CARD GAMES AND CRYSTALS

This is the extended version of a talk I gave at KIDDIE (graduate student colloquium) in April 2011. I wish I could give this version, but there wasn't enough time, so I left out section 1 and parts of section 5. This was written mainly to help myself plan the talk, so it might not work so well read on its own, I've tried to make it feel like a talk by including lots of pictures (so it's not printer-friendly, sorry). Please direct corrections and suggestions to amypang@stanford.edu .

This is based on a lecture by Daniel Bump in his Lie groups course, which tied together things I learnt from Ian Grojnowski, Akshay Venkatesh and Persi Diaconis. Thanks also to Peter and Sam for correcting my patchy knowledge of crystals; to Rick Sommer for lending me his Zome tools to make my 3D crystal; to Ralph, Cary and flatmates for lending me the cards; to Megan, Simon, Charlotte, Jon and Ilya for listening to my practice and giving good suggestions for improvements. If you want to know more about crystals, the textbook by Hong and Kang is apparently pretty good.

Abstract: I'm going to show you a stupid game and some stupid pictures, and then show you how they come together to decompose representations of GL_n.

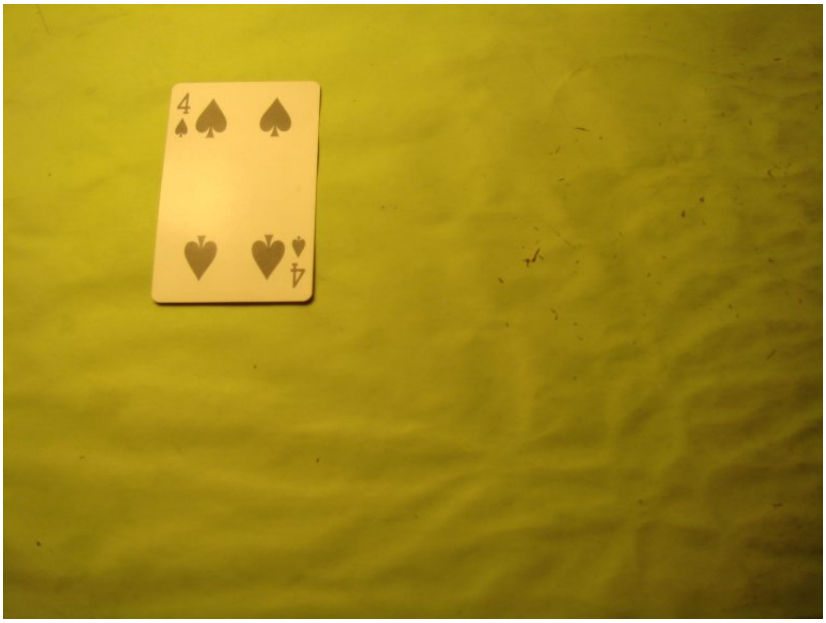
Prerequisites: It'll certainly be useful if you've met some representations of GL_n, although I'm only assuming knowledge of characters and tensor products of representations of finite groups. In particular, I'm doing this all without Lie algebra theory (though I will mention them sometimes to help people who already know about them).

1. RSK Algorithm

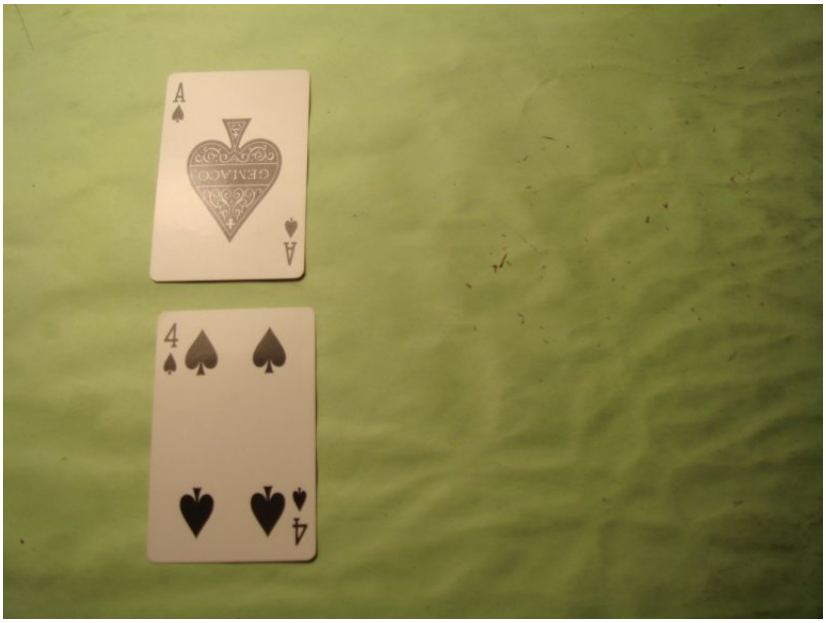
Here is a rather boring card game you can play with a deck of k cards whose face values are $\{1, 2, ...n\}$. (Each face value can appear many times, or not at all.) It's also rather complicated to explain, so I'll do an example in a bit. Start by drawing a card from the deck, and put it in the first row. Draw another card; if its value is higher than or equal to the first card, put it at the end of the first row (ie on the right of the first card you put down). Otherwise, replace the first card with the second card, and put the displaced card in the second row. Now draw the third card. If it's higher than all cards currently in the first row, put it at the end of the first row; otherwise, it replaces the leftmost card on the first

row which is higher than it. In the second scenario, we take the displaced card and it replaces the leftmost card on the second row which is higher than it, or goes at the end of the second row if it's higher than all cards there. If a card was displaced at this stage, we look to the third row... until no card is displaced (because I finally put a card at the end of a row). Then we draw the fourth card and repeat the process, displacing from the first row then the second row...

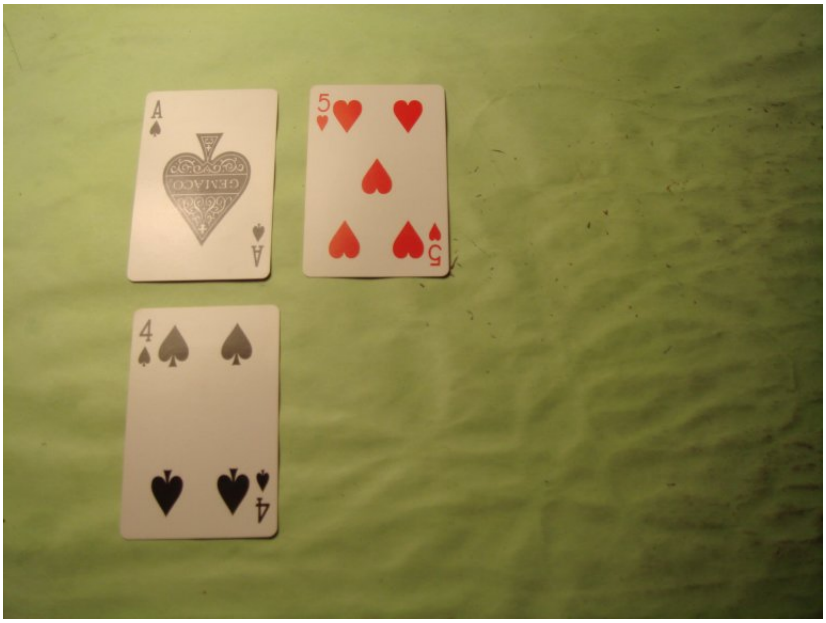
I'll demonstrate with $k = 7$ and $n = 5$ (ignore the suits). The first card I turn over is a 4, which I put in the first row.



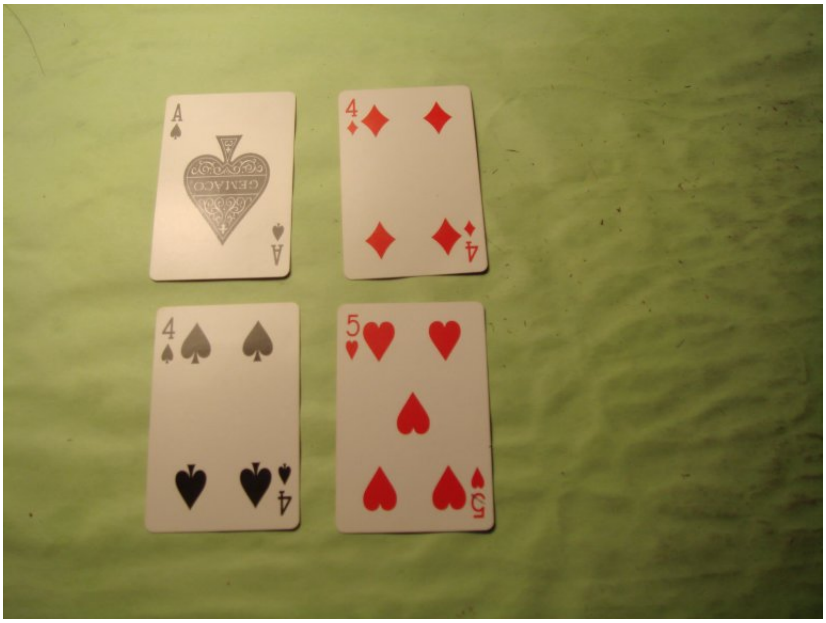
The next card I turn over is a 1, which is lower than 4, so it replaces the 4. There's nothing on the second row so the 4 just goes at the start of the second row (you can think of this as the end of an empty row).



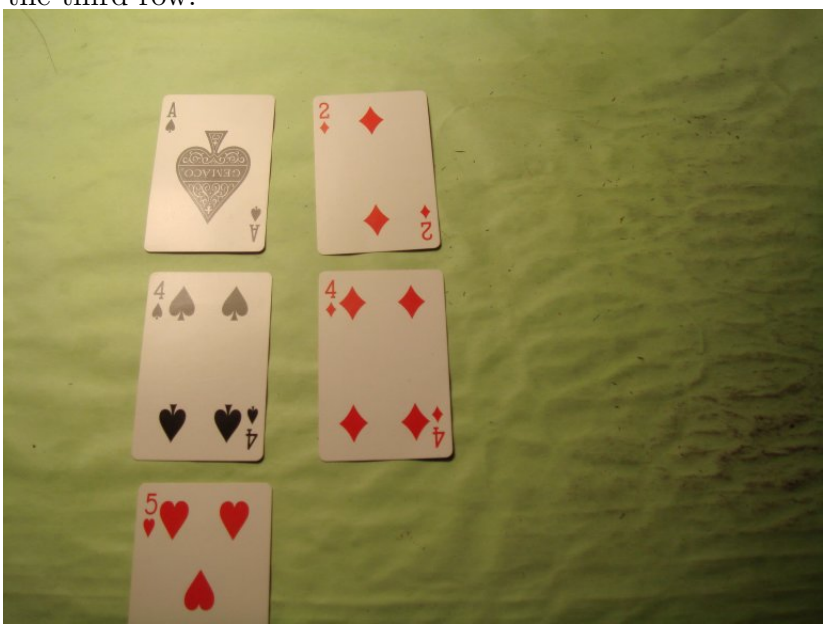
The third card I draw is a 5. It's higher than all the cards in the first row (ie the 1), so it goes on the end of the first row.



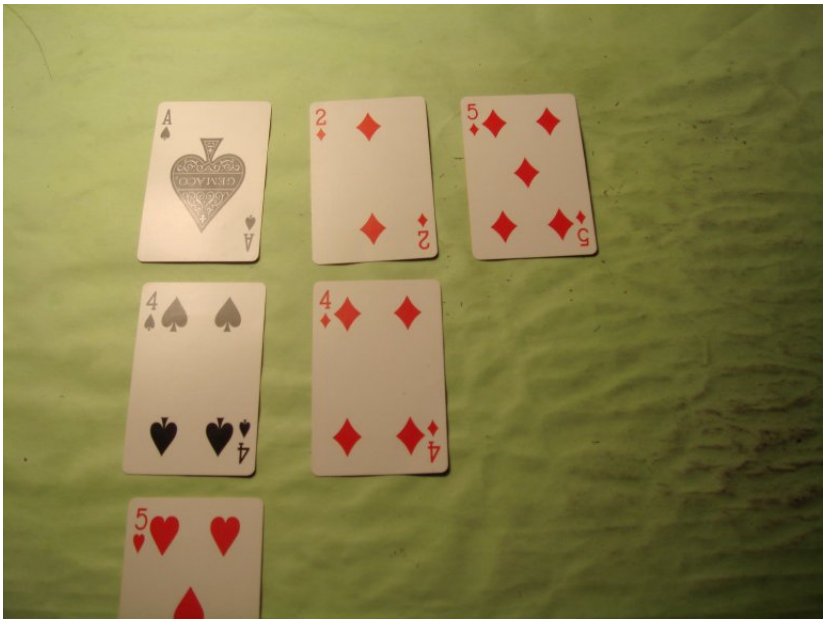
Next, I draw a 4. $4 > 1$ so the 4 can't replace the 1. But $4 < 5$ so the 4 displaces the 5. The 5 is higher than everything in the second row, so it goes at the end of the second row.



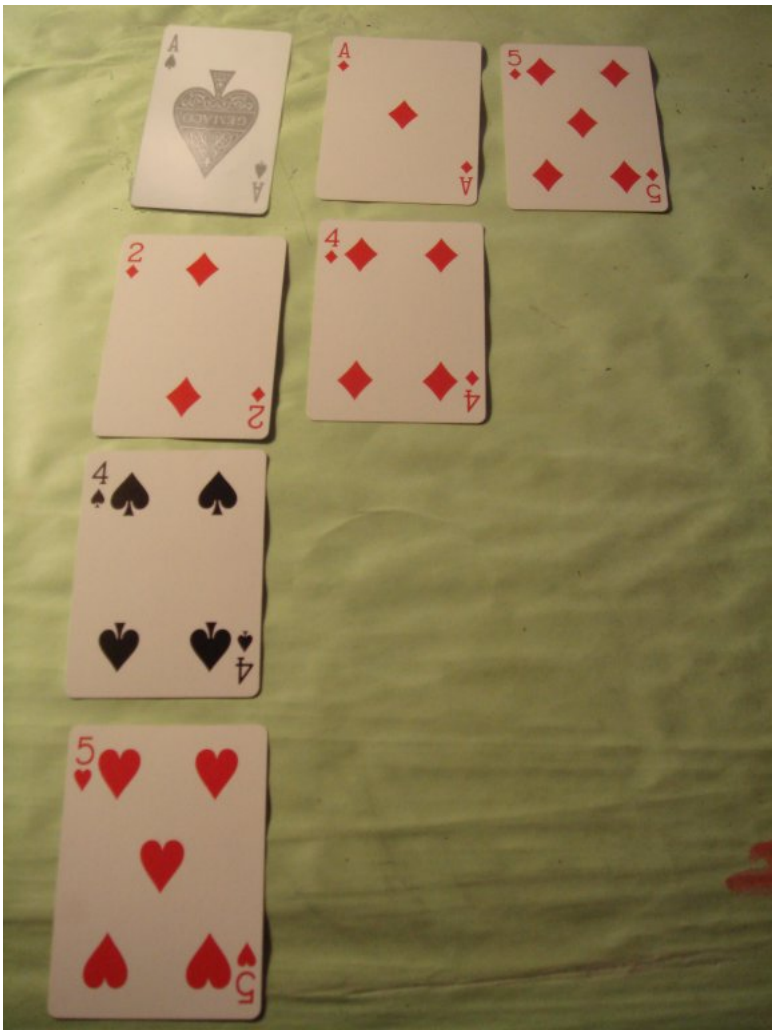
Next, I draw a 2. Again, we check the cards on the first row one by one from the left: the 2 can't replace the 1, because $2 > 1$, but it can replace the 4. Similarly, the displaced (red) 4 can't replace the (black) four on the second row, but it can replace the 5. There's nothing on the third row so the 5 goes in the first column of the third row.



The next card in the deck is a 5. $5 > 1$ and $5 > 2$, so it can't replace anything on the first row, so it goes on the end of the first row.



Finally, I draw a 1. It can't replace the existing 1, only a card of strictly higher value, so it replaces the 2 in the first row. The 2 then displaces the leftmost 4 in the second row, which displaces the 5 below it. Note that this affected cards in two columns; it's not generally true that cards just move down one column.



So I end up with a "left justified" grid of k squares, filled with the face values of the cards in my deck. Observe that each row is weakly increasing and each column is strictly increasing; such an arrangment is called a semi-standard tableaux, and this game always produces one. The shape of the tableaux is the number of squares in the rows, which is a partition of k. For example, in the above play we ended up with a tableaux of shape $(3, 2, 1, 1)$.

We can make another tableaux to keep track of how the semi-standard tableaux was built during the game. This bookkeeping tableaux has the same shape as the tableaux that the game produced, and it's filled with $\{1, 2, ..., k\}$ such that the positions filled by $\{1, 2, ...i\}$ indicate the shape of the game tableaux after i cards have been drawn. In other words, as you play the game, enter i in the position of the last card you moved before drawing the $i + 1$th card. (In contrast to the game tableaux, the numbers in the bookkeeping tableaux do not move once we

have entered them.) So, for the game I played above, the bookkeeping tableaux is

$$\begin{array}{ccc} 1 & 3 & 6 \\ 2 & 4 & \\ 5 & & \\ 7 & & \end{array}$$

which you'll notice is also semi-standard. A semi-standard tableaux filled with $\{1, 2, ..., k\}$ such that each entry occurs only once is called a standard tableaux.

The game tableaux and bookkeeping tableaux together give us enough information to reconstruct the game (ie we can play it backwards, removing the cards one by one) and deduce the order of the cards in the deck we started with. In fact, we can play this backwards-game with any semi-standard tableaux and standard tableaux of the same shape, and we get out a string whose numbers are the entries in the semi-standard tableaux. We'll write this string so that the rightmost number is the top of the deck, so the above example is 1524541. In other words, there is a bijection between strings of length k and pairs of semi-standard tableaux and standard tableaux of same shape a partition of k, and under this bijection, the semi-standard tableaux and the string has the same entries.

Now for something completely different.

2. Crystals

A crystal is a combinatorial gadget that we get by "freezing" a representation of a Lie group. (Kashiwara created them from quantum groups, via deforming the associated universal enveloping algebra then setting the parameter $q = 0$ - I don't know how this works, I can only "use" the crystals without proof.) It doesn't matter if you don't know what a Lie group is; for the whole talk, I'll work with $GL_n(\mathbb{C})$, the n-by-n invertible matrices with complex entries, for which this theory is best-understood. And all my representations will be holomorphic and over the complex numbers.

A (finite, seminormal) GL_n crystal is a finite graph with directed edges in $n-1$ colours, where each vertex b is assigned a weight $wt(b) \in \mathbb{Z}^n$. There are three additional conditions:

(1) At most one edge of each colour can leave a vertex, and at most one edge of each colour can enter a vertex.

(2) Going along an edge of colour i reduces the weight by $\epsilon_i - \epsilon_{i+1}$ (so there can't be cycles, of any combination of colours). (ϵ_i denotes the ith basis vector of $\mathbb{Z}^n$.)
(3) For any vertex b with weight $(\lambda_1, ..., \lambda_n)$, the length of the i-coloured path from b minus the length of the i-coloured path to b is $\lambda_i - \lambda_{i+1}$.

You can check that all these hold in the crystal of the standard (3-dimensional) representation of GL_3, which looks like:

and $wt(b_i) = \epsilon_i$. (red is colour 1, blue is colour 2)

Why is this associated with the standard representation? Here's how to read off the representation-theoretic information conveyed in the crystal: each vertex b in the crystal corresponds to a basis vector v_b of the representation, and the action of $\begin{pmatrix} t_1 & & \\ & \ddots & \\ & & t_n \end{pmatrix}$ on v_b is multiplication by $t_1^{\lambda_1}...t_n^{\lambda_n}$. In particular, the number of vertices is the dimension of the representation. For example, the above example is three dimensional, and $v_{b_i} = \epsilon_i$ correspond to the ith standard basis vector. (It's not usually true that $v_b = wt(b)$, since they're usually in vector spaces of different dimensions.) The character of $\begin{pmatrix} t_1 & & \\ & \ddots & \\ & & t_n \end{pmatrix}$ in this representation is $\sum t_1^{\lambda_1}...t_n^{\lambda_n}$, summing over all weights λ in the crystal. In the example above, it's $t_1 + t_2 + t_3$. This in fact gives the characters of all elements of GL_n, because almost everything in GL_n is diagonalisable and the character is continuous. As with finite groups, the character determines a GL_n representation uniquely (proof is to bootstrap either from the Lie algebra $\mathfrak{sl}_n$ or from the compact group U_n), so "freezing" a representation is injective.

The i-coloured edges represent what happens if you apply the Kashiwara map $\tilde{f}_i$ to these basis vectors; this is like f_i in the Lie algebra $\mathfrak{sl}_n$, but tweaked so a basis vector is sent to a multiple of another of these basis vectors (as opposed to

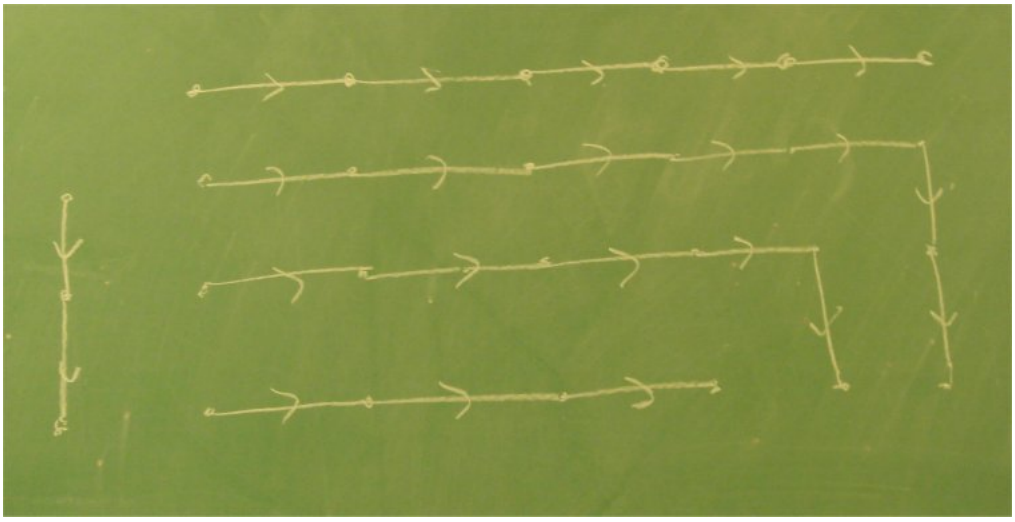
a linear combination). If there's no i-coloured edge coming from a vertex, $\tilde{f}_i$ of that basis vector is zero. There is a complicated formula for $\tilde{f}_i$, but roughly, it describes how the GL_2 formed by the ith and $i+1$th rows and columns of the matrix acts (hence $n-1$ colours). The upshot of this is that we can find v_b by applying $\tilde{f}_i$s repeatedly to the "source", which is b_1 in the example above.

What does it mean if there are no paths from vertex b to vertex b'? An intuitive answer is that GL_n-action cannot send v_b to $v_{b'}$ (because we can write any element of GL_n as a product of matrices in those GL_2s). Hence the connected components of the crystal correspond to the irreducible representations.

You should convince yourself that the trivial representation is represented by a single vertex of weight $(0,0,0)$, and the determinant representation is represented by a single vertex of weight $(1,1,1)$.

3. Tensoring Crystals

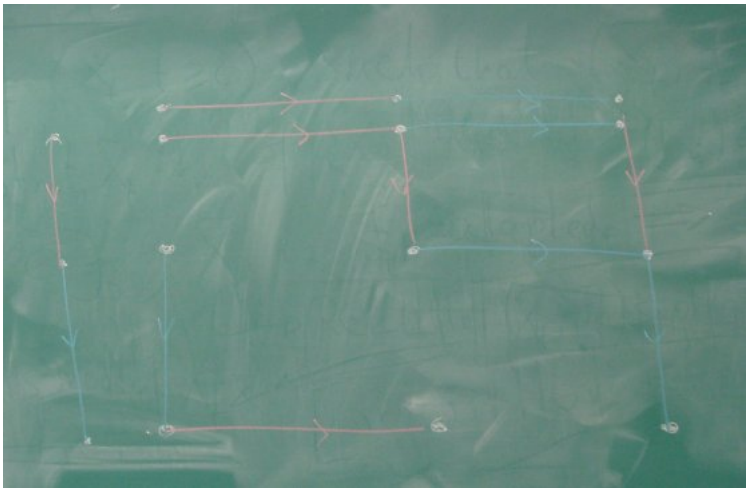
One can tensor two crystals B and C, and that'll correspond to tensoring the two representations. The vertex set of $B \otimes C$ is $b \otimes c$ where b is a vertex of B and c a vertex of C, and $w(b \otimes c) = w(b) + w(c)$. The edges are a little complicated and work like this: for any two strands of colour i in B and in C respectively (including strands of length 0, ie vertices without i-coloured edges going into or out of them), draw the following diagram on their product:



(the top and left lines are not part of the tensor product, they are the strands in B and C and are there to help me draw the product. In this example the strands have length 5 and 2, but you get the idea). This is called the Clebsch-Gordon

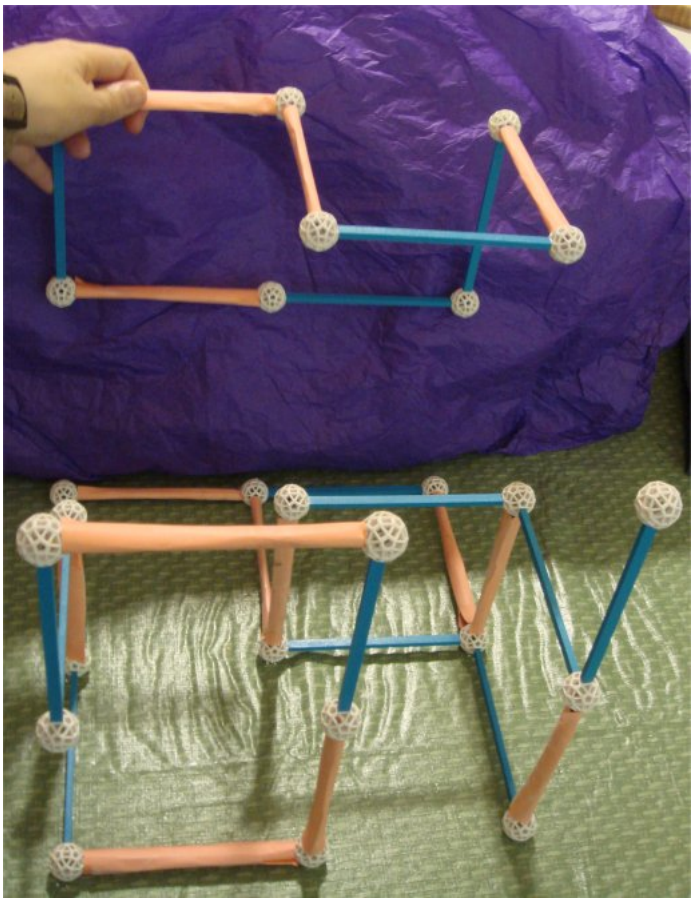
rule, and it comes from the decomposition into irreducibles of a tensor product of GL_2 representations.

If we want to tensor the standard representation crystal of GL_3 with itself, we get:



(Again, the lines on the top and left are not in the product, they're just for guidance.) You see there are two connected components, which means this representation is made up of two irreducible representations - indeed these are the symmetric and exterior square. Observe that $b_1 \otimes b_2$ is in the symmetric square crystal, but $(1,0,0) \otimes (0,1,0)$ isn't in the symmetric square representation: the tensor of two vertices doesn't match up to the tensor of those basis vectors. (To find out which vector $b_1 \otimes b_2$ does represent, we need to apply $\tilde{f}_1$ to $(1,0,0) \otimes (1,0,0)$. We know $b_1 \otimes b_1$ must match $(1,0,0) \otimes (1,0,0)$ since it has weight $(2,0,0)$ and the only line on which $\begin{pmatrix} t_1 & & \\ & t_2 & \\ & & t_3 \end{pmatrix}$ act as multiplication by t_1^2 is $(1,0,0) \otimes (1,0,0)$.)

We can tensor this with the standard representation again; then red arrows should come out of all three dots in the middle row, and the bottom left and bottom right dots. The triple tensor product looks like:



where the bottom layer is what was drawn on the previous board. There is a missing vertex in front of my hand, on the top layer, it isn't connected to anything.

Although it's unrelated to our talk, you might ask how to dualise a crystal. A first guess might be to reverse all the arrows, but then you realise that the weights don't change correctly along arrows. To fix this, put a minus sign in front of every weight.

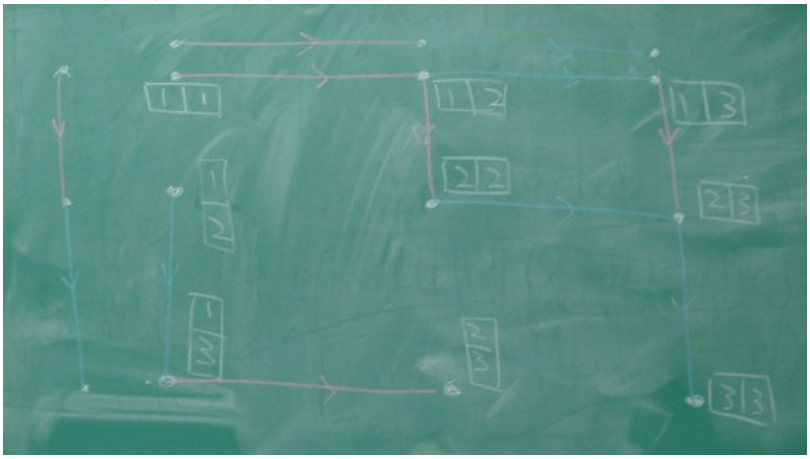
4. Crystals of Tableaux

Big crystals get very hard to draw. So often what we end up doing is indexing the vertex set by some combinatorial object, and describing where the arrows go.

Fix a shape μ (with k boxes). The semi-standard Young tableaux of shape μ and entries in $\{1, 2, ...n\}$ are vertices of a GL_n crystal: the weight of a tableaux is (number of 1s, number of 2s, ...), and $\tilde{f}_i$ either sends the tableaux to 0, or changes one of the is into an $i+1$ - which i gets changed depend on a rather complicated

algorithm, that I won't have time to describe. The confusing thing is that, sometimes even a tableaux containing plenty of is would be sent to zero. (Regardless, the weight changes in the correct way). It turns out the crystal of shape μ is connected; the "source" tableau has 1s in the first row, 2s in the second row etc, and we can obtain any semi-standard tableau of shape μ from this one by repeatedly changing is into $i+1$s in a way allowed by the $\tilde{f}_i$s. We'll call these the μ-irreps of GL_n.

Actually, we've already met two crystals of tableaux: the symmetric square and exterior square of the standard representation happen to be crystals of tableaux, of shape $(1,1)$ and (2):

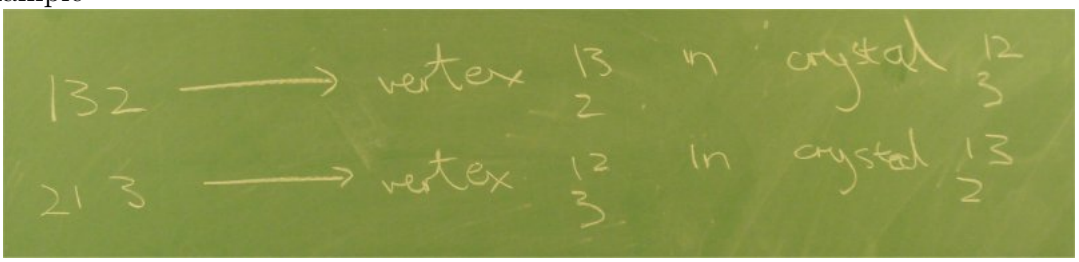


5. Schur Weyl Duality

So you might guess that the k-fold tensor product of the standard representation of GL_n is the direct sum of the μ irreps for all partitions μ of k. But if you try this for $k = n = 3$, you'll see that the dimension of these don't add up to 27. Plus there are only 3 partitions of 3, but my model had 4 connected components. The issue is that these irreps occur with multiplicities - you might've noticed that, in my model, the piece I'm holding and the piece on the bottom left are isomorphic as graphs. In fact, they're isomorphic as crystals (which I'll explain later), so they correspond to the same irrep. To find these multiplicities, we use our card game.

The vertices of the k-fold tensor product of the standard GL_n crystal have the form $b_{i_1} \otimes b_{i_2} \otimes ... \otimes b_{i_k}$, where $i_1, i_2, ...i_k \in \{1, 2, ...n\}$. Let ϕ send such a vertex to the two tableaux given by playing RSK with $i_1 i_2 ... i_k$ (remember, i_k is the top of the deck). Look at the semi-standard tableau as a vertex in a crystal of tableau,

and the standard (bookkeeping) tableau as labelling which crystal it's in. For example



This turns out to be an injective morphism of crystals (which means $w(\phi(b)) = w(b)$ and $\phi\left(\tilde{f}_i b\right) = \tilde{f}_i(\phi(b))$, ie ϕ preserves weights and edges), and its image is the direct sum, over all μ that are partitions of k, of N_μ copies of each μ-irrep, where N_μ is the number of standard tableau of shape μ. (The proof is via Pieri's rule.) So the irreducible constituents of the k-fold tensor product of the standard representation of GL_n are the μ-irreps where μ is a partition of k, and the multiplicity of the μ-irrep is N_μ.

(It is in fact possible to prove a stronger statement of Schur-Weyl duality using only elementary representation theory, without crystals or RSK; see the end of my online scanned notes on representation of GL_n.)

Amy Pang, 2011