

# COMPUTATIONS ON THE KAZHDAN-LUSZTIG REPRESENTATIONS

TIMOTHY J. MCLARNAN

Department of Mathematics, Earlham College  
timm@math.earlham.edu

June 19, 2001

One can imagine at least 3 related conjectures related to the work of Lascoux and Schützenberger on the Kazhdan-Lusztig representations of  $S_n$ :

**The Edge Transport Conjecture.** *Every edge in the Kazhdan-Lusztig graph is the transport of a Bruhat edge.*

**The Bruhat Representation Conjecture.** *The transports of the Bruhat edges may not include all the K-L edges, but taken by themselves, they give rise to a representation of  $S_n$ .*

**The 0-1 Conjecture.** *Every edge in the K-L graph has weight either 0 or 1.*

What my calculations show is that none of these three conjectures is true. The ET and BR Conjectures fail for the first time in  $S_{14}$ ; the 0-1 Conjecture fails for the first time in  $S_{16}$ .

## THE EDGE TRANSPORT CONJECTURE

A counterexample to the ET Conjecture is a pair of permutations  $w = (P_w, Q)$  and  $x = (P_x, Q)$  in  $S_n$  such that  $\{x, w\}$  is an edge which is not the transport of a Bruhat edge. For simplicity, call the transports of Bruhat edges *good* edges. I'll regard  $S_n$  as the group of permutations of  $\{0, 1, 2, \dots, n-1\}$ , and I'll write numbers in base 16. Thus,  $S_{14}$  permutes the set

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d\}.$$

A typical element of  $S_{14}$  might be 83a0921c7b65d4.

We obviously can't look at every pair of permutations even in  $S_{10}$ , but we can cut down the search space a lot if we look only for a minimal counterexample. Let's therefore search for a counterexample  $\{w, x\}$  minimizing in order the following parameters.

- (a) Minimal  $n$ . All edges within L-cells in  $S_{n-1}$  should be good edges.

- (b) For that  $n$ , minimal  $l(w) - l(x)$ . (Here  $l(x)$  is the inversion number (length) of  $x$ .) All edges  $\{y, z\}$  within L-cells of  $S_n$  and with  $|l(y) - l(z)| < l(w) - l(x)$  should be good edges.
- (c) For this  $n$  and this length difference, minimal  $l(w)$ . All edges  $\{y, z\}$  within L-cells of  $S_n$  and with  $l(y) - l(z) = l(w) - l(x)$  and with  $l(y) < l(w)$  should be good edges.

A bad edge minimal in this sense must satisfy 6 conditions.

- (1)  $D(P_w) \subseteq D(P_x)$ . ( $D(P)$  is the descent set of the tableau  $P$ .) Otherwise  $\{w, x\}$  isn't an edge.
- (2)  $(P_w, Q) \succ (P_x, Q)$  (in Bruhat order) for all tableaux  $Q$ . Otherwise either the two are incomparable for some  $Q$ , in which case there is no edge, or else for some  $Q$  the length difference is 1, in which case you have a good edge.
- (3) The largest number,  $n - 1$ , sits strictly higher in  $P_w$  than in  $P_x$  (if the tableaux are written the French way). It can't be lower if the column words satisfy  $\text{cw}(P_w) \succ \text{cw}(P_x)$ ; it can't be the same height or you could delete it and get a counterexample in  $S_{n-1}$ .
- (4) If  $k + 2, k$  is a subword of the row word  $w(P_w)$ , then  $k + 2, k + 1, k$  is a subword of  $w(P_x)$ . This is equivalent to Property (1) plus the fact that

$$L_i P_w \prec P_w \implies P_x \notin \mathcal{L}_i,$$

which is true because otherwise  $\mu[L_i P_x, L_i P_w]$  could be computed by ET, because of our assumptions of minimality.

- (5) No sequence of  $L_i$ 's, say,  $L = L_{i_1}, L_{i_2}, \dots, L_{i_k}$  has the property that

$$l(LP_w) - l(LP_x) < l(P_w) - l(P_x). \tag{I}$$

- (6)  $l(w) - l(x)$  is odd.

A weaker form of (5) which is much faster to compute is also helpful:

- (5') No sequence  $L = L_{i_1}, L_{i_2}, \dots, L_{i_k}$  satisfies both (I) and

$$l(L_{i_j} L_{i_{j-1}} \cdots L_{i_1} P_w) - l(L_{i_j} L_{i_{j-1}} \cdots L_{i_1} P_x) = l(P_w) - l(P_x), \quad \text{all } j < k \tag{II}$$

Further, any edge within an L-cell that is a minimal counterexample to the 0-1 Conjecture must also satisfy

- (7) For no  $i$  is it the case that  $0, 1, \dots, i - 1$  are in identical positions in  $w$  and  $x$ , and is it the case that that when these letters are removed from  $w$  and  $x$ , the resulting words are tableau words. Otherwise you can compute  $\mu[w, x]$  by looking at these tableau words in  $S_{n-i}$ .

It's easy to check whether a given pair  $(w, x)$  satisfies (1), (3), (4), (6), and (7). It's more time-consuming to check (5'). It's hardest of all to check (5) and (2). Of course, you can't just list all the pairs within L-cells of  $S_{16}$  and then test these conditions, since  $S_{16}$  is too big.

It's not hard, however, to generate exactly the pairs of tableaux  $(P_w, P_x)$  that satisfy both (1) and (4). You build up the pairs one letter at a time, checking the descent

relations as you go. These pairs can then be piped through a filter which passes only pairs satisfying both (3) and (6) (the next easiest to test). The results go through a filter passing pairs satisfying (5'), then through a filter testing for compliance with (5), and finally through a filter passing pairs satisfying (2). Schematically, then, we have five programs, and we are doing something like

$$1\text{and}4 \mid 3\text{and}6 \mid 5\text{prime} \mid 5 \mid 2. \tag{III}$$

None of this code is trivial to write (or at least, none of it was trivial on the computers of 1988, when I was doing this). For instance, if you test (2) by computing all  $Q$  tableaux, doing inverse Schensted, and checking the Bruhat relations, the resulting program is way too slow to be useful. Instead, you need to generate the pairs by doing Knuth transformations, and you need to check for the Bruhat relation by seeing whether the Knuth transformation can have destroyed the Bruhat relation which applied before the transformation. This is much faster than checking Bruhat comparability of two completely random permutations. The other programs take similar care. Condition (5), for instance, ended up using a big data structure involving 2-3 trees.

In any event, for  $n \leq 13$ , no pair of tableaux makes it through the pipeline in (III): there is no pair satisfying (1–6). In  $S_{14}$ , the first pairs satisfying (1–6) appear. They are listed in Table 1 by column word.

$\text{cw}(P_w)$	$\text{cw}(P_x)$
b730d951a6284c	7320b651a94d8c
b730d941a52c68	4320b761a95d8c
b730d841952a6c	43208761ba5d9c
b9510da62c7384	b6210a543987dc
b7510d962a83c4	76210b543a98dc
b710c832d94a56	8410c732b65a9d
da610c7328495b	76510da3294c8b
ba610d7328495c	76510ba3294d8c
b510c732d8496a	5410c832b76a9d
9510b732c84d6a	54109832c76bad
d9510b732c84a6	95410d832c76ba
d9510a732b84c6	a54109832d76cb
d9510a632b74c8	65410a932d87cb
a8410c932b5d67	a5410983276cbd
b7310d952a64c8	b32107654a98dc
a8210c943b5d67	a5210984376cbd

**Table 1:** The pairs of tableaux in  $S_{14}$  satisfying conditions 1–6.

To test whether these edges are in fact counterexamples to the Edge Transport Conjecture, one needs to generate all pairs obtainable from them by  $L_i$  and  $R_j$  operations, and verify that none of these are Bruhat edges. One also needs to compute  $\mu[x, w]$  and to verify that it is non-zero. This latter computation is done using the standard recurrence that

$$\mu[x, w] = P_{xw}(q) \Big|_{q^{(l(w)-l(x)-1)/2}},$$

where  $v = sw \prec w$  and where

$$P_{xw}(q) = \begin{cases} P_{sx,w}(q), & \text{if } sx \succ x \\ P_{sx,v}(q) + qP_{xv}(q) - \sum_{sz \prec z} q^{(l(w)-l(z))/2} P_{xz}(q) \mu[z, v], & \text{if } sx \prec x \end{cases}$$

As always, one needs to use some care to make this calculation as efficient as possible, but there aren't any big surprises in the process. Much of the work in programming went into trying to make at every stage the best choice of  $s$ .

When you start with the pairs in Table 1 and apply to them every possible sequence of operations  $L_i$ , the resulting pairs are those shown in Table 2.

$cw(P_w)$	$cw(P_x)$
b730d951a6284c	7320b651a94d8c
b740d951a6283c	7420b651a93d8c
b830d951a6274c	8320b651a94d7c
c730d951a6284b	7320c651a94d8b
b840d951a6273c	8420b651a93d7c
c740d951a6283b	7420c651a93d8b
c830d951a6274b	8320c651a94d7b
c840d951a6273b	8420c651a93d7b
b730d941a52c68	4320b761a95d8c
b830d941a52c67	4320b861a95d7c
c730d941a52b68	4320c761a95d8b
c830d941a52b67	4320c861a95d7b
b730d841952a6c	43208761ba5d9c
c730d841952a6b	43208761ca5d9b
b9510da62c7384	b6210a543987dc
c9510da62b7384	c6210a543987db
b7510d962a83c4	76210b543a98dc
b8510d962a73c4	86210b543a97dc
c7510d962a83b4	76210c543a98db
c8510d962a73b4	86210c543a97db
b710c832d94a56	8410c732b65a9d
b720c831d94a56	8420c731b65a9d
da610c7328495b	76510da3294c8b
da620c7318495b	76520da3194c8b
da610c7428395b	76510da4293c8b
db610c7328495a	76510db3294c8a
db620c7318495a	76520db3194c8a
da620c7418395b	76520da4193c8b
db610c7428395a	76510db4293c8a
cb610d7328495a	76510cb3294d8a
cb620d7318495a	76520cb3194d8a
da630c7418295b	76530da4192c8b
db620c7418395a	76520db4193c8a
cb610d7428395a	76510cb4293d8a

$cw(P_w)$	$cw(P_x)$
ca610d7328495b	76510ca3294d8b
ca620d7318495b	76520ca3194d8b
db630c7418295a	76530db4192c8a
cb620d7418395a	76520cb4193d8a
ca610d7428395b	76510ca4293d8b
ba610d7328495c	76510ba3294d8c
ba620d7318495c	76520ba3194d8c
cb630d7418295a	76530cb4192d8a
ca620d7418395b	76520ca4193d8b
ba610d7428395c	76510ba4293d8c
ca630d7418295b	76530ca4192d8b
ba620d7418395c	76520ba4193d8c
ba630d7418295c	76530ba4192d8c
ba610d7328495c	76510ba3294d8c
ba620d7318495c	76520ba3194d8c
ba610d7428395c	76510ba4293d8c
ca610d7328495b	76510ca3294d8b
ca620d7318495b	76520ca3194d8b
ba620d7418395c	76520ba4193d8c
ca610d7428395b	76510ca4293d8b
cb610d7328495a	76510cb3294d8a
cb620d7318495a	76520cb3194d8a
ba630d7418295c	76530ba4192d8c
ca620d7418395b	76520ca4193d8b
cb610d7428395a	76510cb4293d8a
db610c7328495a	76510db3294c8a
db620c7318495a	76520db3194c8a
ca630d7418295b	76530ca4192d8b
cb620d7418395a	76520cb4193d8a
db610c7428395a	76510db4293c8a
da610c7328495b	76510da3294c8b
da620c7318495b	76520da3194c8b
cb630d7418295a	76530cb4192d8a
db620c7418395a	76520db4193c8a
da610c7428395b	76510da4293c8b
db630c7418295a	76530db4192c8a
da620c7418395b	76520da4193c8b
da630c7418295b	76530da4192c8b
b510c732d8496a	5410c832b76a9d
b520c731d8496a	5420c831b76a9d
b610c732d8495a	6410c832b75a9d
b620c731d8495a	6420c831b75a9d
9510b732c84d6a	54109832c76bad
9520b731c84d6a	54209831c76bad

$cw(P_w)$	$cw(P_x)$
9610b732c84d5a	64109832c75bad
a510b732c84d69	5410a832c76b9d
9620b731c84d5a	64209831c75bad
a520b731c84d69	5420a831c76b9d
a610b732c84d59	6410a832c75b9d
a620b731c84d59	6420a831c75b9d
d9510b732c84a6	95410d832c76ba
d9520b731c84a6	95420d831c76ba
d9610b732c84a5	96410d832c75ba
da510b732c8496	a5410d832c76b9
d9620b731c84a5	96420d831c75ba
da520b731c8496	a5420d831c76b9
da610b732c8495	a6410d832c75b9
da620b731c8495	a6420d831c75b9
d9510a732b84c6	a54109832d76cb
d9520a731b84c6	a54209831d76cb
d9610a732b84c5	a64109832d75cb
d9620a731b84c5	a64209831d75cb
d9510a632b74c8	65410a932d87cb
d9520a631b74c8	65420a931d87cb
a8410c932b5d67	a5410983276cbd
a8420c931b5d67	a5420983176cbd
b8410c932a5d67	b5410983276cad
a8410d932b5c67	a5410983276dbc
a8320c941b5d67	a5320984176cbd
b8420c931a5d67	b5420983176cad
a8420d931b5c67	a5420983176dbc
b8410d932a5c67	b5410983276dac
a8310c942b5d67	a5310984276cbd
b8320c941a5d67	b5320984176cad
a8320d941b5c67	a5320984176dbc
b8420d931a5c67	b5420983176dac
c8410d932a5b67	c5410983276dab
a8210c943b5d67	a5210984376cbd
b8310c942a5d67	b5310984276cad
a8310d942b5c67	a5310984276dbc
b8320d941a5c67	b5320984176dac
c8420d931a5b67	c5420983176dab
b8210c943a5d67	b5210984376cad
a8210d943b5c67	a5210984376dbc
b8310d942a5c67	b5310984276dac
c8320d941a5b67	c5320984176dab
b8210d943a5c67	b5210984376dac
c8310d942a5b67	c5310984276dab

$cw(P_w)$	$cw(P_x)$
c8210d943a5b67	c5210984376dab
b7310d952a64c8	b32107654a98dc
b7410d952a63c8	b42107653a98dc
b8310d952a64c7	b32108654a97dc
c7310d952a64b8	c32107654a98db
b8410d952a63c7	b42108653a97dc
c7410d952a63b8	c42107653a98db
c8310d952a64b7	c32108654a97db
c8410d952a63b7	c42108653a97db
a8210c943b5d67	a5210984376cbd
a8310c942b5d67	a5310984276cbd
b8210c943a5d67	b5210984376cad
a8210d943b5c67	a5210984376dbc
a8320c941b5d67	a5320984176cbd
b8310c942a5d67	b5310984276cad
a8310d942b5c67	a5310984276dbc
b8210d943a5c67	b5210984376dac
a8420c931b5d67	a5420983176cbd
b8320c941a5d67	b5320984176cad
a8320d941b5c67	a5320984176dbc
b8310d942a5c67	b5310984276dac
c8210d943a5b67	c5210984376dab
a8410c932b5d67	a5410983276cbd
b8420c931a5d67	b5420983176cad
a8420d931b5c67	a5420983176dbc
b8320d941a5c67	b5320984176dac
c8310d942a5b67	c5310984276dab
b8410c932a5d67	b5410983276cad
a8410d932b5c67	a5410983276dbc
b8420d931a5c67	b5420983176dac
c8320d941a5b67	c5320984176dab
b8410d932a5c67	b5410983276dac
c8420d931a5b67	c5420983176dab
c8410d932a5b67	c5410983276dab

**Table 2:** Edges in  $S_{14}$  which are not obtained from Bruhat edges by edge transport.

None of these edges are Bruhat edges, and all of them have  $\mu[x, w] = 1$ , so all of them are counterexamples to the Edge Transport Conjecture. I think it is possible that there are additional counterexamples in  $S_{14}$  not related to these multiply minimal ones.

### THE BRUHAT EDGE CONJECTURE

Now let's look at the conjecture that even though not every edge of the K-L graph is a good edge, the good edges by themselves give rise to a representation of  $S_n$ . To

believe this would seem to me to call for much more than normal optimism, but it seemed worth the trouble to try to disprove it anyway. We'll do this by showing that the matrices obtained from the good edges fail to satisfy the Coxeter relations for  $S_n$ .

Let  $B^\lambda(s)$  denote the K-L matrix for the elementary reflection  $s$ , and let  $S^\lambda(s)$  be the corresponding matrix computed using only the good edges.

It's not hard to show that  $A^\lambda(s)^2 = I$ . Indeed, if we take any graph whose vertices are labeled by the tableaux of shape  $\lambda$ , and we construct a matrix for  $s$  using the K-L formulas, then the square of this matrix will be  $I$ .

To show that the  $A$  matrices don't form a representation, let's therefore look for some  $t > s + 1$  with

$$A^\lambda(s)A^\lambda(t) \neq A^\lambda(t)A^\lambda(s). \quad (\text{IV})$$

It's a major undertaking to write these matrices out in their entirety, so we should try to pick a particular  $\lambda$ ,  $s$ , and  $t$  and a particular row and column for which inequality is unlikely. To do this, we need to see how a bad edge  $\{w, x\}$  affects  $B^\lambda(s)B^\lambda(t)$  and  $B^\lambda(t)B^\lambda(s)$ . There are lots of cases to consider, but some playing around shows that one place we might find the broken Coxeter relation in (IV) is where the following conditions are satisfied.

- (1)  $\{w, x\}$  is a bad edge;  $w$  and  $x$  are the column words of  $P_w$  and  $P_x$ , resp.
- (2)  $sx \prec x$  and  $sw \succ w$ .
- (3)  $tx \succ x$  and  $tw \succ w$ .
- (4)  $sw$ ,  $sx$ , and  $tx$  are the column words of tableaux, but  $tw$  is not.

If (1–4) are satisfied then the entries in the K-L matrices in rows and columns indexed by  $sx$ ,  $x$ ,  $tx$ ,  $w$ , and  $sw$  look like this:

$$B^\lambda(s) = \begin{matrix} & sx & x & tx & w & sw \\ \begin{matrix} sx \\ x \\ tx \\ w \\ sw \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{matrix}, \quad B^\lambda(t) = \begin{matrix} & sx & x & tx & w & sw \\ \begin{matrix} sx \\ x \\ tx \\ w \\ sw \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & ? & 0 & 1 \end{pmatrix} \end{matrix}.$$

(All but one of the entries in these positions can be computed easily; the one non-trivial position is marked with a question mark.) The bold face  $\mathbf{1}$  in the  $(4, 2)$  position of the first matrix is contributed by the bad edge  $\{w, x\}$ ; all other edges shown are good edges.

The important thing to notice is that changing the bold face  $\mathbf{1}$  in the first matrix to a 0 decreases the  $(w, tx)$  entry of  $B^\lambda(s)B^\lambda(t)$  by 1, but has no effect on  $(w, tx)$  entry of  $B^\lambda(s)B^\lambda(t)$ . One therefore has reason to believe that perhaps  $(A^\lambda(s)A^\lambda(t))_{w,tx} \neq (A^\lambda(t)A^\lambda(s))_{w,tx}$ . Of course, entries in rows and columns we haven't considered could change the story, so we have to do a calculation.

Inspection of the bad edges in Table 2 led me to the following pair satisfying conditions (1–4) with  $s = 4$  and  $t = b$ :

$$P_w = \begin{matrix} & d & & & & \\ & b & c & & & \\ 6 & 7 & & & & \\ 3 & 4 & 8 & 9 & & \\ 0 & 1 & 2 & 5 & a & \end{matrix} \quad P_x = \begin{matrix} & 7 & & & & \\ & 6 & d & & & \\ 5 & b & & & & \\ 3 & 4 & 9 & c & & \\ 0 & 1 & 2 & 8 & a & \end{matrix}$$

The dimension of the representation  $B^\lambda$  is thus 68640.

It turns out not to be too hard to compute the matrix entries  $(A^\lambda(s)A^\lambda(t))_{w,tx}$  and  $(A^\lambda(t)A^\lambda(s))_{w,tx}$ . The K-L matrices are quite sparse, and in nearly every product  $(A^\lambda(s))_{w,y}(A^\lambda(t))_{y,tx}$  and  $(A^\lambda(t))_{w,y}(A^\lambda(s))_{y,tx}$ , one of the two factors is obviously 0.

When the dust settles,  $(A^\lambda(s)A^\lambda(t))_{w,tx} = 0$  and  $(A^\lambda(t)A^\lambda(s))_{w,tx} = 1$ . The matrices computed from the graphs containing only the good edges therefore fail to be a representation of  $S_n$ .

### THE 0-1 CONJECTURE

Finally, what about the conjecture that all the edges of the K-L graph (or at least, all the edges within any L-cell) have weight 0 or 1? To test this conjecture, one can again make use of the criteria for possible counterexamples listed on page 2. This time, the most efficient approach seemed to be to use a pipe that looked like

```
1and3and4and7 | 6 | 5prime | 5 | 2prime.
```

The program `2prime` checks whether condition (2) from page 2 is satisfied. If (2) is satisfied, then instead of printing the original pair of permutations  $(P_w, Q)$  and  $(P_x, Q)$ , `2prime` chooses a right tableau  $Q'$  that minimizes the length difference between  $(P_w, Q')$  and  $(P_x, Q')$ , and outputs the new pair of permutations.

In  $S_{14}$  and  $S_{15}$ , every pair that makes it through this pipe has  $\mu = 1$ ; so the 0-1 Conjecture is true through  $S_{15}$ .

At this point I was temporarily stymied until I was able to improve my algorithms. I therefore looked at the weights of *all* edges (not just those within L-cells) in  $S_9$ , thinking that bad things might first happen between L-cells. All these edges still had weight 1, however, as did all edges found in a Monte Carlo experiment for larger values of  $n$ .

The break finally happens in  $S_{16}$ . The pair with minimal length and then minimal length difference for which the 0-1 Conjecture fails is

$$\begin{aligned} w &= c810d942fa53b6e7, & l(w) &= 53 \\ x &= 54109832dc76bafe, & l(x) &= 32. \end{aligned}$$

The difference in lengths is 21, and the leading coefficient (the coefficient of degree 10) of  $P_{xw}(q)$  is  $\mu[x, w] = 5$ . The K-L polynomial in its entirety is

$$P_{xw}(q) = 5q^{10} + 72q^9 + 387q^8 + 1039q^7 + 1610q^6 + 1536q^5 + 931q^4 + 365q^3 + 92q^2 + 14q + 1.$$

I believe that the only other counterexample in  $S_{16}$  to the 0-1 Conjecture which is minimal in all these senses is

$$\begin{aligned} w &= ca610fb732d84e95, & l(w) &= 61 \\ x &= 76310cb542a98fed, & l(x) &= 40, \end{aligned}$$

for which, again,  $\mu[x, w] = 5$ .

## REMARKS ON THE COMPUTATIONS

All these results represent calculations done between 1985–1988, initially on an 8 MHz 8086 box with 640kB of RAM, and later on a Sun 3, and finally on a NeXT machine with a 25MHz 68040 processor and 40MB of RAM. I never used more than one machine at a time. The nastiest calculations took a couple of months of CPU time. Midwestern thunderstorms and the associated loss of power therefore meant that saving the state of the calculation to disk periodically was a necessity.

Throughout the project, the code was in a continual state of profiling and rewriting, so that the algorithms I was using at the end were many orders of magnitude faster than those I started out with.

How does one convince oneself of the correctness of these calculations? Partly the answer is that the process is one of bootstrapping. The initial algorithms could be tested for small values of  $n$  against hand computations. Each new refinement of the code was then tested against the largest computations possible for the previous version. Finally, the entire body of work in C was set aside and rewritten in Lisp without reference to the earlier code, and all the computations were repeated with the new software. The only exception to this was, I believe, the work in  $S_{16}$  on the 0-1 Conjecture, where the computational and storage efficiency of C was necessary. That this effort confirmed the original calculations and that the body of results seem consistent with one another are the strongest guarantees of the results reported here.

On the other hand, there are plenty of places where the validity of the calculations relies on the correctness of some simple lemma which could be wrong. Not knowing how to exclude this possibility entirely, and not feeling completely comfortable with the algebra underlying the K-L representations, I set all this aside without publishing it, in an act either of paranoia and insecurity, or of prudence and modesty.

Here it is now, though, in the hope it may still be useful to somebody.

## ACKNOWLEDGMENTS

Adriano Garsia introduced me to the K-L representations and taught me the little I know about them. I would never have started this work were it not for him; I'd have completed it in a better fashion had he had his way. He deserves both my hearty thanks and my sheepish apologies.

My wife Ann, who has also tolerated with remarkable grace the percentage of my time and income lavished on my silicon companions, merits far more thanks than this note can offer.