Building Mathematical Bridges Between Symmetric Functions

George H. Seelinger

Jefferson Scholars Foundation

ghs9ae@virginia.edu

28 November 2018

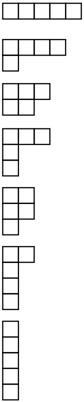
Partitions of 5

How many ways can we write a positive integer as a sum of positive integers?

Partitions of 5

How many ways can we write a positive integer as a sum of positive integers?

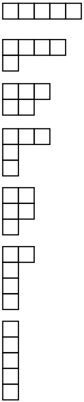
$$5 \to$$
$$4 + 1 \to$$
$$3 + 2 \to$$
$$3 + 1 + 1 \to$$
$$2 + 2 + 1 \to$$
$$2 + 1 + 1 + 1 \to$$
$$1 + 1 + 1 + 1 + 1 \to$$



Partitions of 5

How many ways can we write a positive integer as a sum of positive integers?

$$5 \to$$
$$4+1 \to$$
$$3+2 \to$$
$$3+1+1 \to$$
$$2+2+1 \to$$
$$2+1+1+1 \to$$
$$1+1+1+1+1 \to$$



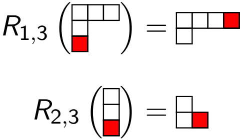
We will use these diagrams to describe a type of symmetric function called a "Schur function."

Raising Operators

To do this, we will need functions that change partition diagrams called "raising operators."

To do this, we will need functions that change partition diagrams called "raising operators."

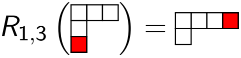
We can change partition diagrams by moving boxes.

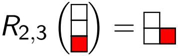


Raising Operators

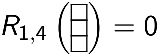
To do this, we will need functions that change partition diagrams called "raising operators."

We can change partition diagrams by moving boxes.





If the result "does not make sense", we get 0:



Schur functions

We define a new class of functions. Given a partition diagram λ with ℓ rows, we have definition

Schur functions

We define a new class of functions. Given a partition diagram λ with ℓ rows, we have definition

Definition

$$\begin{aligned} s_\lambda = & (1 - R_{1,2}) \\ & (1 - R_{1,3})(1 - R_{2,3}) \\ & \dots \\ & (1 - R_{1,\ell})(1 - R_{2,\ell}) \cdots (1 - R_{\ell-2,\ell})(1 - R_{\ell-1,\ell})\lambda \end{aligned}$$

Schur functions

We define a new class of functions. Given a partition diagram λ with ℓ rows, we have definition

Definition

$$
\begin{aligned}
s_\lambda = &(1 - R_{1,2}) \\
&(1 - R_{1,3})(1 - R_{2,3}) \\
&\cdots \\
&(1 - R_{1,\ell})(1 - R_{2,\ell}) \cdots (1 - R_{\ell-2,\ell})(1 - R_{\ell-1,\ell})\lambda
\end{aligned}
$$

Example

$$
s_{\ydiagram{3,2,1}} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\ydiagram{3,2,1}
$$

Example continued

Example

$$s_{\yng(3,2,1)} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\yng(3,2,1)$$

Example continued

Example

$$s_{\ydiagram{3,2,1}} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\ydiagram{3,2,1}$$

Recall the foil method from high school:

$$\begin{aligned}
&(1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3}) \\
=&(1 - R_{1,2} - R_{1,3} + R_{1,2}R_{1,3})(1 - R_{2,3}) \\
=&1 - R_{1,2} - R_{1,3} - R_{2,3} + R_{1,2}R_{1,3} + R_{1,2}R_{2,3} + R_{1,3}R_{2,3} - R_{1,2}R_{1,3}R_{2,3}
\end{aligned}$$

Example continued

Example

$$s_{\yng(3,2,1)} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\yng(3,2,1)$$

Recall the foil method from high school:

$$\begin{aligned}
&(1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3}) \\
=&(1 - R_{1,2} - R_{1,3} + R_{1,2}R_{1,3})(1 - R_{2,3}) \\
=&1 - R_{1,2} - R_{1,3} - R_{2,3} + R_{1,2}R_{1,3} + R_{1,2}R_{2,3} + R_{1,3}R_{2,3} - R_{1,2}R_{1,3}R_{2,3}
\end{aligned}$$

So, we must compute $s_{\yng(3,2,1)} =$

$$(1 - R_{1,2} - R_{1,3} - R_{2,3} + R_{1,2}R_{1,3} + R_{1,2}R_{2,3} + R_{1,3}R_{2,3} - R_{1,2}R_{1,3}R_{2,3})\yng(3,2,1)$$

Example continued

Example

$$s_{\ydiagram{3,2,1}} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\ydiagram{3,2,1}$$

Example

$$s_{\ydiagram{3,2,1}} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\ydiagram{3,2,1}$$

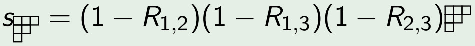
$$\ydiagram{3,2,1}$$

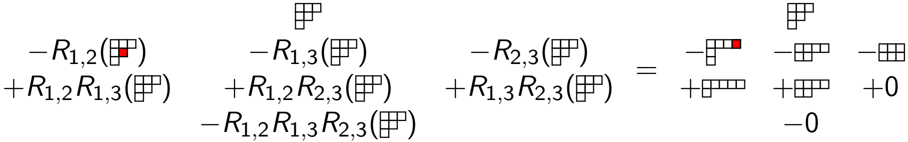
$$-R_{1,2}(\ydiagram{3,2,1}) \qquad -R_{1,3}(\ydiagram{3,2,1}) \qquad -R_{2,3}(\ydiagram{3,2,1})$$

$$+R_{1,2}R_{1,3}(\ydiagram{3,2,1}) \qquad +R_{1,2}R_{2,3}(\ydiagram{3,2,1}) \qquad +R_{1,3}R_{2,3}(\ydiagram{3,2,1})$$

$$-R_{1,2}R_{1,3}R_{2,3}(\ydiagram{3,2,1})$$

Example





Example continued

Example

$$s_{\ydiagram{3,2,1}} = (1 - R_{1,2})(1 - R_{1,3})(1 - R_{2,3})\ydiagram{3,2,1}$$

$$\begin{array}{cccccc}
 & \ydiagram{3,2,1} & & & \ydiagram{3,2,1} & \\
-R_{1,2}(\ydiagram{3,2,1}) & -R_{1,3}(\ydiagram{3,2,1}) & -R_{2,3}(\ydiagram{3,2,1}) & & -\ydiagram{4,1,1} & -\ydiagram{4,2} & -\ydiagram{3,3} \\
+R_{1,2}R_{1,3}(\ydiagram{3,2,1}) & +R_{1,2}R_{2,3}(\ydiagram{3,2,1}) & +R_{1,3}R_{2,3}(\ydiagram{3,2,1}) & = & +\ydiagram{5,1} & +\ydiagram{4,2} & +0 \\
 & -R_{1,2}R_{1,3}R_{2,3}(\ydiagram{3,2,1}) & & & & -0 &
\end{array}$$

Adding it all together, we get

Solution

$$s_{\ydiagram{3,2,1}} = \ydiagram{3,2,1} - \ydiagram{4,1,1} - \ydiagram{3,3} + \ydiagram{5,1}$$

Why Schur functions?

- Schur functions encode the possible ways certain abstract algebraic objects appear in n-dimensional space. (If $n = 3$, we have 3D space.)

Why Schur functions?

- Schur functions encode the possible ways certain abstract algebraic objects appear in n-dimensional space. (If $n = 3$, we have 3D space.)
- Schur functions make computer computations easier.

Why Schur functions?

- Schur functions encode the possible ways certain abstract algebraic objects appear in n-dimensional space. (If $n = 3$, we have 3D space.)
- Schur functions make computer computations easier.

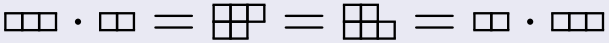
Problem

However, the formula for Schur functions is complicated. If we have another formula for Schur functions, how can we prove they give the same result?

Multiplication for Symmetric Functions

Let us introduce a rule for multiplication of partition diagrams by "stacking."

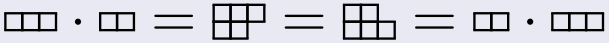
Rule for Multiplication (Example)



Multiplication for Symmetric Functions

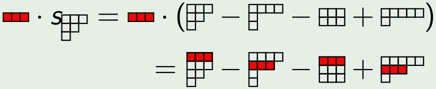
Let us introduce a rule for multiplication of partition diagrams by "stacking."

Rule for Multiplication (Example)



Schur functions are a sum of partition diagrams, so we can compute

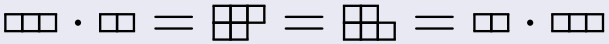
Example



Multiplication for Symmetric Functions

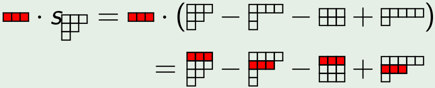
Let us introduce a rule for multiplication of partition diagrams by "stacking."

Rule for Multiplication (Example)



Schur functions are a sum of partition diagrams, so we can compute

Example



Problem

Result is in terms of partition diagrams, but we would like a result in terms of Schur functions.

Example

$$(4) \cdot s_{(2,1)} = s_{(4,2,1)} + s_{(5,1,1)} + s_{(5,2)} + s_{(6,1)}$$

Example

$$\ydiagram{4} \cdot s_{\ydiagram{2,1}} = s_{\ydiagram{4,2,1}} + s_{\ydiagram{5,1,1}} + s_{\ydiagram{5,2}} + s_{\ydiagram{6,1}}$$

- In general, we get the result in terms of Schur functions by finding all ways to add the red boxes such that we only add at most one box to each column.

The Pieri Rule

Example

- In general, we get the result in terms of Schur functions by finding all ways to add the red boxes such that we only add at most one box to each column.
- We call this method *the Pieri rule* and it is a fundamental property of Schur functions.

Proof Technique

One approach to show two formulas for Schur functions are the same:

Proof technique

Base cases are equal → Linear algebra ← Pieri rules are the same

Linear algebra

↓

Functions are the same!

What do I think about?

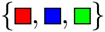
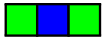
- Most problems about Schur functions are solved.

What do I think about?

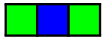
- Most problems about Schur functions are solved.
- Instead, I think about a class of functions called “type C dual affine Stanley symmetric functions” which have similar properties to Schur functions.

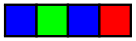
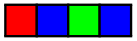
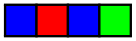
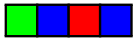
What do I think about?

- Most problems about Schur functions are solved.
- Instead, I think about a class of functions called "type C dual affine Stanley symmetric functions" which have similar properties to Schur functions.
- However, the current formula for these functions is not as concrete as the formula I gave you for Schur functions.

Start with "word" with letters given by colors, . For example, let's use $w =$ .

Type C dual affine Stanley symmetric functions

Start with "word" with letters given by colors, {, , }. For example, let's use $w =$ .

We must find all "subword decompositions" of w that are also subwords of $\rho =$  or any of its "rotations" , , .

Type C dual affine Stanley symmetric functions

Start with “word” with letters given by colors, {🟥, 🟦, 🟩}. For example, let’s use $w =$ 🟩🟦🟩.

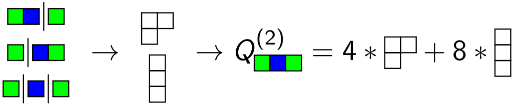
We must find all “subword decompositions” of w that are also subwords of $\rho =$ 🟦🟩🟦🟥 or any of its “rotations” 🟥🟦🟩🟦, 🟦🟥🟦🟩, 🟩🟦🟥🟦.

Example

🟩🟦|🟩 is a subword decomposition of w where each part appears as a subword of $\rho =$ 🟦🟩🟦🟥, but 🟩🟦🟩 is not a subword of ρ or any of its rotations.

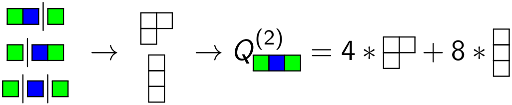
Example continued

Then, you take all such subword decompositions to get a formula



Example continued

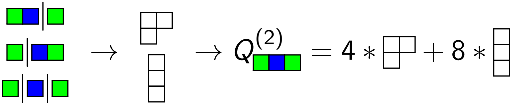
Then, you take all such subword decompositions to get a formula



But, unfortunately, you are not done!

Example continued

Then, you take all such subword decompositions to get a formula



But, unfortunately, you are not done!

Problem

You then have to take the "dual" of this function to get the Type C dual affine Stanley symmetric function, $P^{(2)}$. This process is not direct and not computationally straightforward.

What have I done?

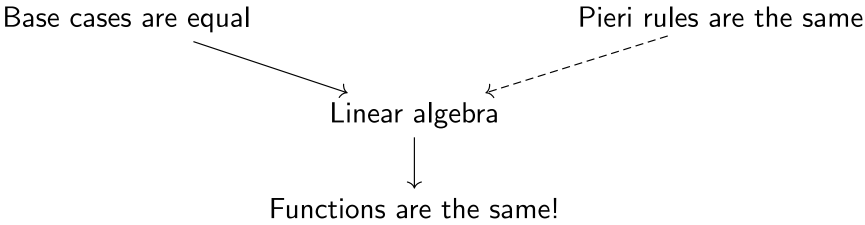
- I have a conjectured formula that describes type C dual affine Stanley symetric functions ($P_w^{(n)}$) directly using raising operators.

What have I done?

- I have a conjectured formula that describes type C dual affine Stanley symetric functions ($P_w^{(n)}$) directly using raising operators.
- Computational evidence suggests my conjecture is correct.

What have I done?

- I have a conjectured formula that describes type C dual affine Stanley symetric functions ($P_w^{(n)}$) directly using raising operators.
- Computational evidence suggests my conjecture is correct.
- However, proving the formulas are the same directly would be quite hard, so instead I am seeking to use the Pieri rule approach



Thank you for your support and for listening!

Jefferson Scholars Foundation

Symmetric Functions?

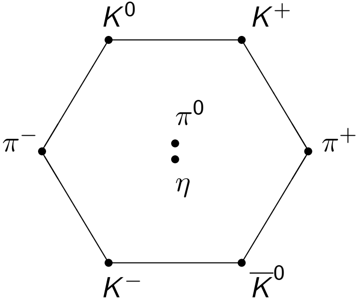
I pulled the wool over your eyes. Our partition diagrams represent polynomial functions with an infinite number of variables and an infinite number of terms.

Dictionary

$$
\begin{aligned}
\square \quad &\rightarrow h_{\square}(x_1, x_2, x_3, \ldots) = && x_1 + x_2 + x_3 + \cdots \\
\square\square \quad &\rightarrow h_{\square\square}(x_1, x_2, x_3, \ldots) = && x_1^2 + x_1x_2 + x_1x_3 + \cdots \\
& && +x_2^2 + x_2x_3 + \cdots \\
& && +x_3^2 + x_3x_4 + \cdots \\
\square\square\square \quad &\rightarrow h_{\square\square\square}(x_1, x_2, x_3, \ldots) = && x_1^3 + x_1^2x_2 + x_1^2x_3 + \cdots \\
&\vdots
\end{aligned}
$$

$$
s_{(3,2,1)} = h_{(3,2,1)} - h_{(4,1,1)} - h_{(3,3)} + h_{(5,1)}
$$

Applications?



The “eightfold way” from particle physics is encoded in Schur functions by

$$s_{\yng(2,1)}(e_{\epsilon_1}, e_{\epsilon_2}, e_{\epsilon_3})$$