Computation in Coxeter groups I. Multiplication

Bill Casselman
Mathematics Department
University of British Columbia
Canada
cass@math.ubc.ca

Abstract. *An efficient and purely combinatorial algorithm for calculating products in arbitrary Coxeter groups is presented, which combines ideas of Fokko du Cloux and myself. Proofs are largely based on geometry. The algorithm has been implemented in practical Java programs, and runs surprisingly quickly. It seems to be good enough in many interesting cases to build the minimal root reflection table of Brink and Howlett, which can be used for a more efficient multiplication routine.*

MR subject classifications: 20H15, 20-04
Submitted March 28, 2001; accepted August 25, 2001.

A **Coxeter group** is a pair (W, S) where W is a group generated by elements from its subset S, subject to relations

$$(st)^{m_{s,t}} = 1$$

for all s and t in S, where (a) the exponent $m_{s,s} = 1$ for each s in S and (b) for all $s \neq t$ the exponent $m_{s,t}$ is either a non-negative integer or ∞ (indicating no relation). Although there some interesting cases where S is infinite, in this paper no harm will be done by assuming S to be finite. Since $m_{s,s} = 1$, each s in S is an **involution**:

$$s^2 = 1 \quad \text{for all } s \in S\,.$$

If we apply this to the other relations we deduce the **braid relations**:

$$st\ldots = ts\ldots \quad (m_{s,t} \text{ terms on each side})\,.$$

The array $m_{s,t}$ indexed by pairs of elements of S is called a **Coxeter matrix**. A pair of distinct elements s and t will commute if and only if $m_{s,t} = 2$. The labeled graph whose nodes are elements of S, with an edge linking non-commuting s and t, labeled by $m_{s,t}$, is called the associated **Coxeter graph**. (For $m_{s,t} = 3$ the labels are often omitted.)

Coxeter groups are ubiquitous. The symmetry group of a regular geometric figure (for example, any of the five Platonic solids) is a Coxeter group, and so is the Weyl group of any Kac-Moody Lie algebra (and in particular any finite-dimensional semi-simple Lie algebra). The Weyl groups of finite-dimensional semi-simple Lie algebras are those associated to the finite root systems $A_n\,(n \geq 1)$, $B_n\,(n \geq 2)$, $C_n\,(n \geq 2)$, $D_n\,(n \geq 4)$, $E_n\,(n = 6, 7, 8)$, F_4, and G_2. The Coxeter groups determined by the affine root systems associated to these are also the Weyl groups of affine Kac-Moody Lie algebras. The other finite Coxeter groups are the remaining dihedral groups $I_p\,(p \neq 2, 3, 4, 6)$, as well as the symmetry group H_3 of the icosahedron and the group H_4, which is the symmetry group of a regular polyhedron in four dimensions called the 120-cell.

In spite of their great importance and the great amount of effort spent on them, there are many puzzles involving Coxeter groups. Some of these puzzles are among the most intriguing in all of mathematics—suggesting, like the Riemann hypothesis, that there are whole categories of structures we haven't imagined yet. This is especially true in regard to the polynomials $P_{x,y}$ associated to pairs of elements of a Coxeter group by Kazhdan and Lusztig in 1981, and the W-graphs determined by these polynomials. In another direction, the structure of Kac-Moody algebras other than the finite-dimensional or affine Lie algebras is still largely uncharted territory. There are, for example, many unanswered questions about the nature of the roots of a non-symmetrizable Kac-Moody Lie algebra which probably reduce to understanding better the geometry of their Weyl groups. The puzzles encountered in studying arbitrary Coxeter groups suggests that it would undoubtedly be useful to be able to use computers to work effectively with them. This is all the more true since many computational problems, such as computing Kazhdan-Lusztig polynomials, overwhelm conventional symbolic algebra packages. Extreme efficiency is a necessity for many explorations, and demands sophisticated programming. In addition to the practical interest in exploring Coxeter groups computationally, there are mathematical problems interesting in their own right involved with such computation.

In this paper, I shall combine ideas of Fokko du Cloux and myself to explain how to program the very simplest of operations in an arbitrary Coxeter group—multiplication of an element by a single generator. As will be seen, this is by no means a trivial problem. The key idea is due to du Cloux, who has used it to design programs for finite Coxeter groups, and the principal accomplishment of this paper is a practical implementation of his idea without the restriction of finiteness. I have not been able to determine the efficiency of the algorithms in a theoretical way, but experience justifies my claims of practicality.

It would seem at first sight that the techniques available for Coxeter groups are rather special. Nonetheless, it would be interesting to know if similar methods can be applied to other groups as well. Multiplication in groups is one place where one might expect to be able to use some of the extremely sophisticated algorithms to be found in language parsing (for example, those devised by Knuth to deal with LR languages), but I have seen little sign of this (in spite of otherwise interesting work done with, for example, automatic groups). For this reason, the results of this paper might conceivably be of interest to those who don't care much about Coxeter groups *per se*.

1. The problem

Every element w of W can be written as a product of elements of S. A **reduced expression** for an element of W is an expression

$$w = s_1 s_2 \dots s_n$$

where n is minimal. The **length** of w is this minimal length n. It is immediate from the definition of W that there exists a unique **parity homomorphism** from W to $\{\pm 1\}$ taking elements of S to -1. This and an elementary argument implies that if w has length n, then sw has length $n+1$ or $n-1$. We write $ws > w$ or $ws < w$, accordingly.

In order to calculate with elements of W, it is necessary to represent each of them uniquely. In this paper, each element of W will be identified with one of its reduced expressions. In order to do this, first put a linear order on S, or equivalently count the elements of S in some order. In this paper I shall call the **normal form** of w that reduced word $NF(w)$ which is lexicographically least if read backwards. In other words, a normal form expression is defined recursively by the conditions (1) the identity element is expressed by the empty string of generators; (2) if w has the normal form

$$w = s_1 s_2 \ldots s_{n-1} s_n$$

then s_n is the least element among the elements s of S such that $ws < w$ and $s_1 s_2 \ldots s_{n-1}$ is the normal form of ws_n. The normal form referred to here, which is called the `InverseShortLex` form, is just one of two used often in the literature. The other is the `ShortLex` form, in which s_1 is the least element of the elements s of S such that $sw < w$, etc. In the `ShortLex` form, w is represented by an expression which is lexicographically least when read from left to right, whereas in `InverseShortLex` when read from right to left (i.e. in inverse order).

For example, the Coxeter group determined by the root system C_2 has two generators $\langle 1\rangle$, $\langle 2\rangle$ and $m_{1,2} = 4$. There are 8 elements in all, whose `InverseShortLex` words are

$$\emptyset,\ \langle 1\rangle,\ \langle 2\rangle,\ \langle 1\rangle\langle 2\rangle,\ \langle 2\rangle\langle 1\rangle,\ \langle 1\rangle\langle 2\rangle\langle 1\rangle,\ \langle 2\rangle\langle 1\rangle\langle 2\rangle,\ \langle 2\rangle\langle 1\rangle\langle 2\rangle\langle 1\rangle\ .$$

The last element has also the reduced expression $\langle 1\rangle\langle 2\rangle\langle 1\rangle\langle 2\rangle$, but this is not in the language of `InverseShortLex` words.

The basic problem addressed by this paper is this:

- *Given any element $w = s_1 s_2 \ldots s_n$, find its* `InverseShortLex` *form.*

By induction, this reduces to a simpler problem:

- *Given any element $w = s_1 s_2 \ldots s_n$ expressed in* `InverseShortLex` *form and an element s in S, find the* `InverseShortLex` *form of sw.*

I will review previous methods used to solve these problems, and then explain the new one. In order to do this, I need to recall geometric properties of Coxeter groups. Since Coxeter groups other than the finite ones and the affine ones are relatively unfamiliar, I will begin by reviewing some elementary facts. The standard references for things not proven here are the books by Bourbaki and Humphreys, as well as the survey article by Vinberg. Also useful are the informal lecture notes of Howlett.

2. Cartan matrices

In this paper, a **Cartan matrix** indexed by a finite set S is a square matrix with real entries $c_{s,t}$ (s, t in S) satisfying these conditions:

(C1) $c_{s,s} = 2$ for all s.
(C2) For $s \neq t$, $c_{s,t} \leq 0$.
(C3) If $c_{s,t} = 0$ then so is $c_{t,s}$.
(C4) For $s \neq t$ let $n_{s,t}$ be the real number $c_{s,t} c_{t,s}$, which according to condition (2) is non-negative. If $0 < n_{s,t} < 4$ then

$$n_{s,t} = 4\cos^2(\pi/m_{s,t})$$

for some integer $m_{s,t} > 2$.

The significance of Cartan matrices is that they give rise to particularly useful representations of Coxeter groups, ones which mirror the combinatorial structure of the group. Suppose V to be a finite-dimensional real vector space, and the α_s for s in S to form a basis of a real vector space V^* dual to V. Then elements $\alpha_s^\vee$ of V are determined uniquely by the conditions

$$\langle \alpha_s, \alpha_t^\vee \rangle = c_{s,t} \ .$$

Since $c_{s,s} = 2$, for each s in S the linear transformation on V

$$\rho_s \colon\ v \mapsto v - \langle \alpha_s, v \rangle \, \alpha_s^\vee$$

is a reflection—that is to say, a linear transformation fixing vectors in the hyperplane $\{\alpha_s = 0\}$, and acting as multiplication by -1 on the transversal line spanned by $\alpha_s^\vee$. The map taking s to ρ_s extends to a representation of a certain Coxeter group whose matrix is determined by the Cartan matrix according to the following conditions:

(1) $m_{s,s} = 1$ for all s;
(2) if $0 < n_{s,t} < 4$ then the integers $m_{s,t}$ are those specified in condition (C4);
(3) if $n_{s,t} = 0$ then $m_{s,t} = 2$;
(4) if $n \geq 4$ then $m_{s,t} = \infty$.

It is essentially condition (C4) that guarantees that the braid relations are preserved by the representation when the $m_{s,t}$ are finite. If its entries $c_{s,t}$ are integers, a Cartan matrix is called **integral**, and for these condition (C4) is redundant. Each integral Cartan matrix gives rise to an associated Kac-Moody Lie algebra, and the Coxeter group of the matrix is the Weyl group of the Lie algebra.

Every Coxeter group arises from at least one Cartan matrix, the **standard** one with

$$c_{s,t} = -2\cos(\pi/m_{s,t}) \ .$$

Given a Cartan matrix and associated representation of W, define the open simplicial cone

$$C = \{ v \mid \langle \alpha_s, v \rangle > 0 \text{ for all } s \} \ .$$

The primary tie between geometry and the combinatorics of Coxeter groups is that for any realization of W (1) $sw > w$ if and only if $\alpha_s > 0$ on wC (i.e. wC lies on the same side of the hyperplane $\alpha_s = 0$ as C); (2) $sw < w$ if and only if $\alpha_s < 0$ on wC (it lies on the opposite side). There are many consequences of this simple geometric criterion for whether sw is longer or shorter than w.

The transforms of $\overline{C}$ by elements of W are called the **closed chambers** of the realization. Let $\mathcal{C}$ be the union of all these. It is clearly stable under non-negative scalar multiplication, and it turns out also to be convex. It is often called the **Tits cone**. The principal result relating geometry and combinatorics was first proved in complete generality in **Tits (1968)**:

Theorem. *The map taking s to ρ_s is a faithful representation of W on V. The group W acts discretely on $\mathcal{C}$, and $\overline{C}$ is a fundamental domain for W acting on this region. A subgroup H of W is finite if and only if it stabilizes a point in the interior of $\mathcal{C}$.*

For each subset T of S define the open face C_T of $\overline{C}$ to be where $\alpha_s = 0$ for s in T and $\alpha_s > 0$ for s not in T. Thus $C = C_\emptyset$ is the interior of $\overline{C}$, and $\overline{C}$ is the disjoint union of the C_T. A special case of this concerns faces of codimension one. If s and t are two elements of S and $wC_{\{s\}} \cap C_{\{t\}} \neq \emptyset$ then $s = t$ and $w = 1$ or $w = s$. As a consequence, each face of codimension one of a closed chamber is a W-transform of a unique face of $\overline{C}$, and hence each such face can be labelled canonically by an element of S. If two chambers $x\overline{C}$ and $y\overline{C}$ share a face labeled by s then $x = ys$.

Recall that the **Cayley graph** of (W, S) is the graph whose nodes are elements w of W, with a link between w and ws. The Cayley graph is a familiar and useful tool in combinatorial investigations of any group with generators. The point of looking at the geometry of the cone $\mathcal{C}$ and the chambers of a realization are that they offer a geometric image of the Cayley graph of (W, S). This is because of the remark made just above. If $w = s_1 s_2 \dots s_n$ then we can track this expression by a sequence of chambers

$$C_0 = C,\ C_1 = s_1 C,\ C_2 = s_1 s_2 C, \dots, C_n = wC$$

where each successive pair C_{i-1} and C_i share a face labeled by $\{s_i\}$. Such a sequence is called a **gallery**. The length of an element w is also the length of a minimal gallery from C to wC.

Geometrically, if D is the chamber wC then the last element s_n of a normal form for w is that element of S least among those s such that the hyperplane containing the face D_s separates D from C.

The **basic roots** associated to a Cartan matrix are the half-spaces $\alpha_s \geq 0$, and we obtain the other (geometric) roots as W-transforms of the basic ones. These geometric roots are distinct but related to the **algebraic roots**, which are the transforms of the functions α_s themselves. Normally, the geometric roots have more intrinsic significance. The positive ones are those containing C, the negative ones their complements. It turns out that all roots are either positive or negative.

For $T \subseteq S$ define W_T to be the subgroup of W generated by elements of T. This is itself a Coxeter group. Every element of W can be factored uniquely as a product xy where y lies in W_T and x has the property that $x\alpha_t > 0$ for all t in T. The set of all such elements x make up canonical representatives of W/W_T, and are called **distinguished** with respect to T.

3. An example

Let $\widetilde{A}_2$ be the Coxeter group associated to the Cartan matrix

$$\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} .$$

The Coxeter matrix has $m_{s,t} = 3$ for all s, t. As its Coxeter graph demonstrates, any permutation of the generators induces an automorphism of the group.

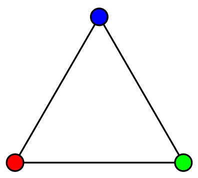


Figure 1. *The Coxeter graph of $\widetilde{A}_2$.*

In the realization determined by this matrix, introduce coordinates through the roots α_i: $v = (x_1, x_2, x_3)$ if $x_i = \langle \alpha_i, v \rangle$. The chamber C is the positive octant $x_i > 0$. The vectors $\alpha_i^\vee$ are

$$\begin{aligned}\alpha_1 &= (2, -1, -1)\\ \alpha_2 &= (-1, 2, -1)\\ \alpha_3 &= (-1, -1, 2)\end{aligned}$$

which turn out in this case to be linearly dependent—they span the plane $x_1 + x_2 + x_3 = 0$. The reflections ρ_i leave the plane $x_1 + x_2 + x_3 = 1$ invariant. This plane contains the three basis vectors

$$\begin{aligned}\varpi_1 &= (1, 0, 0)\\ \varpi_2 &= (0, 1, 0)\\ \varpi_3 &= (0, 0, 1)\end{aligned}$$

and we can picture the geometry of the Coxeter group by looking only at this slice, on which the elements of W act by affine transformations.

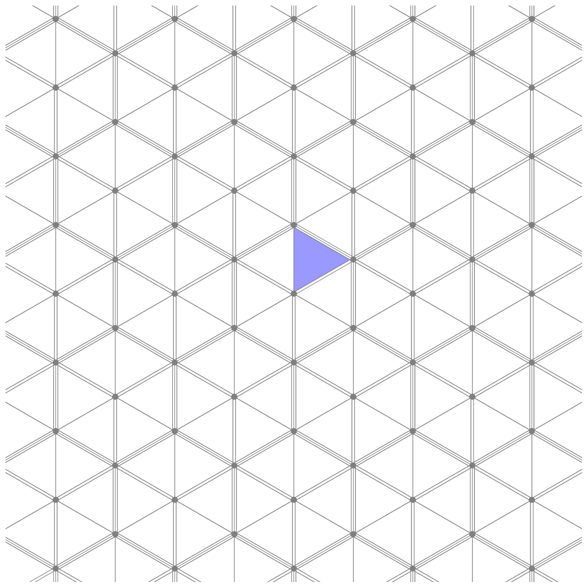


Figure 2. *A slice through chambers of $\widetilde{A}_2$. Edges of chambers are labeled by line multiplicities.*

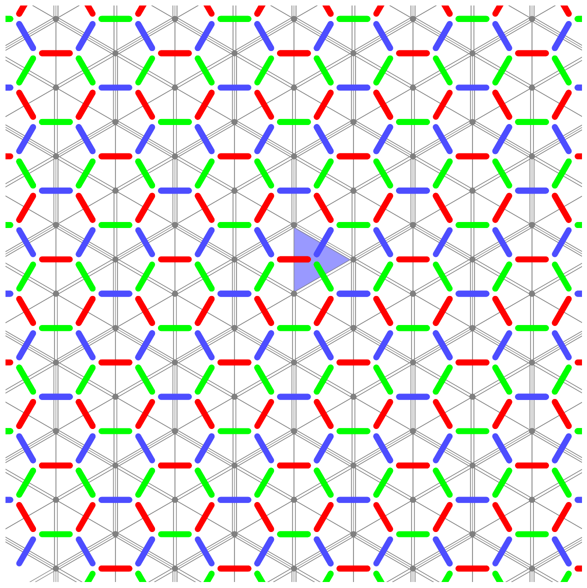


Figure 3. *The Cayley graph of $\widetilde{A}_2$. Generators are labeled by color.*

This group is in fact the affine Weyl group associated to the root system A_2. Below is shown how a typical gallery in the group is constructed in steps.

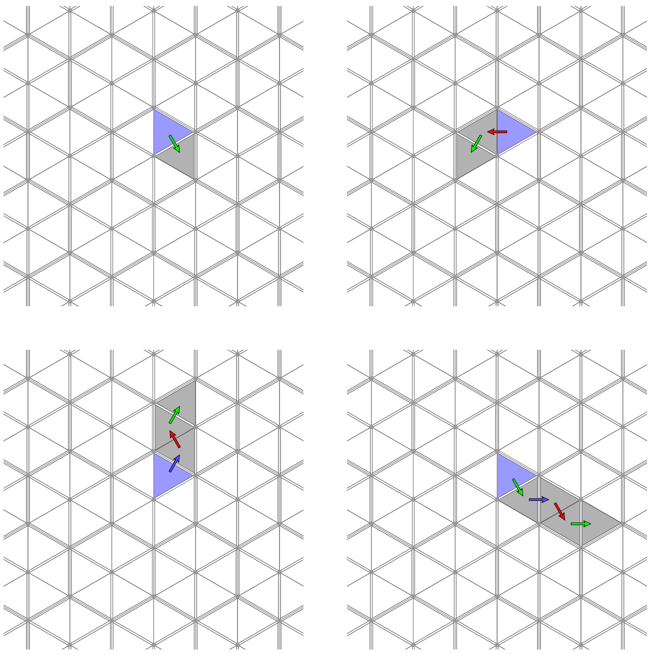


Figure 4. *Building the gallery* $\langle 2\rangle\langle 1\rangle\langle 3\rangle\langle 1\rangle$.

And just below here is the `InverseShortLex` tree for the same group.

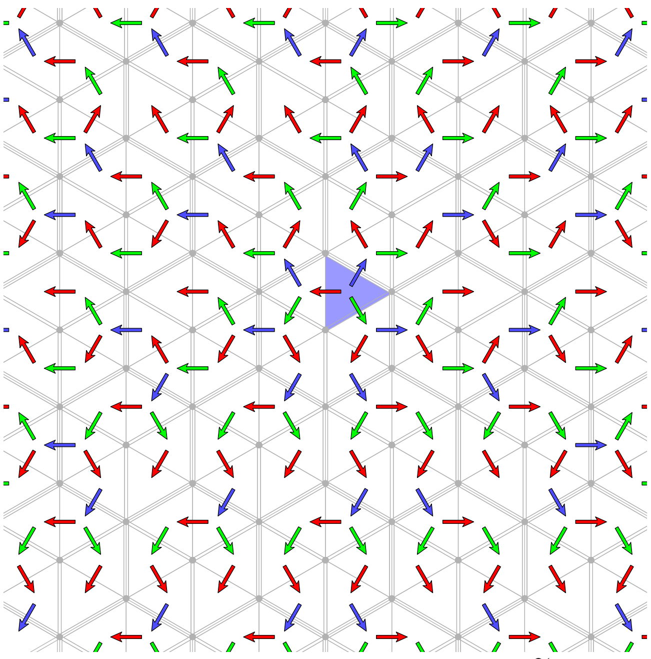


Figure 5. *The* `InverseShortLex` *tree of* $\widetilde{A}_2$*, edges oriented towards greater length. An arrow into an alcove traverses the wall with the least label separating that alcove from* C.

4. The geometric algorithm

One solution to the problem of computing products in W is geometric in nature. For any vector v in V and simple algebraic root α, let

$$v_\alpha = \langle \alpha, v \rangle \ .$$

These are effectively coordinates of v. If β is any simple root, then we can compute the effect of the reflection s_β on these coordinates according to the formula

$$(s_\beta v)_\alpha = \langle \alpha, v - \langle \beta, v \rangle \beta^\vee \rangle = v_\alpha - \langle \alpha, \beta^\vee \rangle v_\beta \ .$$

This is quite efficient since only the coefficients for roots α linked to β in the Dynkin graph will change.

Let ρ be the element of V such that $\rho_\alpha = 1$ for all simple roots α. It lies in C, and for any w in W the vector $w^{-1}\rho$ lies in $w^{-1}C$. We have $ws < w$ if and only if $sw^{-1} < w^{-1}$, or equivalently if and only if $\alpha = 0$ separates C from $w^{-1}C$, or again if

$$(w^{-1}\rho)_\alpha = \langle \alpha, w^{-1}\rho \rangle < 0 \ .$$

Thus the last generator s in an `InverseShortLex` expression for w is the least of those α such that $(w^{-1}\rho)_\alpha < 0$. Since we can calculate all the coordinates $(s_n s_{n-1} ... s_1 \rho)_\alpha$ inductively by the formulas above, we can then use this idea to calculate the `InverseShortLex` form of w. In effect, we are identifying an element w with its vector $w^{-1}\rho$.

There is a catch, however. The reflections s are not in general expressed in terms of integers. In the standard representation, for example, the coordinates of a vector $w^{-1}\rho$ will be sums of roots of unity. For only a very small number of Coxeter groups—those with all $m_{s,t} = 1$, 2, 3, 6, or ∞—can we find representations with rational coordinates. Therefore we can expect the limited precision of real numbers stored in computers to cause real trouble (no pun intended). It is notoriously difficult, for example, to tell whether a sum of roots of unity is positive or negative. The method described here for finding `InverseShortLex` forms looks in principle, at least, quite unsatisfactory. In practice, for technical reasons I won't go into, it works pretty well for finite and affine Coxeter groups, but it definitely looks untrustworthy for others.

5. Tits' algorithm

The first combinatorial method found to derive normal forms of elements of a Coxeter group is due to Jacques Tits, although he didn't explicitly use a notion of normal form. He first defines a partial order among words in S: he says that $x \to y$ if a pair ss in x is deleted, or if one side of a braid relation is replaced by the other, in order to obtain y. Such a deletion or replacement is called by Tits a **simplification**. By definition of a group defined by generators and relations, x and y give rise to the same element of W if and only if there is a chain of words $x_1 = x$, …, $x_n = y$ with either $x_i \to x_{i+1}$ or $x_{i+1} \to x_i$. Tits' basic theorem is a strong refinement of this assertion: *x and y give rise to the same element of W if and only if there exist sequences $x_1 = x$, …, x_m and $y_1 = y$, …, $y_n = x_m$ such that $x_i \to x_{i+1}$ and $y_i \to y_{i+1}$ for all i.* The point is that the lengths of words always decreases, whereas *a priori* one might expect to insert arbitrary expressions ss. In particular, two reduced words of the same length give rise to the same element of W if and only if one can deduce one from the other by a chain of braid relations. As a consequence, if we list all the words one obtains from a given one by successive simplifications, its `InverseShortLex` word will be among them. So one can find it by sorting the subset of all listed words of shortest length according to `InverseShortLex` order and picking out the least one.

This algorithm has the definite advantage that it really is purely combinatorial. For groups where the size of S and the length of w are small, applying it in manual computation is reasonable, and indeed it may be the only technique practical for hand work. Implementing it in in a program requires only well known techniques of string processing to do it as well as could be expected. The principal trick is to apply a fairly standard algorithm first introduced by Alfred Aho and Margaret Corasick for string recognition. Even so, this algorithm is not at all practical for finding the `InverseShortLex` forms of elements of large length, by hand or machine. The principal reason for this is that any element of W is likely to have a large number of reduced expressions—even a huge number—and all of them will be produced. Another major drawback, in comparison with the algorithm to be explained later on, is that there does not seem to be any good way to use the calculations for short elements to make more efficient those for long ones. In finding the `InverseShortLex` form of an element ws where that for w is known, it is not obvious how to use what you know about w to work with ws.

One improvement one might hope to make is to restrict to braid relations going from one word to another which is in `InverseShortLex`. This would allow a huge reduction in complexity, certainly. But we cannot make this improvement, as the finite Weyl group of type A_3 already illustrates. The braid relations in this case are

$$\begin{aligned}\langle 2\rangle\langle 1\rangle\langle 2\rangle &= \langle 1\rangle\langle 2\rangle\langle 1\rangle\\ \langle 3\rangle\langle 2\rangle\langle 3\rangle &= \langle 2\rangle\langle 3\rangle\langle 2\rangle\\ \langle 1\rangle\langle 3\rangle &= \langle 3\rangle\langle 1\rangle\end{aligned}$$

where the terms on the right are in `InverseShortLex`. The word $\langle 1\rangle\langle 2\rangle\langle 3\rangle$ is in `InverseShort-Lex`, since it has no simplifications. What if we multiply it on the left by $\langle 3\rangle$ to get

$$\langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\ ?$$

This reduces to its `InverseShortLex` equivalent through this chain of transformations:

$$\begin{aligned}\langle \mathbf{3}\rangle\langle \mathbf{1}\rangle\langle 2\rangle\langle 3\rangle &= \langle \mathbf{1}\rangle\langle \mathbf{3}\rangle\langle 2\rangle\langle 3\rangle\\ \langle 1\rangle\langle \mathbf{3}\rangle\langle \mathbf{2}\rangle\langle \mathbf{3}\rangle &= \langle 1\rangle\langle \mathbf{2}\rangle\langle \mathbf{3}\rangle\langle \mathbf{2}\rangle\end{aligned}$$

but the first step is not an `InverseShortLex` simplification. Nor is there any chain of `Inverse-ShortLex` simplification which will carry out the reduction.

6. Reflection in the InverseShortLex tree

Recall that for w in W its `InverseShortLex` normal form is $NF(w)$. Denote concatenation of words by $\bullet$.

As already suggested, the `InverseShortLex` language defines a tree whose edges are labeled by elements of S. Its nodes are the elements of W, and there exists an edge $x \xrightarrow{t} y$ if $y = xt > x$ and $NF(y) = NF(x)\bullet t$. Or, equivalently, if $y = xt > x$ and t is the least element of S such that $yt < y$. The root of the tree is the identity element of W, and from the root to any element w of W there exists a unique path whose edges trace out the `InverseShortLex` expression for w.

What is the effect of reflection on this tree? In other words, suppose we have an edge $x \xrightarrow{t} y$, and that s is an element of S. Under what circumstances is there an edge $sx \xrightarrow{t} sy$ in the `InverseShortLex` tree?

Theorem. *Suppose that $x \xrightarrow{t} y$ is an edge in the* `InverseShortLex` *tree and that s is an element of S.*

(a) If yC does not have a face contained in the reflection hyperplane $\alpha_s = 0$ then there will also be an edge $sx \xrightarrow{t} sy$ in the `InverseShortLex` *tree.*

(b) If yC has a face labeled by u in the reflection hyperplane $\alpha_s = 0$, then there will exist an edge $sx \xrightarrow{t} sy$ in the `InverseShortLex` *tree except when $u \le t$.*

In particular, most edges in the `InverseShortLex` tree will certainly be preserved under reflection by s, because relatively few chambers will lie next to the root plane $\alpha_s = 0$.

The theorem's formulation masks an important dichotomy. In (b), the case where $u = t$ is that where xC and yC share a face contained in the root plane $\alpha_s = 0$. Reflection by s simply interchanges xC and yC, or in other words $sx = xt$. We have what is called in the theory of Coxeter groups an **exchange**.

If $u < t$, let $z = sy = yu$. Reflection by s transforms yC into zC. In other words, the edge $y \xrightarrow{u} z$ is an example of the first case. The `InverseShortLex` edge into zC comes from yC across the root hyperplane instead of from sx.

PROOF. Suppose $x \xrightarrow{t} y$ to be an edge in the `InverseShortLex` tree that disappears under reflection by t. In other words, t *is* the least element of S labeling a face of yC separating it from C, but t is *not* the least element of S labeling a face of syC separating it from C.

One of two things can go wrong: (i) The face of syC labeled by t does not separate syC from C; or (ii) it does separate, but it is not the least. We have to analyze what happens to the faces of yC separating it from C upon reflection by s.

If F is a face of yC separating yC from C, then sF is a face of syC separating syC from sC. The union of C and sC is a convex set, and its two components are separated by the single face where $\alpha_s = 0$. Therefore the faces separating syC from C will be the same as the reflections of those separating yC from C, except where yC and syC have a common face in $\alpha_s = 0$. If $sy > y$ and yC has a face in $\alpha_s = 0$, then yC and syC will share a face which also separates syC from C.

Therefore in case (i) above $sy < y$. Furthermore, even in this case only one separating face disappears under reflection by s, and that is the one in $\alpha_s = 0$—the face of syC labeled by t must be that lying in $\alpha_s = 0$. In this case, therefore, xC lies on the positive side of $\alpha_s = 0$ and yC lies on the other, and their common face lies in the root plane $\alpha_s = 0$.

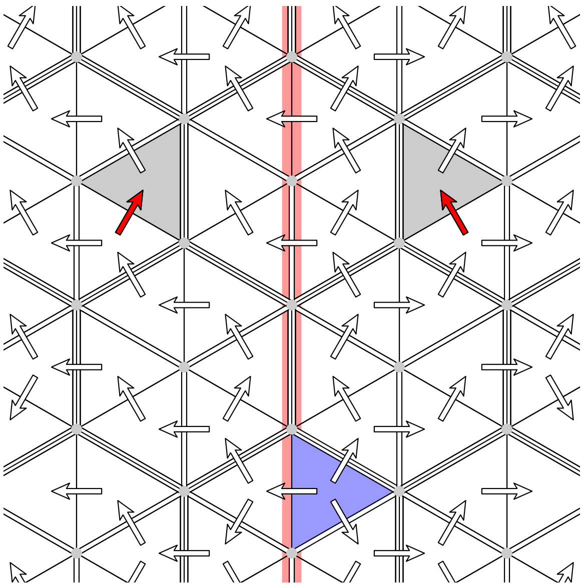


Figure 6a. *If yC does not lie next to the reflection hyperplane, then the faces separating syC from C are the images under s of those separating yC from C.*

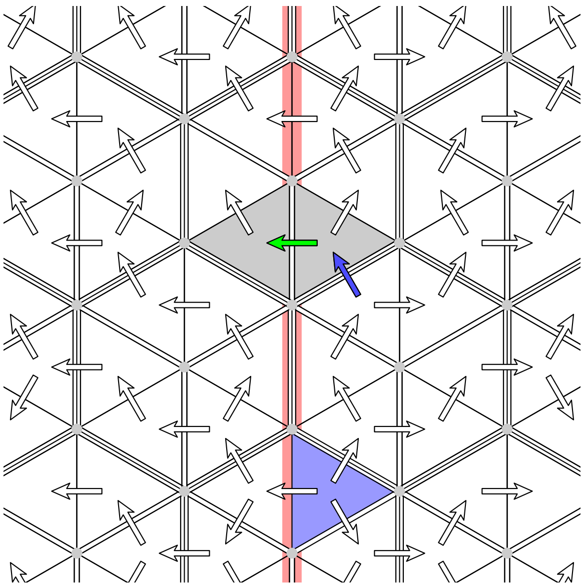


Figure 6b. *If a face of yC does lie in the reflection hyperplane, then its* `ISL` *link may be affected under reflection.*

Suppose that the face labeled by t of syC does separate syC from C, but that it is not least. Let the face which is least be labeled by $u < t$. If the face labeled by u also separated yC from C then the edge $x \xrightarrow{t} y$ would not be in the `InverseShortLex` tree. Therefore this does not happen. By the same reasoning as in the previous paragraph, this means that the face labeled by u lies in the root plane $\alpha_s = 0$.$\square$

A simple analysis will show that all possible cases where reflection does not preserve an edge arise from one of two basic configurations: (i) The case $x = 1$. No edges leading out from C itself are preserved under reflection by any t in S. (ii) We have a succession of edges $x \xrightarrow{t} y$ and $y \xrightarrow{u} z$ where $u < t$, and the face of yC labeled by u crosses from one side of $\alpha_s = 0$ to the other. In this second case, neither of the two edges is preserved under reflection by s. In the second case the node y is called an **exchange node** for s in the `InverseShortLex` tree. To deal with both cases uniformly, we also call the identity element of W an exchange node. The terminology comes from the fact that if y is an exchange node then $sy = yu$—i.e. s and u are exchanged.

Thus y is an exchange node for s if $y = 1$ or if $NF(y) = s_1 \ldots s_n$ and $NF(sy) = NF(y) \bullet t$ with $t < s_n$. I will call t the **exchange token**. The exchange node y will be called **primitive** if there is no other exchange node among the $y_i = s_1 \ldots s_i$ with $i < n$. This means that all the expressions $ss_1 \ldots s_m$ are in normal form for $m < n$.

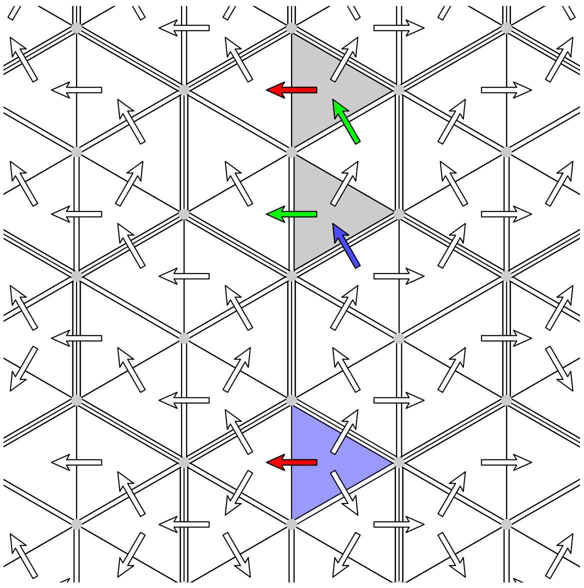


Figure 7. *Some of the exchange nodes for reflection by* $\langle 1 \rangle$ *in* $\widetilde{A}_2$*. Links modified under reflection are colored. The fundamental chamber is at the bottom.*

This theorem has immediate consequences for calculating the normal form of sx, given the normal form of x.

Theorem (`InverseShortLex` exchange). *Suppose x to have the normal form $x = s_1 \ldots s_n$ and let $x_i = s_1 \ldots s_i$ for each i. Suppose that x_m is the last exchange node for s among the x_i, with exchange token t. Then*

$$NF(sx) = \begin{cases} s_1 s_2 \ldots s_m s_{m+2} \ldots s_n & \textit{if } t = s_{m+1} \\ s_1 s_2 \ldots s_m t s_{m+1} s_{m+2} \ldots s_n & \textit{otherwise} \end{cases}$$

PROOF. Since $sx_m = x_m t$ and in the first case $ts_{m+1} = 1$, the products on left and right are identical in the group. It remains to see that they are in `InverseShortLex`, or that each symbol in one of the strings corresponds to an edge in the `InverseShortLex` tree. There is no problem for the initial segment $s_1 \ldots s_m$.

When $t = s_{m+1}$, $sx_m = x_{m+1}$ and the chain

$$x_m \xrightarrow{s_{m+2}} sx_{m+2} \xrightarrow{s_{m+3}} sx_{m+3} \ \ldots$$

is the reflection under s of the `InverseShortLex` chain

$$x_{m+1} \xrightarrow{s_{m+2}} x_{m+2} \xrightarrow{s_{m+3}} x_{m+3} \ \ldots$$

There are no exchange nodes here since all terms in the first chain lie in $\alpha_s < 0$. So `InverseShortLex` edges are preserved.

When $t \neq s_{m+1}$, there is certainly an edge from x_m to $x_{m+1}t$ in the `InverseShortLex` tree, by definition of t. The rest of the chain is the reflection under t of the chain

$$x_m \xrightarrow{s_{m+1}} x_{m+1} \ \ldots$$

and since there are no exchange nodes here by choice of x_m, `InverseShortLex` links are again preserved.◻

One simple consequence of the theorem is this, equivalent to Proposition 3.4 of **Brink and Howlett (1993)**:

Corollary. *If $sw > w$, then the normal form of sw is obtained from that of w by insertion of a single element of S. If $sw < w$, then the normal form of sw is obtained from that of w by deletion of a single element.*

Thus the theorem itself is an explicit version of the familiar exchange lemma for Coxeter groups.

A few examples are illustrated in the following figures, which ought to explain the proof better than words can do:

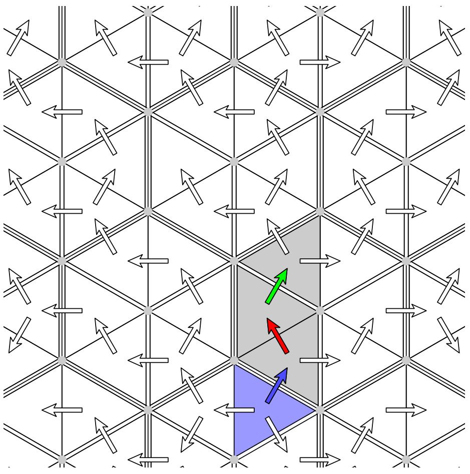


Figure 8a. *The element* $w = \langle 3\rangle\langle 1\rangle\langle 2\rangle$.

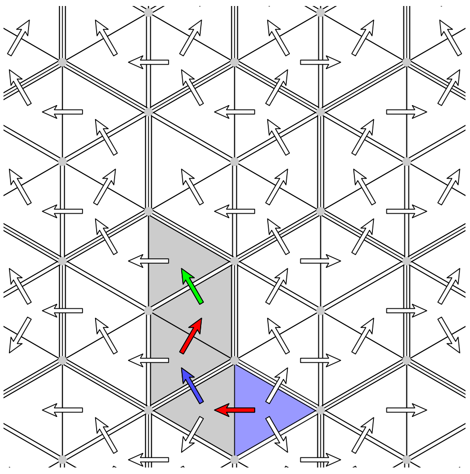


Figure 8b. *The element* $\langle 1\rangle\, w = \langle \mathbf{1}\rangle\langle 3\rangle\langle 1\rangle\langle 2\rangle$.

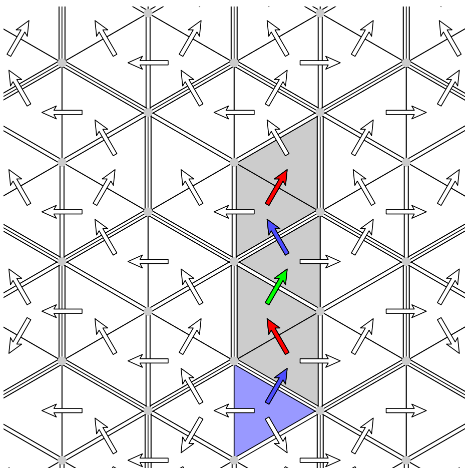


Figure 9a. *The element* $w = \langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\langle 1\rangle$.

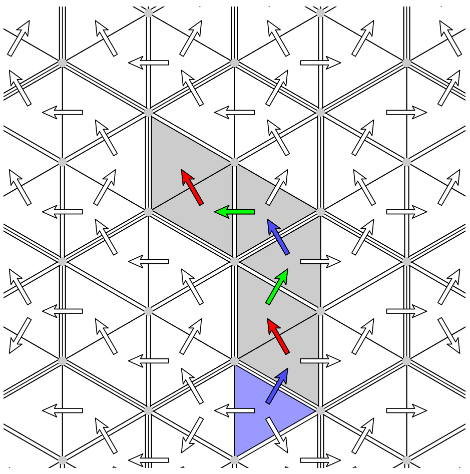


Figure 9b. *The element* $\langle 1\rangle\, w = \langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\langle \mathbf{2}\rangle\langle 1\rangle$.

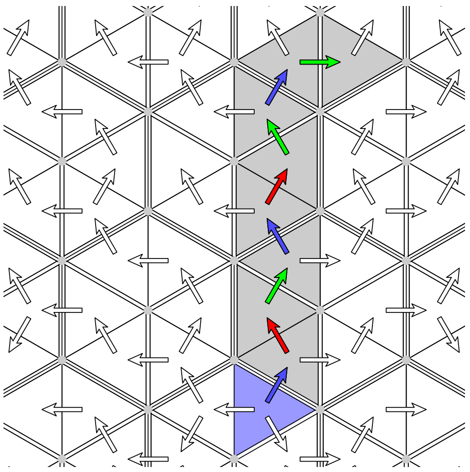


Figure 10a. *The element* $w = \langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\langle 2\rangle$.

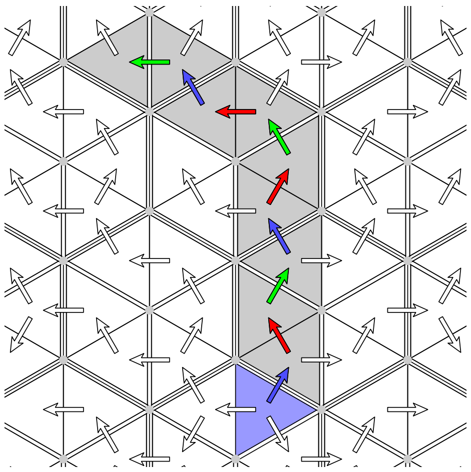


Figure 10b. *The element* $\langle 1\rangle\, w = \langle 3\rangle\langle 1\rangle\langle 2\rangle\langle 3\rangle\langle 1\rangle\langle 2\rangle\langle \mathbf{1}\rangle\langle 3\rangle\langle 2\rangle$.

We can carry out this process in a slightly simplified fashion. We do not have to figure out ahead of time what the last exchange node in $NF(x)$ is, but recognize exchange nodes as we go along, each time carrying out the exchange and starting over again with the remaining terminal string in x.

Theorem. *If x has normal form $s_1 s_2 \dots s_n$ and $x_m = s_1 \dots s_m$ is an exchange node for s with exchange token t, then $NF(sx) = s_1 \dots s_m NF(ty)$ where $y = s_{m+1} \dots s_n$.*

This is immediate from the previous result.

Given this result, we must now ask: *How do we tell when we have a primitive exchange node y? If we have one, how do we compute the exchange token?*

One simple criterion relies on the coset factorization $W = W^T \times W_T$ when T is a subset of S. If we take T to be one of the subsets $T_m = [m, r]$ (r is the rank of W), then this factorization is compatible with `InverseShortLex`:

Proposition. *If $w = xy$ with x in W^{T_m} and y in W_{T_m} then $NF(w) = NF(x)\bullet NF(y)$.*

PROOF. Recall that in these circumstances $\ell(w) = \ell(x) + \ell(y)$. The proof is by induction on $\ell(y)$.□

This factorization extends to give

$$W = W_r^{[r]} \times \dots W_1^{[1]}$$

where $W_m^{[m]}$ is the subset of the elements w in the group generated by the first m elements of S whose `InverseShortLex` word ends in m. If w lies in $W_m^{[m]}$ for some m I call it from now on **distinguished**. This means that $NF(w) = s_1 \dots s_n$ with $s_n \geq s_i$ for all i.

Lemma. *Suppose y to be distinguished, $NF(y) = s_1 \dots s_n$, and s in S. Exactly one of these holds:*

(a) *$sy > y$ and $NF(sy)$ is obtained from $NF(y)$ by insertion somewhere before s_n;*
(b) *$sy > y$ and $NF(sy) = NF(y)\bullet t$ with $t < s_n$;*
(c) *$y = s$ and $sy = 1$;*
(d) *$sy < y$ and $NF(sy)$ is obtained from $NF(y)$ by a deletion before s_n.*

PROOF. If $sy > y$ then $NF(sy)$ is obtained from $NF(y)$ by insertion. It has to be shown that if the insertion puts t at the end then $t < s_n$. This is clear from the `InverseShortLex` exchange theorem. Similarly, if $sy < y$ then $NF(sy)$ is obtained from $NF(y)$ by deletion. In this case it has to be shown that either $y = s$ or the deletion does not take place at the end. But if the deletion takes place at the end and $y \neq s$ then

$$ss_1 \dots s_n = s_1 \dots s_{n-1}, \quad ss_1 \dots s_{n-1} = s_1 \dots s_n = y$$

which contradicts the hypothesis that s_n is the least s in S such that $sy < y$.□

Lemma. *If $w \neq 1$ is a primitive exchange node for s and $NF(w) = s_1 \dots s_n$, then $s_n \geq s_i$ for all i and $s_n \geq s$.*

PROOF. Suppose $w \neq 1$ with normal form $s_1 \dots s_n$ is a primitive exchange node for $s_0 = s$. This means that (i) $s_0 s_1 \dots s_{n-1}$ is in normal form; (ii) $NF(s_0 w) = NF(w) \bullet t$ with $t < s_n$. It is to be shown that $s_n \geq s_i$ for all i.

If not, suppose that $s_n < s_i$ for some i, so that the maximum element occurs somewhere than at the right end. Let s_m to be the last occurrence of the maximum, with $m < n$, so that $s_m > s_k$ for all $k > m$. The element $x = s_0 s_1 \dots s_m$ is in normal form, because $s_0 s_1 \dots s_{n-1}$ is assumed to be in normal form, and similarly $y = s_{m+1} \dots s_n$ is in normal form since $s_1 \dots s_n$ is assumed to be in normal form. But the element $s_0 w$ has the coset decomposition

$$s_0 w = xy, \quad NF(x) = s_0 s_1 \dots s_m, \quad y = s_{m+1} \dots s_n \ .$$

Then $NF(w) = NF(x) \bullet NF(y)$, contradicting that the normal form of sw is wt.□

7. Du Cloux's trick

So now we are reduced to the following problem: We are given $w = s_1 s_2 \dots s_n$ in normal form, we know that $ss_1 \dots s_{n-1}$ is also in normal form, and we want to compute $NF(sw)$. We also know that s_n is maximal among s and the s_i. There are only two possibilities: • sw is already in normal form, or • $sw = wt$ with t less than the terminal element in w. We must decide which, and in the second case we must determine t.

We can do what we want to easily enough in two very special cases: • we have $s = s_1 = w$, or • we have $s = s_2$ and sw is one half of a braid relation $s_2 s_1 s_2 \dots$ where the other half is in `InverseShortLex` form. These we can handle easily. In the first case, we can even stop all further work; cancelling ss gives us exactly the `InverseShortLex` form we are looking for. From now on, we shall assume we are not dealing with either of these two cases. In particular, we may assume that $n \geq 2$.

This is exactly where a marvelous trick due to du Cloux (Lemma 3.1 of **du Cloux (1999)**, explained in a more leisurely fashion in §3 of the unpublished note **du Cloux (1990)**) comes in. du Cloux's remarkable idea is that we can tell what happens by finding the `InverseShortLex` form of $s_1 s s_1 s_2 \dots s_{n-1}$. On the face of it, this is just as difficult as the problem of finding $NF(ss_1 \dots s_n)$. But in fact either the new maximum element in this string occurs before s_{n-1}, in which case because of coset factorization we have broken our problem into one involving two smaller expressions, or s_{n-1} is the new maximum, in which case we have decreased the maximum without lengthening the expression involved. In either case, we are led by recursion to a strictly simpler problem.

So let $y = s_1 s_2 \ldots s_{n-1}$. Calculate the normal form of

$$s_1 s y = s_1 s s_1 s_2 \ldots s_{n-1} \ .$$

There are a couple of possible outcomes:

(1) $s_1 sy < sy$. We can calculate $s_1 s y s_n = s_1 s x$ by recursion, since the length of $s_1 sy$ is that of y, one less than that of x. It turns out that sx is distinguished if and only if $s_1 sx$ is. For if sx is distinguished then by a Proposition above so is $s_1 sx$ since $s_1 sx < sx$ as well. And if sx is not distinguished, then

$$\begin{aligned} sx &= xt \\ s_1 sx &= s_1 xt \\ &= s_1 s_1 s_2 \ldots s_{n-1} s_n t \\ &= s_2 \ldots s_n t \end{aligned}$$

is not distinguished. Furthermore, we can find the t we are looking for by calculating $s_1 sx$.

(2) $s_1 sy > sy$. In this case, either (a) sx is distinguished or (b) we are looking at a braid relation in a rank two group. For say $s_1 s s_1 \ldots s_{n-1}$ is reduced but sx is not distinguished. Then $sx = xt$ and hence

$$s_1 s s_1 \ldots s_{n-1} s_n = s_1 s_1 \ldots s_{n-1} s_n t = s_2 \ldots s_n t \ .$$

On the other hand, $s_1 s s_1 s_2 \ldots s_{n-1}$ is reduced but $s_1 s s_1 \ldots s_{n-1} s_n$ is not, which means that we must have a deletion when we multiply $s_1 s s_1 s_2 \ldots s_{n-1}$ on the right by s_n. Since $s s_1 \ldots s_n$ is reduced, the only exchange possible is with the first s_1:

$$s_1 s s_1 s_2 \ldots s_{n-1} s_n = s s_1 s_2 \ldots s_{n-1} \ .$$

For the one element we now have two expressions:

$$s_1 s s_1 s_2 \ldots s_{n-1} s_n = s s_1 s_2 \ldots s_{n-1} = s_2 \ldots s_n t$$

Both are normal forms. But since normal forms are unique, these must match element by element: $s = s_2$, $s_1 = s_3$, etc.$\square$

It is instructive to see how du Cloux's procedure works out in the example we looked at in examining Tits' algorithm. We want to find sw where $s = \langle 3 \rangle$ and $w = \langle 1 \rangle \langle 2 \rangle \langle 3 \rangle$. We read from left to right in w: getting normal forms $\langle 3 \rangle \langle 1 \rangle$ and $\langle 3 \rangle \langle 1 \rangle \langle 2 \rangle$ before arriving at w itself, which is a possible primitive exchange node for s. Applying du Cloux's trick, we have to calculate first the normal form of $\langle 1 \rangle \langle 3 \rangle \langle 1 \rangle \langle 2 \rangle$. We apply a braid relation exchange $\langle 1 \rangle \langle 3 \rangle = \langle 3 \rangle \langle 1 \rangle$ to get $\langle 3 \rangle \langle 1 \rangle \langle 1 \rangle \langle 2 \rangle = \langle 3 \rangle \langle 2 \rangle$ for this. Then we calculate the normal form of $\langle 3 \rangle \langle 2 \rangle \langle 3 \rangle$, getting an exchange to $\langle 2 \rangle \langle 3 \rangle \langle 2 \rangle$. We read off $t = \langle 2 \rangle$, then $sw = wt$ with $t = \langle 2 \rangle$.

8. The full algorithm

There are three basic function procedures, linked by a somewhat intricate recursion:

- `NF(x,y)`. Here x and y are arrays of elements of S, and y is assumed to be in normal form. The return value is the array of the full product in normal form.
- `NF(s,w)`. Here s is a single generator, and again y is an array in normal form. The return value is the normal form of the product.
- `Exchange(s,w)`. Here s is a single generator,

$$w = (s_1, s_2, \dots, s_n)$$

  an array in normal form, and it is assumed that $ss_1 \dots s_{n-1}$ is also in normal form. The return value is either a new dummy generator, say $\langle 0 \rangle$, if the normal form of sw is $ss_1 \dots s_n$, or otherwise the generator t such that $NF(sw) = w \bullet t$.

In more detail:

• `NF(x,y)`

Suppose $x = (s_1, \dots, s_n)$. Recall that y is in normal form. For $i = n, \dots, 1$ let

$$y := NF(s_i, y)$$

Then return the final value of y.

• `NF(s,w)`

The rough idea here is that we scan left to right for possible primitive exchange nodes, and when we find them we call `Exchange(s,w)`, which for the most part carries out du Cloux's calculations. Depending on the outcome, we either continue on or restart the scan with new data.

As we scan we keep track of the current left-hand end ℓ of what possible exchange we are dealing with; the index of the current element r to be scanned; the current maximum element σ; and the current value of s. At each entry to the principal loop: $1 \le \ell \le r \le n$; $NF(sw) = s_1 \dots s_{\ell-1} NF(ss_\ell \dots s_n)$; σ is the maximum of s and the s_i with $\ell \le i < r$;

$$NF(ss_1 \dots s_{r-1}) = [s_1, \dots, s_{\ell-1}, s, s_\ell, \dots, s_{r-1}] \ ;$$

and we are in the process of finding $NF(ss_\ell \dots s_r)$. Start off by setting $\sigma = s$, $\ell = r = 1$.

```
While $r \le n$:
    Let $c = s_r$ and increment $r$.
    If $c < \sigma$ just re-enter the loop.
    Otherwise $c \ge \sigma$ and we are at a possible primitive exchange node. Set $\sigma = c$.
    Let $t$ be the return value of Exchange($s, [s_\ell, \dots, s_{r-1}]$).
    If $t = \langle 0 \rangle$ (the dummy generator), there was no exchange. Reenter the loop.
    Else there are two possibilities, a deletion or an exchange.
    If $s = s_\ell$, there is a deletion at $s_\ell$. Return immediately with
```

$$NF(sw) = [s_1, \dots, s_{\ell-1}, s_{\ell+1}, \dots, s_n] \ .$$

Otherwise, there is an exchange. Set $s = t$, $\ell = r$, $\sigma = t$, and re-enter the loop.

After exiting from the loop, we return

$$NF(sw) = s_1 \dots s_{\ell-1} s s_\ell \dots s_{r-1} \ .$$

• `Exchange(s,w)`

Here w is given in normal form $w = s_1 \dots s_n$ with $n > 0$, $ss_1 \dots s_{n-1}$ is in normal form, and $s_n \geq s$, all s_i. There are three distinct cases:

(i) If $\ell(w) = 1$ and $s_1 = s$, this is an exchange. Return s.

(ii) If s and w lie in a subgroup with two generators from S and sw is the left hand side of a braid relation $sw = wt$ with the right hand side in `InverseShortLex`, this also is an exchange. Return t.

(iii) Otherwise, let $y = ss_1 \dots s_{n-1}$ and find $\mathtt{NF}(s_1, y)$ (a recursive call). (a) If $s_1 y < y$, then its normal form is obtained by a deletion from that of y. Say $s_1 y = s \dots \widehat{s_m} \dots s_{n-1}$. We can calculate the normal form of $s_1 y s_n$ by a recursion call to find

$$\mathtt{NF}([s_1, s, s_1, \dots, s_{m-1}], [s_{m+1}, \dots, s_n]) \ .$$

If this is distinguished then so is sw. If it is not, and $s_1 y s_n = zt$ with $t < s_n$, then we have an exchange $sw = wt$. (b) If $s_1 y > y$ then sw is distinguished—there is no exchange.

In any practical implementation of these, it will be efficient to keep a registry of exchanges and potential exchanges, starting with the braid relations and building the rest of the registry as calculations proceed.

9. The minimal roots of Brink and Howlett

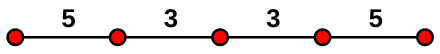
du Cloux's idea is clever, but the recursion involved in it means it is not likely to be efficient. It would be good if there were some other, more efficient way, to recognize primitive exchanges. Ideally, we would read the normal form of w from left to right, adjusting something as we go, and recognize without recalculation when we arrive at a primitive exchange node. In fact, there is such a procedure. It involves the notion of **minimal root** of **Brink and Howlett (1993)**.

A minimal root is a positive root (half-space) which does not contain the intersection of $\mathcal{C}$ and another positive root—in the terminology of Brink and Howlett does not **dominate** another positive root. It corresponds to a root hyperplane which is not walled off in $\mathcal{C}$ from C by another root hyperplane. For finite Coxeter groups all root hyperplanes intersect in the origin, there is no dominance, and all positive roots are minimal. For affine Weyl groups, the minimal roots are the positive roots in the corresponding finite root system, together with the $-\lambda + 1$ where λ is positive in the finite root system. For other Coxeter groups, they are more interesting, less well understood, and probably more important. The principal theorem of Brink and Howlett, and in my opinion one of the most remarkable facts about general Coxeter groups, is that the number of minimal roots is finite. That this has not been applied much, for example to the theory of Kac-Moody algebras, just shows how little we know about general Kac-Moody algebras. In addition, Brink and Howlett proved that if λ is a minimal root and s an element of S, then exactly one of these is true: (i) $\lambda = \alpha_s$ and $s\lambda < 0$; (ii) $s\lambda$ dominates α_s; (iii) $s\lambda$ is again a minimal root. I interpret this by introducing the **extended set** of minimal roots, the usual ones together with $\oplus$ and $\ominus$. I define the **minimal root reflection table** to be that which tabulates the $s\lambda$ for λ an extended minimal root (setting $s\lambda = \ominus$ in case (i), $s\lambda = \oplus$ in case (iii)). We incorporate also the simple rules $s\ominus = \ominus$, $s\oplus = \oplus$. This minimal root reflection table is one of the most important tools for computational work with Coxeter groups.

For a distinguished $w = s_1 \dots s_n$ primitive with respect to s we can calculate $w^{-1}\alpha_s$ from the table. *The element w will be an exchange node if and only if $w^{-1}\alpha_s = \alpha_u$ with $u < s_n$, and the exchange token will be u.* This is explained in **Casselman (1994)**.

Of course we can calculate $w^{-1}\alpha_s$ as we read the `InverseShortLex` expression for w from left to right, since $(w_i s_{i+1})^{-1}\alpha_s = s_{i+1} w_i^{-1}\alpha_s$. Therefore, in order to calculate products efficiently, we want to calculate the minimal root reflection table. This involves constructing a list of all minimal roots, and calculating the effect of reflections on them. The algorithm I now use pretty much follows that devised first by Howlett, and explained in more detail in **Casselman (1994)**. There is an important difference, however: in the earlier routine, floating point numbers were used to check an inequality involving real roots, in order to decide when dominance occurred. In my current programs, I check dominance by checking whether a certain element in the Coxeter group has finite order or not. In doing this, products are calculated using the algorithm described earlier. The lengths of the elements encountered are quite large, and it is perhaps surprising that du Cloux's recursion is indeed practical enough to allow that check. In other words, although it is not good enough to carry out serious calculations on its own, it is good enough to bootstrap itself up to a more efficient algorithm. For crystallographic groups one can also legitimately use exact integer arithmetic and the geometric algorithm to construct the minimal root reflection table. It is interesting to see that the general method, even on large groups like $\widetilde{E}_8$, is only about 10 times slower.

I thought of putting in a table of program run times for various systems, but it would be hardly worth it—even for quite large groups like the affine E_8, it takes less than a second or two to build the minimal root table. The toughest group I have looked at so far is that with Coxeter graph



With this group, construction of the minimal root table encounters about 20,000 exchange nodes. In my programs these are all registered in a dictionary as they are found, for subsequent look-up. This takes up a lot of space, and in finding a practical implementation that would deal with this particular group I had to sacrifice a small amount of speed. The number of minimal roots for this group, incidentally, is 135, and on one machine takes 2.3 seconds to construct the minimal root reflection table. By comparison, the group affine E_8 has 240 minimal roots, encounters about 1,000 exchange nodes, and on the same machine takes 0.3 seconds. In practice, the group '5335' marks a kind of frontier. Although one can manage to construct lots of interesting structures for it, doing really interesting computations, for example of non-trivial Kazhdan-Lusztig polynomials, seems to be definitely beyond present capability.

10. References

1. A. Aho and M. Corasick, 'Efficient string matching: an aid to bibliographic research', *Comm. ACM* **18** (1975), 333–340.

2. Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullmann, **Compilers—Principles, Techniques, and Tools**, Addison-Wesley, 1986.

3. N. Bourbaki, *Groupes et algèbres de Lie, Chapitres 4,5, et 6*, Hermann, Paris, 1968.

4. Brigitte Brink and Robert Howlett, 'A finiteness property and an automatic structure for Coxeter groups', *Math. Ann.* **296** (1993), 179–190.

5. W. Casselman, 'Automata to perform basic calculations in Coxeter groups', in *Representations of groups*, *CMS Conference Proceedings 16*, A.M.S., 1994.

6. W. Casselman, 'Multiplication in Coxeter groups—applet and source code', at
`http://www.math.ubc.ca/people/faculty/cass/java/coxeter/multiplication`

7. Fokko du Cloux, 'Un algorithme de forme normale pour les groupes de Coxeter', preprint, Centre de Mathématiques à l'École Polytechnique, 1990.

8. Fokko du Cloux, 'A transducer approach to Coxeter groups', *J. Symb. Computation* **27** (1999), 311–324.

9. Robert Howlett, 'Introduction to Coxeter groups', preprint, 1997. Available from the web site

`http://www.maths.usyd.edu.au`

of the Mathematics and Statistics Department at the University of Sydney, N.S.W.

10. James E. Humphreys, *Reflection groups and Coxeter groups*, Cambridge University Press, 1990.

11. D. E. Knuth, 'On the translation of languages from left to right', *Information and Control* **8:6** (1965), 607–639.

12. J. Tits, 'Le problème des mots dans les groupes de Coxeter', *Symposia Math.* **1** (1968), 175–185.

13. E. Vinberg, 'Discrete linear groups generated by reflections', *Mathematics of the USSR—Izvestia* **5** (1971), 1083–1119.