Some positivity results for the Hecke algebra in type H_4

Fokko du Cloux

June 22, 2005

Institut Camille Jordan[1]
UMR 5208 CNRS
Université Lyon-I
69622 Villeurbanne Cedex FRANCE
`ducloux@math.univ-lyon1.fr`

Abstract. This paper is a report on a computer check of some positivity properties of the Hecke algebra in type H_4, including the positivity of the structure constants in the Kazhdan-Lusztig basis, thus answering a long-standing question of Lusztig's.

1 Statement of the problem

1.1. Let W be a Coxeter group, S its set of distinguished generators, and denote $\leq$ the Bruhat ordering on W. Denote $\mathcal{H}$ the Hecke algebra of W over the ring of Laurent polynomials $A = \mathbf{Z}[v, v^{-1}]$, where v is an indeterminate. We refer to [3] for basic results about Coxeter groups and Hecke algebras; we just recall here that $\mathcal{H}$ is a free A-module with basis $(t_y)_{y \in W}$, where the algebra structure is the unique one which satisfies

$$t_s.t_y = \begin{cases} t_{sy} & \text{if } sy > y \\ (v - v^{-1})t_y + t_{sy} & \text{if } sy < y \end{cases}$$

(here we use $t_y = v^{-l(y)}T_y$, where the T_y satisfy the more familiar relation $T_s.T_y = (q-1)T_y + qT_{sy}$ when $sy < y$, with $q = v^2$.) Then there is a unique ring involution i on $\mathcal{H}$ such that $i(v) = v^{-1}$, and $i(t_y) = t_{y^{-1}}^{-1}$.

[1]The author gratefully acknowledges the hospitality and technical assistance of the Centre de Calcul Medicis, Laboratoire STIX, FRE 2341 CNRS, and of Maply, UMR 5528 CNRS, where the computations for this paper have been carried out.

The Kazhdan-Lusztig basis of $\mathcal{H}$ is the unique family $(c_y)_{y\in W}$ such that *(a)* $i(c_y) = c_y$ and *(b)* $c_y = t_y + \sum_{x<y} p_{x,y} t_x$, with $p_{x,y} \in v^{-1}\mathbf{Z}[v^{-1}]$; in particular we find that $c_s = t_s + v^{-1}$ for all $s \in S$. It turns out that $P_{x,y} = v^{l(y)-l(x)} p_{x,y}$ is a polynomial in q, the *Kazhdan-Lusztig polynomial* for the pair x, y. For any pair (x, y) of elements in W, write

$$c_x.c_y = \sum_{z\in W} h_{x,y,z} c_z$$

(in other words, the $h_{x,y,z} \in A$ are the structure constants of the Hecke algebra in the Kazhdan-Lusztig basis).

A number of the deeper results in the theory of Hecke algebras depend on positivity properties of the polynomials $P_{x,y}$ and $h_{x,y,z}$. More precisely, consider the following properties :

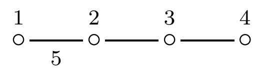
P_1: the polynomials $P_{x,y}$ have non-negative coefficents;

P_2: the $P_{x,y}$ are decreasing for fixed y, in the sense that if $x \leq z \leq y$ in W, the polynomial $P_{x,y} - P_{z,y}$ has non-negative coefficents;

P_3: the polynomials $h_{x,y,z}$ have non-negative coefficients.

Properties P_1 and P_3 are basic tools in the study of Kazhdan-Lusztig cells and the asymptotic Hecke algebra; they have been proved in [8] for crystallographic W using deep properties of intersection cohomology. Property P_2 is proved for finite Weyl groups in [4], using the description of Kazhdan-Lusztig polynomials in terms of filtrations of Verma modules.

None of these geometric or representation-theoretic tools are available for non-crystallographic Coxeter groups, and the validity in general of the positivity properties above are among the main open problems in the theory of Coxeter groups. Let us concentrate on the case where W is finite. It is easy to see that for the validity of P_1–P_3 we may reduce to the case where W is irreducible. Leaving aside the easy case of dihedral groups, this leaves us only with the two groups H_3 and H_4, and since the former is contained in the latter, the only case we need to consider is the Coxeter group of type H_4, of order 14400, with Coxeter diagram



Despite its rather modest size, the group H_4 poses a redoubtable computational challenge, even for present-day computers. It is all the more remarkable that property P_1 was checked already in 1987 by Dean Alvis [1] by explicitly computing all the Kazhdan-Lusztig polynomials. Quite a feat with the hardware of the time! Unfortunately, Alvis' programs have never been made available; to my knowledge, the only publicly available computer program capable of carrying out this computation is my own program `Coxeter` [2], which does it in about one minute on a modern-day personal computer. This still leaves open properties P_2 and P_3; the purpose of this paper is to report on a computation, carried out as one of the first applications of version 3 of `Coxeter`, by which we prove :

1.2 Theorem. — *Properties P_2 and P_3 hold for the Coxeter group of type H_4.*

1.3 Corollary. — (see [7] Theorem 5.4.) *The a-function on the Coxeter group of type H_4 is increasing w.r.t. the $\leq_{\mathrm{LR}}$ preorder; in particular it is constant on two-sided cells.*

2 Methodology

2.1. The verification of P_1 and P_2 is straightforward : one simply runs through the computation of the $P_{x,y}$ for all $x \leq y$ in the group. In fact, from the well-known property $P_{x,y} = P_{sx,y}$ whenever $sx > x$, $sy < y$ (cf. [5] (2.3.g)), and the analogous property on the right, for P_1 it suffices to consider the cases where $\mathrm{LR}(x) \supset \mathrm{LR}(y)$, where we denote $L(x) = \{s \in S \mid sx < x\}$ (resp. $R(x) = \{s \in S \mid xs < x\}$) the left (resp. right) descent set of x, and $LR(x) = L(x) \coprod R(x) \subset S \coprod S$; we call such pairs (x, y) *extremal pairs.* Taking also into account the symmetry $P_{x,y} = P_{x^{-1},y^{-1}}$, there are 2 348 942 cases to consider, which are easily tabulated by the program. The non-negativity of the polynomials is checked as they are found. For P_2, one also easily reduces to extremal pairs (x, y) and (z, y). The tough computation is for P_3; here there are *a priori* $14\,400^3 = 2\,985\,984\,000\,000$ (almost three trillion!) polynomials $h_{x,y,z}$ to be computed, and the only obvious symmetry is $h_{x,y,z} = h_{y^{-1},x^{-1},z^{-1}}$ (but, as explained below, we don't even use that.)

2.2. The algorithm used in the computation is straightforward. For a fixed y, we compute the various $c_x.c_y$ by induction on the length of x, starting with $c_e.c_y = c_y$, where e denotes the identity element of W. To carry out the induction, we choose any generator $s \in S$ such that $sx < x$, and write :

$$c_x = c_s.c_{sx} - \sum_{\substack{z<sx \\ sz<z}} \mu(z, sx)c_z \tag{1}$$

where as usual $\mu(x, y)$ denotes the coefficient of v^{-1} in $p_{x,y}$ (which is also the coefficient of degree $\frac{1}{2}(l(y) - l(x) - 1)$ in $P_{x,y}$, and in particular is zero when the length difference of x and y is even.)

Then we may assume that $c_{sx}.c_y$ is already known, and similarly for the various $c_z.c_y$, so we are reduced to multiplications of the form $c_s.c_u$, for $s \in S$ and $u \in W$. When $su > u$ this is read off from formula (1) with $x = su$; and when $su < u$ one simply has $c_s.c_u = (v + v^{-1})c_u$ (see for instance [5]).

The information which is required for this computation is encoded in the W-graph of the group; once this graph is known, everything else just involves elementary operations on polynomials. Actually, it is obvious that the $h_{x,y,z}$ are in fact polynomials in $(v+v^{-1})$, so they are determined by their positive-degree part; it is this part that we compute and keep in memory. The only complication arises from the fact that we need to be careful about memory overflow; it turns out in fact that for H_4 all the coefficients of the $h_{x,y,z}$ fit into a 32-bit unsigned (and even signed) integer, but not by all that much: the largest coefficient which occurs is 710 904 968, which is only about six times smaller than $2^{32} = 4\,294\,967\,296$.

2.3. From the computational standpoint, this procedure has a number of desirable features. First and foremost, once the W-graph of the group has been determined, the problem splits up into 14400 independent computations, one for each y, so we can forget about the computation for a given y when passing to the next (this would not be true if we tried to use the symmetry $h_{x,y,z} = h_{y^{-1},x^{-1},z^{-1}}$.) This is advantageous in terms of memory usage, could be used to parallelize the computation if necessary (it turns out that it hasn't been), and also means that the computation can be harmlessly interrupted (either voluntarily or involuntarily), at least if its progress is recorded somewhere : basically, the only penalty to pay for picking up an interrupted computation is the recomputation of the W-graph, which takes about half a minute. This is very valuable for computations running over several days, where there is always the risk of a system crash or power failure.

On the other hand, it is essential that for a fixed y, the full table of $h_{x,y,z}$ be stored in memory. In practice, there are many repetitions among these polynomials; so we store them in the form of a table of $14400^2 = 207\,360\,000$ pointers. Initially, this is the main memory requirement of the program; it is interesting to note that the cost doubles, from about 800 MB to about 1.6 GB, when we pass from a 32-bit to a 64-bit computer. It turns out that for a fixed y, the additional memory required to store the actual polynomials is small, and never exceeds 300 MB. So the full computation runs confortably in 2 GB of memory, and barely exceeds 1 GB on a 32-bit machine.

It is much more difficult to try to write down a full table of all the polynomials that occur as $h_{x,y,z}$. I have done this a number of times, but with the memory available on the machines to which I have had access, it has been necessary to split up the computation in about a hundred pieces to avoid memory overflow in the polynomial store, to keep the corresponding files in compressed form to avoid overflowing the hard disk, and then to merge those one hundred compressed files into a single compressed file. At some point I have needed to store about 30 GB of compressed files—not something administrators are very happy about! In view of these difficulties, I have chosen not to make that version of the program available for the time being.

2.4. The computation has been done a number of times (writing files of all the distinct polynomials): at the Ecole Polytechnique, Centre de Calcul Médicis, Laboratoire STIX, FRE 2341 CNRS, on several computers, including a Compaq Alpha EV68 and an AMD Opteron server with 4 and 8 GB of main memory respectively, where it has taken about 5 days of CPU time, and twice at the Université Lyon I, Maply, UMR 5585 CNRS, on a Xeon processor with 4 GB of main memory, where it took 80 hours (not counting the final file merging pass, which took another 80 hours or so.) The program as presented here has run on our 2.7 GHz AMD Opteron server at the Institut Camille Jordan, using less than 2 GB of memory, in a little under 85 hours.

On the technological side, it seems that the time was just right for this computation : it makes full use of the 3Ghz processors, at least 2GB central memories, and 100+ GB hard disks that are found on typical low-end servers today. It would still be beyond the grasp of most present-day personal computers, however, although that, too, is changing fast!

3 Verification

3.1. Let's come now to the thorny issue of verification : what is the amount of trust that can be put in a result like this ? An obvious prerequisite is the availablility of the source code of the program that carries out the computation; this may be downloaded at `http://igd.univ-lyon1.fr/~ducloux/coxeter/coxeter3/positivity`

This is actually the source code of an especially modified version of `Coxeter3`. All the extra code is contained in the file `special.cpp`; all the other files are identical to the ones in `Coxeter3`.

3.2. In addition to those already available in `Coxeter3`, the following commands are defined :

- `klplist` : prints out a list of all the distinct Kazhdan-Lusztig polynomials which occur in the group (so that in particular, one may re-check property P_1, although this was done already by the computer in the course of the computation);
- `decrklpol` : checks property P_2;
- `positivity` : checks the non-negativity of the $h_{x,y,z}$;
- `cycltable` : prints out the table of the $c_x.c_y$ for a fixed y;
- `cprod` : prints out a single product $c_x.c_y$;

For type H_4, on a decent server, the `cycltable` command should not take more than two minutes; `cprod` should usually take less than a minute (Note that the first call will take longer than subsequent ones, because the W-graph must be computed the first time.) So these commands give local access to the multiplication table of the Hecke algebra, thus opening up the "black box" a little.

The `positivity` command for type H_4 can also be executed through the little stand-alone command `coxbatch` that I have included; this will run the computation in the background. It writes any errors in `error_log`, and records the progress of the computation in `positivity_log`. After a successful run, `error_log` should be empty, and `positivity_log` should end with the line :

```
14399:  maxcoeff = 710904968
```

(elements of the group are numbered from 0 to 14399.)

3.3. As was explained in 2.2, the computation is entirely elementary once the W-graph of the group is known. The trust that one may place in this ingredient should be rather high, in my opinion, as it is computed with an algorithm which in simpler cases has been checked against other programs, and which even for H_4 has been checked against other algorithms for the same computation. (The latter check is perhaps more convincing than the former, for it may very well happen that some of the nastiest configurations occur for the first time in type H_4, or even only in type H_4, as it is such an exceptional group.)

For the actual non-negativity check, we are as far as I know in entirely uncharted territory. However, the code for the computation of the $h_{x,y,z}$ from the W-graph is really quite simple, so it can rather convincingly be checked by inspection. Another check is as follows : the order in which the computations are performed depends on the choice of a descent generator for x. By default, we always choose the first such generator in the internal numbering of `Coxeter`; however, it is easy enough to replace this by other "descent strategies" (for instance, choosing the first generator in some other ordering.) This will lead to a very different flow of recursion. I have done this replacing the first descent generator by the last one, and obtained the same output file, which is as it should be.

3.4. There is one check that I have *not* been able to perform, namely to verify the statement of Corollary 1.3 by a direct computation of the values of the a-function. The problem is that this involves the knowledge of all $h_{x,y,z}$ for a fixed value of z, instead of y, and this does not seem to be a local computation. The validity of Corollary 1.3 stems from the fact that its proof in [7] uses only the non-negativity of the coefficients of the $h_{x,y,z}$.

4 Questions

4.1. On looking at the lists of polynomials which occur as $h_{x,y,z}$, one immediately notices that they are not only non-negative, but have a much stronger positivity property : if we denote d the degree of $h_{x,y,z}$, then $v^d h_{x,y,z}$ is a polynomial in $q = v^2$, which is *unimodal* (recall that this means that the coefficients increase up to a point, which in our case has to be the middle because of the symmetry of the $h_{x,y,z}$, and decrease from there.)

4.2. For Weyl groups, there is one case where it is easy to prove that the unimodality property from 4.1 holds : *viz.*, the case where $y = w_0$ is the longest element in the group. From the properties of the $\leq_{\mathrm{LR}}$ preorder it is clear that $A.c_{w_0}$ is a two-sided ideal in $\mathcal{H}$; so we may write $c_x.c_{w_0} = h_x c_{w_0}$, where $h_x = h_{x,w_0,w_0}$. Now it is clear that $t_s.c_{w_0} = v.c_{w_0}$ for all s, hence $t_x.c_{w_0} = v^{l(x)}c_{w_0}$, and

$$c_x.c_{w_0} = \sum_{z \leq x} p_{z,x} v^{l(z)}$$

from which it follows immediately, using the expression of the intersection homology Poincaré polynomial in terms of Kazhdan-Lusztig polynomials ([6] Theorem 4.3.) that $v^d h_x$ is equal to the Poincaré polynomial of the intersection (hyper)cohomology of the Schubert variety X_x. The unimodality then follows from the so-called hard Lefschetz theorem.

4.3. Clearly, all the results about the Hecke algebra of type H_4 which are stated in this paper point to the fact that there is a hidden geometry here that is begging to be discovered. Hopefully, the facts about this geometry which the program opens up will help us understand what is going on, and serve as a guide towards the solution. I should be very happy if this turns out to be the case!

References

[1] D. Alvis. The Left Cells of the Coxeter Group of Type H_4. *J. Algebra*, **107**:160–168, 1987.

[2] F. du Cloux. `Coxeter`. available from `http://igd.univ-lyon1.fr/~ducloux/coxeter/coxeter3/english/coxeter3.html`.

[3] M. Geck and G. Pfeiffer. *Characters of finite Coxeter Groups and Iwahori-Hecke Algebras.* London Mathematical Society Monographs (N.S.) 21. Clarendon Press, Oxford, 2000.

[4] R.S. Irving. The socle filtration of a Verma module. *Ann. Sci. Ec. Norm. Sup. (4e série)*, **21**:47–65, 1988.

[5] D. Kazhdan and G. Lusztig. Representations of Coxeter groups and Hecke algebras. *Invent. Math.*, **53**:165–184, 1979.

[6] D. Kazhdan and G. Lusztig. Schubert Varieties and Poincare Duality. In *Geometry of the Laplace operator (Proc. Sympos. Pure Math., Univ. Hawaii, Honolulu, Hawaii, 1979)*, volume 37 of *Proc. Sympos. Pure Math.*, pages 185–203, Providence, R.I., 1980. Amer. Math. Soc.

[7] G. Lusztig. Cells in Affine Weyl Groups. In *Algebraic Groups and related topics (Kyoto/Nagoya 1983)*, volume 6 of *Adv. Stud. Pure Math.*, pages 255–287, Amsterdam, 1985. North-Holland.

[8] T.A. Springer. Quelques applications de la cohomologie d'intersection. In *Séminaire Bourbaki vol. 1981/82*, Astérisque 92–93, pages 249–273. Société Mathématique de France, 1982.