Proofs of the Hook Length Formula

Arleigh Dickerson

December 8, 2014

Abstract

The hook-length formula of Frame, Robinson, and Thrall yields the number of possible standard tableau for a given partition of an integer. Although the formula itself is straightforward, proofs of the formula can be complicated. We will explore some of the different methods combinatorialists have used to prove the formula and discuss the merits and implications associated with each.

1 Introduction

Young tableaux and the hook-length formula are fascinating to combinatorialists because young tableaux are indices on the basis of S^λ, where S^λ is a *Specht module*. Specht modules are irreducible representations which correspond to fillings in the context of tableaux. Although the Young tableau have a direct algebraic representation, they are also combinatorial objects. The hook length formula counts these tableaux and was first proved algebraically. Combinatorial, particularly bijective, proofs were elusive for many years.

The formula was first conjectured by Frame[1] at Michigan State University during a visit from his colleague, Robinson. Frame first conjectured the formula on a Thursday while the two were discussing the work of Staal, one of Robinson's students. Together, the two proved the formula algebraically. The following Saturday, Frame presented their proof at the University of Michigan where Thrall, a member of the audience, had proven the same result on the same day.

Novelli, Pak, and Stoyanovskii[3] use a different method to prove the formula both directly and bijectively. By assembling two well-defined algorithms that

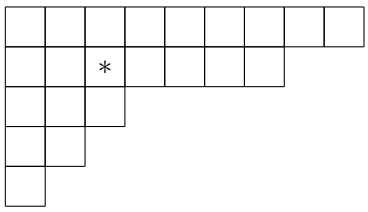

Figure 1: A Ferrers diagram with the shape $\lambda = \{9, 7, 3, 2, 1\}$ and an asterisk placed on coordinates $(i = 2, j = 3)$.

map from the set of all standard tableaux to and from pairs of a filling and a hook function, the authors are able to establish a bijection between the two sets. Once this bijection has been established, they establish the cardinality of the second set, which must be equal to the cardinality of the first.

2 Definitions

In essence, a tableau is a way of visualizing values associated an *integer partition*. A partition of a positive integer, n, is a sequence of non-increasing positive integer that sum to n. For example $\lambda = \{4, 2, 1, 1\}$ is a partition of 8 because $4 \geq 2 \geq 1 \geq 1$ and $4 + 2 + 1 + 1 = 8$. Subscripts on λ refer to an element of the partition. For example: $\lambda_1 = 4, \lambda_2 = 2, \lambda_3 = 1$, etc. In our context, we refer to λ as the *shape* of the diagram.

An integer partition, λ, can be represented visually as a *Ferrers diagram*. A Ferrers diagram is a collection of one or more left-justified rows of cells where the number of cells in the top row is equal λ_1, the number of cells in the second-to-top row is equal to λ_2, etc. Cells of a Ferrers diagram are given ordered-pair coordinates (i, j) where $i, j \in \mathbb{Z}^+$, $1 \leq i \leq |\lambda|$, and $1 \leq j \leq \lambda_i$. The pair (i, j) refers to the cell in the jth column of the ith row of the Ferrers diagram being considered.

A *tableau* is a Ferrers diagram in which each box has been assigned a value. We refer to the value of tableau T at coordinate (i, j) by writing T_{ij}. In our context, a tableau will contained no repeated values and all values $v \in T_{ij}$ satisfy $1 \leq v \leq n$. For instance, a tableau with shape $\lambda = \{3, 2\}$ will have $n = 5$ and therefore only contain the values 1,2,3,4,and 5.

Definition 1. A tableau is a *standard tableau* if its cell values are strictly

3	21	7	20	8	18	19	9	10
11	1	13	15	16	17	22		
2	4	14						
5	6							
12								

Figure 2: A tableau for the partition $\lambda = \{9, 7, 3, 2, 1\}$.

1	3	5
2	4	

1	2	5
3	4	

1	3	4
2	5	

1	2	4
3	5	

1	2	3
4	5	

Figure 3: All standard tableaux for $\lambda = \{3, 2\}$.

increasing on all rows when read from left to right and on all columns when read from top to bottom. Figure 2 shows a tableau which is standard and figure 2 shows a tableau which is not.

A tableau is said to be *ordered to* coordinates (i, j) if definition 1 holds for the cells below, to the right of, or indexed by (i, j).

The *hook* of a cell contained in a Ferrers diagram is the set containing the union of the arm of the cell and leg of the cell. The arm of a cel is defined as all cells in the same row and to the right of the cell in question. The leg of a cell is defined as all cells in the same column and below the cell in question. We will let $h(i, j)$ define the cardinality of this selection. The *hook length* of a cell is the cardinality of the set containing the hook of the cell and the

	1	13	15	16	17	22		
	4	14						
	6							

Figure 4: A tableau with values to the left of or above $(2, 2)$ removed. The tableau is said to be ordered to $(2, 2)$ because the remaining values are increasing from left to right and from top to bottom.

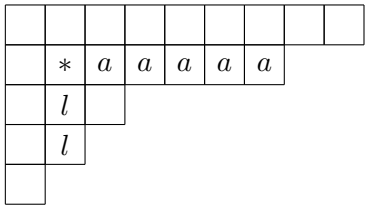

Figure 5: The hook of cell $(2, 2)$. Cells in the arm are marked with an a, cells in the leg is marked with l, and cell $(2, 2)$ is marked with an asterisk.

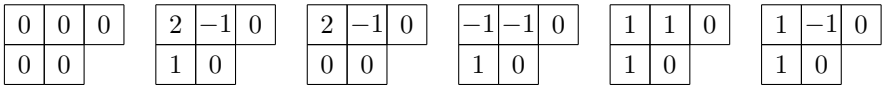

Figure 6: Some hook functions for $\lambda = (3, 2)$

cell itself.

We can use the hook of all cells of a Ferrers diagram to establish a *hook function* for the diagram. A hook function for a diagram shape, λ, takes any coordinate of λ as input and returns an integer value such that $-(\lambda_j - i) \leq f(i, j) \leq (\lambda_i - j)$. Finally, the hook length formula yields the cardinality of the set containing all possible standard fillings of the shape λ [2].

Theorem 1. $f_\lambda = \dfrac{n!}{\prod_{i,j \in \lambda} h(i, j)}$

3 Jeu de Taquin

The Jeu de Taquin (teasing game) was first introduced by Schutzenberg. A *Jeu de Taquin* slide provides us with a standardized method to move a tableau closer to standard form by swapping the position of two values[4]. To perform a forward slide on a cell (i, j), we look to candidate cells $T_{i,j+1}$ and $T_{i+1,j}$. If both candidates exist, we exchange (i, j) with the smaller of the two candidate cell values. If only one candidate exists, we choose that candidate. Finally, if no candidates exist, no slide is possible. Forward slides can be used to move a tableau closer to standard form. First, we find top and leftmost cell to which the tableau is ordered. For example, the first tableau of figure 7 is ordered to cell value 4. We perform our slide on the

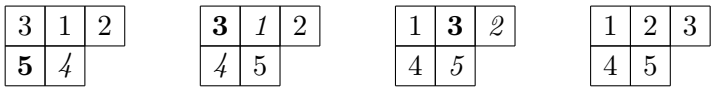

Figure 7: Performing a forward Jeu de Taquin slide. The cell we are sliding is emboldened and the candidate cells are italicized.

cell directly to the left of this cell, which in the previous example is cell 5. This procedure is repeated until the tableau is in standard form.

Forward Jeu de Taquin establishes a surjection between the set of all tableaux and the set of standard tableaux for a given λ. The set of all possible tableaux includes all standard tableaux by definition. Performing forward Jeu de Taquin on these standard tableaux will map them to themselves as the algorithm will terminate immediately, returning the input values unchanged. We can also see that forward Jeu de Taquin will take any tableau of a given shape and produce a standard tableau of that same shape. It is also clear that some nonstandard tableaux will map to the same standard tableau. To establish a bijection between these two sets, we will need information in the codomain that reflects the value of the domain.

4 A direct bijective proof

Novelli, Pak, and Stoyanovskii[3] establish a bijection between two sets defined for any given tableau shape λ.

I: The set of pairs (A, f) where A is a standard tableau f is a hook function.

II: The set of all possible tableaux.

If a bijection has been established between two sets then by definition the two sets must have equal cardinality. After the authors establish their bijection, they use this fact to prove that the hook length formula is indeed valid.

4.1 Mapping II to I

To map an element of II to an element of I, the authors establish three algorithms: Algorithm P, Algorithm 1, and Algorithm 2. These algorithms perform forward Jeu de Taquin and record which element was slid in what direction by modifying the hook function.

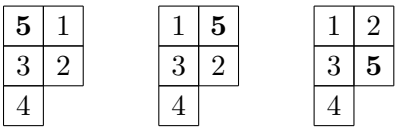

Figure 8: Algorithm P applied to cell $a = 5$.

4.1.1 Algorithm P

Algorithm P accepts a tableau and a member element as input and produces a tableau which is *closer* to, or at least not farther from, standard form. Specifically, it is the repeated application of forward Jeu de Taquin to the same value until no candidates remain.
Input: a pair (A, a) where A is a tableau and $a \in \{1, 2, \ldots, n\}$
. **Output**: A tableau
begin
  $(i0, j0) \leftarrow$ the position of a in A;
  $b \leftarrow A(i_0, j_0 + 1)$ if $j_0 < \lambda_{i_0}$ else $n + 1$;
  $c \leftarrow A(i_0 + 1, j_0)$ if $i_0 < \lambda'_{j_0}$ else $n + 1$;
  if $a > b$ *or* $a > c$ **then**
    $B \leftarrow A$ with element a swapped with the least of b and c;
    return *AlgorithmP(B,a)*
  else
    return A
  end if
end

4.1.2 Algorithm 1

Algorithm 1 accepts a tableau, a hook function, and coordinates as input. It then calls algorithm P with the tableau and the value of the input coordinates as arguments. Then, information pertaining to how the result of algorithm P differs from the original input is stored in the hook function. Finally, it returns the modified tableau, the hook function reflecting the modifications, and the successors to the input coordinates.

For a given tableau shape λ, we define the successor to a coordinate (i, j) as $s(i, j)$ where $(i, j) \neq (1, 1)$ and $(i, j) \in \lambda$ as the coordinate of the next entry in the *successor map*, s, of λ. To find $s(i, j)$, first look to the cell directly above (i, j). If this cell exists, then it is the successor. Otherwise, choose

22	17	13	10	8	6	4	2	1
21	16	12	9	7	5	3		
20	15	11						
19	14							
18								

Figure 9: The successor map for $\lambda = \{9, 7, 3, 2, 1\}$.

Tableau	6	2	6	2	6	1	6	1	6	1	1	4
	4	3	4	1	4	2	4	2	2	4	2	5
	5	1	5	3	5	3	3	5	3	5	3	6
Hook Function	0	0	0	0	0	−2	0	−2	0	−2	0	−2
	0	0	0	−1	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	1	0	1	0	1	0

Figure 10: Recursive applications of algorithm 1. Each tableau-hook pair is the result of algorithm 1 applied to the previous pair.

the bottom-most cell of the column to the left of (i, j). The concept of a successor map is important because algorithm 2 uses the output coordinates from 1 as the input coordinates for the next call to 1.

Algorithm 1 calls Algorithm P across all coordinates in the tableau in the order in which they appear in the successor map. If the call to algorithm P results in a modification to the tableau, the hook function is updated to record information the slide. In other words, the hook function serves as a manner in which to record how the tableau was changed.

Input: A triple $(A, f, (i_0, j_0))$ where A is a tableau, f is a hook function, and $(i_0, j_0) \neq (1, 1)$ is a cell in A.
Output: A triple $(B, g, s(i_0, j_0))$ where B is a tableau, g is a hook function, and $s(i_0, j_0)$ is the successor to (i_0, j_0).

begin
  $(i_1, j_1) \leftarrow s(i_0, j_0)$;
  $a \leftarrow A(i_1, j_1)$;
  $B \leftarrow AlgorithmP(A, a)$;
  $(i_2, j_2) \leftarrow$ the position of a in B;
  for $i_1 \leq i \leq i_2$ *and* $1 \leq j \leq |\lambda|$ **do**
    if $j = j_1$ **then**
      if $i = i_2$ **then**
        $g(i_2, j_1) = j_2 - j_1$
      else
        $g(i, j_1) = f(i + 1, j_1) + 1$
      end if
    else
      $g(i, j) = f(i, j)$
    end if
  end for
  return $(B, g, (i_1, j_1)$
end

4.1.3 Algorithm 2

Algorithm 2 is the "main" algorithm mapping elements of II to elements of I. Algorithms 1 and P actually do the work, Algorithm 2 just calls Algorithm 1 across the coordinates of the successor map. For example, the first tableau and the last tableau-hook pair in figure 4.1.2 are input and output values for algorithm 2.

Input: $A \in II$
Output: $(A, f) \in I$
begin
  $f = 0$;
  $(i_0, j_0) \leftarrow$ position of least cell;
  while $(i_0, j_0) \neq (1, 1)$ **do**
    $(A, f, (i_0, j_0)) \leftarrow \text{Algorithm1}(A, f, (i_0, j_0))$
  end while
  return $(A, f, (i_0, j_0))$
end

4.2 Mapping I to II

To map an element of I to an element of II, the authors establish another sequence of three algorithms: Algorithm P', Algorithm 1', and Algorithm 2'. These algorithms perform backwards Jeu de Taquin and use the directions encoded in the hook function to determine which elements to slide where.

4.2.1 Algorithm P'

Algorithm P' works analogously to algorithm P, but in reverse. It uses information encoded in the hook function provided as input to return output closer to the original element of II. In other words, Algorithm P' performs reverse Jeu de Taquin on P.

Input: a triple $(T, a', (i'_0, j'_0))$ where T is a tableau, $a' \in \{1, 2, \ldots, n\}$, and (i'_0, j'_0) is a cell in T.
Output: A tableau
begin
  $(i'_1, j'_1) \leftarrow$ the position of a' in T;
  if $(i'_1, j'_1) \geq (i'_0, j'_0)$ **then**
    return T
  else
    $b' \leftarrow T(i'_1, j'_1 - 1)$ if $j'_1 > j'_0$ else 0;
    $c' \leftarrow T(i'_1 - 1, j'_1)$ if $i'_1 > i'_0$ else 0;
    $U \leftarrow T$ with a' swapped with the greatest of b' and c'.;
    return *AlgorithmP'(U)*
  end if
end

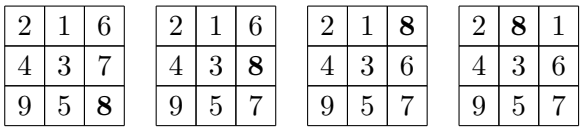

Figure 11: Algorithm P' applied recursively to cell value 8.

4.2.2 Algorithm 1'

To define algorithm 1', we will first need a manner in which to track the movement of a cell value across the tableau during the recursive P' calls. To this end, we will define the *direction* the cell value moves as N if it exchanges position with the value directly above it, W if it exchanges position with the value directly to the left of it, and $\emptyset$ if it does not move. We define the concatenation of all of these letters as the *path* of the cell, and it is read from **right to left**. For example, the directions of the movement of cell value 8 in figure 4.2.1 is NNW, so the value's path is WNN. These paths are ordered lexicographically such that $N < \emptyset < W$. The *maximum candidate element* is the cell appearing first in the lexicographical ordering of the paths of all *candidates elements* in the tableau. Finally, the set of *candidate elements* associated with the tableau-hook-coordinate triple $(U, g', (i'_1, j'_1)$ are all cells with (i, j) coordinates satisfying the conditions $i \geq i'_1$, $g'(i, j'_1) \geq 0$, and $j = j'_1 +$ $g'(i, j'_1)$.

Input: A triple $(T, g', (i'_0, j'_0))$ where T is a tableau ordered up to (i'_0, j'_0) , g' is a hook function, and (i_0, j_0) is a cell in T which is not the least in the successor map.

Output: A triple $(U, f', s^{-1}(i_0, j_0))$ where U is a tableau, f' is a hook function, and $s^{-1}(i_0, j_0)$ is the predecessor to (i_0, j_0).

begin

  $(i'_1, j'_1, p) \leftarrow$ maximum candidate element;

  $U \leftarrow$ AlgorithmP'(T, p, i'_0, j'_0);

  for $i'_0 < i \leq i'_1$ **do**

    $f(i, j'_0) = g'(i - 1, j'_0) + 1$;

    $f'(i'_0, j'_0) = 0$;

    $f'(i, j) = g(i, j)$ otherwise;

  end

  Return$(U, f', s^{-1}(i'_0, j'_0))$

end

4.2.3 Algorithm 2'

Algorithm 2' works across the successor map analagously to Algorithm 2, but in the reverse order. It calls Algorithm 1' at each location of the successor map and "rewinds" the hook function as it is interpreted.

Input: $(T, g') \in I$
Output: $T \in II$
begin
  $(i'_0, j'_0) \leftarrow (1, 1)$;
  while $(i'_0, j'_0) \neq$ *the least cell in* T **do**
    $(T, g', (i'_0, j'_0)) \leftarrow$ Algorithm1'$(T, g', (i'_0, j'_0))$;
  end
  Return$(T, g', (i'_0, j'_0))$
end

4.3 Remarks

To establish their bijection, the authors needed to prove that the two main algorithms (2 and 2') are inverses of each other. Because both of them are actually just Algorithm 1 and 1' calls with initial and halting conditions, the authors assert that proving 1 and 1' are inverses is sufficient. Although algorithms 1 and 1' both accept the same type of input, they do not function as inverses across all possible input values. The authors note this and proceed to define a "correct" set C of input values for which 1 and 1' are inverses.

$(A, f, (i_0, j_0)) \in C$:
  (i_0, j_0) are coordinates in λ
  A is ordered to(i_0, j_0)
  $f(i > i_0, j > j_0) \equiv 0$
  p moves to (i_0, j_0) after calling AlgorithmP' for any $p \in$ candidates.

They then show that for any element of C provided as input to algorithms 1 or 1', an element of C is produced as output. Therefore, algorithms 1 and 1' stabilize C and the bijection holds. It is interesting to note that often some elements of II will map to the same standard tableau. However, they

will not actually map to the same elements of I because the slides required to get the elements of I into standard form will not be the same and the hook function will reflect this.

5 A probabilistic proof

Greene, Nijenhuis, and Wilf[2] take a different approach to prove the formula's validity. By calculating all possible standard tableaux of a given shape probabilistically, they show that the formula must be valid because the probabilities of generating each standard filling sum to one.

The definition of a *corner cell* is central to this proof. A corner cell of a Ferrers diagram is a cell that is located both at the end of a row and the end of a column. The authors propose a manner in which to obtain random corner cells. First, randomly choose cell in (a, b) in λ. If the cell is not a corner cell, make another random selection from the hook of the cell and repeat the process until you reach a corner cell (α, β). We call this cell the *terminal* cell of the trial and we denote the probability of landing on (α, β) as $P(\alpha\beta)$. Figure 5 shows how we can create random standard fillings of λ by filling random corner cells of the empty portion of a tableau with n.

Corner cells are a useful tool because when we remove a corner cell from a Ferrers diagram, the remaining diagram is still a Ferrers diagram (albeit for $n-1$). The authors define a function which gives the number of standard tableaux for a shape λ with a corner cell removed from row α.

$$f_\alpha = f_{\lambda=(\lambda_1,\lambda_2,\ldots,\lambda_\alpha-1,\lambda_\alpha-1,\lambda_{\alpha+1},\ldots,\lambda_m)} \text{ if } \lambda \text{ is a partition else } 0 \qquad (1)$$

By showing that $F = \sum_\alpha f_\alpha$ throughout α, the authors prove the validity of the hook length theorem. In other words, by showing $1 = \sum_\alpha \frac{f_\alpha}{f_\lambda}$, the authors prove that the sum of the probabilities of generating all possible tableau via the following method is equal to one.

To acquire these corner cells randomly, consider the following trial procedure. Choose a cell (a_1, b_1) in the diagram with uniform probability $P(a, b) = \frac{1}{n}$. If this cell is not a corner cell, choose another cell (a_2, b_2) in the hook of (a_1, b_1). If this cell is not a corner cell, continue the process with (a_3, b_3), a cell in the hook of (a_2, b_2) etc. until a corner cell is reached. We call the initial cell (a_1, b_1) the *initial cell* of the trial and the corner cell

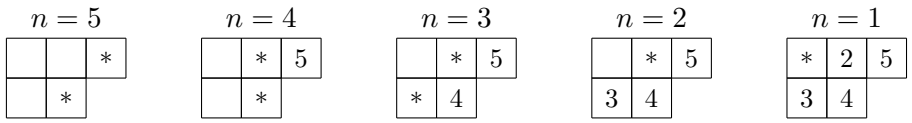

Figure 12: The construction of a standard tableau. Asterisks denote corner cells.

the *terminal cell* of the trial. We will denote the initial cells as (a, b) and terminal cells as (α, β).

The authors assert that $P(\alpha, \beta) = \frac{f_\alpha}{f_\lambda}$. Via an "easy calculation", we can see that the factorials in numerator of fraction cancel, yielding $\frac{1}{n}$ and quotient of the hook products of the original diagram and the trimmed diagram.

$$\begin{aligned} P(\alpha, \beta) = \frac{f_\alpha}{f_\lambda} =& \frac{1}{n} \prod_{1 \le i < \alpha} \frac{h_{i\beta}}{h_{i\beta} - 1} \prod_{1 \le j < \beta} \frac{h_{\alpha j}}{h_{\alpha j} - 1} \\ =& \frac{1}{n} \prod_{1 \le i < \alpha} (1 + \frac{1}{h_{i\beta} - 1}) \prod_{1 \le j < \beta} (1 + \frac{1}{h_{\alpha j} - 1}) \end{aligned} \tag{2}$$

Pressing forward, we will strive to give each term in this expansion a probabilistic interpretation. We will need to consider both the path $P : (a_1 b_1) \to (a_2, b_2) \to, \ldots, \to (\alpha, \beta)$ and its projections $A = (a_1, a_2, \ldots, \alpha)$ and $B = (b_1, b_2, \ldots, \beta)$. The authors show via induction the probability of taking any possible path given the initial cells of the trial.

$$P(A, B | a, b) = \prod_{i \in (A \setminus \alpha)} \frac{1}{h_{i\beta} - 1} \prod_{j \in (B \setminus \beta)} \frac{1}{h_{\alpha j} - 1} \tag{3}$$

Therefore, we can sum over all possible and initial cells and paths paths to calculate $P(\alpha, \beta) = \frac{1}{n} \sum P(A, B | a, b)$. By (2), this is the identical to 2. This establishes that that $\Sigma P(\alpha, \beta) = 1$ and the authors thusly conclude their proof.

6 Conclusion

Greene's probabilistic proof was completed soon after the discovery of the formula, but the direct bijective proof of Novelli, Pak and Stoyanovskii was

not obtained until many years later. We can see that proofs of this formula often require both ingenuity and creativity, which may explain why different authors have chosen different tools to utilize in their proofs. In combinatorics, counting partitions is often very difficult. Perhaps this is why proofs of the hook length formula have historically proven to be elusive.

Each new proof bring to light different mathematical features of the standard young tableaux. Efforts at acquiring elegant proofs of this and related formulas continue.

References

[1] JS Frame, G de B Robinson, and RM Thrall. The hook graphs of the symmetric group. *Canad. J. Math*, 6(316):C324, 1954.

[2] Curtis Greene, Albert Nijenhuis, and Herbert S Wilf. A probabilistic proof of a formula for the number of young tableaux of a given shape. *Advances in Mathematics*, 31(1):104–109, 1979.

[3] Jean-Christophe Novelli, Igor Pak, Alexander V Stoyanovskii, et al. A direct bijective proof of the hook-length formula. *Discrete Mathematics & Theoretical Computer Science*, 1(1):53–67, 1997.

[4] Bruce E Sagan. *The symmetric group: representations, combinatorial algorithms, and symmetric functions*, volume 203. Springer, 2001.