Technische Universität Braunschweig
Institut Computational Mathematics

Posets of twisted involutions in Coxeter groups

Christian Hoffmeister

August 10, 2012

Contents

1 Coxeter groups

A Coxeter group, named after Harold Scott MacDonald Coxeter, is an abstract group generated by involutions with specific relations between these generators. A simple class of a Coxeter groups are the symmetry groups of regular polyhedras in the Euclidean space.
The symmetry group of the square for example can be generated by two reflections s, t, whose stabilized hyperplanes enclose an angle of $\pi/4$. In this case the map st is a rotation in the plane by $\pi/2$. So we have $s^2 = t^2 = (st)^4 = \mathrm{id}$. In fact this reflection group is determined up to isomorphy by s, t and these three relations [4, Theorem 1.9]. Furthermore it turns out, that the finite reflection groups in the Euclidean space are precisely the finite Coxeter groups [4, Theorem 6.4].
In this chapter we will compile some basic facts on Coxeter groups, based on [4].

1.1 Introduction to Coxeter groups

Definition 1.1. Let $S = \{s_1, \ldots, s_n\}$ be a finite set of symbols and

$$R = \{m_{ij} \in \mathbb{N} \cup \infty : 1 \leq i, j \leq n\}$$

a set numbers (or ∞) with $m_{ii} = 1$, $m_{ij} > 1$ for $i \neq j$ and $m_{ij} = m_{ji}$. Then the free represented group

$$W = \langle S \mid (s_i s_j)^{m_{ij}} \rangle$$

is called a ***Coxeter group*** and (W, S) the corrosponding ***Coxeter system***. The cardinality of S is called the ***rank*** of the Coxeter system (and the Coxeter group).

From the definiton we see, that Coxeter groups only depend on the cardinality of S and the relations between the generators in S. A common way to visualize this information are Coxeter graphs.

Definition 1.2. Let (W, S) be a Coxeter system. Create a graph by adding a vertex for each generator in S. Let $(s_i s_j)^m = 1$. In case $m = 2$ the two corrosponding vertices have no connecting edge. In case $m = 3$ they are connected by an unlabed edge. For $m > 3$ they have an connecting edge with label m. This graph we call the ***Coxeter graph*** of our Coxeter system (W, S).

Definition 1.3. For an arbitrary element $w \in W$, (W, S) a Coxeter system, we call a product $s_{i_1} \cdots s_{i_n} = w$ of generators $s_{i_1} \ldots s_{i_n} \in S$ an ***expression*** of w. Any expression that can be obtained from $s_{i_1} \cdots s_{i_n}$ by omitting some (or all) factors is called a ***subexpression*** of w.

The present relations between the generators of a Coxeter group allow us to rewrite expressions. Hence an element $w \in W$ can have more than one expression. Obviously any element $w \in W$ has infinitly many expressions, since any expression $s_{i_1} \cdots s_{i_n} = w$ can be extended by applying $s_1^2 = 1$ from the right. But there must be a smallest number of generators needed to receive w. For example the neutral element e can be expressed by the empty expression. Or each generator $s_i \in S$ can be expressed by itself, but any expression with less factors (i.e. the empty expression) is unequal to s_i.

Definition 1.4. Let (W, S) be a Coxeter system and $w \in W$ an element. Then there are some (not neseccarily distince) generators $s_i \in S$ with $s_1 \cdots s_r = w$. We call r the ***expression length***. The smallest number $r \in \mathbb{N}_0$ for that w has an expression of length r is called the ***length*** of w and each expression of w, that is ob minimal length, is called ***reduced expression***. The map

$$l : W \to \mathbb{N}_0$$

that maps each element in W to its length is called ***length function***.

Definition 1.5. Let (W, S) be a Coxeter system. We define

$$D_R(w) := \{s \in S : l(ws) < l(w)\}$$

as the ***right descending set*** of w. The analogue left version

$$D_L(w) := \{s \in S : l(sw) < l(w)\}$$

is called ***left descending set*** of w. The right descending set will also just be called ***descending set*** of w.

The next lemma yields some useful identities and relations for the length function.

Lemma 1.6. *Let (W, S) be a Coxeter system, $s \in S$, $u, w \in W$ and $l : W \to \mathbb{N}$ the length function. Then*

1. $l(w) = l(w^{-1})$,
2. $l(w) = 0$ *iff* $w = e$,
3. $l(w) = 1$ *iff* $w \in S$,
4. $l(uw) \leq l(u) + l(w)$,
5. $l(uw) \geq l(u) - l(w)$ *and*
6. $l(ws) = l(w) \pm 1$.

Proof. See [4, Section 5.2]. □

1.2 Exchange and Deletion Condition

We now obtain a way to get a reduced expression of an arbitrary element $s_1 \cdots s_r = w \in W$. But first we define what a reflection is. Any element $w \in W$ that is conjugated to an generator $s \in S$ is called ***reflection***. Hence the set of all reflections in W is

$$T = \bigcup_{w \in W} wSw^{-1}.$$

Theorem 1.7 (Strong Exchange Condition)**.** *Let (W, S) be a Coxeter system, $w \in W$ an arbitrary element and $s_1 \cdots s_r = w$ with $s_i \in S$ a not neseccarily reduced expression for w. For each reflection $t \in T$ with $l(wt) < l(w)$ there exists an index i for which $wt = s_1 \cdots \hat{s}_i \cdots s_r$, where $\hat{s}_i$ means omission. In case we started from a reduced expression, then i is unique.*

Proof. See [4, Theorem 5.8]. □

The Strong Exchange Condition can be weaken when insisting on $t \in S$ to receive the following corollary.

Corollary 1.8 (Exchange Condition)**.** *Let (W, S) be a Coxeter system, $w \in W$ an arbitrary element and $s_1 \cdots s_r = w$ with $s_i \in S$ a not neseccarily reduced expression for w. For each generator $s \in S$ with $l(ws) < l(w)$ there exists an index i for which $ws = s_1 \cdots \hat{s}_i \cdots s_r$, where $\hat{s}_i$ means omission.*

Proof. Directly from Strong Exchange Condition. □

The Exchange Condition immediatly yields another corollary for Coxeter groups:

Corollary 1.9 (Deletion Condition)**.** *Let (W, S) be a Coxeter system, $w \in W$ and $w = s_1 \cdots s_r$ with $s_i \in S$ a unreduced expression of w. Then there exist two indices $i, j \in \{1, \cdots, r\}$ with $i < j$, such that $w = s_1 \cdots \hat{s}_i \cdots \hat{s}_j \cdots s_r$, where $\hat{s}_i$ and $\hat{s}_j$ mean omission.*

Proof. Since the expression is unreduced there must be an index j for that the twisted length shrinks. That means for $w' = s_1 \cdots s_{j-1}$ is $l(w's_j) < l(w')$. Using the Exchange Condition we get $w's_j = s_1 \cdots \hat{s}_i \cdots s_{j-1}$ yielding $w = s_1 \cdots \hat{s}_i \cdots \hat{s}_j \cdots s_r$. □

This corollary is called ***Deletion Condition*** and allows us to reduce expressions, i.e. to find a subexpression that is reduced. Due to the Deletion Condition any unreduced expression can be reduced by omitting a even number of generators (we just have to apply the Deletion Condition inductively).

The Strong Exchange Condition, the Exchange Condition and the Deletion Condition, are some of the most powerful tools when investigating properties of Coxeter groups. We can use the second to prove a very handy property of Coxeter groups. The intersection of two parabolic subgroups is again a parabolic subgroup.

Definition 1.10. Let (W, S) be a Coxeter system. For a subset of generators $I \subset S$ we call the subgroup $W_I \leq W$ that is generated by the elements in I with the corrosponding relations a ***parabolic subgroup*** of W.

Lemma 1.11. *Let (W, S) be a Coxeter system and $I, J \subset S$ two subsets of generators. Then $W_I \cap W_J = W_{I \cap J}$.*

Proof. Let $w \in W_{I \cap J}$. Then $w \in W_I$ and $w \in W_J$. To show the other inclusion we induce over the length r. For $r = 0$ we have $w = e$ and so $w \in W_{S'}$ for any $S' \subset S$. So suppose we have proven the assumption for all lengths up to $r - 1$. Let $w \in W_I \cap W_J$ with $l(w) = r$. Then we have two reduced expressions $w = s_1 \cdots s_r = t_1 \cdots t_r$ with $s_i \in I$ and $t_i \in J$. By applying s_r from the right we get $ws_r = s_1 \cdots s_{r-1} = t_1 \cdots t_r s_r$. The expression $t_1 \cdots t_r s_r$ is of length $r - 1$, so Exchange Condition yields $ws_r = s_1 \cdots s_{r-1} = t_1 \cdots \hat{t}_i \cdots t_r$, hence $ws_r \in W_I \cap W_J$. Due to induction we know that $ws_r \in W_{I \cap J}$. **TODO** □

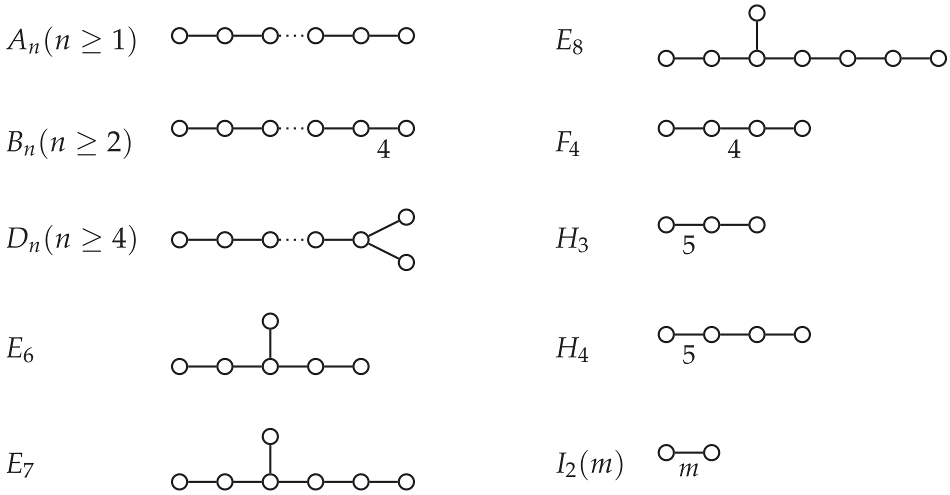

Figure 1.1: All types of irreducible finite Coxeter systems

1.3 Finite Coxeter groups

Coxeter groups can be finite and infinite. A simple example for the former category is the following. Let $S = \{s\}$. Due to definition it must be $s^2 = e$. So W is isomorph to $\mathbb{Z}_2$ and finite. An example for an infinite Coxeter group can be obtained from $S = \{s, t\}$ with $s^2 = t^2 = e$ and $(st)^\infty = e$ (so we have no relation between s and t). Obviously the element st has infinite order forcing W to be infinite. But there are also infinite Coxeter groups without an ∞-relation between two generators. An example for this is W obtained from $S = \{s_1, s_2, s_3\}$ with $s_1^2 = s_2^2 = s_3^2 = (s_1 s_2)^3 = (s_2 s_3)^3 = (s_3 s_1)^3 = e$. But how can it be seen that this W is infinite?
To provide a general answer to this question we fallback to a certain class of Coxeter groups, the irreducible ones.

Definition 1.12. A Coxeter system is called ***irreducible***, if the corrosponding Coxeter graph is connected. Else it is called ***reducible***.

If a Coxeter system is reducible, then its graph has more than one connection component and each connection component corrosponds to a parabolic subgroup of W.

Proposition 1.13. *Let (W, S) be a reducible Coxeter system. Then there exists a partition of S into I, J with $(s_i s_j)^2 = e$ whenever $s_i \in I, s_j \in J$ and W is isomorph to the direct product of the two parabolic subgroups W_I and W_J.*

Proof. See [4, Proposition 6.1]. □

This proposition tells us, that an arbitray Coxeter system is finite iff its irreducible parabolic subgroups are finite. Therefor we can indeed fallback to irreducible Coxeter systems without loss of generality. If we could categorize all irreducible finite Coxeter systems, we could categorize all finite Coxeter systems. This is done by the following theorem:

Theorem 1.14. *The irreducible finite Coxeter systems are exactly the ones in Figure 1.1.*

Proof. [4, Theorem 6.4] □

Finally we can decide with ease, if a given Coxeter system is finite. Take its irreducible parabolic subgroups and check, if each is one of A_n, B_n, D_n, E_6, E_7, E_8, F_4, H_3, H_4 or $I_2(m)$.

1.4 Bruhat ordering

We now investigate ways to partially order the elements of a Coxeter group. Futhermore this ordering should be compatible with the length function. The most useful way to achieve this is the Bruhat ordering [4, Section 5.9].

Definition 1.15. Let M be a set. A binary relation, in this case often denoted as "$\leq$", is called a ***partial order*** over M, if fullfills the following conditions for all $a, b, c \in M$:

1. $a \leq a$, called ***reflexivity***
2. if $a \leq b$ and $b \leq a$ then $a = b$, called ***antisymmetry***
3. if $a \leq b$ and $b \leq c$ then $a \leq c$, called ***transitivity***

In this case $(M, \leq)$ is called a ***poset***. If two elements $a \leq b \in M$ are immediate neighbours, i.e. there is no third element $c \in M$ with $a \leq c \leq b$ we say that b ***covers*** a. A poset is called ***graded poset*** if there is a map $\rho : M \to \mathbb{N}$ so that $\rho(b) - 1 = \rho(a)$ whenever b covers a. In this case ρ is called the ***rank function*** of the graded poset.

Definition 1.16. Let $(M, \leq)$ be a poset. The ***Hasse diagram*** of the poset is the graph obtained in the following way: Add a vertex for each element in M. Then add a directed edge from node a to b whenever b covers a.

Example 1.17. Suppose we have an arbitrary set M. Then the powerset $\mathcal{P}(M)$ can be partially ordered by the subset relation, so $(\mathcal{P}(M), \subseteq)$ is a poset. Indeed this poset is always graded with the cardinality function as rank function. In Figure 1.2 we see the Hasse diagram of this poset with $M = \{x, y, z\}$.

Definition 1.18. Let (W, S) be a Coxeter system and $T = \cup_{w \in W} wSw^{-1}$ the set of all reflections in W. We write $w' \to w$ if there is a $t \in T$ with $w't = w$ and $l(w') < l(w)$. If there is a sequence $w' = w_0 \to w_1 \to \ldots \to w_m = w$ we say $w' < w$. The resulting relation $w' \leq w$ is called ***Bruhat ordering***, denoted as $\mathrm{Br}(W)$.

Lemma 1.19. *Let (W, S) be a Coxeter system. Then $\mathrm{Br}(W)$ is a poset.*

Proof. The Bruhat ordering is reflexive by definition. Since the elements in sequences $e \to w_1 \to w_2 \to \ldots$ are strictly ascending in length, it must be antisymmetric. By concatenation of sequences we get the transitivity. □

$$\begin{tikzcd}
 & \{x,y,z\} & \\
\{x,y\} \arrow[ur] & \{x,z\} \arrow[u] & \{y,z\} \arrow[ul] \\
\{x\} \arrow[u] \arrow[ur] & \{y\} \arrow[ul] \arrow[ur] & \{z\} \arrow[ul] \arrow[u] \\
 & \{\} \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 1.2: Hasse diagram of the set of all subsets of $\{x, y, z\}$ order by the subset relation

What we really want is the Bruhat ordering to be graded with the length function as rank function. By definition we already have $v < w$ iff $l(v) < l(w)$, but its not that obvious that two immediately adjacent elements differ in length by exactly 1. Before lets just mention two other partial orderings, where this property is obvious by definition:

Definition 1.20. Let (W, S) be a Coxeter system. The ordering $\leq_R$ defined by $u \leq_R w$ iff $uv = w$ for some $u \in W$ with $l(u) + l(v) = l(w)$ is called the ***right weak ordering***. The left sided version $u \leq_L w$ iff $vu = w$ is called the ***left weak ordering***.

So lets ensure that the Bruhat ordering is graded as well. For this we need another characterization of the Bruhat ordering with subexpressions. As we will show it is $u \leq w$ iff there is a reduced expression of u that is a subexpression of a reduced expression of w.

Proposition 1.21. *Let (W, S) be a Coxeter system, $u, w \in W$ with $u \leq w$ and $s \in S$. Then $us \leq w$ or $us \leq ws$ or both.*

Proof. We can reduce the proof (**TODO**why?) to the case $u \to w$, i.e. $ut = w$ for a $t \in T$ with $l(v) < l(u)$. Let $s = t$. Then $us \leq w$ and we are done. In case $s \neq t$ there are two alternatives for the lengths. We can have $l(us) = l(u) - 1$ which would mean $us \to u \to w$, so $us \leq w$.
So assume $l(us) = l(u) + 1$. For the reflection $t' = sts$ we get $(us)t' = ussts = uts = ws$. So it is $us \leq ws$ iff $l(us) < l(ws)$. Assume this is not the case. Since we have assumed $l(us) = l(u) + 1$ any reduced expression $u = s_1 \cdots s_r$ for u yields a reduced expression $us = s_1 \cdots s_r s$ for us. With the Strong Exchange Condition we can obtain $ws = ust'$ from us by omitting one factor. This omitted factor cannot be s since $s \neq t$. This means $ws = s_1 \cdots \hat{s}_i \cdots s_r s$ and so $ws = s_1 \cdots \hat{s}_i \cdots s_r$, contradicting to our assumption $l(u) < l(w)$ □

Theorem 1.22. *Let (W, S) be a Coxeter system and $w \in W$ with any reduced expression $w = s_1 \cdots s_r$ and $s_i \in S$. Then $u \leq w$ (in the Bruhat ordering) iff u can be obtained as a subexpression of this reduced expression.*

Proof. TODO ☐

This characterization of the Bruhat ordering is very handy. With it and the following short lemma we will be in the position to show, that $\mathrm{Br}(W)$ is graded with rank function l.

Lemma 1.23. *Let (W, S) be a Coxeter system, $u, w \in W$ with $u < w$ and $l(w) = l(u) + 1$. In case there is a generator $s \in S$ with $u < us$ but $us \neq w$, then both $w < ws$ and $us < ws$.*

Proof. Due to Proposition 1.21 we have $us \leq w$ or $us \leq ws$. Since $l(us) = l(w)$ and $us \neq w$ the first case is impossible. So $us \leq ws$ and because of $u \neq w$ already $us < ws$. In turn, $l(w) = l(us) < l(ws)$, forcing $w < ws$. ☐

Proposition 1.24. *Let (W, S) be a Coxeter system and $u < w$. Then there are elements $w_0, \ldots, w_m \in W$ such that $u = w_0 < w_1 < \ldots < w_m = w$ with $l(w_i) = l(w_{i-1}) + 1$ for $1 \leq i \leq m$.*

Proof. We will induce on $r = l(u) + l(w)$. In case $r = 1$ we have $u = e$ and $w = s$ for an $s \in S$ and are done. So suppose $r > 1$. Then there is a reduced expression $w = s_1 \cdots s_r$ for w. Lets fix this expression. Then $l(ws_r) < l(w)$. Thanks to Theorem 1.22 there must be a subexpression of w with $u = s_{i_1} \cdots s_{i_q}$ for some $i_1 < \ldots < i_q$. We distinguish between two cases:

$u < us$ If $i_q = r$, then $us = s_{i_1} \cdots s_{i_q} s = s_{i_1} \cdots s_{i_{q-1}}$ which is also a subexpression of ws. This yields $u < us \leq ws < w$. Since $l(ws) < r$ there is, by induction, a sequence of the desired form. The last step from ws to w also differs in length by exactly 1, so we are done. If $i_q < r$ then u is itself already a subexpression of ws and we can again find a sequence from u to ws strictly ascending length by 1 in each step and have one last step from ws to w also increasing length by 1.

$us < u$ Then by induction we can find a sequence from us to w, say $us = w_0 < \ldots < w_m = w$, where the lengths of neighboured elements differ by exactly 1. Since $w_0 s = u > us = w_0$ and $w_m s = ws < w = w_m$ there must be a smallest index $i \geq 1$, such that $w_i s < w_i$, which we choose. Suppose $w_i \neq w_{i-1}s$. It is $w_{i-1} < w_{i-1}s \neq w_i$ and due to Lemma 1.23 we get $w_i < w_i s$. This contradicts to the minimality of i. So $w_i = w_{i-1}s$. For all $1 \leq j < i$ we have $w_j \neq w_{j-1}s$, because of $w_j < w_j s$. Again we apply Lemma 1.23 to receive $w_{j-1}s < w_j s$. Alltogether we can construct a sequence

$$u = w_0 s < w_1 s < \ldots < w_{i-1}s = w_i < w_{i+1} < \ldots w_m = w,$$

which matches our assumption. ☐

Corollary 1.25. *Let (W, S) be a Coxeter system and $\mathrm{Br}(W)$ the Bruhat ordering poset of W. Then $\mathrm{Br}(W)$ is graded with $l : W \to \mathbb{N}$ as rank function.*

Proof. Let $u, w \in W$ with w covering u. Then Proposition 1.24 says there is a sequence $u = w_0 < \ldots < w_m = w$ with $l(w_i) = l(w_{i-1}) + 1$ for $1 \leq i \leq m$. Since w covers u it must be $m = 1$ and so $u < w$ with $l(w) = l(u) + 1$. ☐

Theorem 1.26 (Lifting Property). *Let* (W, S) *be a Coxeter system and* $v, w \in W$ *with* $v \leq w$. *Suppose* $s \in S$ *with* $s \in D_R(w)$. *Then*

1. $vs \leq w$,
2. $s \in D_R(v) \Rightarrow vs \leq ws$.

Proof. We use the alternative subexpression characterization of the Bruhat ordering from Theorem 1.22.

1. Since $s \in D_R(w)$ there exists a reduced expression $w = s_1 \cdots s_r$ with $s_r = s$. Due to $v \leq w$ we can obtain v as a subexpression $v = s_{i_1} \cdots s_{i_q}$ from w. If $i_q = r$ then $vs = s_{i_1} \cdots s_{i_q} s = s_{i_1} \cdots s_{i_{q-1}}$ is also a subexpression of w. Else if $i_q \neq r$ then v is a subexpression of $ws = s_1 \cdots s_{r-1}$ and so again vs is a subexpression of $w = s_1 \cdots s_{r-1} s$. In both cases we get $vs \leq w$.
2. If we additionally assume $s \in D_R(v)$ then we can always find a reduced expression $w = s_1 \cdots s_r$ with $s_r = s$ having $u = s_{i_1} \cdots s_{i_q}$ as subexpression with $s_{i_q} = s$. This yields $vs = s_{i_1} \cdots s_{i_{q-1}} \leq s_1 \cdots s_{r-1} = ws$. □

The Lifting Property seems quite innocent, but when trying to investigate facts around the Bruhat ordering it proofs to be one of the key tools in many cases.

1.5 Compact hyperbolic Coxeter groups

TODO

2 Twisted involutions in Coxeter groups

In this section our interest will focus on a certain subset of elements in Coxeter groups, the so called twisted involutions. From now on (and in the next sections) we will fix some symbols to have always the same meaning (some definitions will follow later):

S	A set of generators.
s	A generator in S.
W	A Coxeter group with generators S.
u, v, w	A element in the Coxeter group W.
m_{ij}	The order of the element $(s_i s_j)$ with s_i the i-th generator of W.
(W, S)	The Coxeter system obtained from W and S.
θ	A Coxeter system automorphism of (W, S) with $\theta^2 = \mathrm{id}$.
$\mathcal{I}_\theta$	The set of twisted involutions of W regarding θ.
$\underline{S}$	A set of symbols, $\underline{S} = \{\underline{s} : s \in S\}$.

2.1 Introduction to twisted involutions

Definition 2.1. An automorphism $\theta : W \to W$ with $\theta(S) = S$ is called a ***Coxeter system automorphism*** of (W, S). We will always assume $\theta^2 = \mathrm{id}$.

Definition 2.2. Each $w \in W$ with $\theta(w) = w^{-1}$ is called a θ***-twisted involution*** or just ***twisted involution***, if θ is clear from the context. The set of all twisted involutions in W regarding θ is denoted with $\mathcal{I}_\theta(W)$. Often we will just omit the Coxeter group and write $\mathcal{I}_\theta$, when it is clear from the context which W is meant.

Lets take a quick look at some examples. First of all the trivial one.

Example 2.3. Let $\theta = \mathrm{id}_W$. Then θ is an Coxeter system automorphism and

$$\mathcal{I}_\theta = \{w \in W : w = w^{-1}\}.$$

The next example is more helpfull, since it reveals a way to think of $\mathcal{I}_\theta$ as a generalization of ordinary Coxeter groups.

Example 2.4. Let θ be a automorphism of $W \times W$ with

$$\theta : W \times W \to W \times W : (u, w) \mapsto (w, u).$$

Note that θ is no Coxeter system automorphism, but we can think of it as one if we identify $S \subset W$ with $S \times S \subset W \times W$. Then the set of twisted involutions is

$$\mathcal{I}_\theta = \{(w, w^{-1}) \in W \times W : w \in W\}.$$

This yields a canonical bijection between $\mathcal{I}_\theta$ and W.

The map we will define right now is of superior importance to this whole paper, since it is needed to define the poset, the main thesis is about.

Definition 2.5. Let $\underline{S} := \{\underline{s} : s \in S\}$ be a set of symbols. Each element in $\underline{S}$ acts from the right on W by the following definition:

$$w\underline{s} = \begin{cases} ws & \text{if } \theta(s)ws = w \\ \theta(s)ws & \text{else} \end{cases}$$

This action can be extended on the whole free monoid over $\underline{S}$ by

$$w\underline{s_1 s_2 \ldots s_k} = (\ldots((w\underline{s_1})\underline{s_2})\ldots)\underline{s_k}.$$

If $w\underline{s} = \theta(s)ws$, then we say s ***acts bothsided*** on w. Else we say s ***acts onesided*** on w.

Definition 2.6. Let $k \in \mathbb{N}$ and $s_{i_j} \in S$ for all $1 \leq j \leq k$. Then an expression $w\underline{s_{i_1} \cdots s_{i_k}}$ is called ***twisted*** w***-expression***. In case $w = e$ we will omit w, just write $\underline{s_{i_1} \cdots s_{i_k}}$ and call it ***twisted expression***.

There is another characterization of this action, distinguishing between one- and both-sided actions by length.

Lemma 2.7. *Let $w \in \mathcal{I}_\theta$ and $s \in S$. Then*

$$w\underline{s} = \begin{cases} ws & \text{if } l(\theta(s)ws) = l(w), \\ \theta(s)ws & \text{else.} \end{cases}$$

Proof. Suppose s acts oneside on w. Then $\theta(s)ws = w$ and so $l(\theta(s)ws) = l(w)$. So let the other way around $l(\theta(s)ws) = l(w)$. **TODO** □

Lemma 2.8. *It is $l(ws) < l(w)$ iff $l(w\underline{s}) < l(w)$.*

Proof. Suppose s acts onesided on w. Then $w\underline{s} = ws$ and there is nothing to prove. So suppose s acts bothsided on w. If $l(ws) < l(w)$, then Lemma 1.6 yields $l(ws) + 1 = l(w)$. Assuming $l(w\underline{s}) = l(\theta(s)ws) = l(w)$ would imply, that s acts oneside on w due to Lemma 2.7, which is a contradiction. So let $l(\theta(s)ws) < l(w)$. Then Lemma 1.6 yields $l(\theta(s)ws) + 2 = l(w)$ and so $l(ws) + 1 = l(w)$. □

Lemma 2.9. *For all $w \in W$ and $s \in S$ it is $w\underline{ss} = w$.*

Proof. For $w\underline{s}$ there are two cases. Suppose s acts onesided on w, i.e. $\theta(s)ws = w$. For $ws\underline{s}$ there are again two possible options.

$$ws\underline{s} = \begin{cases} wss = w & \text{if } \theta(s)wss = ws \\ \theta(s)wss = ws & \text{else} \end{cases}$$

The second option contradicts itself.

So lets now suppose s acts bothsided on w. This means $\theta(s)ws \neq w$ and for $(\theta(s)ws)\underline{s}$ there are again two possible options.

$$(\theta(s)ws)\underline{s} = \begin{cases} \theta(s)wss = \theta(s)w & \text{if } \theta(s)\theta(s)wss = \theta(s)ws \\ \theta(s)\theta(s)wss = w & \text{else} \end{cases}$$

The first option is impossible since $\theta(s)\theta(s)wss = w$ and we have assumed $\theta(s)ws \neq w$. So the only cases possible yield $w\underline{ss} = w$. □

Remark 2.10. This lemma allows us to to rewrite equations of twisted expressions. For example

$$u = w\underline{s} \iff u\underline{s} = w\underline{ss} = w.$$

This can be iterated to get

$$u = w\underline{s}_1 \dots \underline{s}_k \iff u\underline{s}_k \dots \underline{s}_1 = w.$$

Lemma 2.11. *For all θ, $w \in W$ and $s \in S$ it is $w \in \mathcal{I}_\theta$ iff $w\underline{s} \in \mathcal{I}_\theta$.*

Proof. Let $w \in \mathcal{I}_\theta$. For $w\underline{s}$ there are two cases. Suppose s acts onesided on w. Then we get

$$\theta(ws) = \theta(\theta(s)wss) = \theta^2(s)\theta(w) = sw^{-1} = (ws^{-1})^{-1} = (ws)^{-1}.$$

Suppose s acts bothsided on w. Then we get

$$\theta(\theta(s)ws) = \theta^2(s)\theta(w)\theta(s) = sw^{-1}\theta(s) = (\theta^{-1}(s)ws^{-1})^{-1} = (\theta(s)ws)^{-1}.$$

In both cases $w\underline{s} \in \mathcal{I}_\theta$.
Now let $w\underline{s} \in \mathcal{I}_\theta$. Suppose s acts onesided on w. Then

$$\theta(w) = \theta(\theta(s)ws) = \theta^2(s)\theta(ws) = s(ws)^{-1} = ss^{-1}w^{-1} = w^{-1}.$$

Suppose s acts twosided on w. Then

$$\begin{aligned} \theta(w) &= \theta(\theta(s)\theta(s)wss) = \theta^2(s)\theta(\theta(s)ws)\theta(s) \\ &= s(\theta(s)ws)^{-1}\theta(s) = s(s^{-1}w^{-1}\theta(s^{-1})\theta(s) = w^{-1}. \end{aligned}$$

In both cases $w \in \mathcal{I}_\theta$. □

A remarkable property of the action from Definition 2.5 is its e-orbit. As the following lemma will shows, it coincides with $\mathcal{I}_\theta$.

Lemma 2.12. *Fix θ. Then the set of twisted involutions regarding θ coincides with the set of all twisted expressions regarding θ.*

Proof. As already seen in Lemma 2.11, each twisted expression is in $\mathcal{I}_\theta$, since $e \in \mathcal{I}_\theta$. So let $w \in \mathcal{I}_\theta$. If $l(w) = 0$, then $w = e \in \mathcal{I}_\theta$. Lets induce on the length of w and let $l(w) = r > 0$. Suppose w has a twisted expression ending with $\underline{s}$. Then w also has a reduced expression (in S) ending with s and so $l(ws) < l(w)$. With Lemma 2.8 we get $l(w\underline{s}) < l(w)$. By induction $w\underline{s}$ has twisted expression and hence $w = (w\underline{s})\underline{s}$ has one, too. □

In the same way, we can use regular expressions to define the length of an element $w \in W$, we can use the twisted expressions to define the twisted absolute length of an element $w \in \mathcal{I}_\theta$.

Definition 2.13. Let $\mathcal{I}_\theta$ be the set of twisted involutions. Then we define $\rho(w)$ as the smallest $k \in \mathbb{N}$ for that a twisted expression $w = \underline{s_1 \dots s_k}$ exists. This is called the ***twisted length*** of w.

Lemma 2.14. *The set of twisted involutions $\mathcal{I}_\theta$ together with the Bruhat ordering, denoted with $\mathrm{Br}(\mathcal{I}_\theta)$, is a graded poset with ρ as rank function.*

Proof. See [2, Theorem 4.8]. □

We will now establish many properties from Section 1 for twisted expressions and $\mathrm{Br}(\mathcal{I}_\theta)$. As seen in Example 2.4 it is $\mathrm{Br}(W) \cong \mathrm{Br}(\mathcal{I}_\theta)$. So the hope, that many properties can be transfered, is eligible.

Lemma 2.15. *Let $w \in \mathcal{I}_\theta$ and $s \in S$. Then $\rho(w\underline{s}) = \rho(w) \pm 1$. In fact it is $\rho(w\underline{s}) = \rho(w) - 1$ iff $s \in D_R(w)$.*

Proof. Since $\mathrm{Br}(\mathcal{I}_\theta)$ is graded with rank function ρ and either $w\underline{s}$ covers w or w covers $w\underline{s}$ it is $\rho(w\underline{s}) = \rho(w) \pm 1$. Now suppose $w\underline{s} < w$. Then $l(w\underline{s}) < l(w)$ and with Lemma 2.8 we have $l(ws) < l(w)$ yielding $s \in D_R(w)$. The other way around suppose $w\underline{s} > w$. Then $l(w\underline{s}) > l(w)$ and again with Lemma 2.8 we have $l(ws) > l(w)$ yielding $s \notin D_R(w)$. □

Proposition 2.16 (Lifting Property for $\underline{S}$)**.** *Let $v, w \in W$ with $v \leq w$. Suppose $s \in S$ with $s \in D_R(w)$. Then*

1. $v\underline{s} \leq w$,
2. $s \in D_R(v) \Rightarrow v\underline{s} \leq w\underline{s}$.

Proof. We will distinguish between the four cases of one- and bothsided action of s on u and w. Whenever a relation comes from the ordinary Lifting Property, we will denote it with $<_{LP}$ in this proof.

$v\underline{s} = vs \wedge w\underline{s} = ws$ Same situation as in Lifting Property.

$v\underline{s} = vs \wedge w\underline{s} = \theta(s)ws$ The first part $v\underline{s} = vs \leq_{LP} w$ is immediate. Suppose $s \in D_R(v)$. Then $vs \leq_{LP} ws \Rightarrow v = \theta(s)vs \leq ws \Rightarrow v\underline{s} = vs \leq \theta(s)ws = w\underline{s}$.

$v\underline{s} = \theta(s)vs \wedge w\underline{s} = ws$ **TODO**

$v\underline{s} = \theta(s)vs \wedge w\underline{s} = \theta(s)ws$ **TODO**

□

TODOExchange property, ...

2.2 Twisted weak ordering

In this section we introduce the twisted weak ordering $Wk(\theta)$ on the set $\mathcal{I}_\theta$ of θ-twisted involutions.

Definition 2.17. Let $\mathcal{I}_\theta$ be the set of twisted involutions. For $v, w \in \mathcal{I}_\theta$ we define $v \preceq w$ iff there are $\underline{s}_1, \ldots, \underline{s}_k \in \underline{S}$ with $w = \underline{v s_1 \ldots s_k}$ and $\rho(v) = \rho(w) - k$. We denote the poset $(\mathcal{I}_\theta, \preceq)$ with $Wk(\theta)$.

Lemma 2.18. *The poset $Wk(\theta)$ is a graded poset with rank function $\preceq$.*

Proof. Follows immediatly from the definition of $\preceq$. □

Definition 2.19. Let $w, u \in W$ with $\rho(u) - \rho(w) = n$. Each sequence $w = w_0 \prec w_1 \prec \ldots \prec w_n = u$ is called a ***geodesic*** from w to u.

TODO

2.3 Residuums

Definition 2.20. Let $w \in W$ and $I \subseteq S$ be a subset of generators. Then we define

$$wC_I := \{w\underline{s_1} \ldots \underline{s_k} : k \in \mathbb{N}_0, s_i \in S\}$$

as the I***-residuum*** of w or just ***residuum***. To emphasize the size of I, say $|I| = n$, we will also speak of a ***rank-n-residuum***.

Example 2.21. Let $w \in W$. Then $wC_\emptyset = \{w\}$ and $wC_S = \mathcal{I}_\theta$.

Lemma 2.22. *Let $w \in W$ and $I \subset S$. If $v \in wC_I$, then $vC_I = wC_I$.*

Proof. Suppose $v \in wC_I$. Then $v = w\underline{s}_1 \ldots \underline{s}_n$ for some $s_i \in I$. Suppose $u = w\underline{t}_1 \ldots \underline{t}_m \in wC_I$ is any other element in wC_I with $t_i \in I$. Then

$$u = w\underline{t}_1 \ldots \underline{t}_m = (v\underline{s}_n \ldots \underline{s}_1)\underline{t}_1 \ldots \underline{t}_m$$

and so $u \in vC_I$. This yields $wC_I \subset vC_I$. Since $w \in vC_I$ we can swap v and w to get the other inclusion. □

Corollary 2.23. *Let $v, w \in W$ and $I \subset S$. Then either $vC_I \cap wC_I = \emptyset$ or $vC_I = wC_I$.*

Proof. Immediatly follows from Lemma 2.22. □

We proceed with some properties of rank-2-residuums. These are needed later in Section 2.4 to construct an effective algorithm for calculating the twisted weak ordering, i.e. calculating the Hasse diagram of $Wk(\theta)$ for arbitrary Coxeter systems (W, S) and Coxeter system automorphisms θ.

Definition 2.24. Let $s, t \in S$ be two distinct generators. We define:

$$[st]^n := \begin{cases} (st)^{\frac{n}{2}} & n \text{ even,} \\ (st)^{\frac{n-1}{2}} s & n \text{ odd.} \end{cases}$$

This definition lets us rewrite rank-2-residuums. Suppose we have a fixed start element $w \in \mathcal{I}_\theta$ and two distinct generators $s, t \in S$. Then

$$wC_{\{s,t\}} = \{w\} \cup \{w\underline{[st]^n} : n \in \mathbb{N}\} \cup \{w\underline{[ts]^n} : n \in \mathbb{N}\}.$$

With the following propositions and corollaries we will get a much better idea of the structure of rank-2-residuums.

Proposition 2.25. **TODO***Let $w \in W$ and let $s, t \in S$ two distinct generators. Then $wC_{\{s,t\}}$ does not contain three elements of equal twisted length.*

Proof. Let (W, S) be a Coxeter system, $w \in W$ with $\operatorname{rank} w = k$, $s, t \in S$ with $s \neq t$. Without loss of generality we can choose w such that $w < w\underline{s}$ and $w < w\underline{t}$. Assume the existence of an element $u \in wC_{\{s,t\}}$ with $u\underline{s} < u$ and $u\underline{t} < u$. Then [3, Lemma 3.8] yields $s, t \in D_R(u)$. By using [3, Lemma 3.9] we conclude that $w\underline{s} \leq u$ and $w\underline{t} \leq u$. Hence there cannot exist more than two Elements of same twisted length.
If no such u exists, then $wC_{\{s,t\}} = w \,\dot\cup\, \{w\underline{[st]^n} : n \in \mathbb{N}\} \,\dot\cup\, \{w\underline{[ts]^n} : n \in \mathbb{N}\}$ and the assumption still holds. □

Corollary 2.26. **TODO***Let $w \in W$ and $s, t \in S$ two distinct generators. Then $wC_{\{s,t\}}$ contains exactly one element v with $v < v\underline{s}$ and $v < v\underline{t}$ and at most one element u with $u > u\underline{s}$ and $u > u\underline{t}$.*

Proof. If there is any $w' \in wC_{\{s,t\}}$ with $w'\underline{s} = w'\underline{t}$, then $wC_{\{s,t\}} = \{w, w\underline{s}\}$ and we are done. So suppose there is no such element.
Since twisted length cannot be lower than 0 there must be at least one element v with $s, t \notin D_R(v)$. Suppose there is another element $v' \neq v$ with $s, t \notin D_R(v')$. If $\rho(u) = \rho(u')$ then $\rho(u\underline{s}) = \rho(u\underline{t}) = \rho(u'\underline{s}) = \rho(u'\underline{t})$. These four expressions must describe at least three distinct elements, since else we would have $u = u'$. So we have three distinct elements of same twisted length contradicting to Proposition 2.25. If $\rho(u) < \rho(u')$ we can conclude a contradiction with similar arguments.
If $|wC_{\{s,t\}}| < \infty$ there must be a u with $s, t \in D_R(u)$. We repeat the previous steps to get, that there is no other $u' \neq u$ with $s, t \in D_R(u')$. If $|wC_{\{s,t\}}| = \infty$ there cannot be a u with $s, t \in D_R(u)$. □

Proposition 2.27. **TODO***Let $w \in S$ and $s, t \in S$ two distinct generators. If s operates onesided on w and $w\underline{s} < w$, then either $w\underline{st} < w\underline{s}$ or $w\underline{t} > w$.*

Proof. We have $\theta(s)ws = w$ and $s \in D_R(w)$. If $t \notin D_R(w)$, then we are done. So suppose $t \in D_R(w)$. This means $w\underline{s} \leq w$ and $w\underline{t} \leq w$ and [3, Lemma 3.9] yields $w\underline{st} < w$ and $w\underline{ts} < w$. If $t \in D_R(w\underline{s})$, then we are done. So suppose $t \notin D_R(w\underline{s})$. Then $t \in D_R(w\underline{st})$. Together with $w\underline{st} \leq w$ [3, Lemma 3.9(2)] says $(w\underline{st})\underline{t} \leq w\underline{t}$. Finally we get

$$ws = w\underline{s} = (w\underline{st})\underline{t} \leq w\underline{t} = wt.$$

Since $w\underline{s}$ and $w\underline{t}$ are of same twisted length they have to be equal and therefore $s = t$ which contradicts to our assumption of two distinct generators s and t. □

Corollary 2.28. TODO*Let $w \in S$ and $s, t \in S$ two distinct generators. If w is neither the unique element in $wC_{\{s,t\}}$ of smallest twisted length nor the unique (but not neseccarily existing) element of largest twisted length, then s and t act twosided on w.*

Proof. Follows immediatly from Proposition 2.27. □

Lemma 2.29. *Let $w \in S$, $s, t \in S$ two distinct generators and $m = \operatorname{ord}(st) < \infty$. Then $|wC_{\{s,t\}}| \leq 2m$.*

Proof. TODO □

Example 2.30. In Figure 2.1 we see Hasse diagram of $Wk(\mathrm{id})$ on the involutions in the Coxeter group A_4. Solid edges represent bothsided actions and dashed edges represent onesided actions.

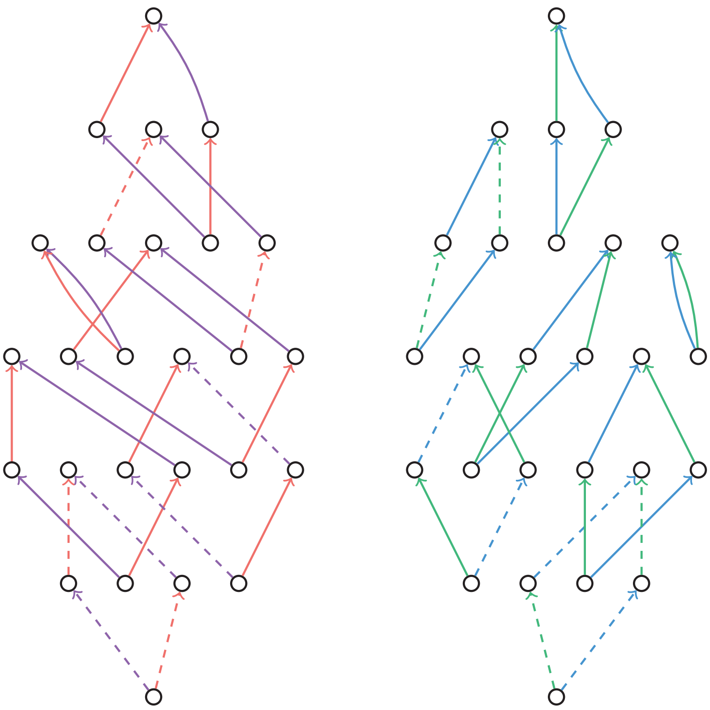

Figure 2.1: Hasse diagrams of $Wk(\mathrm{id})$, $W = A_4$ with only s_1, s_3 edges on the left and only s_2, s_4 edges on the right side

2.4 Twisted weak ordering algorithms

3 Twisted weak ordering

Wir wollen nun einen Algorithmus zur Berechnung der getwisteten schwachen Ordnung $Wk(\theta)$ einer beliebigen Coxetergruppe W erarbeiten. Also Ausgangspunkt werden wir den Algorithmus aus [1, Algorithm 3.1.1] verwenden, der im wesentlichen benutzt, dass für jede getwistete Involution $w \in \mathcal{I}_\theta$ entweder $w\underline{s} < w$ oder aber $w\underline{s} > w$ gilt.

Algorithm 3.1 (Algorithmus 1)**.**

```
procedure TwistedWeakOrderingAlgorithm1(W)          ▷ W sei die Coxetergruppe
    V ← {(e,0)}
    E ← {}
    for k ← 0 to k_max do
        for all (w, k_w) ∈ V with k_w = k do
            for all s ∈ S with ∄(·, w, s) ∈ E do     ▷ Nur die s, die nicht schon nach w führen
                y ← ws
                z ← θ(s)y
                if z = w then
                    x ← y                            ▷ s operiert ungetwistet auf w
                    t ← s
                else
                    x ← z                            ▷ s operiert getwistet auf w
                    t ← s̲
                end if
                isNew ← true
                for all (w', k_w') ∈ V with k_w' = k + 1 do   ▷ Prüfen, ob x nicht schon in V liegt
                    if x = w' then
                        isNew ← false
                    end if
                end for
                if isNew = true then
                    V ← V ∪ {(x, k + 1)}
                    E ← E ∪ {(w, x, t)}
                else
                    E ← E ∪ {(w, x, t)}
                end if
            end for
        end for
        k ← k + 1
    end for
    return (V, E)                                    ▷ The poset graph
```

			Timings		Element compares	
W	$\lvert Wk(\mathrm{id}, W)\rvert$	$\rho(w_0)$	TWOA1	TWOA2	TWOA1	TWOA2
A_9	9496	25	00:02.180	00:01.372	13,531,414	42,156
A_{10}	35696	30	00:31.442	00:06.276	185,791,174	173,356
A_{11}	140152	36	11:04.241	00:29.830	2,778,111,763	737,313
E_6	892	20	00:03.044	00:00.268	85,857	2,347
E_7	10208	35	06:11.728	00:02.840	7,785,186	29,687
E_8	199952	64	–	11:03.278	–	682,227

Table 3.1: Benchmark

end procedure

Dieser Algorithmus berechnet alle getwisteten Involutionen und deren getwistete Länge (w, k_w) und deren Relationen (w', w, s) bzw. $(w', w, \underline{s})$. Zu bemerken ist, dass zur Berechnung der getwisteten Involutionen der Länge k nur die Knoten aus V benötigt werden, mit der getwisteten Länge $k-1$ und k sowie die Kanten aus E, die Knoten der Länge $k-2$ und $k-1$ bzw. $k-1$ und k verbinden. Alle vorherigen Ergebnisse können schon persistiert werden, so dass nie das komplette Ergebnis im Speicher gehalten werden muss.
Eine Operation, die hier als elementar angenommen wurde ist der Vergleich von Elementen in W. Für bestimmte Gruppen wie z.B. die A_n, welche je isomorph zu $Sym(n+1)$ sind, lässt sich der Vergleich von Element effizient implementieren. Will man jedoch mit Coxetergruppen im Allgemeinen arbeiten, so liegt W als frei präsentierte Gruppe vor und der Vergleich von Element is eine sehr aufwendige Operation. Bei Algorithm 3.1 muss jedes potentiell neue Element x mit allen schon bekannten w' von gleicher getwisteter Länge verglichen werden um zu bestimmen, ob x wirklich ein noch nicht bekanntes Element aus $\mathcal{I}_\theta$ ist.

Algorithm 3.2 (Algorithmus 2)**.**

```
procedure TwistedWeakOrderingAlgorithm2(W)        ▷ W sei die Coxetergruppe
    V ← {(e, 0)}
    E ← {}
    for k ← 0 to k_max do
        TODO
    end for
    return (V, E)                                  ▷ The poset graph
end procedure
```

Im Anhang findet sich eine Implementierung von Algorithm 3.1 und Algorithm 3.2 in GAP 4.5.4. Table 3.1 zeigt ein Benchmark anhand von fünf ausgewählten Coxetergruppen. Dabei sind die A_n als symmetrische Gruppen implementiert und die E_n als frei präsentierte Gruppen. Ausgeführt wurden die Messungen auf einem Intel Core i5-3570k mit vier Kernen zu je 3,40 GHz. Der Algorithmus ist dabei aber nur single threaded und kann

so nur auf einem Kern laufen. Um die Messergebnisse nicht durch Limitierungen des Datenspeichers zu beeinflussen, wurden die Daten in diesem Benchmark nicht stückweise persistiert sondern ausschließlich berechnet.
Wie zu erwarten ist der Geschwindigkeitsgewinn bei den Coxetergruppen vom Typ E_n deutlich größer, da in diesem Fall die Elementvergleiche deutlich aufwendiger sind als bei Gruppen vom Typ A_n.

4 Miscellaneous

Question 4.1. *Let (W, S) be a Coxeter system, $\theta : W \to W$ an automorphism of W with $\theta^2 = \mathrm{id}$ and $\theta(S) = S$, and $K \subset S$ a subset of S generating a finite subgroup of W with $\theta(K) = K$. Futhermore let $T, S_1, S_2, S_3 \subset S$ be four pairwise disjoint sets of generators. For which Coxeter groups W does the implication*

$$w \in w_K C_{T \cup S_i}, i = 1, 2, 3 \Rightarrow w \in w_K C_T \tag{4.1.1}$$

hold for any possible $K, \theta, T, S_1, S_2, S_3$ and w?

Proposition 4.2. *Let (W, S) be a Coxeter system and K, T, S_1, S_2, S_3 be like in Question 4.1. Suppose we have $w \in W$ and $a_1, \ldots, a_n \in T \cup S_1$, $b_1, \ldots, b_n \in T \cup S_2$, $c_1, \ldots, c_n \in T \cup S_3$ with*

$$\begin{aligned} w &= w_K \underline{a_1 \cdots a_n} \\ &= w_K \underline{b_1 \cdots b_n} \\ &= w_K \underline{c_1 \cdots c_n} \end{aligned}$$

and (4.1.1) does not hold for these three expressions, i.e. $w \notin w_K C_T$. Then there exist $t_1, \ldots, t_m \in T$ and $a'_1, \ldots, a'_{n-m} \in T \cup S_1$, $b'_1, \ldots, b'_{n-m} \in T \cup S_2$, $c'_1, \ldots, c'_{n-m} \in T \cup S_3$ such that

$$\begin{aligned} w\underline{t_1 \ldots t_m} &= w_K \underline{a'_1 \cdots a'_{n-m}} \\ &= w_K \underline{b'_1 \cdots b'_{n-m}} \\ &= w_K \underline{c'_1 \cdots c'_{n-m}} \end{aligned}$$

with $a'_{n-m}, b'_{n-m}, c'_{n-m} \notin T$.

Proof. Suppose at least one element of a_n, b_n, c_n to be in T, for example $a_n \in T$. Then we can apply $\underline{a_n}$ to all three expressions. Since $\rho(w\underline{a_n}) < \rho(w)$ the exchange condition for $\mathcal{I}_\theta$ [3, Proposition 3.10]yields

$$\begin{aligned} w\underline{a_n} &= w_K \underline{a_1 \cdots a_n a_n} = w_K \underline{a_1 \cdots a_{n-1}} \\ &= w_K \underline{b_1 \cdots b_n a_n} = w_K \underline{b_1 \cdots \hat{b}_i \cdots b_n} \\ &= w_K \underline{c_1 \cdots c_n a_n} = w_K \underline{c_1 \cdots \hat{c}_j \cdots c_n} \end{aligned}$$

where $\hat{}$ means omission. The omission cannot occur within w_K since all three expressions are still of same twisted length and in the first expression we can see, that $w_K \leq w\underline{a_n}$ still holds. This step can be repeated until $w = w_K$ or $a_n, b_n, c_n \notin T$. □

Lemma 4.3. *A counterexample to Question 4.1 can only exist, if there is an element $u \in wC_T$ and three distinct generators $s_1, s_2, s_3 \in D_r(u)$ such that $u\underline{s_i} \notin wC_T$ for $i = 1, 2, 3$.*

Proof. According to Proposition 4.2. □

Lemma 4.4. *A counterexample to Question 4.1 can only exist, if there are three not neseccarily distinct elements $a, b, c \in w_K C_{S \setminus T}$, three distinct generators $s_1 \in A_r(a)$, $s_2 \in A_r(b)$, $s_3 \in A_r(c)$ and an element $u \notin w_K C_{S \setminus T}$ such that*

$$a\underline{s_1} = b\underline{s_2} = c\underline{s_3} = u.$$

Proof. If there is a counterexample, then the two residuums $w_K C_{S \setminus T}$ and wC_T are disjunct. Since we are only interested in w with $w_K \leq w$ it follows, that any geodesic from w_K to w is contained in the union set of both residuums. Hence having one element in $u \in wC_T$ with three distinct generators s_1, s_2, s_3 with $u\underline{s_i} \notin wC_T$ is equivalent to having three elements $a, b, c \notin wC_T$ and the same three generator s_1, s_2, s_3 with $a\underline{s_1} = b\underline{s_2} = c\underline{s_3} = u \in wC_T$. $\square$

A Source codes

```
LoadPackage("io");

Read("misc.gap");
Read("coxeter.gap");
Read("twistedinvolutionweakordering-persist.gap");

TwistedInvolutionDeduceNodeAndEdgeFromGraph := function(matrix, startNode, startLabel,
     labels)
    local rank, comb, trace, possibleEqualNodes, e, k, n;

    rank := -1/2 + Sqrt(1/4 + 2*Length(matrix)) + 1;
    possibleEqualNodes := [];

    for comb in List(Filtered(labels, label -> label <> startLabel), label -> rec(
         startNode := startNode, s := [startLabel, label], m := CoxeterMatrixEntry(
         matrix, rank, startLabel, label))) do
        trace := [];
        k := 1;
        n := comb.startNode;

        Add(trace, rec(node := n, edge := rec(label := comb.s[1], type := -1)));

        while k < comb.m do
            e := FindElement(n.inEdges, e -> e.label = comb.s[k mod 2 + 1]);
            if e = fail then break; fi;
            n := e.source;

            Add(trace, rec(node := n, edge := e));
            k := k + 1;
        od;

        while k > 0 do
            e := FindElement(n.outEdges, e -> e.label = comb.s[k mod 2 + 1]);
            if e = fail then break; fi;
            n := e.target;

            Add(trace, rec(node := n, edge := e));
            k := k - 1;
        od;

        if k <> 0 then continue; fi;

        if Length(trace) = 2*comb.m then
            return rec(result := 0, node := trace[Length(trace)].node, type := trace[
                 comb.m + 1].edge.type, trace := trace);
        fi;

        if Length(trace) >= 4 then
            if trace[Length(trace) / 2 + 1].edge.type <> trace[Length(trace) / 2].edge.
                 type then
                # cannot be equal
            else
                if trace[Length(trace)].edge.type = 0 then
                    return rec(result := 0, node := trace[Length(trace)].node, type :=
                         0, trace := trace);
                else
                    Add(possibleEqualNodes, trace[Length(trace)].node);
                fi;
```

```
            fi;
        else
            Add(possibleEqualNodes, trace[Length(trace)].node);
        fi;
    od;

    return rec(result := -1, possibleEqualNodes := possibleEqualNodes);
end;

# Calculates the poset Wk(theta).
TwistedInvolutionWeakOrdering := function (filename, W, matrix, theta)
    local persistInfo, maxOrder, nodes, edges, absNodeIndex, absEdgeIndex, prevNode, currNode, newEdge,
        label, type, deduction, startTime, endTime, S, k, i, s, x, y, n;

    persistInfo := TwistedInvolutionWeakOrderingPersistResultsInit(filename);

    S := GeneratorsOfGroup(W);
    maxOrder := Minimum([Maximum(Concatenation(matrix, [1])), 5]);
    nodes := [ [], [ rec(element := One(W), twistedLength := 0, inEdges := [], outEdges := [], absIndex := 1) ] ];
    edges := [ [], [] ];
    absNodeIndex := 2;
    absEdgeIndex := 1;
    k := 0;

    while Length(nodes[2]) > 0 do
        if not IsFinite(W) then
            if k > 200 or absNodeIndex > 10000 then
                break;
            fi;
        fi;

        for i in [1..Length(nodes[2])] do
            Print(k, " ", i, "          \r");

            prevNode := nodes[2][i];
            for label in Filtered([1..Length(S)], n -> Position(List(prevNode.inEdges, e -> e.label), n) = fail) do
                deduction := TwistedInvolutionDeduceNodeAndEdgeFromGraph(matrix, prevNode, label, [1..Length(S)]);

                if deduction.result = 0 then
                    type := deduction.type;
                    currNode := deduction.node;
                elif deduction.result = 1 then
                    type := deduction.type;

                    currNode := rec(element := y, twistedLength := k + 1, inEdges := [], outEdges := [], absIndex := absNodeIndex);
                    Add(nodes[1], currNode);

                    absNodeIndex := absNodeIndex + 1;
                else
                    x := prevNode.element;
                    s := S[label];

                    type := 1;
                    y := s^theta*x*s;
                    if (CoxeterElementsCompare(x, y)) then
                        y := x * s;
```

```
                        type := 0;
                    fi;

                    currNode := FindElement(deduction.possibleEqualNodes, n -> CoxeterElementsCompare(n.element, y));

                    if currNode = fail then
                        currNode := rec(element := y, twistedLength := k + 1, inEdges := [], outEdges := [], absIndex := absNodeIndex);
                        Add(nodes[1], currNode);

                        absNodeIndex := absNodeIndex + 1;
                    fi;
                fi;

                newEdge := rec(source := prevNode, target := currNode, label := label, type := type, absIndex := absEdgeIndex);

                Add(edges[1], newEdge);
                Add(currNode.inEdges, newEdge);
                Add(prevNode.outEdges, newEdge);

                absEdgeIndex := absEdgeIndex + 1;
            od;
        od;

        TwistedInvolutionWeakOrderingPersistResults(persistInfo, nodes[2], edges[2]);

        Add(nodes, [], 1);
        Add(edges, [], 1);
        if (Length(nodes) > maxOrder + 1) then
            for n in nodes[maxOrder + 2] do
                n.inEdges := [];
                n.outEdges := [];
            od;
            Remove(nodes, maxOrder + 2);
            Remove(edges, maxOrder + 2);
        fi;
        k := k + 1;
    od;

    TwistedInvolutionWeakOrderingPersistResultsInfo(persistInfo, W, matrix, theta, absNodeIndex - 1, k - 1);
    TwistedInvolutionWeakOrderingPersistResultsClose(persistInfo);

    return rec(numNodes := absNodeIndex - 1, numEdges := absEdgeIndex - 1, maxTwistedLength := k - 1);
end;

# Calculates the poset Wk(theta).
TwistedInvolutionWeakOrdering1 := function (filename, W, matrix, theta)
    local persistInfo, maxOrder, nodes, edges, absNodeIndex, absEdgeIndex, prevNode, currNode, newEdge,
        label, type, deduction, startTime, endTime, S, k, i, s, x, y, n;

    persistInfo := TwistedInvolutionWeakOrderingPersistResultsInit(filename);

    S := GeneratorsOfGroup(W);
    maxOrder := Minimum([Maximum(Concatenation(matrix, [1])), 5]);
    nodes := [ [], [ rec(element := One(W), twistedLength := 0, inEdges := [], outEdges := [], absIndex := 1) ] ];
```

```
    edges := [ [], [] ];
    absNodeIndex := 2;
    absEdgeIndex := 1;
    k := 0;

    while Length(nodes[2]) > 0 do
        if not IsFinite(W) then
            if k > 200 or absNodeIndex > 10000 then
                break;
            fi;
        fi;

        for i in [1..Length(nodes[2])] do
            Print(k, " ", i, "         \r");

            prevNode := nodes[2][i];
            for label in Filtered([1..Length(S)], n -> Position(List(prevNode.inEdges,
                e -> e.label), n) = fail) do
                x := prevNode.element;
                s := S[label];

                type := 1;
                y := s^theta*x*s;
                if (CoxeterElementsCompare(x, y)) then
                    y := x * s;
                    type := 0;
                fi;

                currNode := FindElement(nodes[1], n -> CoxeterElementsCompare(n.element
                    , y));

                if currNode = fail then
                    currNode := rec(element := y, twistedLength := k + 1, inEdges :=
                        [], outEdges := [], absIndex := absNodeIndex);
                    Add(nodes[1], currNode);

                    absNodeIndex := absNodeIndex + 1;
                fi;

                newEdge := rec(source := prevNode, target := currNode, label := label,
                    type := type, absIndex := absEdgeIndex);

                Add(edges[1], newEdge);
                Add(currNode.inEdges, newEdge);
                Add(prevNode.outEdges, newEdge);

                absEdgeIndex := absEdgeIndex + 1;
            od;
        od;

        TwistedInvolutionWeakOrderingPersistResults(persistInfo, nodes[2], edges[2]);

        Add(nodes, [], 1);
        Add(edges, [], 1);
        if (Length(nodes) > maxOrder + 1) then
            for n in nodes[maxOrder + 2] do
                n.inEdges := [];
                n.outEdges := [];
            od;
            Remove(nodes, maxOrder + 2);
            Remove(edges, maxOrder + 2);
```

```
        fi;
        k := k + 1;
    od;

    TwistedInvolutionWeakOrderingPersistResultsInfo(persistInfo, W, matrix, theta,
        absNodeIndex - 1, k - 1);
    TwistedInvolutionWeakOrderingPersistResultsClose(persistInfo);

    return rec(numNodes := absNodeIndex - 1, numEdges := absEdgeIndex - 1,
        maxTwistedLength := k - 1);
end;

TwistedInvolutionWeakOrderungResiduum := function (vertex, labels)
    local visited, queue, residuum, current, edge;

    visited := [ vertex ];
    queue := [ vertex ];
    residuum := [];

    while Length(queue) > 0 do
        current := queue[1];
        Remove(queue, 1);
        Add(residuum, current);

        for edge in current.outEdges do
            if edge.label in labels and not edge.target in visited then
                Add(visited, edge.target);
                Add(queue, edge.target);
            fi;
        od;
    od;

    return residuum;
end;

TwistedInvolutionWeakOrderungLongestWord := function (vertex, labels)
    local current;

    current := vertex;

    while Length(Filtered(current.outEdges, e -> e.label in labels)) > 0 do
        current := Filtered(current.outEdges, e -> e.label in labels)[1].target;
    od;

    return current;
end;
```

B References

[1] Ruth Haas and Aloysius G. Helminck. Algorithms for twisted involutions in weyl groups. *Algebra Colloquium 19*, 2012.

[2] Axel Hultman. Fixed points of involutive automorphisms of the bruhat order. *Adv. Math. 195*, pages 283–296, 2005. Also available at `http://www.ams.org/mathscinet-getitem?mr=2145798`.

[3] Axel Hultman. The combinatorics of twisted involutions in coxeter groups. *Transactions of the American Mathematical Society, Volume 359*, pages 2787–2798, 2007.

[4] James E. Humphreys. *Reflection groups and Coxeter groups.* Cambridge University Press, 1992.