1) = 2$ for $i = 4$. So our candidate cells are $(1, 2)$, $(2, 3)$, and $(4, 2)$, which are the positions of the 2, 8, and 11. Then, to move 2 to position $(1, 1)$, we swap the 2 and 1, so the reading word is W, to move the 8 to position $(1, 1)$, the reading word is SWW, and to move the 11 to position $(1, 1)$, the reading word is SWSS. Therefore, reading right to left, 8 has the largest lexicographic coding.

Now we have all the information we in order to state the algorithm in the second direction.

Algorithm 2.17.

1. Let (i'_0, j'_0) be the largest cell in our total order.
2. Find the maximum candidate element p of the position (i'_0, j'_0). Denote the location of p in the SYT by (i'_1, j'_1).
3. Exchange p with the greatest of the cell to the left of it and below it until p is in position (i'_0, j'_0).
4. Adjust the hook function according to the following conditions:
   - for all $i'_0 < i \leq i'_1$, put $f'(i, j'_0) = f(i - 1, j'_0) + 1$
   - $f'(i'_0, j'_0) = 0$
   - $f'(i, j) = f(i, j)$ otherwise.

   Where f is the original input hook function and f' is the adjusted hook function that goes through the next iteration of the algorithm.
5. Repeat steps 1-4 using the tableau that is made after step 3, f', and the next largest cell in total order.

Example 2.18. We know that for the SYT in example 2.11, the candidate cell is 5, which is in position $(2, 2)$. So $(i'_0, j'_0) = (1, 1)$ and $(i'_1, j'_1) = (2, 2)$; and, therefore, moving 5 to position $(1, 1)$ according to step 3 maps the tableau:

2	5	
1	3	4

$\rightarrow$

2	3	
1	5	4

$\rightarrow$

2	3	
5	1	4

And we adjust the hook function so that for $1 < i \leq 2$, then $f(2, 1) = f'(2 - 1, 1) + 1 = -1 + 1 = 0$, $f'(1, 1) = 0$, and $f'(i, j) = f(i, j)$ otherwise. So the resulting hook function is:

0	0	
0	−1	0

Finally, we let (i'_0, j'_0) be the next largest cell in our total order. So in total, the triple we started with in our algorithm maps to:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline 5 & 1 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & -1 & 0 \\ \hline \end{array} \ , \ (2,1) \right)$$

And the algorithm iterates triples through it until (i'_0, j'_0) is the cell 1 (or the smallest cell) in our total order. And in this case, the algorithm iterates through the triples:

$$\left(\begin{array}{|c|c|c|}\hline 2 & 5 & \\ \hline 1 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 1 & 0 & \\ \hline -1 & -1 & 0 \\ \hline \end{array} \ , \ (1,1) \right) \rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline 5 & 1 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & -1 & 0 \\ \hline \end{array} \ , \ (2,1) \right)$$

$$\rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 3 & \\ \hline 5 & 1 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & -1 & 0 \\ \hline \end{array} \ , \ (1,2) \right) \rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 1 & \\ \hline 5 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & 0 & 0 \\ \hline \end{array} \ , \ (2,2) \right)$$

$$\rightarrow \left(\begin{array}{|c|c|c|}\hline 2 & 1 & \\ \hline 5 & 3 & 4 \\ \hline \end{array} \ , \ \begin{array}{|c|c|c|}\hline 0 & 0 & \\ \hline 0 & 0 & 0 \\ \hline \end{array} \ , \ (1,3) \right)$$

It at least appears, in our examples, that the forward and reverse algorithms between the triples we looked at were inverses and that we have enough information to state the hook length formula. And in order to prove a bijection between sets of triples, Novelli, Pak, and Stoyanovskii showed that the sets of triples used in the algorithm include the correct set for the hook length formula and that the general algorithms in both directions are inverses.

3 Richard Stanley Proof

Richard Stanley has a number of proofs stated in his textbook. The proof that we will focus on seems to be the most straightforward. This proof will require the use of a lemma which will be used to prove his main theorem, from this theorem a corollary arises that will give the proof for the hook-length formula.

3.1 Definitions

Definition 3.1. For $u \in \lambda$ let $c(u) = j - i$ the *content* of square $u = (i, j)$. Where i is the *i-th* row and j is the *j-th* column.

Example 3.2. Semi-Standard Young Tableau: $\begin{array}{|c|c|c|}\hline 2 & 4 & \\ \hline 1 & 3 & 5 \\ \hline \end{array}$ → Content Tableau: $\begin{array}{|c|c|c|}\hline -1 & 0 & \\ \hline 0 & 1 & 2 \\ \hline \end{array}$

Definition 3.3. Define *hook-length*, $h(u)$ of λ at u by $h(u) = \lambda_i + \lambda'_j - i - j + 1$.

Example 3.4. Semi-Standard Young Tableau: $\begin{array}{|c|c|c|}\hline 2 & 4 & \\ \hline 1 & 3 & 5 \\ \hline \end{array}$ → Hook Length Tableau: $\begin{array}{|c|c|c|}\hline 2 & 1 & \\ \hline 4 & 3 & 1 \\ \hline \end{array}$

Definition 3.5. Define $b(\lambda)$ to be the smallest possible sum of entries of a semi-standard young tableau of shape λ. Obtained by placing $i-1$ in all of the squares of the i-th row of λ.

Example 3.6. Notice there are a number of ways to calculate $b(\lambda)$ so let's see;

2	4	
1	3	5

$\rightarrow$ Replacing first row with 0's

2	4	
0	0	0

$\rightarrow b(\lambda) = 6.$

2	4	
1	3	5

$\rightarrow$ Replacing second row with 1's

1	1	
1	3	5

$\rightarrow b(\lambda) = 11.$

Notice that $b(\lambda)$ is correctly obtained in this example by replacing the first row with 0's to obtain $b(\lambda) = 6$.

Definition 3.7. Define $l(\lambda)$ to be the length of λ. (for $\lambda = (3,2), l(\lambda) = 2$)

Definition 3.8. Let $|\lambda/\mu| = n$. then

$$s_{\lambda/\mu}(1, q, q^2, ...) = \frac{\sum_T q^{maj(T)}}{(1-q)(1-q^2)...(1-q^n)}$$

Lemma 3.9. *Let $\lambda = (\lambda_1, ..., \lambda_n)$ be a partition and $\mu_i = \lambda_i + n - i$. Then*

$$\prod_{u \in \lambda} [h(u)] = \frac{\prod_{i \geq 1} [\mu_i]!}{\prod_{i \leq j} [\mu_i - \mu_j]} \qquad (1)$$

$$\prod_{u \in \lambda} [n + c(u)] = \prod_{i=1}^{n} \frac{[\mu_i]!}{[n-i]!} \qquad (2)$$

Where $[k] = 1 - q^k$ and $[k]! = [1][2]...[k]$.

Proof. We will proceed by example; let $\lambda = (5, 3, 2, 2)$, notice that for this specific choice of λ that $n = 4$. The content of each cell in λ is given by the hook length of that particular cell. For this example the hook length tableau looks like;

2	1			
3	2			
5	4	1		
8	7	4	2	1

The shape of μ is given in the statement of the lemma so we have;

$$\begin{aligned} \mu_1 &= 5 + 4 - 1 = 8 \\ \mu_2 &= 3 + 4 - 2 = 5 \\ \mu_3 &= 2 + 4 - 3 = 3 \\ \mu_4 &= 2 + 4 - 4 = 2 \\ \mu &= (8, 5, 3, 2) \end{aligned}$$

Define the content of cell $(i, j) \in \mu$ to be $\mu_i - j + 1$, then μ looks like;

2	1						
3	2	**1**					
5	4	**3**	**2**	1			
8	7	**6**	**5**	4	**3**	2	1

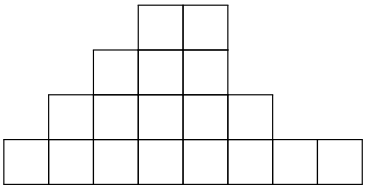
Notice that if we remove the boldface cells from μ we will obtain the hook length tableau of shape λ. Also, each row in μ corresponds with a $\mu_i!$, the process of removing the cells would be equivalent to dividing by them. However, the boldface cells correspond directly with a particular $[\mu_i - \mu_j]$. We can notice that the right hand side of (1) follows directly from this. The left hand side of (1) is simply the product of the entries in the hook length tableau. Let us verify this calculation;

$$\begin{aligned}
\prod_{u\in\lambda} h(u) &= \frac{\prod_{i\geq 1}\mu_i!}{\prod_{i<j}(\mu_i-\mu_j)} \\
8\cdot 7\cdot 5\cdot 4^2\cdot 3\cdot 2^3 &= \frac{8!5!3!2!}{(8-2)(8-3)(8-5)(5-2)(5-3)(3-2)} \\
&= \frac{8!5!3!2!}{6\cdot 5\cdot 3\cdot 3\cdot 2\cdot 1} \\
&= \frac{8\cdot 7\cdot 5!\cdot 4!\cdot 2!}{3} \\
&= 8\cdot 7\cdot 5\cdot 4^2\cdot 3\cdot 2^3
\end{aligned}$$

For (2) let λ remain the same, the $n + c(u)$ tableau of λ looks like;

1	2			
2	3			
3	4	5		
4	5	6	7	8

The shape of μ has already been calculated however, this will be a degenerate tableau shape;



The content of this μ in cell (i, j) will be given by $j - i + 1$ resulting in the tableau;

		1	2				
	1	2	3				
1	**2**	3	4	5			
1	**2**	**3**	4	5	6	7	8

Notice that if we remove the boldface entries from μ we will obtain the $n + c(u)$ tableau of shape λ. Again, notice that each row of μ corresponds to a $\mu_i!$ and again the process of removing the cells would be equivalent to dividing by them. These boldface entries correspond directly with a particular $(n-i)!$, for example the third column corresponds with $(4-1)!$. The right hand side of (2) follows from these observations and the left hand side of (2) is simply the product of the entries in the $n + c(u)$ tableau of shape λ.

□

3.2 The Main Theorem

Theorem 3.10. *For any λ, partition and $n \in \mathbb{Z}^+$ we have;*

$$s_\lambda(1, q, ..., q^{n-1}) = q^{b(\lambda)} \prod_{u \in \lambda} \frac{[n + c(u)]}{[h(u)]}$$

Proof. If $n < l(\lambda)$, then this equation becomes trivial by the definition of Schur Functions. Consider $n \geq l(\lambda)$. From the algebraic definition of Schur functions we can write;

$$s_\lambda(1, q, ..., q^{n-1}) = \frac{a_{\lambda+\rho}(1, q, ..., q^{n-1})}{a_\rho(1, q, ..., q^{n-1})}$$

Now, we have that $a_\rho = \prod_{i<j}(x_i - x_j)$ thus $a_\rho(1, q, ..., q^{n-1}) = \prod_{i<j}(q^{i-1} - q^{j-1})$
Notice also that the numerator $a_{\lambda+\rho}(1, q, ..., q^{n-1}) = (-1)^{\binom{n}{2}} \prod_{i<j}(q^{\mu_i} - q^{\mu_j})$ where $\mu_j = \lambda_j + n - j$[7]. Thus we can rewrite,

$$s_\lambda(1, q, ..., q^{n-1}) = (-1)^{\binom{n}{2}} \prod_{i<j} \frac{q^{\mu_i} - q^{\mu_j}}{q^{i-1} - q^{j-1}}$$

And by Lemma 3.9 we can now write;

$$s_\lambda(1, q, ..., q^{n-1}) = \frac{q^{\sum_{i<j} \mu_j} \prod_{i<j}[\mu_i - \mu_j] \prod_{i \geq 1}[\mu_i]!}{q^{\sum_{i<j}(i-1)} \prod_{i<j}[j-i] \prod_{i \geq 1}[\mu_i]!}$$

In which we can simplify some things;

$$s_\lambda(1, q, ..., q^{n-1}) = q^{b(\lambda)} \frac{\prod_{i<j}[\mu_i - \mu_j]}{\prod_{i<j}[j-i]}$$

One last simplification gives;

$$s_\lambda(1, q, ..., q^{n-1}) = q^{b(\lambda)} \prod_{u \in \lambda} \frac{[n + c(u)]}{[h(u)]}$$

□

Corollary 3.11. *For any λ that is a partition;*

$$\sum_T q^{maj(T)} = \frac{q^{b(\lambda)}[n]!}{\prod_{u \in \lambda}[h(u)]}$$

where T is all standard Young tableau of shape λ.

Proof. The proof of this corollary follows directly from Definition 3.8 and the algebraic definition of Schur Functions. □

3.3 The Hook Length Formula

Remark 3.12. Notice that if we set $q = 1$ in Corollary 3.11, that the left hand side will count the number of Standard Young tableau, T, of shape λ. On the right hand side, we will lose the term $q^{b(\lambda)}$ and also we will lose any q-analogues. Thus we can simplify this corollary, when $q = 1$ to be;

$$f^{\lambda} = \frac{n!}{\prod h_{\lambda}(i,j)}$$

4 Catalan Numbers and Hook Length

One of the beautiful things about mathematics is the ability to connect two seemingly different ideas into one magnificent statement. This is the case for the Catalan Numbers and Hook Length, although once the connection is shown it seems rather obvious. The discovery of this connection must have been the most interesting to discover.

Remark 4.1. Consider the number of Standard Young Tableau of shape $\lambda = (1,1)$, with content given from [2]. This is relatively simple to construct; $\begin{array}{|c|}\hline 2 \\ \hline 1 \\ \hline \end{array}$. We also know that $C_n = \frac{1}{n+1}\binom{2n}{n}$ is the *n-th* Catalan Number, and $C_1 = \frac{1}{2}\binom{2}{1} = 1$.
Let's make this more formal and then construct a claim.

Example 4.2. Let $n = 2$ then $\lambda_2 = (2,2)$ and we are given content from [4]. Let's construct all of the possible standard Young tableau of this shape and content; $\begin{array}{|c|c|}\hline 2 & 4 \\ \hline 1 & 3 \\ \hline \end{array}$ and $\begin{array}{|c|c|}\hline 3 & 4 \\ \hline 1 & 2 \\ \hline \end{array}$ also we have that $C_2 = \frac{1}{3}\binom{4}{2} = 2$.

Remark 4.3. As n increases, the position of $2n$ and 1 in the Young Tableau must not be changed to preserve standard-ness.

Conjecture 4.4. *Let $\lambda_n = (n,n)$ with content given from $[2n]$, the number of standard Young Tableau of shape λ_n is the n-th Catalan Number.*

Proof. Without loss of generality consider the standard Young Tableau given by;

2	4	6	...	$2n$
1	3	5	...	$2n-1$

if we construct the hook length tableau given by this SYT we have;

n	$n-1$	$n-2$	...	1
$n+1$	n	$n-1$	...	2

Then we have;

$$f^{\lambda_n} = \frac{(2n)!}{\prod h(u)} = \frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1}\frac{(2n)!}{n!n!} = \frac{1}{n+1}\binom{2n}{n} = C_n$$

□

5 References

[1] B.E. Sagan, *The symmetric group: Representations, combinatorial algorithms, and symmetric functions*, Springer Science+Business Media New York, (2001).

[2] J. S. Frame, G. B. Robinson, and R. M. Thrall, *The hook graphs of the symmetric group.* Canad. J. Math. (1954).

[3] B.E. Sagan, *The ubiquitous young tableau.* Michigan State University, (1988).

[4] C. Greene, A. Nijenhuis, and H. S. Wilf, *A probabilistic proof of a formula for the number of Young tableaux of a given shape.* Adv. in Math. (1979).

[5] D. S. Franzblau and D. Zeilberger, *A bijective proof of the hook-length formula.* J. Algorithms, Vol. 3, (1982).

[6] J. C. Novelli, I. Pak, A. V. Stoyanovskii, *A direct bijective proof of the hook-length formula.* Discrete Mathematics and Theoretical Computer Science, Vol. 1, (1997).

[7] R. Stanley, *Enumerative Combinatorics, Vol. 2*, Cambridge University Press, Cambridge (1999).