Technische Universität Braunschweig

Posets of twisted involutions in Coxeter groups

by

Christian Hoffmeister

A thesis submitted to the Carl-Friedrich-Gauß-Faculty
in conformity with the requirements
for the degree of Master of Science (Mathematics)

Technische Universität Braunschweig
Brunswick, Lower Saxony, Germany
October 2012

E-Mail: choffmeister@choffmeister.de

Student number: 2944344

Contents

List of Figures

1 Introduction

TODO

2 Preliminaries

We start up with collecting some definitions and facts to ensure a uniform terminology and state of knowledge.

2.1 Posets

Definition 2.1. Let M be a set. A binary relation $\leq$ is called a ***partial order*** over M, if for all $a, b, c \in M$ it satisfies the conditions

1. $a \leq a$ (***reflexivity***),
2. $a \leq b \wedge b \leq a \Rightarrow a = b$ (***antisymmetry***) and
3. $a \leq b \wedge b \leq c \Rightarrow a \leq c$ (***transitivity***).

In this case $(M, \leq)$ is called a ***poset***. If two elements $a \leq b \in M$ are immediate neighbors, i.e. there is no third element $c \in M$ with $a \leq c \leq b$ we say that b ***covers*** a.

Definition 2.2. A poset is called ***graded poset*** if there is a map $\rho : M \to \mathbb{N}$ such that for all $a, b \in M$ with b covers a we have $\rho(b) = \rho(a) + 1$. In this case ρ is called the ***rank function*** of the graded poset.

Definition 2.3. A poset is called ***directed poset***, if for any two elements $a, b \in M$ there is an element $c \in M$ with $a \leq c$ and $b \leq c$. It is called ***bounded poset***, if it has a unique minimal and maximal element, denoted by $\hat{0}$ and $\hat{1}$.

Definition 2.4. Let $(M, \leq)$ be a poset and $a, b \in M$. Then we call $\{c \in M : a \leq c \leq b\}$ an ***interval*** and denote it by $[a, b]_{\leq}$. The set $\{c \in M : a < c < b\}$ is called an ***open interval*** and is denoted by $(a, b)_{\leq}$. In both cases we can omit the $\leq$, if the relation is clear from context.

Definition 2.5. The ***Hasse diagram*** of the poset $(M, \leq)$ is the graph obtained in the following way: Add a vertex for each element in M. Then add a directed edge from vertex a to b whenever b covers a.

Example 2.6. 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 2.1 we see the Hasse diagram of this poset with $M = \{x, y, z\}$.

Definition 2.7. Let $(M_i, \leq_i), i = 1, \ldots, n$ be a finite set of posets. We call the poset

$$(M_1 \times \ldots \times M_n, \leq) \text{ with } (a_1, \ldots, a_n) \leq (b_1, \ldots, b_n) \iff a_i \leq_i b_i \text{ for } i = 1, \ldots, n$$

a ***direct product of posets*** and denote it by $(M_n, \leq_n) \times \ldots \times (M_n, \leq_n)$.

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

Figure 2.1: Hasse diagram of the set of all subsets of $\{x, y, z\}$ order by the subset relation

2.2 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 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 [9, Theorem 1.9]. Furthermore it turns out, that the finite reflection groups in the Euclidean space are precisely the finite Coxeter groups [9, Theorem 6.4].

In this chapter we compile some basic well-known facts on Coxeter groups, based on [9].

Definition 2.8. 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 2.9. 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. We call this graph the ***Coxeter graph*** of our Coxeter system (W, S).

Definition 2.10. Let (W,S) be a Coxeter system. For an arbitrary element $w \in W$ 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 2.11. Let (W,S) be a Coxeter system and $w \in W$ an element. Then there are some (not necessarily distinct) 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 of 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 2.12. 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. Since the left descending set is not need in this paper, we will often call the right descending just ***descending set*** of w.

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

Lemma 2.13. [9, Section 5.2] *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. $l(ws) = l(w) \pm 1$.

2.2.1 Exchange and Deletion Condition

We now obtain a way to get a reduced expression of an arbitrary element $s_1 \cdots s_r = w \in W$.

Definition 2.14. Let (W, S) be a Coxeter system. 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 2.15 (Strong Exchange Condition)**.** [9, Theorem 5.8] *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 necessarily 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 start from a reduced expression, then i is unique.*

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

Corollary 2.16 (Exchange Condition)**.** [9, Theorem 5.8] *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 necessarily 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. In case we start from a reduced expression, then i is unique.*

Proof. Directly from Strong Exchange Condition. □

Remark 2.17. Note that both, Strong Exchange Condition and Exchange Condition have an analogues left-sided version

$$l(tw) < l(w) \Rightarrow tw = ts_1 \cdots s_k = s_1 \cdots \hat{s}_i \cdots s_k$$

for all reflections $t \in T$, hence for all generators $s \in S$ in particular.

Corollary 2.18 (Deletion Condition)**.** [9, Corollary 5.8] *Let (W, S) be a Coxeter system, $w \in W$ and $w = s_1 \cdots s_r$ with $s_i \in S$ an 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 an 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 2.19. 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 2.20. [9, Section 5.8] *Let (W, S) be a Coxeter system and $I, J \subset S$ two subsets of generators. Then $W_I \cap W_J = W_{I \cap J}$.*

A related fact, is the following lemma.

Lemma 2.21. [9, Section 5.8] *Let (W, S) be a Coxeter system and $w \in W$. Let $w = s_1 \cdots s_k$ any reduced expression for w. Then $\{s_1, \ldots, s_k\} \subset S$ is independent of the particular choosen reduced expression. It only depends on w itself.*

This means, that two reduced expressions for an element $w \in W$ use exactly the same generators.

2.2.2 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 infinite Coxeter groups without an ∞-relation between two generators, as well. 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 one decide weather W is finite or not?

To provide a general answer to this question we fallback to a certain class of Coxeter groups, the irreducible ones.

Definition 2.22. A Coxeter system is called ***irreducible***, if the corresponding Coxeter graph is connected. Else, it is called ***reducible***.

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

Proposition 2.23. [9, Proposition 6.1] *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.*

This proposition tells us, that an arbitray Coxeter system is finite iff its irreducible parabolic subgroups are finite. Therefore 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 2.24. [9, Theorem 6.4] *The irreducible finite Coxeter systems are exactly the ones in Figure 2.2.*

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

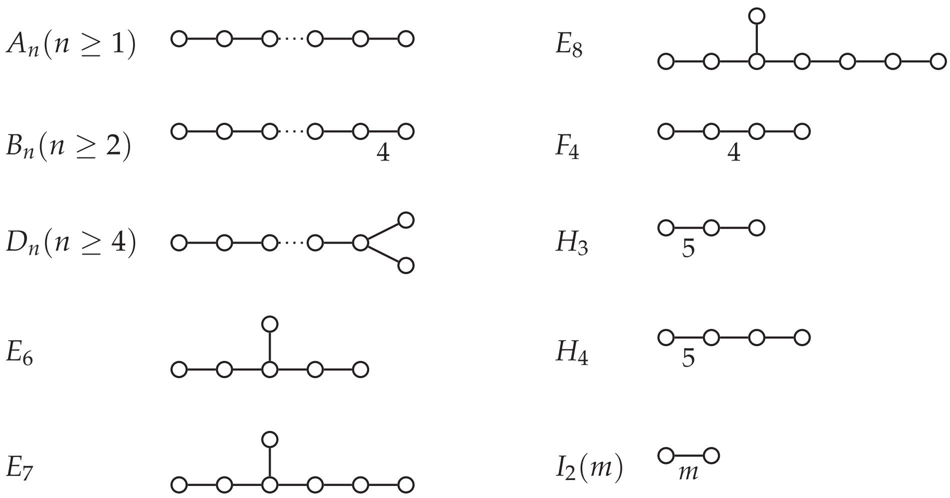

Figure 2.2: All types of irreducible finite Coxeter systems

2.2.3 Compact hyperbolic Coxeter groups

TODO

2.3 Bruhat ordering on Coxeter groups

We now investigate ways to partially order the elements of a Coxeter group. Futhermore, this ordering should be compatible with the length function, i.e. for $w, v \in W$ we have $l(w) < l(v)$ whenever $w < v$.

Definition 2.25. 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 by $\mathrm{Br}(W)$.

Lemma 2.26. [9, Section 5.9] *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. □

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. Beforehand let us just mention two other partial orderings, where this property is obvious by definition:

Definition 2.27. 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***.

To ensure the Bruhat ordering is graded as well, we need another characterization of the Bruhat ordering in terms of subexpressions.

Proposition 2.28. [9, Proposition 5.9] *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 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 $us \leq w$.

Assume $l(us) = l(u) + 1$. For the reflection $t' = sts$ we get $(us)t' = ussts = uts = ws$. So we have $us \leq ws$ iff $l(us) < l(ws)$. Suppose 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 2.29 (Subword property)**.** [9, Theorem 5.10] *Let (W,S) be a Coxeter system and $w \in W$ with a fixed, but arbitrary, 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. First we show, that any $w' < w$ occurs as a subexpression. For that we start with the case $w' \to w$, say $w't = w$. We have $l(w') < l(w)$ and hence by Strong Exchange Condition we get

$$w' = w'tt = wt = s_1 \cdots \hat{s}_i \cdots s_r$$

for some i. This step can be iterated. In return, suppose we have a subexpression $w' = s_{i_1} \cdots s_{i_q}$. We induce on $r = l(w)$. For $r = 0$ we have $w = e$, hence $w' = e$, too and so $w' \leq w$. Now suppose $r > 0$. If $i_q < r$, then $s_{i_1} \cdots s_{i_q}$ is a subexpression of $ws_r = s_1 \cdots s_{r-1}$, too. Since $l(ws_r) = r - 1 < r$, we can conclude

$$s_{i_1} \cdots s_{i_q} \leq s_1 \cdots s_{r-1} = ws_r < w$$

by induction hypothesis. If $i_q = r$, then we use our induction hypothesis to get $s_{i_1} \cdots s_{i_{q-1}} \leq s_1 \cdots s_{r-1}$. By Proposition 2.28 we get

$$s_{i_1} \cdots s_{i_q} \leq s_1 \cdots s_{r-1} < w$$

or

$$s_{i_1} \cdots s_{i_q} \leq s_1 \cdots s_r = w,$$

both finishing our induction. ☐

Corollary 2.30. *Let $u,w \in W$. Then the interval $[u,w]$ in the Bruhat order $\mathrm{Br}(W)$ is finite.*

Proof. We have $[u,w] \subseteq [e,w]$. All elements $v \in [e,w]$ can be obtained as subexpressions of one fixed reduced expression for w. Let $s_1 \ldots s_k = w$ be such an reduced expression. Then there are at most 2^k many subexpressions, hence $[u,w]$ is finite. ☐

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 2.31. [9, Lemma 5.11] *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 2.28 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 2.32. [9, Proposition 5.11] *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 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. Conversely 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 Subword property 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 neighbored 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 $w_i \neq w_{i-1}s$. We have $w_{i-1} < w_{i-1}s \neq w_i$ and due to Lemma 2.31 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 2.31 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 2.33. *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 2.32 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 2.34 (Lifting Property). [4, Theorem 1.1] *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 Subword property.

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 vs is again 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 proves to be one of the key tools in many cases.

Proposition 2.35. [3, Proposition 7] *The poset* $\mathrm{Br}(W)$ *is directed.*

Proof. Let $u, v \in W$. We need to find an element $w \in W$ with $u \leq w$ and $v \leq w$. For that, we induce on $r = l(u) + l(w)$. For $r = 0$ we have $u = v = e$ and can choose $w = e$. So let $r > 0$. Because of symmetry we can assume $l(u) > 0$, hence $u \neq e$ and so there is a $s \in S$ with $us < u$. By induction hypothesis there is a $w \in W$ with $us \leq w$ and $v \leq w$. Consider two cases:

$ws < w$**:** Then $s \in D_R(w)$ and with Lifting Property we have $u = uss \leq w$, so both $u \leq w$ and $v \leq w$.

$ws > w$**:** Then $s \in D_R(ws)$ and $us \leq w < ws$, hence again by Lifting Property we have $u = uss \leq ws$, so both $u \leq ws$ and $v \leq w < ws$. □

Corollary 2.36. [3, Proposition 8]

1. *Let W be finite, then there exists an unique element $w_0 \in W$ with $w \leq w_0$ for all $w \in W$.*
2. *If W contains an element w, with $D_R(w) = S$, then W is finite and w is the unique element w_0.*

Proof. 1. Assume there are two elements $u, v \in W$ of maximal rank. Since $\mathrm{Br}(W)$ is directed, there is an element $w \in W$ with $u \leq w$ and $v \leq w$. Because $\mathrm{Br}(W)$ is graded, we have $l(w) > l(u) = l(v)$, contradicting to the maximality of u and v.

2. We want to show, that $v < w$ for all $v \in W$. For that, we induce on $r = l(v)$. If $r = 0$, then $v = e \leq w$. Let $r > 0$. Then there is a $s \in S$ with $us < u$. By induction, $us \leq w$. Since $s \in D_R(w)$, we have $uss = u \leq w$ by Lifting Property and are done with our induction. This yields $W = [e, w]$ and since by Corollary 2.30 intervals in the Bruhat order are finite, W is finite, too. □

Theorem 2.37. *Let (W, S) be a finite Coxeter system. Then $\mathrm{Br}(W)$ is graded, directed and bounded.*

Proof. $\mathrm{Br}(W)$ is graded due to Corollary 2.33, directed due to Proposition 2.35 and bounded due to Corollary 2.36. □

Corollary 2.38. *Let (W, S) be a Coxeter system and $w, v \in W$ with $w < v$. Then the interval $[w, v]$ is a finite poset, which is finite, graded, directed and bounded.*

Proof. The poset structure and the graduation transfers directly from $Br(W)$. By Corollary 2.30 intervals in $\mathrm{Br}(W)$ are finite. Since v is the unique maximal element and w the unique minimal element, it is bounded. By definition of intervals we have $u \leq v$ for every element $u \in [w, v]$, hence it is directed. □

3 Twisted involutions in Coxeter groups

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

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

3.1 Introduction to twisted involutions

Definition 3.1. An automorphism $\theta : W \to W$ with $\theta(S) = S$ is called a ***Coxeter system automorphism*** of (W, S). We always assume $\theta^2 = \mathrm{id}$.

Definition 3.2. 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 θ-twisted involutions in W is denoted by $\mathcal{I}_\theta(W)$. Often we just omit the Coxeter group and write $\mathcal{I}_\theta$, when it is clear from the context which W is meant.

Example 3.3. Let $\theta = \mathrm{id}_W$. Then θ is an Coxeter system automorphism and the set of all id-twisted involutions coincides with the set of all ordinary involutions of W.

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 3.4. [8, Example 3.2] Let θ be a automorphism of $W \times W$ with $\theta : (u, w) \mapsto (w, u)$. Then θ is an Coxeter system automorphism of the Coxeter system $(W \times W, S \times S)$ and 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 define right now is of great importance to this whole paper, since it is needed to define the poset, the main thesis is about.

Definition 3.5. 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

$$w\underline{s}_1\underline{s}_2 \ldots \underline{s}_k = (\ldots((w\underline{s}_1)\underline{s}_2)\ldots)\underline{s}_k.$$

If $w\underline{s} = \theta(s)ws$, then we say $\underline{s}$ ***acts by twisted conjugation*** on w. Else we say $\underline{s}$ ***acts by multiplication*** on w.

Note that this is no group action. For example let W be a Coxeter group with two generators s, t satisfying $\mathrm{ord}(st) = 3$ and let $\theta = \mathrm{id}$. Then $sts = tst$, but

$$e\underline{sts} = s\underline{ts} = tst\underline{s} = sts\underline{s} = t \neq s = tst\underline{t} = sts\underline{t} = t\underline{st} = e\underline{tst}.$$

Definition 3.6. Let $k \in \mathbb{N}$ and $s_i \in S$ for all $1 \leq i \leq k$. Then an expression $e\underline{s}_1 \ldots \underline{s}_k$, or just $\underline{s}_1 \ldots \underline{s}_k$, is called θ- ***twisted expression***. If θ is clear from the context, we omit θ and call it ***twisted expression***. A twisted expression is called ***reduced twisted expression***, if there is no $k' < k$ with $\underline{s}'_1 \ldots \underline{s}'_{k'} = \underline{s}_1 \ldots \underline{s}_k$.

Lemma 3.7. [8, Lemma 3.4] *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 $\underline{s}$ acts by multiplication on w. Then $\theta(s)ws = w$ and so $l(\theta(s)ws) = l(w)$. Conversely, suppose $l(\theta(s)ws) = l(w)$. If $w\underline{s} = ws$, then we are done. So assume $\theta(s)ws \neq w$. Then w must have a reduced expression beginning with $\theta(s)$ or ending with s (else, we could not have $l(\theta(s)ws) = l(w)$). Without loss of generality suppose that $\theta(s)s_1 \cdots s_k$ is such an expression for w. Since w is a θ-twisted involution, i.e. $\theta(w) = w^{-1}$, we have $l(ws) < l(w)$. Since $l(\theta(s)ws) = l(w)$, no reduced expression for w both begins with $\theta(s)$ and ends with s and hence Exchange Condition yields $ws = s_1 \cdots s_k$, which implies $\theta(s)ws = w$, contradicting to our assumption. $\square$

Lemma 3.8. *We have $l(ws) < l(w)$ iff $l(w\underline{s}) < l(w)$.*

Proof. Suppose $\underline{s}$ acts by multiplication on w. Then $w\underline{s} = ws$ and there is nothing to prove. So suppose $\underline{s}$ acts by twisted conjugation on w. If $l(ws) < l(w)$, then Lemma 2.13 yields $l(ws) + 1 = l(w)$. Assuming $l(w\underline{s}) = l(\theta(s)ws) = l(w)$ would imply, that $\underline{s}$ acts by multiplication on w due to Lemma 3.7, which is a contradiction. So $l(w\underline{s}) = l(\theta(s)ws) < l(w)$. Conversely, suppose $l(w\underline{s}) < l(w)$. Then Lemma 2.13 says $l(w\underline{s}) = l(\theta(s)ws) = l(w) - 2$ and so $l(ws) = l(w) - 1$. □

Lemma 3.9. *For all $w \in W$ and $s \in S$ we have $w\underline{s}\underline{s} = w$.*

Proof. For $w\underline{s}$ there are two cases. Suppose $\underline{s}$ acts by multiplication 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.

Now suppose $\underline{s}$ acts by twisted conjugation 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 $\theta(s)ws \neq w$. Hence the only possible cases yield $w\underline{s}\underline{s} = w$. □

Remark 3.10. Lemma 3.9 allows us to to rewrite equations of twisted expressions. For example

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

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 3.11. *For all $\theta, w \in W$ and $s \in S$ it holds that $w \in \mathcal{I}_\theta$ iff $w\underline{s} \in \mathcal{I}_\theta$.*

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

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

Suppose $\underline{s}$ acts by twisted conjugation 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 $\underline{s}$ acts by multiplication on w. Then

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

Suppose $\underline{s}$ acts by twisted conjugation 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 3.5 is its e-orbit. As the following lemma shows, it coincides with $\mathcal{I}_\theta$.

Lemma 3.12. [8, Proposition 3.5] *The set of θ-twisted involutions coincides with the set of all θ-twisted expressions.*

Proof. By Lemma 3.11, 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 $w = e \in \mathcal{I}_\theta$. So assume $l(w) = r > 0$ and that we have already proven, that every twisted involution $w' \in \mathcal{I}_\theta$ with $\rho(w') < r$ has a twisted expression. If w has a reduced 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 3.8 we get $l(w\underline{s}) < l(w)$. By induction $w\underline{s}$ has a 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 length of an element $w \in \mathcal{I}_\theta$.

Definition 3.13. 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 $w = \underline{s}_1 \dots \underline{s}_k$ exists. This is called the ***twisted length*** of w.

Lemma 3.14. [7, Theorem 4.8] *The Bruhat ordering, restricted to the set of twisted involutions $\mathcal{I}_\theta$, is a graded poset with ρ as rank function. We denote this poset by* $\mathrm{Br}(\mathcal{I}_\theta)$.

We now establish many properties from ordinary Coxeter groups for twisted expressions and $\mathrm{Br}(\mathcal{I}_\theta)$. As seen in Example 3.4 there is a Coxeter system (W', S') and an Coxeter system automorphism θ with $\mathrm{Br}(W) \cong \mathrm{Br}(\mathcal{I}_\theta(W'))$. So the hope, that many properties can be transfered, is eligible.

Lemma 3.15. [8, Lemma 3.8] *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 $w\underline{s}$ we have $\rho(w\underline{s}) = \rho(w) \pm 1$. Now suppose $w\underline{s} < w$. Then we have $\rho(w\underline{s}) < \rho(w)$ iff $w\underline{s} < w$ iff $l(w\underline{s}) < l(w)$ iff $l(ws) < l(w)$ iff $s \in D_R(w)$. □

Lemma 3.16 (Lifting property 2)**.** [8, Lemma 3.9] *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. Whenever a relation comes from the ordinary Lifting Property, we denote it by $<_{LP}$ 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** □

Proposition 3.17 (Exchange property for twisted expressions)**.** [8, Proposition 3.10] *Suppose $\underline{s}_1 \ldots \underline{s}_k$ is a reduced twisted expression. If $\rho(\underline{s}_1 \ldots \underline{s}_k\underline{s}) < k$ for some $s \in S$, then $\underline{s}_1 \ldots \underline{s}_k\underline{s} = \underline{s}_1 \ldots \hat{\underline{s}}_i \ldots \underline{s}_k$ for some $i \in \{1, \ldots, k\}$.*

Proof. Let $w = s_1 \ldots \underline{s}_k$ and $v = s_1 \ldots \underline{s}_k\underline{s}$. Assume $v\underline{s}_k \ldots \underline{s}_{i+1}\underline{s}_i < v\underline{s}_k \ldots \underline{s}_{i+1}$ for all i. Then we would get $\rho(v\underline{s}_k \ldots s_1) < k - k = 0$. Hence there is an index i with $v\underline{s}_k \ldots \underline{s}_{i+1}\underline{s}_i > v\underline{s}_k \ldots \underline{s}_{i+1}$ and we choose i maximal with this property. Since $w > v$ we conclude by repetition of Lifting property 2, that $w\underline{s}_k \ldots \underline{s}_{i+1} \geq v\underline{s}_k \ldots \underline{s}_i$. By Lemma 3.15 we have $\rho(v) = k - 1$ and so $\rho(w\underline{s}_k \ldots \underline{s}_{i+1}) = \rho(v\underline{s}_k \ldots \underline{s}_i)$. Because $\mathrm{Br}(\mathcal{I}_\theta)$ is graded with rank function ρ, both twisted expresssions must represent the same element. Therefore we have $w\underline{s}_k \ldots \underline{s}_{i+1} = v\underline{s}_k \ldots \underline{s}_i$ yielding $v = w\underline{s}_k \ldots \underline{s}_{i+1}\underline{s}_i \ldots \underline{s}_k = \underline{s}_1\hat{\underline{s}}_i \ldots \underline{s}_k$. □

Proposition 3.18 (Deletion property for twisted expressions)**.** [8, Proposition 3.11] *Let $w = s_1 \ldots \underline{s}_k$ be a not reduced twisted expression. Then there are two indices $1 \leq i < j \leq k$ such that $w = \underline{s}_1 \ldots \hat{\underline{s}}_i \ldots \hat{\underline{s}}_j \ldots \underline{s}_k$.*

Proof. Choose j minimal, so we have $\underline{s}_1 \ldots \underline{s}_j$ is not reduced. By Exchange property for twisted expressions there is an index i with $\underline{s}_1 \ldots \underline{s}_j = s_1 \ldots \hat{\underline{s}}_i \ldots \underline{s}_{j-1}$ yielding our hypothesis $w = \underline{s}_1 \ldots \underline{s}_j \ldots \underline{s}_k = \underline{s}_1 \ldots \hat{\underline{s}}_i \ldots \hat{\underline{s}}_j \ldots \underline{s}_k$. □

When applying the Exchange property for twisted expressions to a twisted expression, there is no hint which $\underline{s}_i$ can be omitted. Consider the following situation: Let $w \in \mathcal{I}_\theta$ and $w\underline{s}_1 \ldots \underline{s}_k = w\underline{t}_1 \ldots \underline{t}_k$ two reduced twisted expressions. Then in the twisted expression $w\underline{s}_1 \ldots \ldots \underline{s}_k\underline{t}_k$ we can omit the $\underline{t}_k$ and one other $\underline{s}$ by Exchange property for twisted expressions and get still the same element. It would be nice, when the second omitted $\underline{s}$ is one of the $\underline{s}_i$ in general, but unfortunately this proves to be false:

Example 3.19. Let $W = A_3$, $\theta = \mathrm{id}$ and $w = \underline{s}_3$. Then $w\underline{s}_2\underline{s}_1\underline{s}_2 = w\underline{s}_1\underline{s}_2\underline{s}_3$, but $w\underline{s}_1\underline{s}_2\underline{s}_3\underline{s}_2 \notin \{w\underline{s}_1\underline{s}_2, w\underline{s}_1\underline{s}_3, w\underline{s}_2\underline{s}_3\}$. Hence the omission cannot be choosen after the prefix w, but at least $w\underline{s}_1\underline{s}_2\underline{s}_3\underline{s}_2 = \underline{s}_1\underline{s}_2\underline{s}_3$ works, as guaranteed by Exchange property for twisted expressions.

3.2 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 3.20. We define the set of ***θ-twisted involutions*** as $\{w \in W : \theta(w) = w^{-1}\}$, denoted by $\mathcal{I}_\theta$. If θ is clear from the context we just say ***twisted involutions***. Every element $w \in \mathcal{I}_\theta$ is called a θ-twisted involution, resp. twisted involution.

Definition 3.21. 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 $w = v\underline{s}_1 \ldots \underline{s}_k$ and $\rho(v) = \rho(w) - k$. We call the poset $(\mathcal{I}_\theta, \preceq)$ ***twisted weak ordering***, denoted by $Wk(W, \theta)$. When the Coxeter group W is clear from the context, we just write $Wk(\theta)$.

Lemma 3.22. *The poset $Wk(\theta)$ is a graded poset with rank function ρ.*

Proof. Follows immediately from the definition of $\preceq$. $\square$

By a diagram of a poset $Wk(\theta)$, we do not just mean the ordinary Hasse diagram. Suppose $w, v \in Wk(\theta)$ with $w\underline{s} = v$. We encode the information, if s acts as twisted involution or as multiplication on w, by drawing either a solid or a dashed edge from w to v. For simplification of terminology we still just speak of the Hasse diagram of $Wk(\theta)$. The next example shows such a (extended) Hasse diagram.

Example 3.23. In Figure 3.1 we see the Hasse diagram of $Wk(A_4, \mathrm{id})$. Solid edges represent twisted congujations and dashed edges represent multiplications.

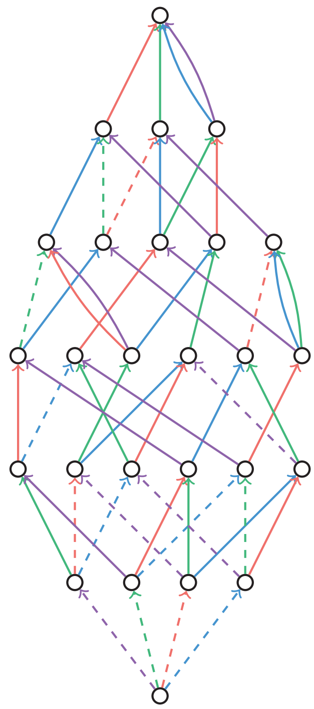

Figure 3.1: Hasse diagram of $Wk(A_4, \mathrm{id})$

Lemma 3.24. *The poset* $\mathrm{Wk}(\theta)$ *is a subposet of* $\mathrm{Br}(\mathcal{I}_\theta)$.

Proof. Both posets are defined on $\mathcal{I}_\theta$. Let $w, v \in \mathcal{I}_\theta$ be two twisted involutions. Assume $w \preceq v$ with $w\underline{s} = v$ for some $s \in S$. If $\underline{s}$ acts by multiplication on w, then $ws = v$ and since $s \in T$ (T the set of all reflections in W) and $l(w\underline{s}) = l(w) + 1$ we have $w \leq v$. If conversely $\underline{s}$ acts by twisted conjugation on w, then $v = \theta(s)ws = w(w^{-1}\theta(s)w)(e^{-1}se)$ and since $w^{-1}\theta(s)w, s \in T$ and $l(w\underline{s}) = l(\theta(s)w) + 1 = l(w) + 2$ we have again $w \leq v$. □

Proposition 3.25. *For all* $w \in \mathcal{I}_\theta$ *and* $s \in S$ *we have* $w\underline{s} \prec w$ *iff* $s \in D_R(w)$ *and* $w\underline{s} \succ w$ *iff* $s \notin D_R(w)$ *as well as* $w\underline{s} < w$ *iff* $s \in D_R(w)$ *and* $w\underline{s} > w$ *iff* $s \notin D_R(w)$.

Proof. We have $w\underline{s}\underline{s} = w$ and $\rho(w\underline{s}) = \rho(w) - 1$ iff $s \in D_R(w)$ and $\rho(w\underline{s}) = \rho(w) + 1$ iff $s \notin D_R(w)$ by Lemma 3.15. By Lemma 3.24 both statements are true for $\mathrm{Br}(\mathcal{I}_\theta)$, too. □

Definition 3.26. Let $v, w \in W$ with $\rho(w) - \rho(v) = n$. A sequence $v = w_0 \prec w_1 \prec \ldots \prec w_n = w$ is called a ***geodesic*** from v to w.

Proposition 3.27. *Let* $v, w \in W$ *with* $v \prec w$. *Then all geodesics from* v *to* w *have the same count of twisted conjugated and multiplicative steps.*

Proof. Suppose we have two geodesics from v to w, where the first has n and the second m multiplicative steps. Then $l(w) + n + 2(k - n) = l(v) = l(w) + m + 2(k - m)$, hence $n = m$. □

Proposition 3.28. *Let* $w \in W$ *and* $w\underline{s} \succ w$. *Then* $|\{t \in S \setminus D_R(w) : w\underline{t} = w\underline{s}\}| \in \{1, 2\}$.

Proof. Suppose $t \in S \setminus D_R(w)$ with $w\underline{t} = w\underline{s}$. Because of the ordinary length either both $\underline{s}$ and $\underline{t}$ act by multiplication on w, or both act by twisted conjugation on w. Suppose they act by multiplication, then $ws = w\underline{s} = w\underline{t} = wt$, hence $s = t$. Conversely, assume they act by twisted conjugation. Then $\theta(s)ws = w\underline{s} = w\underline{t} = \theta(t)wt$. Because of $\theta(t)wtt = \theta(t)w = \theta(s)wst$ we have $l(\theta(s)wst) < l(\theta(s)ws)$ and so by Exchange Condition there are three possible cases

$$\theta(t)w = \theta(s)wst = \begin{cases} \theta(s)w & \Rightarrow s = t, \\ ws & \Rightarrow \theta(t) = wsw^{-1} \text{ or} \\ \theta(s)\overline{w}s & \Rightarrow w = \theta(t)\theta(s)\overline{w}s, \end{cases}$$

where $\overline{w}$ denotes a well choosen subexpression of w. The first case is trivial, the second determines t unambiguously. The third case is impossible, since by Exchange Condition and Remark 2.17 we would have a reduced expression for w beginning with $\theta(s)$ or ending with s (or both), yielding $l(\theta(s)ws) \leq l(w)$, which contradicts to $\rho(w\underline{s}) = \rho(\theta(s)ws) > \rho(w)$. Therefore, there cannot be more than two distinct $s, t \in S \setminus D_R(w)$ with $w\underline{s} = w\underline{t}$. □

Corollary 3.29. *Let* $w \in \mathcal{I}_\theta$ *and* $s, t \in S$ *be two distinct generators. If* $w\underline{s} = w\underline{t}$, *then* $\mathrm{ord}(st) = 2$.

Proof. By the proof of Proposition 3.28 we see, that $w\underline{s} = w\underline{t}$ for two distinct $s, t \in S$ implies, that $\theta(t)w = ws$ holds and that $\underline{s}$ and $\underline{t}$ act by twisted conjugation on w. Since $\theta(w) = w^{-1}$, we also have $\theta(s)w = wt$ by

$$\theta(t)w = ws \iff \theta(\theta(t)w) = \theta(ws) \iff tw^{-1} = w^{-1}\theta(s) \iff wt = \theta(s)w.$$

Hence we have $wts = \theta(s)ws = \theta(t)wt = wst$, yielding $st = ts$ and $\mathrm{ord}(st) = 2$. □

TODO

3.3 Residues

Definition 3.30. Let $w \in \mathcal{I}_\theta$ and $I \subseteq S$ be a subset of generators. Then we define

$$wC_I := \{w\underline{s}_1 \dots \underline{s}_k : k \in \mathbb{N}_0, s_i \in S\}$$

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

Example 3.31. Let $w \in \mathcal{I}_\theta$. Then $wC_\emptyset = \{w\}$ and $wC_S = \mathcal{I}_\theta$.

Lemma 3.32. *Let $w \in \mathcal{I}_\theta$ and $I \subset S$. If $v \in wC_I$, then $vC_I = wC_I$.*

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

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

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 3.33. *Let $v, w \in \mathcal{I}_\theta$ and $I \subset S$. Then either $vC_I \cap wC_I = \emptyset$ or $vC_I = wC_I$.*

Proof. Immediately follows from Lemma 3.32. □

Proposition 3.34. [8, Lemma 5.6] *Let $w \in \mathcal{I}_\theta$, $I \subseteq S$ be a set of generators. Then there exists a unique element $w_0 \in wC_I$ with $w_0 \preceq w_0\underline{s}$ for all $s \in I$.*

Proof. Suppose there is no such element. Then for each $w \in wC_I$ we can find a $s \in I$ with $w' = w\underline{s} \preceq w$ and $e' \in wC_I$. By repetition of Deletion property for twisted expressions we get, that $e \in wC_I$, but e has the property, which we assumed, that no element in wC_I has. Hence there must be at least one such element. Now suppose there are two distinct elements u, v with the desired property. Note that this means, that u and w have no reduced twisted expression ending with some $\underline{s} \in I$. Let v have a reduced twisted expression $v = \underline{s}_1 \dots \underline{s}_k$. Since u and v are both in wC_I there must be a twisted v-expression for u

$$u = v\underline{s}_{k+1} \dots \underline{s}_{k+l} = \underline{s}_1 \dots \underline{s}_{k+l}$$

with $s_n \in I$ for $k+1 \leq n \leq k+l$. This twisted expression cannot be reduced, since it ends with $\underline{s}_{k+l} \in I$. Then Deletion property for twisted expressions yields that this twisted expression contains a reduced twisted subexpression for u. It cannot end with $\underline{s}_n$ for $k+1 \leq n \leq k+l$. Hence, it is a twisted subexpression of $\underline{s}_1 \dots \underline{s}_k = v$, too. So $u \leq v$ by Subword property. Because of symmetry we have $v \leq u$ and so $u = v$, contradicting to our assumption $u \neq v$. □

Corollary 3.35. *Let $w \in \mathcal{I}_\theta$, $I \subseteq S$ be a set of generators and let $\rho_{min} := \min\{\rho(v) : v \in wC_I\}$ be the minimal twisted length within the residue wC_I. Then there is a unique element $w_{min} \in wC_I$ with $\rho(w_{min}) = \rho_{min}$. We denote this element by $\min(w, I)$.*

Proof. The minimal rank ρ_{min} exists, since the image of ρ is in $\mathbb{N}_0$, which is well-ordered, and $wC_I \neq \emptyset$. Suppose we have an element w_{min} with $\rho(w_{min}) = \rho_{min}$. This means, that in particular all $w_{min}\underline{s}$ with $s \in I$ must be of larger twisted length, i.e. $w_{min} \prec w_{min}\underline{s}$ for all $s \in I$. With Proposition 3.34 this element must be unique. □

We proceed with some properties of rank-2-residues. Our interest in these residues stems from the fact, that their properties are needed later in Section 3.4 to construct an effective algorithm for calculating the twisted weak ordering, i.e. calculating the Hasse diagram of $Wk(W, \theta)$ for arbitrary Coxeter systems (W, S) and Coxeter system automorphisms θ.

Definition 3.36. Let $s, t \in S$ be two distinct generators. We define:

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

This definition allows us to express rank-2-residues differently. Suppose we have an element $w \in \mathcal{I}_\theta$ and two distinct generators $s, t \in S$. Thanks to Lemma 3.32 and Corollary 3.35 we can assume, that $w = min(w, \{s, t\})$. Then

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

This encourages the following definition.

Definition 3.37. Let $w \in \mathcal{I}_\theta$ and let $s, t \in S$ be two distinct generators. Suppose $w = min(w, \{s, t\})$. Then we call $\{w[\underline{st}]^n : n \in \mathbb{N}\}$ the s***-branch*** and $\{w[\underline{ts}]^n : n \in \mathbb{N}\}$ the t***-branch*** of $wC_{\{s,t\}}$.

One question arises immediately: Are the s- and the t-branch disjoint? With the following propositions, corollaries and lemmas we will get a much better idea of the structure of rank-2-residues and answer this question.

Proposition 3.38. *Let $w \in \mathcal{I}_\theta$ and let $s, t \in S$ be two distinct generators. Without loss of generality suppose $w = \min(w, \{s, t\})$. If there is a $v \in wC_{\{s,t\}}$ with $v\underline{s} \prec v$ and $v\underline{t} \prec v$, then it is unique with this property in $wC_{\{s,t\}}$. Hence $wC_{\{s,t\}}$ consists of two geodesics from w to v intersecting only in these two elements. Else, the s- and t-branch are disjoint, strictly ascending in twisted length and of infinite size.*

Proof. Suppose there is a v in the s-branch with $v\underline{s} \prec v$ and $v\underline{t} \prec v$, say $v = w[\underline{st}]^n$ and n is minimal with this property. Because of the uniqueness of a minimal element from Proposition 3.34 we have $w[\underline{st}]^{m+1} \prec w[\underline{st}]^m$ for all $m \in \mathbb{N}$ with $n \leq m \leq 2n-1$. With the same argument we have $w[\underline{st}]^{2n} = w$. If no such v exists, then the s- and t-branch must be disjoint, strictly ascending in twisted length and so of infinite size. □

The assertion that Proposition 3.38 makes can be thought of some kind of convexity of rank-2-residues. A rank-2-residue cannot have a concave structure like in Figure 3.2.

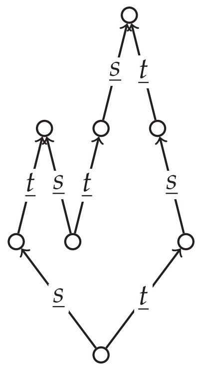

Figure 3.2: Impossible concave structure of a rank-2-residues

Proposition 3.39. *Let $w \in S$ and $s, t \in S$ be two distinct generators with $w\underline{s} \prec w$. If $\underline{s}$ acts by multiplication on w, then $w\underline{st} \succ w\underline{s}$ or $w\underline{t} \prec w$.*

Proof. Suppose $w\underline{st} \prec w\underline{s} \prec w$, hence $l(w\underline{st}) < l(w\underline{s}) < l(w)$ in particular. If $\underline{t}$ acts by multiplication on $w\underline{s}$, then we have $l(w\underline{st}) = l(\theta(s)(wt)) = l(w) - 2$. If it acts by twisted conjugation, then we have $l(w\underline{st}) = l(\theta(t)\theta(s)(wt)) = l(w) - 3$. In both cases we have $l(wt) < l(w)$, hence $t \in D_R(w)$ and so $w\underline{t} \prec w$. □

Note that this proposition could be strengthen by insisting on an exclusive or, since we cannot have both cases at the same time. By the proof of Proposition 3.28 we see that we cannot have $w\underline{st} = w$, since double edges are always twisted conjugations. Hence having $w\underline{st} \succ w\underline{s} \prec w \succ w\underline{t}$ would contradict to the convexity from Proposition 3.38. The next corollary ensures that multiplicative actions in $Wk(\theta)$ can only occur at the top or bottom end of rank-2-residues.

Corollary 3.40. *Let $w \in S$ and let $s, t \in S$ be two distinct generators and suppose $\underline{s}$ acts by multiplication on w. Then w or $w\underline{s}$ is the unique minimal or maximal element in $wC_{\{s,t\}}$.*

Proof. Suppose w is not maximal, i.e. $w\underline{t} \succ w$. Then by Proposition 3.39 we have $w\underline{st} \succ w\underline{s}$, hence $w\underline{s}$ is minimal. Suppose w is not minimal, i.e. $w\underline{st} \prec w\underline{s}$. Then with the same argument we have $w\underline{t} \prec w$, hence w is maximal. Supposing $w\underline{s}$ not to be maximal or not to be minimal yields analogue results. □

Again, this corollary can be strengthen by insisting on an exclusive or with the same arguments as before.

Definition 3.41. Let $w \in \mathcal{I}_\theta$, $s, t \in S$ be two distinct generators with $\mathrm{ord}(st) < \infty$ and $C := wC_{\{s,t\}}$ the corrosponding rank-2-residue. We classify rank-2-residues according to Figure 3.3.

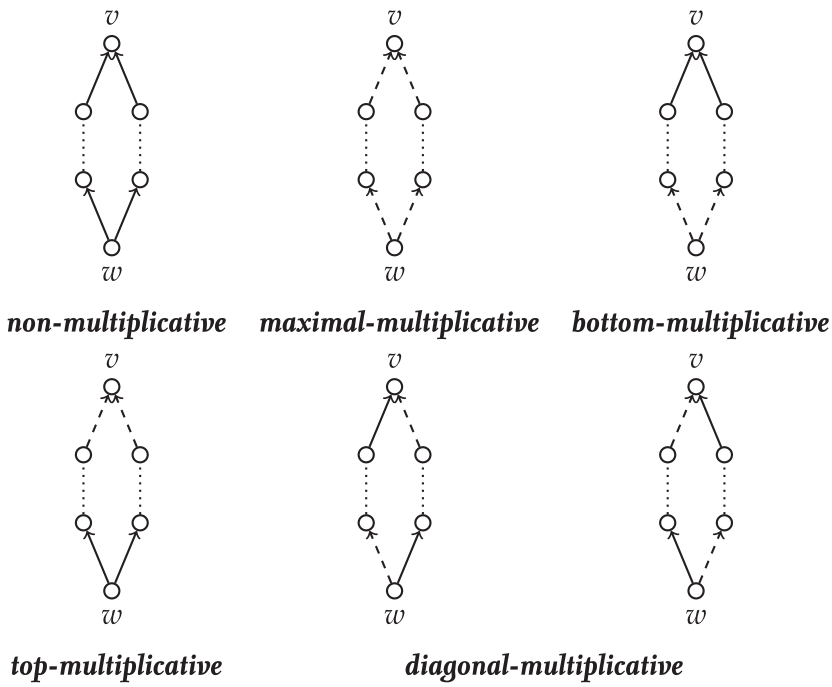

Figure 3.3: Classification of rank-2-residues

Lemma 3.42. *Let $s, t \in S$ be two distinct generators and $w \in S$ with $w = min(w, \{s, t\})$. Suppose $v \in wC_{\{s,t\}}$ with $v\underline{s} \prec v$ and $v\underline{t} \prec v$. Then $wC_{\{s,t\}}$ is either non-, maximal-, bottom-, top- or diagonal-multiplicative. In particular the twisted conjugations and mulitplications are distributed axisymmetrically or pointsymmetrically.*

Proof. If u covers w, then there are only two edges and the assumption holds. So suppose $wC_{\{s,t\}}$ contains at least four edges. Due to Corollary 3.40 the actions by multiplication can only occure next to w and v. Hence there are $2^4 = 16$ configurations possible. Proposition 3.27 wipes out ten out of the 16 configurations. The remaining are those from Figure 3.3. □

Example 3.43. In Figure 3.4 we see two Hasse diagrams of $Wk(A_4, \mathrm{id})$. The left one only contains edges with labels s_1, s_2, the middle one only edges with labels s_1, s_3 and the right one only edges with labels s_1, s_4.

Corollary 3.44. *Let $w \in \mathcal{I}_\theta$ with $\rho(w) = k$, s, t be two distinct generators and $s \notin D_R(w)$. Suppose $w[\underline{ts}]^{2n-1} = w\underline{s}$ and suppose n to be the smallest number with this property. Then $w[\underline{ts}]^{n-1}$*

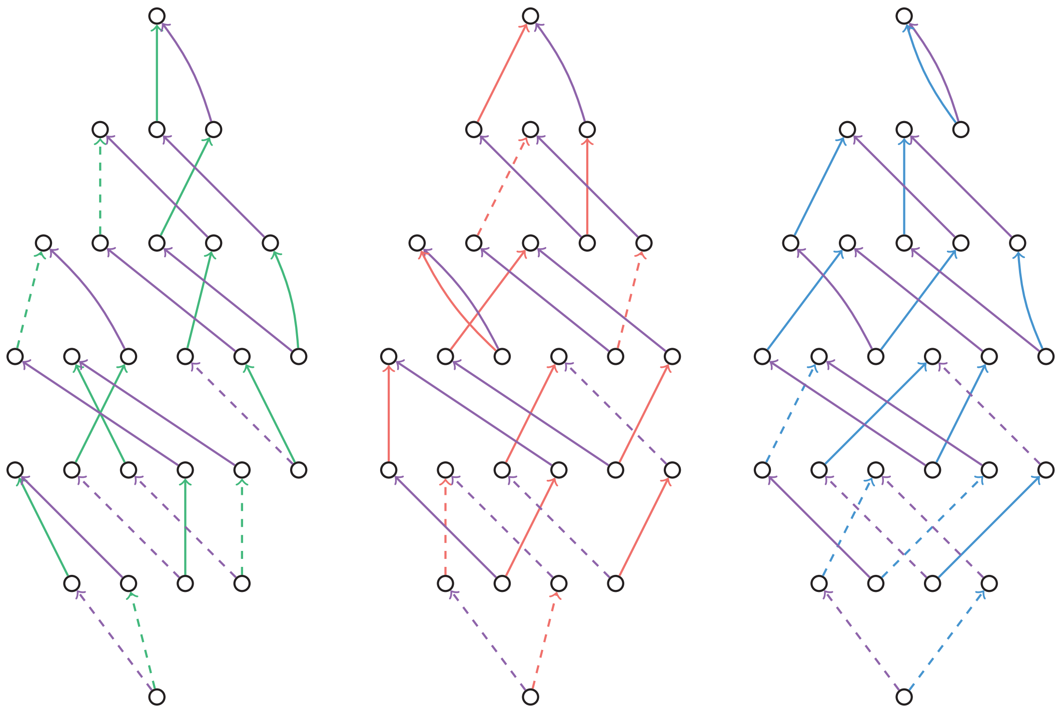

Figure 3.4: Hasse diagrams of $Wk(A_4, \mathrm{id})$ after removing s_3, s_4 edges in the left, s_2, s_4 edges in the middle and s_2, s_3 edges in the right diagram

is the minimal element $\min(w, \{s,t\})$ *and* $w[\underline{ts}]^{2n-1}$ *is the maximal element. Define*

$$\begin{aligned} a &= l(w\underline{s}) - l(w), \\ b &= l(w[\underline{ts}]^{n-1}) - l(w[\underline{ts}]^{n-2}), \\ c &= l(w[\underline{ts}]^{n}) - l(w[\underline{ts}]^{n-1}) \text{ and} \\ d &= l(w[\underline{ts}]^{2n-1}) - l(w[\underline{ts}]^{2n-2}). \end{aligned}$$

Note that $a, b, c, d \in \{1, 2\}$ *contain the information, if edges next to the minimal and the maximal element of* $wC_{\{s,t\}}$ *are twisted conjugations or multiplications. Then each can be deduced from the three remaining ones with the equation* $a + b = c + d$.

Proof. The minimality of $w[\underline{ts}]^{n-1}$ and the maximality of $w[\underline{ts}]^{2n-1}$ is due to Proposition 3.38. The soundness of the equation follows from the symmetric distribution of twisted conjugations and mutipliations from Lemma 3.42. □

Lemma 3.45. *Let* $w \in S$, $s, t \in S$ *be two distinct generators and* $m = \mathrm{ord}(st) < \infty$. *Then* $|wC_{\{s,t\}}| \leq 2m$.

Proof. Let w be the Wk-minimal element and v be the Wk-maximal element in our residue. Due to Lemma 3.42 there are five different cases we have to consider:

Non-multiplicative: We have $w(\underline{st})^m = (ts)^m w(st)^m = w$.

Maximal-multiplicative: Due to $\theta(s)w = ws$ and $\theta(t)w = wt$ we have

$$w(\underline{st})^{m/2+1} = \theta(\hat{t}(st)^{m/2-1}\hat{s})w(st)^{m/2+1} = w(st)^m = w.$$

(**TODO** Show that this situation only occurs for even m)

Bottom-multiplicative: Again we are in a case, where $\theta(s)w = ws$ and $\theta(t)w = wt$ hold. Hence we have

$$w(\underline{st})^{(m+1)/2} = \theta(\hat{t}(st)^{(m-1)/2}\hat{s})w(st)^{(m+1)/2} = w(st)^m = w.$$

(**TODO** Show that this situation only occurs for odd m)

Top-multiplicative: Analogue to the previous case, if we start from u instead of w.

Diagonal-multiplicative: Suppose m is even. Then we have

$$w(\underline{st})^m = \theta(\underbrace{ts\cdots st}_{m-1}\hat{s}\underbrace{ts\cdots st}_{m-1}\hat{s})w(st)^m = \theta(\underbrace{ts\cdots s}_{m-2}\underbrace{s\cdots st}_{m-2})w = \ldots = w.$$

If m is odd, then we have the completely analogue situation

$$w(\underline{st})^m = \theta(\underbrace{ts\cdots ts}_{m-1}\hat{t}\underbrace{st\cdots st}_{m-1}\hat{s})w(st)^m = \theta(\underbrace{ts\cdots t}_{m-2}\underbrace{t\cdots st}_{m-2})w = \ldots = w.$$

So in all cases we have $w(\underline{st})^k = w$ for a $k \leq \operatorname{ord}(st)$ and hence the residue can have at most $2 \cdot \operatorname{ord}(st)$ many distinct elements. $\square$

Proposition 3.46. *Let $w \in S$ and $s, t \in S$ be two distinct generators with $\operatorname{ord}(st) < \infty$. Suppose $k \in \mathbb{N}$ to be the smallest number with $w = w(\underline{st})^k$. Then for any $n \in \mathbb{N}$ with $w = w(\underline{st})^n$ we have $k \mid n$.*

Proof. Let $n = qk + r$ for $q \in \mathbb{N}_0$ and $r \in \{0, \ldots, k-1\}$. Then

$$w = w(\underline{st})^n = w(\underline{st})^{qk+r} = w(\underline{st})^{qk}(\underline{st})^r = w(\underline{st})^{q(k-1)}(\underline{st})^r = \ldots = w(\underline{st})^r.$$

For $r > 0$ we would have a contradiction to the minimality of k, hence $r = 0$, $q > 0$ and therefore $k \mid n$. $\square$

Corollary 3.47. *Let $w \in S$ and $s, t \in S$ be two distinct generators with $w\underline{s} \neq w\underline{t}$. Suppose $w = w(\underline{st})^m = w(\underline{st})^n$. Then $\gcd(m, n) > 1$.*

Proof. Let k be the same as in Proposition 3.46. Since $w\underline{s} \neq w\underline{t}$ we have $k > 1$. Both, $k \mid n$ and $k \mid m$, hence $\gcd(m, n) \geq k > 1$. $\square$

This constraints the possible size of rank-2-residues.

3.4 Twisted weak ordering algorithms

Now we address the problem of calculating $Wk(\theta)$ for an arbitrary Coxeter group W, given in form of a set of generating symbols $S = \{s_1, \ldots s_n\}$ and the relations in form of $m_{ij} = \mathrm{ord}(s_i s_j)$. From this input we want to calculate the Hasse diagram, i.e. the vertex set $\mathcal{I}_\theta$ and the edges labeled with $\underline{s}$. Thanks to Lemma 3.12 the vertex set can be obtained by walking the e-orbit of the action from Definition 3.5. The only element of twisted length 0 is e. Suppose we have already calculated the Hasse diagram until the twisted length k, i.e. we know all vertices $w \in \mathcal{I}_\theta$ with $\rho(w) \leq k$ and all edges connecting two vertices u, v with $\rho(u) + 1 = \rho(v) \leq k$. Let $\rho_k := \{w \in \mathcal{I}_\theta : \rho(w) = k\}$. Then all vertices in ρ_{k+1} are of the form $w\underline{s}$ for some $w \in \rho_k, s \in S$. For each $(w, s) \in \rho_k \times S$, we calculate $w\underline{s}$. If $\rho(w\underline{s}) = k+1$ then $w \prec w\underline{s}$. To avoid having to check the twisted length we use Lemma 3.15. We already know the set $S_w \subseteq S$ of all generators yielding an edge into w. Due to the lemma we have $\rho(w\underline{s}) = k - 1$ for all $s \in S_w$ and $\rho(w\underline{s}) = k + 1$ for all $s \in S \setminus S_w$. Hence we only calculate $w\underline{s}$ for $s \in S \setminus S_w$ and know $w \prec w\underline{s}$ without checking the twisted length explicitly. The last problem to solve is the possibility of two different $(w, s), (v, t) \in \rho_k \times S$ with $w\underline{s} = v\underline{t}$. To deal with this, we have to compare a potential new twisted involution $w\underline{s}$ with each element of twisted length $k+1$, already calculated. The concrete problem of comparing two elements in a free presented group, called ***word problem for groups***, will not be addressed here. We suppose, that whatever computer system is used to implement our algorithm, supplies a suitable way to do that. The only thing to note is that solving the wordproblem is not a cheap operation. Reducing the count of element comparisions is a major demand to any algorithm, calculating $Wk(\theta)$.

The steps discussed have been compiled in to an algorithm by [1, Algorithm 2.4] and [5, Algorithm 3.1.1]. We take this as our starting point. Since the runtime is far from being optimal, we use the structural properties of rank-2-residues from Section 3.3 to improve the algorithm. As we will show, these optimizations yield an algorithm with an asymptotical perfect runtime behavior. TWOA1 shows this algorithm.

Algorithm 3.48 (TWOA1).

procedure TwistedWeakOrderingAlgorithm1($(W, S), k_{max}$)
  $V \leftarrow \{(e, 0)\}$
  $E \leftarrow \{\}$
  for $k \leftarrow 0$ **to** k_{max} **do**
    for all $(w, k_w) \in V$ **with** $k_w = k$ **do**
      for all $s \in S$ **with** $\nexists(\cdot, w, s) \in E$ **do** ▷ Only for $s \notin D_R(w)$
        $y \leftarrow ws$
        $z \leftarrow \theta(s)y$
        if $z = w$ **then**
          $x \leftarrow y$
          $t \leftarrow s$
        else
          $x \leftarrow z$

14: $t \leftarrow \underline{s}$
15: **end if**
16: $isNew \leftarrow$ **true**
17: **for all** $(w', k_{w'}) \in V$ **with** $k_{w'} = k+1$ **do** ▷ Check if x already known
18: **if** $x = w'$ **then**
19: $isNew \leftarrow$ **false**
20: **end if**
21: **end for**
22: **if** $isNew =$ **true then**
23: $V \leftarrow V \cup \{(x, k+1)\}$
24: **end if**
25: $E \leftarrow E \cup \{(w, x, t)\}$
26: **end for**
27: **end for**
28: $k \leftarrow k+1$
29: **end for**
30: **return** (V, E) ▷ The poset graph
31: **end procedure**

Note, that if W is finite, k_{max} does not have to be evaluated explicitly. When k reaches the maximal twisted length in $Wk(\theta)$, then the only vertex of twisted length k is the unique element $w_0 \in W$ of maximal ordinary length. Since $s \in D_R(w_0)$ for all $s \in S$, there is no $s' \in S$ remaining to calculate $w_0\underline{s}'$ for. This condition can be checked to terminate the algorithm without knowing k_{max} before. When W is infinite, there is no maximal element and $\mathcal{I}_\theta$ is infinite, too. In this case k_{max} is used to terminate after having calculated a finite part of $Wk(\theta)$.

Lemma 3.49. *TWOA1 is a deterministic algorithm.*

Proof. The outer loop (line 4) is strictly ascending in $k \in \{0, \dots, k_{max}\}$ and so finite. The innermost loop (line 6) is finite since S is finite and the inner loop (line 5) is finite, since V starts as finite set and in each step there are added at most $|V| \cdot |S|$ many new vertices. Therefore the algorithm terminates. The soundness is due to the arguments at the beginning of Section 3.4. □

Lemma 3.50. *Let $k \in \mathbb{N}$, $n = |\{w \in \mathcal{I}_\theta : \rho(w) \leq k\}|$. Then $TWOA1 \in \mathcal{O}(n^2/k)$.*

Proof. Let $\rho_i = |\{w \in \mathcal{I}_\theta : \rho(w) = i\}|$ for $0 \leq i \leq k$. Our algorithm has to do at least $\rho_i(\rho_i - 1)/2$ many element comparisons (line 17) for each $0 \leq i \leq k$. Set $m = \lfloor \frac{n}{k} \rfloor$. In the most optimistic case we have $\rho_i \geq m$ for all i. In practice the situation will be worse, since some ρ_i will be smaller than m (for example $\rho_0 = 1$) and so some ρ_i will be much larger than m. This optimistic case yields at least $m(m-1)/2 \cdot k$ many element comparisons. Hence regarding the most delimiting operation, the element comparison, our algorithm is in $\Omega(m^2 k) = \Omega(n^2/k)$. The element comparison at line 9 done at most $n \cdot |S|$. Other operations, like for example insertion into or searching in sets can be considered super linear, if for example sets are ordered immediately at insertion and then searching is done with binary search. So the algorithm is in $\mathcal{O}(n^2/k)$. □

Any algorithm calculating $Wk(\theta)$ must be at least linear in the size of $Wk(\theta)$. Our goal is to improve TWOA1 so that we get an algorithm in $\mathcal{O}(|Wk(\theta)|)$, i.e. an asymptotical perfect algorithm for calculating $Wk(\theta)$. As already seen the element comparison of a potential new element with all already known elements of same twisted length (line 17) is the bottleneck. Here the rank-2-residues become key. Suppose we have a $w \in \mathcal{I}_\theta$ with $\rho(w) = k$ and $s \in S$. In TWOA1 we would now check, if $w\underline{s}$ is a new vertex, or if we already calculated it by comparing it with all already known vertices of twisted length $k+1$. Assume we have already calculated it. This means there is another twisted involution v with $\rho(v) = k$ and another generator $t \in S$ with $v\underline{t} = w\underline{s}$. With Proposition 3.38 $w\underline{s}$ is the unique element of maximal twisted length in the rank-2-residue $wC_{\{s,t\}}$. This yields a necessary condition for $w\underline{s}$ to be equal to a already known vertex, allowing us to replace the ineffective search all method in TWOA1 at line 17.

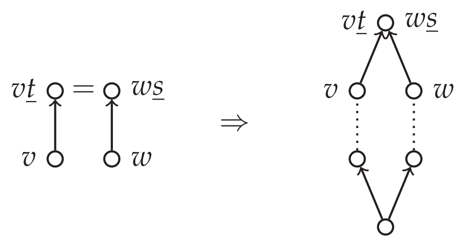

Figure 3.5: Optimization of TWOA1

Lemma 3.51. *Let $k \in \mathbb{N}$ and suppose we are in the sitation described at the beginning of Section 3.4. Let $\rho_i := \{w \in \mathcal{I}_\theta : \rho(w) = i\}$ and ρ'_{k+1} the set of the already calculated vertices with twisted length $k+1$. If $w\underline{s} \in \rho'_{k+1}$ for some $w \in \rho_k, s \in S$, say $w\underline{s} = v\underline{t}$ with $v \in \rho_k$ and $t \in S \setminus \{s\}$, then $w\underline{s} = w[\underline{ts}]^n$ for some $n \in \mathbb{N}$ with $w[\underline{ts}]^j \in \rho_0 \cup \ldots \cup \rho_k \cup \rho'_{k+1}$ for $1 \le j \le n$.*

Proof. The equality $w\underline{s} = w[\underline{ts}]^n$ for some $n \in \mathbb{N}$ is due to Proposition 3.38. All vertices in this rank-2-residue except $v\underline{t}$ have a twisted length of k or lower. For $v\underline{t}$ we supposed it is already known, hence $v\underline{t} \in \rho'_{k+1}$. Therefore all vertices $w[\underline{ts}]^j$, $1 \le j \le n$ are in $\rho_0 \cup \ldots \cup \rho_k \cup \rho'_{k+1}$. □

This can be checked effectively. Both, w and s are fixed. Start with $M = \emptyset$. For all already known edges from or to w being labeled with $\underline{t} \in \underline{S} \setminus \{\underline{s}\}$ we do the following: Walk $w[\underline{ts}]^i$ for $i = 0, 1, \ldots$ until $\rho(w[\underline{ts}]^i) = k+1$. Note that walking in this case really means walking the graph. All involved vertices and edges have already been calculated. So there is no need for more calculations in W to find $w[\underline{ts}]^i$. By Proposition 3.38 such a path must exist (in a completely calculated graph). But we could be in the case, where the last step from $w[\underline{ts}]^{i-1}$ to $w[\underline{ts}]^i$ has not been calculated yet. If it is already calculated, then add this element to M by setting $M = M \cup \{w[\underline{ts}]^i\}$. If not, do not add it to M.

Now M contains all already known elements of twisted length $k+1$, satisfying the necessary condition from Lemma 3.51. Furthermore $|M| < |S|$. So for each pair (w, s) we have to do at most $|S| - 1$ many element comparisons the determine, if $w\underline{s}$ is new or already known, no matter how many elements of twisted length $k+1$ are already known. This can be used to massively improve TWOA1:

Algorithm 3.52 (TWOA2)**.** Exactly like TWOA1 expect for line 17: Here we do not iterate over all already calculated vertices with twisted length $k+1$, but just over those, that can be reached by $w(\underline{st})^n$ for some $t \in S \setminus \{s\}$ and $n \in \mathbb{N}$.

Lemma 3.53. *TWOA2 is a deterministic algorithm.*

Proof. It terminates since TWOA1 terminates. In comparison to TWOA1 we do not compare x to all already known elements w' with $\rho(w') = k+1$, but just with a few. The soundness of this improvement is due to Lemma 3.51. □

Lemma 3.54. *Let $k \in \mathbb{N}$, $n = |\{w \in \mathcal{I}_\theta : \rho(w) \leq k\}|$. Then $TWOA2 \in \mathcal{O}(n)$.*

Proof. We can consider the rank of W to be a constant, since it is tiny in comparison to n. The loop of TWOA1, that increased it runtime above linear, was the one in line 17. We restricted it to have at most $|S|$ cases, hence it can be considered to have constant runtime, too. □

Many more explicit element comparisons can be avoided. In some cases we can deduce the equality $v\underline{t} = w\underline{s}$ as well as $l(w\underline{s}) - l(w)$ just from the already calculated structure of the rank-2-residue $wC_{\{s,t\}}$, while in other cases we can preclude that $v\underline{t}$ equals $w\underline{s}$. The following two corollaries show examples of restrictions, that rank-2-residues are subjected to:

Corollary 3.55. *Let $w \in \mathcal{I}_\theta$ with $\rho(w) = k$, s, t be two distinct generators and $s \notin D_R(w)$. Suppose $n \in \mathbb{N}$ to be the smallest number for that $\rho(w[\underline{ts}]^{2n-1}) = k+1$ holds. Then:*

1. *If $n = \operatorname{ord}(st)$, then $w[\underline{ts}]^{2n-1} = w\underline{s}$.*
2. *If $n \geq 2$ and $l(w[\underline{ts}]^{2n-1}) - l(w[\underline{ts}]^{2n-2}) = 1$, then $w[\underline{ts}]^{2n-1} = w\underline{s}$.*

Proof.
1. Follows immediately from Lemma 3.45.
2. Because of the length difference the step from $w[\underline{ts}]^{2n-2}$ to $w[\underline{ts}]^{2n-1}$ is a multiplication, not a twisted conjugation, and because of $n \geq 1$ this step cannot be next to the smallest element in $wC_{\{s,t\}}$. Hence $w[\underline{ts}]^{2n-1} = w\underline{s}$ by Corollary 3.40. □

Corollary 3.56. *Let $w \in S$ and $s, t \in S$ be two distinct generators. Then the following table shows all possible $n \in \mathbb{N}$ with $w(\underline{st})^n = w$ regarding $\operatorname{ord}(st)$ and the distribution of multiplications and twisted conjugations in $wC_{\{s,t\}}$ (see Figure 3.3).*

	ord(st)						
	2	3	4	5	6	7	8
non-multiplicative	1,2	3	2,4	5	2,3,4,6	7	2,4,6,8
diagonal-multiplicative	2	3	2,4	5	2,3,4,6	7	2,4,6,8
maximal-multiplicative	2	–	3	–	2,4	–	5
bottom- and top-multiplicative	–	2	–	3	–	2,4	–

Proof. In each case we get a m with $w = (\underline{s}\underline{t})^m$ from the proof of Lemma 3.45. By Corollary 3.47 any n with this property has a non trivial divisor in common with m, if $w\underline{s} \neq w\underline{t}$. The situation $w\underline{s}\underline{t} = w$ for $s \neq t$ can only occur, if $\operatorname{ord}(st) = 2$ and if $\underline{s}$ and $\underline{t}$ act by twisted conjugation on w due to Corollary 3.29 and the proof of Proposition 3.28. □

We use these restrictions to further improve TWOA2:

Algorithm 3.57 (TWOA3). **TODO**

Lemma 3.58. *TWOA3 is a deterministic algorithm.*

Proof. **TODO** □

Lemma 3.59. *Let $k \in \mathbb{N}$, $n = |\{w \in \mathcal{I}_\theta : \rho(w) \leq k\}|$. Then $TWOA3 \in \mathcal{O}(n)$.*

Proof. **TODO** □

3.5 Implementing the twisted weak ordering algorithms

In this section we will look at a concrete implementation of the algorithm TWOA1 from [1] and [5] and of the improved versions TWOA2 and TWOA3, that we have just introduced. The source codes of the test implementations can be found in the appendix, Section A. They are written in GAP[1], a System for Computational Discrete Algebra. It supplies a powerful programming language and can handle with free represented groups, in particular it allows comparisons of elements in such groups. The following algorithm benchmarks have been executed on a computer running Debian Linux in Verion 6.0.5 with an Intel® Core™ i7-965 CPU with four cores at 3.2 GHz and 8 GiB RAM. The version 4.5.5 of GAP is used. Note that our implementations do not support multithreding.

At first we compare the count of element comparisons needed for our three algorithms. For this we calculate $Wk(W, \mathrm{id})$ for a selection of finite Coxeter systems and count the comparisons. In Figure 3.6 we see the count of needed element comparisons plotted against the size of the set of id-twisted involutions.

The first observation is the much lower count of needed element comparisons of TWOA2 and TWOA3 in comparison to TWOA1, just as we intended it with our improvements. Our implementations represents Coxeter systems of type A_n as $\mathrm{Sym}(n+1)$ while representing the Coxeter systems of other types as arbitrary free represented groups. Hence in our case element comparison in A_n is very effective, while the element comparison in other types is very ineffective and therefore comparing the runtimes for A_n with the runtimes of

[1] See `http://www.gap-system.org/`.

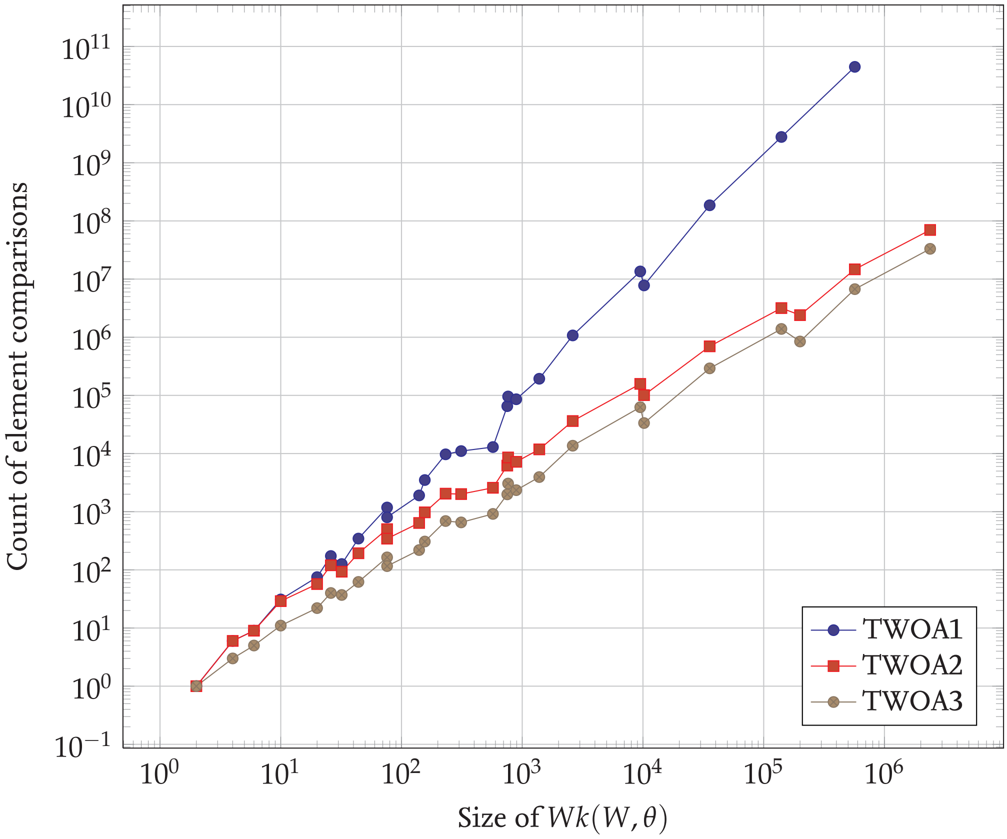

Figure 3.6: Element comparisons needed in TWOA1/2/3 with $\theta = \mathrm{id}$

other types is senseless. Figure 3.7 plots the runtimes against the size of $Wk(\theta)$ for Coxeter groups of type A_n and Figure 3.8 for the other types. The complete table of benchmark results can be found in the appendix, Section B.

For $W = A_n$ with $n < 9$ TWOA1 is faster than our improved versions. But as already seen, TWOA1 is quadratic in the size of $Wk(\theta)$, while TWOA2 and TWOA3 are linear and so for larger n our improvements start to pay off. In case $W \neq A_n$ we have essentially the situation that TWOA3 is faster than TWOA2 while TWOA2 is faster than TWOA1.

4 Main Thesis

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 $\theta(K) = K$. Denote the largest element in $\langle K \rangle \leq W$ by w_K. Futhermore let $S_1, S_2, S_3 \subset S$ be three sets of generators. Define $S_{ij} = S_i \cap S_j$ and $T = S_1 \cap S_2 \cap S_3$. For which Coxeter groups W does the implication*

$$\forall\, 1 \leq i < j \leq 3 : w \in w_K C_{S_{ij}} \quad \Rightarrow \quad w \in w_K C_T \tag{$*$}$$

hold for any possible K, θ, S_1, S_2, S_3 and w?

The reader might wonder, why we handle with intersections of sets of generators and not just with arbitrary sets of generators. The reason for that is also the main reason, why

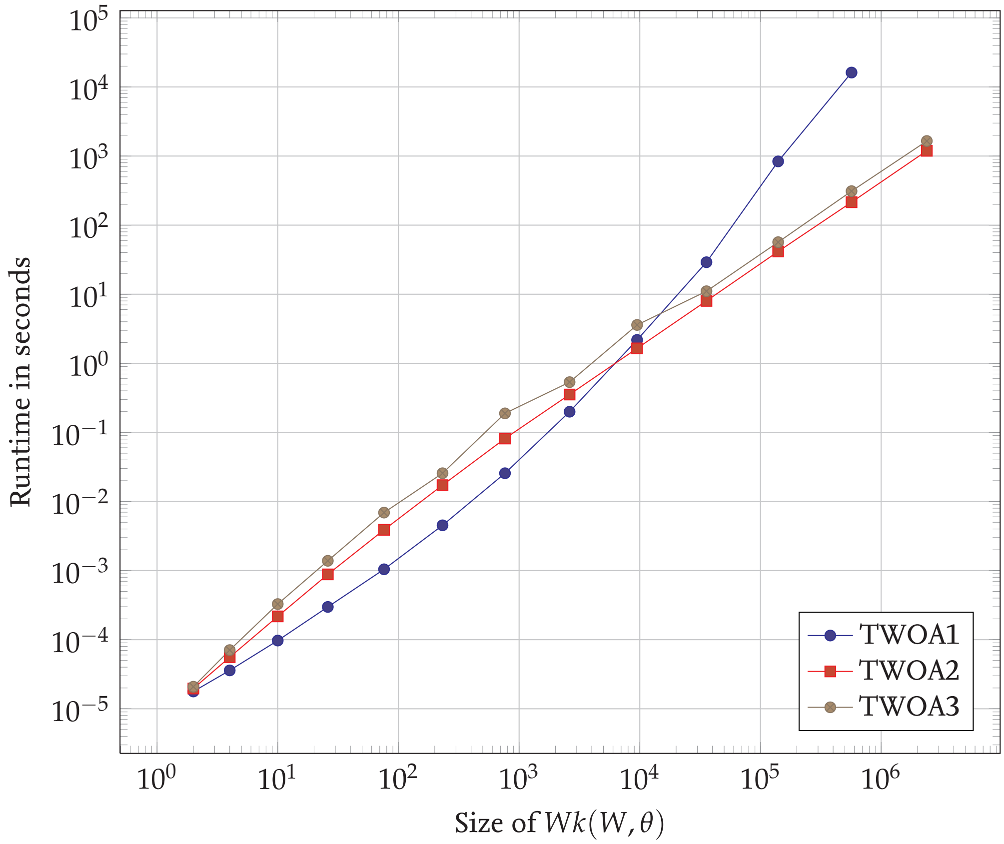

Figure 3.7: Runtime for TWOA1/2/3 in seconds with $W = A_n$, $\theta = \mathrm{id}$

$Wk(\theta)$ is less accessible than $\mathrm{Br}(W)$: In $Wk(\theta)$ there is the possibilty for $w\underline{s} = w\underline{t}$ for two distinct generators $s, t \in S$. Within the Hasse diagram this situation appears in form of double edges between two vertices. For example, let $W = A_3$ and θ be the Coxeter system automorphism swapping s_1 with s_3. Then we have $e\underline{s_1} = s_3s_1 = s_1s_3 = e\underline{s_3}$. Double edges can also occur for $\theta = \mathrm{id}$, but in this situation they cannot appear next to the neutral element e, since $\theta(s)es = e$ for all $s \in S$, hence $e\underline{s} = s \neq t = e\underline{t}$ for all $s, t \in S$ with $s \neq t$. Therefore, if we had written $(*)$ with arbitrary sets S_{12}, S_{23}, S_{31}, then it would be false immediately for any Coxeter system automorphism, that swaps two commutating generators, as seen in Example 4.3.

The following corollary shows us, what distinguishes our special configuration of sets of generators from the arbitrary configuration.

Corollary 4.2. *Let M be a set and $S_{12}, S_{23}, S_{31} \subseteq M$ three subsets. Then there are three sets $S_1, S_2, S_3 \subseteq M$ with $S_{ij} = S_i \cap S_j$ iff no element $x \in M$ is precisely in two of the sets S_{ij}.*

Proof. Let S_{12}, S_{23}, S_{31} be the pairwise intersection of three sets S_1, S_2, S_3. If an element $x \in M$ is in none or in one of the sets S_i, then it is in none of the sets S_{ij}. If it is in two of the sets S_i, say $x \in S_1, S_2$, then $x \in S_{12}$, but x is not in one of the other two S_{ij}. If x is in all three S_i, then it is in all three S_{ij}, too. Hence there is no $x \in M$, that is in precisely two

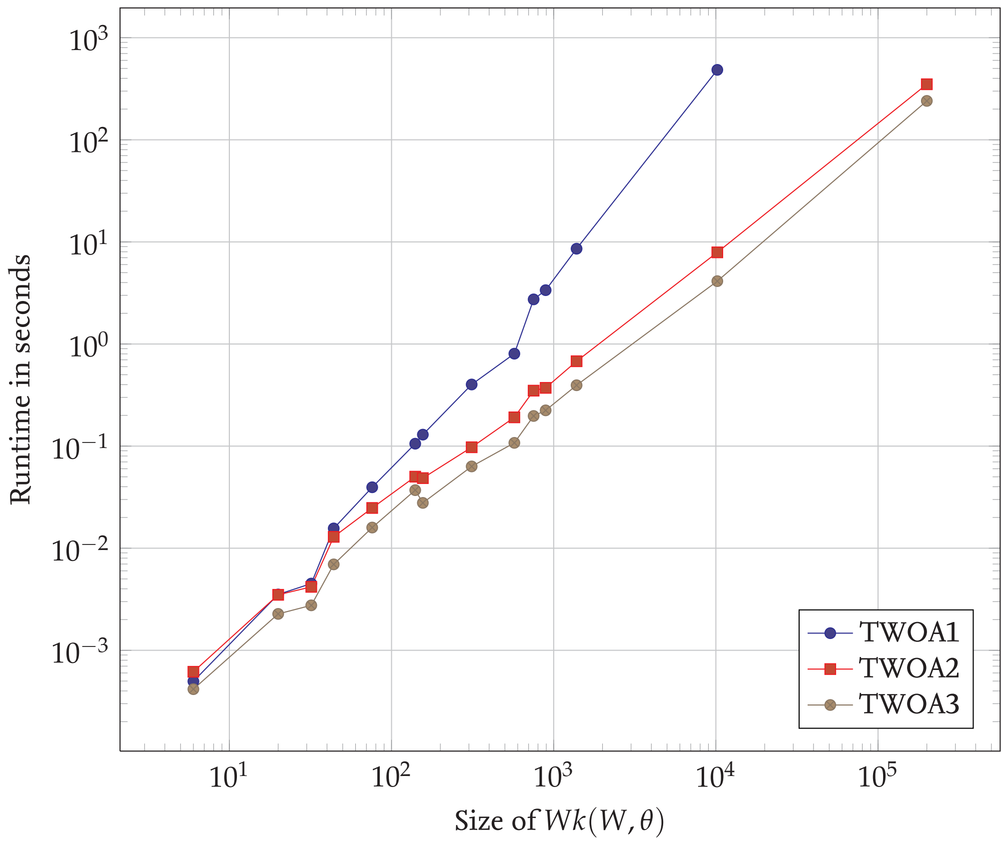

Figure 3.8: Runtime for TWOA1/2/3 in seconds with $W \neq A_n, \theta = \mathrm{id}$

of the sets S_{ij}. Conversely, suppose S_{12}, S_{23}, S_{31} to be arbitrary with the constraint, that there is no element $x \in M$ in precisely two of them. Then we can construct three sets S_1, S_2, S_3, whose pairwise intersections coincides with the sets S_{ij} by $x \in S_i \wedge x \in S_j$ iff $x \in S_{ij}$. With this construction and the previous considerations, it is clear that these S_i have the S_{ij} as pairwise intersection. Note that this construction is not unique in general, since when there is a $x \in M$, that is in none of the sets S_{ij}, then we could add it to S_1, S_2 or S_3 or just omit it without changing there pairwise intersection. □

4.1 Results in less and more specific cases

In this section we investigate some results and examples, in situations that are less or more specific than the situation from Question 4.1.

Example 4.3. Let $W = A_3$ and θ be the Coxeter system autmorphism swapping s_1 and s_3 and let $w = s_1 s_3 = s_3 s_1$. We have $e\underline{s}_1 = s_3 s_1 = w = s_1 s_3 = e\underline{s}_3$. Hence $w \in eC_{\{s_1\}}$ and $w \in eC_{\{s_3\}}$ but $w \notin eC_{\{s_1\} \cap \{s_1\} \cap \{s_3\}} = eC_{\emptyset} = \{e\}$.

Such a trivial counterexample like in Example 4.3 can not occur in the situation from Question 4.1.

Proposition 4.4. *Consider the situation from Question 4.1. Let $w, v \in \mathcal{I}_\theta$ with $\rho(v) - \rho(w) = 1$ and let $v \in wC_{S_{ij}}$ for $1 \leq i < j \leq 3$. Then we have $v \in wC_T$.*

Proof. By Proposition 3.28 there are at most two (not necessarily distinct) $s, t \in S$ with $w\underline{s} = v$ and $w\underline{t} = v$. Each set S_{12}, S_{23}, S_{31} must at least contain s or t, hence s or t is at least in two sets, say $s \in S_{12}, S_{23}$. Hence $s \in S_1, S_2, S_3$ and therefore $v \in wC_T$. □

A hypothesis, that is much stronger than Question 4.1, reads $wC_I \cap wC_J = wC_{I\cap J}$. If this would be true, Question 4.1 could be concluded immediately. Unfortunately it proves to be false. Again, double-edges yield a simple counterexample.

Example 4.5. Let $w \in \mathcal{I}_\theta$ and s, t two distinct generators with $w\underline{s} = w\underline{t} = v$. Then $wC_{\{s\}} \cap wC_{\{t\}} = \{w, v\} \neq \{w\} = wC_\emptyset = wC_{\{s\}\cap\{t\}}$.

Proposition 4.6. *Consider the situation from Question 4.1. Suppose one set of S_1, S_2, S_3 is contained in another. Then*

$$\forall\, 1 \leq i < j \leq 3 : v \in wC_{S_{ij}} \quad \Rightarrow \quad v \in wC_T.$$

Proof. Without loss of generality let $S_1 \subset S_2$. Then we have $S_{12} = S_1$. By this we get the identity

$$T = S_1 \cap S_2 \cap S_3 = S_{12} \cap S_3 = S_1 \cap S_3.$$

Hence $v \in wC_T = wC_{S_{31}}$. □

Lemma 4.7. *Let $(W, S_1 \dot\cup S_2)$ be a reducible Coxeter system with $\operatorname{ord}(st) = 2$ for $s \in S_1, t \in S_2$. Let $\theta = \mathrm{id}$, $s_1, \dots, s_m, s \in S_1$ and $t_1, \dots, t_n, t \in S_2$. Then*

1. *$\underline{s}$ acts by twisted conjugation on $\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n$ if and only if it acts by twisted conjugation on $\underline{s}_1 \dots \underline{s}_m$,*
2. *$\underline{t}$ acts by twisted conjugation on $\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n$ if and only if it acts by twisted conjugation on $\underline{t}_1 \dots \underline{t}_m$, and*
3. *$\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \underline{s} = \underline{s}_1 \dots \underline{s}_m \underline{s} \underline{t}_1 \dots \underline{t}_n$.*

Proof. We have $\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n = t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n$ for some well chosen indices $1 \leq i_1 < \dots < i_q \leq m$ and $1 \leq j_1 < \dots < j_r \leq n$.

1. We prove this by a straight forward chain of equivalences.

$$\begin{aligned}
& s(\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n)s = \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \\
\iff & s(t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n)s = t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n \\
\iff & (t_{i_q} \cdots t_{i_1} t_1 \cdots t_n) s s_{j_r} \cdots s_{j_1} s_1 \cdots s_m s = (t_{i_q} \cdots t_{i_1} t_1 \cdots t_n) s_{j_r} \cdots s_{j_1} s_1 \cdots s_m \\
\iff & s s_{j_r} \cdots s_{j_1} s_1 \cdots s_m s = s_{j_r} \cdots s_{j_1} s_1 \cdots s_m \\
\iff & s(\underline{s}_1 \dots \underline{s}_m)s = \underline{s}_1 \dots \underline{s}_m
\end{aligned}$$

2. This part is almost the same as before.

$$
\begin{aligned}
& t(\underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n)t = \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \\
\iff & t(t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n)t = t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n \\
\iff & t t_{i_q} \cdots t_{i_1} t_1 \cdots t_n t (s_{j_r} \cdots s_{j_1} s_1 \cdots s_m) = t_{i_q} \cdots t_{i_1} t_1 \cdots t_n (s_{j_r} \cdots s_{j_1} s_1 \cdots s_m) \\
\iff & t t_{i_q} \cdots t_{i_1} t_1 \cdots t_n t = t_{i_q} \cdots t_{i_1} t_1 \cdots t_n \\
\iff & t(\underline{t}_1 \dots \underline{t}_n)t = \underline{t}_1 \dots \underline{t}_n
\end{aligned}
$$

Note that the last equivalence is not true in general. Suppose $v \in \mathcal{I}_\theta$ to be an arbitrary twisted expression. In general we cannot deduce the action of $\underline{s}$ on a subexpression of v from the action of $\underline{s}$ on v itself. But with the first part of this lemma we can first conclude, that $\underline{t}_1$ acts by twisted conjugation on e if and only if it acts by twisted conjugation on $\underline{s}_1 \dots \underline{s}_m$. Again with the same argument $\underline{t}_2$ acts by twisted conjugation on $\underline{t}_1$ iff it acts by twisted conjugation on $\underline{s}_1 \dots \underline{s}_m \underline{t}_1$ and so forth.

3. To avoid having to repeat the proof for twisted conjugative and multiplicative action of $\underline{s}$ we set $s' = s$ if $\underline{s}$ acts by twisted conjugation and else $s' = e$.

$$
\begin{aligned}
& \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \underline{s} \\
&= s'(t_{i_q} \cdots t_{i_1} s_{j_r} \cdots s_{j_1} s_1 \cdots s_m t_1 \cdots t_n)s \\
&= t_{i_q} \cdots t_{i_1} (s' s_{j_r} \cdots s_{j_1} s_1 \cdots s_m s) t_1 \cdots t_n \\
&= t_{i_q} \cdots t_{i_1} (\underline{s}_1 \dots \underline{s}_m \underline{s}) t_1 \cdots t_n \\
&= \underline{s}_1 \dots \underline{s}_m \underline{s} \underline{t}_1 \dots \underline{t}_n
\end{aligned}
$$

Again note that the last to equalities need the two previous parts of this lemma. □

Corollary 4.8. *Let $(W, S_1 \dot\cup S_2)$ be Coxeter system with $\operatorname{ord}(st) = 2$ whenever $s \in S_1, t \in S_2$. In particular W is reducible. Let $W := W_{S_1}$ and $W_2 := W_{S_2}$ be the parabolic subgroups of W corrosponding to S_1 and S_2. Then we have $Wk(W, \mathrm{id}) \cong Wk(W_1, \mathrm{id}) \times Wk(W_2, \mathrm{id})$.*

Proof. We denote the relation in W (resp. in W_1, W_2) by $\preceq_W$ (resp. by $\preceq_{W_1}$, $\preceq_{W_2}$). By Lemma 4.7 for every element $w \in \mathcal{I}_{\mathrm{id}}(W)$ we can find a twisted expression like $w = \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n$ with $s \in S_1, t \in S_2$. Hence the map

$$
\varphi : \mathcal{I}_{\mathrm{id}}(W_1) \times \mathcal{I}_{\mathrm{id}}(W_2) \to \mathcal{I}_{\mathrm{id}}(W) : (\underline{s}_1 \dots \underline{s}_m, \underline{t}_1 \dots \underline{t}_n) \mapsto \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n
$$

is surjective. The injectivity is due to Proposition 2.23. It remains to show that $\preceq_W$ satisfies Definition 2.7. Let $v_1, w_1 \in \mathcal{I}_{\mathrm{id}}(W_1)$, $v_2, w_2 \in \mathcal{I}_{\mathrm{id}}(W_2)$ and $v = v_1 v_2 = \varphi(v_1, v_2), w = w_1 w_2 = \varphi(w_1, w_2) \in \mathcal{I}_{\mathrm{id}}(W)$. Suppose $v_i \preceq_{W_i} w_i$ for $i = 1, 2$. Then we have

$$
\begin{aligned}
v_1 &= \underline{s}_1 \dots \underline{s}_m, & w_1 &= \underline{s}_1 \dots \underline{s}_m \dots \underline{s}_{m'} = v_1 \underline{s}_{m+1} \dots \underline{s}_{m'}, \\
v_2 &= \underline{t}_1 \dots \underline{t}_n \text{ and} & w_2 &= \underline{t}_1 \dots \underline{t}_n \dots \underline{t}_{n'} = v_2 \underline{t}_{n+1} \dots \underline{t}_{n'}
\end{aligned}
$$

for some well chosen generators $s_i \in S_1, t_i \in S_2$ and $0 \leq m \leq m', 0 \leq n \leq n'$. Hence

$$\begin{aligned} v = v_1 v_2 &= \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \preceq_W \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \underline{s}_{m+1} \dots \underline{s}_{m'} \underline{t}_{n+1} \dots \underline{t}_{n'} \\ &= \underline{s}_1 \dots \underline{s}_m \underline{s}_{m+1} \dots \underline{s}_{m'} \underline{t}_1 \dots \underline{t}_n \underline{t}_{n+1} \dots \underline{t}_{n'} = w_1 w_2 = w. \end{aligned}$$

In return suppose $v \preceq_W w$. Then we have

$$\begin{aligned} v &= \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \text{ and} \\ w &= \underline{s}_1 \dots \underline{s}_m \underline{t}_1 \dots \underline{t}_n \underline{s}_{m+1} \dots \underline{s}_{m'} \underline{t}_{n+1} \dots \underline{t}_{n'} \end{aligned}$$

for some well chosen generators $s_i \in S_1, t_i \in S_2$ and $0 \leq m \leq m', 0 \leq n \leq n'$. Again with similar arguments we have

$$\begin{aligned} v_1 &= \underline{s}_1 \dots \underline{s}_m \preceq_{W_1} \underline{s}_1 \dots \underline{s}_m \underline{s}_{m+1} \dots \underline{s}_{m'} = w_1 \text{ and} \\ w_1 &= \underline{t}_1 \dots \underline{t}_n \preceq_{W_2} \underline{t}_1 \dots \underline{t}_n \underline{t}_{n+1} \dots \underline{t}_{n'} = w_2. \end{aligned}$$ □

Remark 4.9. Note that Lemma 4.7 and Corollary 4.8 still hold, if we drop the premise $\theta = \text{id}$ and instead insist on $\theta(S_i) = S_i$ for $i = 1, 2$. They also remain true, if we have a partition of the generator set in more than two subsets. Hence for $(W, S_1 \dot\cup \dots \dot\cup S_n)$ with $\operatorname{ord}(st) = 2$ whenever $s \in S_i, t \in S_j, i \neq j$ we have

$$Wk(W, \text{id}) = Wk(W_{S_1}, \text{id}) \times \dots \times Wk(W_{S_n}, \text{id}).$$

Theorem 4.10. *Let (W, S) be a reducible Coxeter system with $S = S' \cup S''$ and $\operatorname{ord}(st) = 2$ whenever $s \in S', t \in S''$ and let $\theta = \text{id}$. Then $(*)$ holds for (W, S) if and only if it holds for $(W_{S'}, S')$ and $(W_{S''}, S'')$, too.*

Proof. If $(*)$ holds for (W, S), then it holds for $(W_{S'}, S')$ and $(W_{S''}, S'')$ in particular. In return suppose $(*)$ to hold for $(W_{S'}, S')$ and $(W_{S''}, S'')$ and assume we are in the situtation from Question 4.1. For a set $M \subseteq S$ we define $M' := M \cap S'$ and $M'' := M \cap S''$, hence $M = M' \dot\cup M''$. This is compatible with our definition of S_{ij} and T:

$$\begin{aligned} S_{ij} &= S_i \cap S_j = (S'_i \dot\cup S''_i) \cap (S'_j \dot\cup S''_j) = (S'_i \cap S'_j) \dot\cup (S''_i \cap S''_j) = S'_{ij} \dot\cup S''_{ij} \\ T &= S_1 \cap S_2 \cap S_3 = (S'_{12} \dot\cup S''_{12}) \cap (S'_3 \dot\cup S''_3) = (S'_{12} \cap S'_3) \dot\cup (S''_{12} \cap S''_3) = T' \dot\cup T'' \end{aligned}$$

Let $w_K = \underline{s}'_1 \dots \underline{s}'_{m'} \underline{s}''_1 \dots \underline{s}''_{m''}$ with $s'_i \in K', s''_i \in K''$. Then $w_{K'} = \underline{s}'_1 \dots \underline{s}'_{m'}$ (resp. $w_{K''} = \underline{s}''_1 \dots \underline{s}''_{m''}$) is the corrosponding longest elements in $\langle K' \rangle \leq W_{S'} \leq W$ (resp. $\langle K'' \rangle \leq W_{S''} \leq W$). We have three twisted expressions

$$\begin{aligned} w &= w_K \underline{a}'_1 \dots \underline{a}'_{n'} \underline{a}''_1 \dots \underline{a}''_{n''} \\ &= w_K \underline{b}'_1 \dots \underline{b}'_{n'} \underline{b}''_1 \dots \underline{b}''_{n''} \\ &= w_K \underline{c}'_1 \dots \underline{c}'_{n'} \underline{c}''_1 \dots \underline{c}''_{n''} \end{aligned}$$

with $a_i', a_i'' \in S_1$, $b_i', b_i'' \in S_2$ and $c_i', c_i'' \in S_3$. Thanks to Lemma 4.7 we can assume without loss of generality that $a', b', c' \in S'$ and $a'', b'', c'' \in S''$. Hence we have also

$$\begin{aligned} w' &= w_{K'}\underline{a}_1' \dots \underline{a}_{n'}' = s_1' \dots \underline{s}_m'\underline{a}_1' \dots \underline{a}_{n'}' \\ &= w_{K'}\underline{b}_1' \dots \underline{b}_{n'}' = s_1' \dots \underline{s}_m'\underline{b}_1' \dots \underline{b}_{n'}' \\ &= w_{K'}\underline{c}_1' \dots \underline{c}_{n'}' = s_1' \dots \underline{s}_m'\underline{c}_1' \dots \underline{c}_{n'}' \end{aligned}$$

and so $w' \in w_{K'}C_{T'}$, since $(*)$ holds in $(W_{S'}, S')$. Analogue we get $w'' \in w_{K''}C_{T''}$. Hence

$$w' = \underline{s}_1' \dots \underline{s}_{m'}'\underline{d}_1' \dots \underline{d}_{l'}' \text{ and } w'' = \underline{s}_1'' \dots \underline{s}_{m''}''\underline{d}_1'' \dots \underline{d}_{l''}''$$

for $d_i' \in T'$ and $d_i'' \in T''$. This yields a twisted expression

$$\begin{aligned} w = w'w'' &= \underline{s}_1' \dots \underline{s}_{m'}'\underline{d}_1' \dots \underline{d}_{l'}'\underline{s}_1'' \dots \underline{s}_{m''}''\underline{d}_1'' \dots \underline{d}_{l''}'' \\ &= \underline{s}_1' \dots \underline{s}_{m'}'\underline{s}_1'' \dots \underline{s}_{m''}''\underline{d}_1' \dots \underline{d}_{l'}'\underline{d}_1'' \dots \underline{d}_{l''}'' \\ &= w_K\underline{d}_1' \dots \underline{d}_{l'}'\underline{d}_1'' \dots \underline{d}_{l''}'' \end{aligned}$$

with $d_i', d_i'' \in T' \dot\cup T'' = T$. Thus $w \in w_K C_S$. □

5 Application

See [6].

Definition 5.1. A ***chamber system over*** I is a pair $\mathcal{C} = (C, (\sim_i, i \in I))$, with a nonempty set C, whose members are called ***chambers*** and a family of equivalence relations $\sim_i$, indexed by $i \in I$, that satisfies the implication

$$c \sim_i d \wedge c \sim_j d \Rightarrow c = d \vee i = j$$

for all $c, d \in C$ and $i, j \in I$. The cardinality $|I|$ is called the ***rank*** of $\mathcal{C}$. If for two chambers c, d we have $c \sim_i d$, then c is called ***i-adjacent*** to d or just ***adjacent***.

So the main assertion for chamber systems is, that two distinct chambers $c, d \in C$ are at most adjacent by one $i \in I$. For the rest of this paper $\mathcal{C} = (C, (\sim_i, i \in I))$ will always denote a chamber system and we will always assume, that chamber systems are of finite rank.

Example 5.2. For an arbitrary Coxeter system let W act as set of chambers and for each generator $s \in S$ define a equivalence relation $w \sim_s v$ if and only if either $w = v$ or $ws = v$. That this are really equivalence relations is easy to check. So suppose $w \sim_s v$, $w \sim_t v$ for two distinct generators $s, t \in S$. The assumption $w \neq v$ immediately yields a contradiction by $ws = v = wt \iff s = t$. Hence this is indeed a chamber system.

The previous example is just a special case of a quite general recipe to create chamber systems from groups, the so-called coset chamber systems.

Definition 5.3. [2, Definition 3.6.3] Let G be an arbitrary group with a subgroup B and a family of subgroups $(G_i, i \in I)$ such that $B \subseteq G_i$ for $i \in I$. Choose the chamber set C as the set of all B-cosets gB for some $g \in G$ and define the equivalence relations $(\sim_i, i \in I)$ by $gB \sim_i hB$ iff $gG_i = hG_i$. Then we call this chamber system the ***coset chamber system*** of G on B with respect to $(G_i, i \in I)$.

Lemma 5.4. *Coset chamber systems are chamber systems.*

Proof. As easy to check the $\sim_i$ are equivalence relations. So suppose $gB \sim_i hB$ and $gB \sim_j hB$ and let $gB \neq hB$, i.e. $h^{-1}g \notin B$. **TODO** Different definitions of chamber system at Horn and Buekenhout/Cohen? $\square$

If two chambers $c, d \in C$ in a chamber system are not adjacent, then there might be a chain of subsequent adjacent chambers with c as first and d as last chamber.

Definition 5.5. Let $G = (c_0, \dots, c_k)$ be a finite sequence of chambers $c_i \in C$ with c_{i-1} adjacent to c_i for all $1 \leq i \leq k$. Then G is called a ***gallery*** in $\mathcal{C}$ whereas the integer k is called the ***length*** of G. The first element c_0 of a gallery G is denoted by $\alpha(G)$ and the last by $\omega(G)$. If for two chambers $c, d \in C$ there is a gallery G with $\alpha(G) = c$ and $\omega(G) = d$, then we say that G ***joins*** c and d. A gallery with G with $\alpha(G) = \omega(G)$ is called ***closed*** and a gallery $G = (c_0, \dots, c_k)$ with $c_{i-1} \neq c_i$ for all $1 \leq i \leq k$ is called ***simple***.

Note, that two chambers are adjacent if and only if they can be joined by a gallery of length 1.

Definition 5.6. The chamber system $\mathcal{C}$ is called ***connected*** if any two chambers $c, d \in C$ can be joined by a gallery.

Definition 5.7. Let $G = (c_0, \dots, c_k)$ be a gallery and let $J \subset I$ be a subset. If for $1 \leq i \leq k$ there is a $j \in J$ with $c_{i-1} \sim_j c_i$, then we call G a ***J-gallery***. Two chambers $c, d \in C$, that have a J-gallery joining them, are called ***J-equivalent***, denoted by $c \sim_J d$.

Definition 5.8. For a chamber $c \in C$ and a subset $J \subseteq I$, we call the set $R_J(c) := \{d \in C : c \sim_J d\}$ a ***J-residue***.

Note that for any chamber system $(C, (\sim_i, i \in I))$, $c \in C$ and $J \subseteq I$, the chamber system $(R_J(c), (\sim_j, j \in J))$ is connected by construction.

Definition 5.9. Let $\mathcal{C}$ be a chamber system over I. We call it a ***residually connected*** chamber system if the following holds: For every $J \subseteq I$ and every family of residues $(R_{I \setminus \{j\}}, j \in J)$ with pairwise nonempty intersection we have

$$\bigcap_{j \in J} R_{I \setminus \{j\}} = R_{I \setminus J}(c)$$

for some $c \in C$.

Definition 5.10. A ***building*** of type (W, S) is a pair $(\mathcal{C}, \delta)$ with a nonempty set $\mathcal{C}$ and a map $\delta : \mathcal{C} \times \mathcal{C} \to W$, called ***distance function***, so that for $x, y \in \mathcal{C}$ and $w = \delta(x, y)$ we have

(Bu1) $w = e \iff x = y$;

(Bu2) for $z \in \mathcal{C}$ with $\delta(y,z) = s \in S$ we have $\delta(x,z) \in \{w, ws\}$, and if in addition $l(ws) = l(w) + 1$ then we have $\delta(x,z) = ws$;

(Bu3) for $s \in S$ there exists a $z \in \mathcal{C}$ with $\delta(y,z) = s$ and $\delta(x,z) = ws$.

We can associate a chamber system to a building.

Definition 5.11. Let $(\mathcal{C}, \delta)$ be a building of type (W, S). For each $s \in S$ we define $c, d \in C$ to be s-adjacent, if and only iff $\delta(c,d) \in \{e, s\}$. Then $(\mathcal{C}, (\sim_s, s \in S))$ is the to our building $(\mathcal{C}, \delta)$ associated chamber system.

Lemma 5.12. *Let $(\mathcal{C}, \delta)$ be a building of type (W, S). Then the associated chamber system is a chamber system.*

Proof. $\square$

A Source codes

File misc.gap

```
GroupAutomorphismByPermutation := function (G, generatorPermutation)
    local automorphism, generators;

    generators := GeneratorsOfGroup(G);

    if generatorPermutation = "id" or generatorPermutation = [1..Length(generators)] then
        automorphism := IdentityMapping(G);
        SetName(automorphism, "id");

        return automorphism;
    elif generatorPermutation = "-id" then
        generatorPermutation := Reversed([1..Length(GeneratorsOfGroup(G))]);
    fi;

    automorphism := GroupHomomorphismByImages(G, G, generators, generators{generatorPermutation});
    SetName(automorphism, Concatenation("(", JoinStringsWithSeparator(generatorPermutation, ","), ")"));

    return automorphism;
end;

GroupAutomorphismIdNeg := function (G)
    return GroupAutomorphismByPermutation(G, Reversed([1..Length(GeneratorsOfGroup(G))]));
end;

GroupAutomorphismId := function (G)
    return GroupAutomorphismByPermutation(G, [1..Length(GeneratorsOfGroup(G))]);
end;

FindElement := function (list, selector)
    local i;

    for i in [1..Length(list)] do
        if (selector(list[i])) then
            return list[i];
        fi;
    od;

    return fail;
end;

StringToFilename := function(str)
    local result, c;

    result := "";

    for c in str do
        if IsDigitChar(c) or IsAlphaChar(c) or c = '-' or c = '_' then
            Add(result, c);
        else
            Add(result, '_');
        fi;
    od;
```

```

    return result;
end;

IO_ReadLinesIterator := function (file)
    local IsDone, Next, ShallowCopy;

    IsDone := function (iter)
        return iter!.nextLine = "" or iter!.nextLine = fail;
    end;

    Next := function (iter)
        local line;

        line := iter!.nextLine;

        if line = fail then
            Error(LastSystemError());
            return fail;
        fi;

        iter!.nextLine := IO_ReadLine(iter!.file);

        return Chomp(line);
    end;

    ShallowCopy := function (iter)
        return fail;
    end;

    return IteratorByFunctions(rec(IsDoneIterator := IsDone, NextIterator := Next,
        ShallowCopy := ShallowCopy, file := file, nextLine := IO_ReadLine(file)));
end;

IO_ReadLinesIteratorCSV := function (file, seperator)
    local IsDone, Next, ShallowCopy;

    IsDone := function (iter)
        return iter!.nextLine = "" or iter!.nextLine = fail;
    end;

    Next := function (iter)
        local line, lineSplitted, result, i;

        line := iter!.nextLine;
        if line = fail then
            Error(LastSystemError());
            return fail;
        fi;
        iter!.nextLine := IO_ReadLine(iter!.file);

        lineSplitted := SplitString(Chomp(line), iter!.seperator);
        result := rec();

        for i in [1..Minimum(Length(iter!.headers), Length(lineSplitted))] do
            result.(iter!.headers[i]) := EvalString(lineSplitted[i]);
        od;

        return result;
    end;

```

```
    ShallowCopy := function (iter)
        return fail;
    end;

    return IteratorByFunctions(rec(IsDoneIterator := IsDone, NextIterator := Next,
        ShallowCopy := ShallowCopy, file := file, seperator := seperator,
        headers := SplitString(Chomp(IO_ReadLine(file)), seperator),
        nextLine := IO_ReadLine(file)));
end;
```

File coxeter.gap

```
Read("coxeter-generators.gap");

coxeterElementComparisons := 0;

CoxeterElementsCompare := function (w1, w2)
    coxeterElementComparisons := coxeterElementComparisons + 1;
    return w1 = w2;
end;

CoxeterMatrixEntry := function(matrix, i, j)
    local temp, rank;
    rank := -1/2 + Sqrt(1/4 + 2*Length(matrix)) + 1;

    if (i = j) then
        return 1;
    fi;

    if (i > j) then
        temp := i;
        i := j;
        j := temp;
    fi;

    return matrix[(rank-1)*(rank)/2 - (rank-i)*(rank-i+1)/2 + (j-i-1) + 1];
end;
```

File coxeter-generators.gap

```
# Generates a coxeter group with given rank and relations. The relations have to
# be given in a linear list of the upper right entries (above diagonal) of the
# coxeter matrix.
#
# Example:
# To generate the coxeter group A_4 with the following coxeter matrix:
#
# | 1 3 2 2 |
# | 3 1 3 2 |
# | 2 3 1 3 |
# | 2 2 3 1 |
#
# A4 := CoxeterGroup(4, [3,2,2, 3,2, 3]);
CoxeterGroup := function (rank, upperTriangleOfCoxeterMatrix)
    local generatorNames, relations, F, S, W, i, j, k;

    generatorNames := List([1..rank], n -> Concatenation("s", String(n)));

    F := FreeGroup(generatorNames);
    S := GeneratorsOfGroup(F);
```

```

    relations := [];

    Append(relations, List([1..rank], n -> S[n]^2));

    k := 1;
    for i in [1..rank] do
        for j in [i+1..rank] do
            Add(relations, (S[i]*S[j])^(upperTriangleOfCoxeterMatrix[k]));
            k := k + 1;
        od;
    od;

    W := F / relations;

    return W;
end;

CoxeterGroup_An := function (n)
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := Flat(List(Reversed([1..n-1]), m -> Concatenation
        ([3], List([1..m-1], o -> 2))));

    #W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);
    W := GroupWithGenerators(List([1..n], s -> (s,s+1)));

    SetName(W, Concatenation("A_{", String(n), "}"));
    SetSize(W, Factorial(n + 1));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_BCn := function (n)
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := Flat(List(Reversed([1..n-1]), m -> Concatenation
        ([3], List([1..m-1], o -> 2))));
    upperTriangleOfCoxeterMatrix[Length(upperTriangleOfCoxeterMatrix)] := 4;

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, Concatenation("BC_{", String(n), "}"));
    SetSize(W, 2^n * Factorial(n));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_Dn := function (n)
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := Flat(List(Reversed([1..n-1]), m -> Concatenation
        ([3], List([1..m-1], o -> 2))));
    upperTriangleOfCoxeterMatrix[Length(upperTriangleOfCoxeterMatrix)] := 2;
    upperTriangleOfCoxeterMatrix[Length(upperTriangleOfCoxeterMatrix) - 1] := 3;
    upperTriangleOfCoxeterMatrix[Length(upperTriangleOfCoxeterMatrix) - 2] := 3;

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, Concatenation("D_{", String(n), "}"));
    SetSize(W, 2^(n-1) * Factorial(n));
```

```
    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_E6 := function ()
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [3, 2, 2, 2, 2,  3, 2, 2, 2,  3, 3, 2,  2, 2,  3];

    W := CoxeterGroup(6, upperTriangleOfCoxeterMatrix);

    SetName(W, "E_6");
    SetSize(W, 2^7 * 3^4 * 5);

    return rec(group := W, rank := 6, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_E7 := function ()
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [3, 2, 2, 2, 2, 2,  3, 2, 2, 2, 2,  3, 3, 2, 2,  2,
         2, 2,  3, 2,  3];

    W := CoxeterGroup(7, upperTriangleOfCoxeterMatrix);

    SetName(W, "E_7");
    SetSize(W, 2^10 * 3^4 * 5 * 7);

    return rec(group := W, rank := 7, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_E8 := function ()
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [3, 2, 2, 2, 2, 2, 2,  3, 2, 2, 2, 2, 2,  3, 3, 2,
        2, 2,  2, 2, 2, 2,  3, 2, 2,  3, 2,  3];

    W := CoxeterGroup(8, upperTriangleOfCoxeterMatrix);

    SetName(W, "E_8");
    SetSize(W, 2^14 * 3^5 * 5^2 * 7);

    return rec(group := W, rank := 8, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_F4 := function ()
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [3, 2, 2, 4, 2, 3];

    W := CoxeterGroup(4, upperTriangleOfCoxeterMatrix);

    SetName(W, "F_4");
    SetSize(W, 2^7 * 3^2);

    return rec(group := W, rank := 4, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_H3 := function ()
    local upperTriangleOfCoxeterMatrix, W;

```

```
    upperTriangleOfCoxeterMatrix := [5, 2, 3];

    W := CoxeterGroup(3, upperTriangleOfCoxeterMatrix);

    SetName(W, "H_3");
    SetSize(W, 120);

    return rec(group := W, rank := 3, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_H4 := function ()
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [5, 2, 2, 3, 2, 3];

    W := CoxeterGroup(4, upperTriangleOfCoxeterMatrix);

    SetName(W, "H_4");
    SetSize(W, 14400);

    return rec(group := W, rank := 4, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_I2m := function (m)
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := [m];

    W := CoxeterGroup(2, upperTriangleOfCoxeterMatrix);

    SetName(W, Concatenation("I_2(", String(m), ")"));
    SetSize(W, 2*m);

    return rec(group := W, rank := 2, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_TildeAn := function (n)
    local upperTriangleOfCoxeterMatrix, W;

    upperTriangleOfCoxeterMatrix := Flat(List(Reversed([1..n]), m -> Concatenation([3],
         List([1..m-1], o -> 2))));

    if n = 1 then
        upperTriangleOfCoxeterMatrix[1] := 0;
    else
        upperTriangleOfCoxeterMatrix[n] := 3;
    fi;

    W := CoxeterGroup(n + 1, upperTriangleOfCoxeterMatrix);

    SetName(W, Concatenation("\\tilde A_{", String(n), "}"));
    SetSize(W, infinity);

    return rec(group := W, rank := n + 1, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A1xA1 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 2;
    upperTriangleOfCoxeterMatrix := [2];
```

```

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_1 \\times A_1");
    SetSize(W, Factorial(2)*Factorial(2));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A2xA2 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 4;
    upperTriangleOfCoxeterMatrix := [3,2,2, 2,2, 3];

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_2 \\times A_2");
    SetSize(W, Factorial(3)*Factorial(3));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A3xA3 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 6;
    upperTriangleOfCoxeterMatrix := [3,2,2,2,2, 3,2,2,2, 2,2,2, 3,2, 3];

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_3 \\times A_3");
    SetSize(W, Factorial(4)*Factorial(4));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A1xA1xA1 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 3;
    upperTriangleOfCoxeterMatrix := [2,2, 2];

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_1 \\times A_1 \\times A_1");
    SetSize(W, Factorial(2)*Factorial(2)*Factorial(2));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A2xA2xA2 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 6;
    upperTriangleOfCoxeterMatrix := [3,2,2,2,2, 2,2,2,2, 3,2,2, 2,2, 3];

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_2 \\times A_2 \\times A_2");
    SetSize(W, Factorial(3)*Factorial(3)*Factorial(3));
```

```

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;

CoxeterGroup_A3xA3xA3 := function ()
    local upperTriangleOfCoxeterMatrix, W, n;

    n := 9;
    upperTriangleOfCoxeterMatrix := [3,2,2,2,2,2,2,2, 3,2,2,2,2,2,2, 2,2,2,2,2,2,
        3,2,2,2,2, 3,2,2,2, 2,2,2, 3,2, 3];

    W := CoxeterGroup(n, upperTriangleOfCoxeterMatrix);

    SetName(W, "A_3 \\times A_3 \\times A_3");
    SetSize(W, Factorial(4)*Factorial(4)*Factorial(4));

    return rec(group := W, rank := n, matrix := upperTriangleOfCoxeterMatrix);
end;
```

File twistedinvolutionweakordering.gap

```
LoadPackage("io");

Read("misc.gap");
Read("coxeter.gap");
Read("twoa-persist.gap");
Read("twoa-misc.gap");
Read("twoa1.gap");
Read("twoa2.gap");
Read("twoa3.gap");

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

    visited := [ vertex.absIndex ];
    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.absIndex in visited then
                Add(visited, edge.target.absIndex);
                Add(queue, edge.target);
            fi;
        od;

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

    return residuum;
end;

```

```
TwistedInvolutionWeakOrderingLongestWord := 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;
```

File twoa-misc.gap

```
DetectPossibleRank2Residuums := function(startVertex, startLabel, labels)
    local comb, trace, v, e, k, possibleTraces;
    possibleTraces := [];

    for comb in List(Filtered(labels, label -> label <> startLabel), label -> rec(
        startVertex := startVertex, st := [startLabel, label])) do
        trace := [ rec(vertex := startVertex, edge := rec(label := comb.st[1], type :=
            -1)) ];

        v := startVertex;
        e := fail;
        k := 1;

        while true do
            e := FindElement(v.inEdges, e -> e.label = comb.st[k mod 2 + 1]);
            if e = fail then
                break;
            fi;

            v := e.source;
            k := k + 1;
            Add(trace, rec(vertex := v, edge := e));
        od;

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

            v := e.target;
            k := k - 1;
            Add(trace, rec(vertex := v, edge := e));
        od;

        if k = 0 then
            Add(possibleTraces, trace);
        fi;
    od;

    return possibleTraces;
end;
```

File twoa-persist.gap

```
TwistedInvolutionWeakOrderingPersistReadResults := function(filename)
```

```
    local fileD, fileV, fileE, csvLine, data, vertices, edges, newEdge, source, target,
        i;

    fileD := IO_File(Concatenation("results/", filename, "-data"), "r");
    fileV := IO_File(Concatenation("results/", filename, "-vertices"), "r", 1024*1024);
    fileE := IO_File(Concatenation("results/", filename, "-edges"), "r", 1024*1024);

    data := NextIterator(IO_ReadLinesIteratorCSV(fileD, ";"));
    vertices := [];
    edges := [];

    i := 1;
    for csvLine in IO_ReadLinesIteratorCSV(fileV, ";") do
        Add(vertices, rec(absIndex := i, twistedLength := csvLine.twistedLength, name
            := csvLine.name, inEdges := [], outEdges := []));
        i := i + 1;
    od;

    i := 1;
    for csvLine in IO_ReadLinesIteratorCSV(fileE, ";") do
        source := vertices[csvLine.sourceIndex + 1];
        target := vertices[csvLine.targetIndex + 1];
        newEdge := rec(absIndex := i, source := source, target := target, label :=
            csvLine.label, type := csvLine.type);

        Add(source.outEdges, newEdge);
        Add(target.inEdges, newEdge);
        Add(edges, newEdge);
        i := i + 1;
    od;

    IO_Close(fileD);
    IO_Close(fileV);
    IO_Close(fileE);

    return rec(data := data, vertices := vertices, edges := edges);
end;

TwistedInvolutionWeakOrderingPersistResultsInit := function(filename)
    local fileD, fileV, fileE;

    if (filename = fail) then return fail; fi;

    fileD := IO_File(Concatenation("results/", filename, "-data"), "w");
    fileV := IO_File(Concatenation("results/", filename, "-vertices"), "w", 1024*1024);
    fileE := IO_File(Concatenation("results/", filename, "-edges"), "w", 1024*1024);
    IO_Write(fileD, "name;rank;size;generators;matrix;automorphism;wk_size;
        wk_max_length\n");
    IO_Write(fileV, "twistedLength;name\n");
    IO_Write(fileE, "sourceIndex;targetIndex;label;type\n");

    return rec(fileD := fileD, fileV := fileV, fileE := fileE);
end;

TwistedInvolutionWeakOrderingPersistResultsClose := function(persistInfo)
    if (persistInfo = fail) then return; fi;

    IO_Close(persistInfo.fileD);
    IO_Close(persistInfo.fileV);
    IO_Close(persistInfo.fileE);
end;
```

```

TwistedInvolutionWeakOrderingPersistResultsInfo := function(persistInfo, W, matrix, theta, numVertices, maxTwistedLength)
    if (persistInfo = fail) then return; fi;

    IO_Write(persistInfo.fileD, "\"", ReplacedString(Name(W), "\\", "\\\\"), "\";");
    IO_Write(persistInfo.fileD, Length(GeneratorsOfGroup(W)), ";");
    if (Size(W) = infinity) then
        IO_Write(persistInfo.fileD, "\"infinity\";");
    else
        IO_Write(persistInfo.fileD, Size(W), ";");
    fi;
    IO_Write(persistInfo.fileD, "[", JoinStringsWithSeparator(List(GeneratorsOfGroup(W), n -> Concatenation("\"", String(n), "\"")), ","), "];");
    IO_Write(persistInfo.fileD, "[", JoinStringsWithSeparator(matrix, ","), "];");
    IO_Write(persistInfo.fileD, "\"", Name(theta), "\";");

    if (Size(W) = infinity) then
        IO_Write(persistInfo.fileD, "\"infinity\";");
        IO_Write(persistInfo.fileD, "\"infinity\"");
    else
        IO_Write(persistInfo.fileD, numVertices, ";");
        IO_Write(persistInfo.fileD, maxTwistedLength, "");
    fi;
end;

TwistedInvolutionWeakOrderingPersistResults := function(persistInfo, vertices, edges)
    local n, e, i, tmp, bubbles;

    if (persistInfo = fail) then return; fi;

    # bubble sort the edges, to make sure, that double edges are neighbours in the list
    bubbles := 1;
    while bubbles > 0 do
        bubbles := 0;
        for i in [1..Length(edges)-1] do
            if edges[i].source.absIndex = edges[i+1].source.absIndex and edges[i].target.absIndex > edges[i+1].target.absIndex then
                tmp := edges[i];
                edges[i] := edges[i+1];
                edges[i+1] := tmp;
                bubbles := bubbles + 1;
            fi;
        od;
    od;

    for n in vertices do
        if n.absIndex = 1 then
            IO_Write(persistInfo.fileV, n.twistedLength, ";\"e\"\n");
        else
            IO_Write(persistInfo.fileV, n.twistedLength, ";\"", String(n.element), "\"\n");
        fi;
    od;

    for e in edges do
        IO_Write(persistInfo.fileE, e.source.absIndex-1, ";", e.target.absIndex-1, ";", e.label, ";", e.type, "\n");
    od;
end;
```

File twoal.gap

```
# Calculates the poset Wk(theta).
TwistedInvolutionWeakOrdering1 := function (filename, W, matrix, theta)
    local persistInfo, maxOrder, vertices, edges, absVertexIndex, absEdgeIndex,
        prevVertex, currVertex, 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]);
    vertices := [ [], [ rec(element := One(W), twistedLength := 0, inEdges := [],
        outEdges := [], absIndex := 1) ] ];
    edges := [ [], [] ];
    absVertexIndex := 2;
    absEdgeIndex := 1;
    k := 0;

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

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

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

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

                currVertex := fail;
                for n in vertices[1] do
                    if CoxeterElementsCompare(n.element, y) then
                        currVertex := n;
                        break;
                    fi;
                od;

                if currVertex = fail then
                    currVertex := rec(element := y, twistedLength := k + 1, inEdges :=
                        [], outEdges := [], absIndex := absVertexIndex);
                    Add(vertices[1], currVertex);

                    absVertexIndex := absVertexIndex + 1;
                fi;

                newEdge := rec(source := prevVertex, target := currVertex, label :=
                    label, type := type, absIndex := absEdgeIndex);

```

```
                Add(edges[1], newEdge);
                Add(currVertex.inEdges, newEdge);
                Add(prevVertex.outEdges, newEdge);

                absEdgeIndex := absEdgeIndex + 1;
            od;
        od;

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

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

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

    return rec(numVertices := absVertexIndex - 1, numEdges := absEdgeIndex - 1, maxTwistedLength := k - 1);
end;
```

File twoa2.gap

```
# Calculates the poset Wk(theta).
TwistedInvolutionWeakOrdering2 := function (filename, W, matrix, theta)
    local persistInfo, maxOrder, vertices, edges, absVertexIndex, absEdgeIndex, prevVertex, currVertex, newEdge, possibleResiduums,
        label, type, deduction, startTime, endTime, S, k, i, s, x, y, n, h, res;

    persistInfo := TwistedInvolutionWeakOrderingPersistResultsInit(filename);

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

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

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

```

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

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

                possibleResiduums := DetectPossibleRank2Residuums(prevVertex, label, [1..Length(S)]);
                currVertex := fail;
                for res in possibleResiduums do
                    h := Length(res) / 2;

                    if CoxeterElementsCompare(res[h*2].vertex.element, y) then
                        currVertex := res[h*2].vertex;
                        break;
                    fi;
                od;

                if currVertex = fail then
                    currVertex := rec(element := y, twistedLength := k + 1, inEdges := [], outEdges := [], absIndex := absVertexIndex);
                    Add(vertices[1], currVertex);

                    absVertexIndex := absVertexIndex + 1;
                fi;

                newEdge := rec(source := prevVertex, target := currVertex, label := label, type := type, absIndex := absEdgeIndex);

                Add(edges[1], newEdge);
                Add(currVertex.inEdges, newEdge);
                Add(prevVertex.outEdges, newEdge);

                absEdgeIndex := absEdgeIndex + 1;
            od;
        od;

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

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

    TwistedInvolutionWeakOrderingPersistResultsInfo(persistInfo, W, matrix, theta,
```

```
        absVertexIndex - 1, k - 1);
    TwistedInvolutionWeakOrderingPersistResultsClose(persistInfo);

    return rec(numVertices := absVertexIndex - 1, numEdges := absEdgeIndex - 1,
        maxTwistedLength := k - 1);
end;
```

File twoa3.gap

```
# Calculates the poset Wk(theta).
TwistedInvolutionWeakOrdering3 := function (filename, W, matrix, theta)
    local persistInfo, maxOrder, vertices, edges, absVertexIndex, absEdgeIndex,
        prevVertex, currVertex, newEdge, possibleResiduums,
        label, type, deduction, startTime, endTime, endTypes, S, k, i, s, x, _y, y, n,
            m, h, res;

    persistInfo := TwistedInvolutionWeakOrderingPersistResultsInit(filename);

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

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

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

            prevVertex := vertices[2][i];
            for label in Filtered([1..Length(S)], n -> Position(List(prevVertex.inEdges
                , e -> e.label), n) = fail) do
                x := prevVertex.element;
                s := S[label];
                y := x*s;
                _y := s^theta*y;
                type := -1;

                possibleResiduums := DetectPossibleRank2Residuums(prevVertex, label,
                    [1..Length(S)]);
                currVertex := fail;
                for res in possibleResiduums do
                    m := CoxeterMatrixEntry(matrix, res[1].edge.label, res[2].edge.
                        label);
                    h := Length(res) / 2;

                    if h = 1 then
                        if m = 2 and res[h*2].edge.type = 1 and CoxeterElementsCompare(
                            res[h*2].vertex.element, _y) then
                            currVertex := res[h*2].vertex;
                            type := 1;
                            break;
                        fi;
```

```
                else
                    endTypes := [-1, res[h].edge.type, res[h+1].edge.type, res[h
                        *2].edge.type];
                    endTypes[1] := endTypes[3] + endTypes[4] - endTypes[2];

                    if endTypes[4] = 0 then
                        currVertex := res[h*2].vertex;
                        type := endTypes[1];
                        break;
                    elif endTypes = [1,1,1,1] then
                        if m = h or (Gcd(m,h) > 1 and CoxeterElementsCompare(res[h
                            *2].vertex.element, _y)) then
                            currVertex := res[h*2].vertex;
                            type := 1;
                            break;
                        fi;
                    elif endTypes = [0,1,0,1] then
                        if m = h or (Gcd(m,h) > 1 and CoxeterElementsCompare(res[h
                            *2].vertex.element, y)) then
                            currVertex := res[h*2].vertex;
                            type := 0;
                            break;
                        fi;
                    elif endTypes = [1,0,0,1] and m mod 2 = 1 then
                        if (m+1)/2 = h or (Gcd((m+1)/2,h) > 1 and
                            CoxeterElementsCompare(res[h*2].vertex.element, _y))
                            then
                            currVertex := res[h*2].vertex;
                            type := 1;
                            break;
                        fi;
                    fi;
                fi;
            od;

            if currVertex = fail then
                if CoxeterElementsCompare(x, _y) then
                    type := 0;
                    _y := y;
                else
                    type := 1;
                fi;

                currVertex := rec(element := _y, twistedLength := k + 1, inEdges :=
                     [], outEdges := [], absIndex := absVertexIndex);
                Add(vertices[1], currVertex);

                absVertexIndex := absVertexIndex + 1;
            fi;

            newEdge := rec(source := prevVertex, target := currVertex, label :=
                label, type := type, absIndex := absEdgeIndex);

            Add(edges[1], newEdge);
            Add(currVertex.inEdges, newEdge);
            Add(prevVertex.outEdges, newEdge);

            absEdgeIndex := absEdgeIndex + 1;
        od;
    od;

```

```
        TwistedInvolutionWeakOrderingPersistResults(persistInfo, vertices[2], edges[2])
            ;

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

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

    return rec(numVertices := absVertexIndex - 1, numEdges := absEdgeIndex - 1,
        maxTwistedLength := k - 1);
end;
```

B Benchmark results

W	$\lvert Wk(W, \mathrm{id})\rvert$	Time in seconds	Element comparisons
A_1	2	1.779_{-5}	1
A_2	4	3.591_{-5}	6
BC_2	6	4.968_{-4}	9
A_3	10	9.711_{-5}	31
BC_3	20	3.525_{-3}	75
A_4	26	2.978_{-4}	173
H_3	32	4.505_{-3}	126
D_4	44	1.563_{-2}	345
A_5	76	1.044_{-3}	1,181
BC_4	76	3.954_{-2}	802
F_4	140	1.056_{-1}	1,906
D_5	156	1.295_{-1}	3,502
A_6	232	4.520_{-3}	9,700
BC_5	312	4.013_{-1}	11,024
H_4	572	8.040_{-1}	12,938
D_6	752	2.736_{0}	65,308
A_7	764	2.564_{-2}	95,797
E_6	892	3.368_{0}	85,857
BC_6	1,384	8.577_{0}	193,218
A_8	2,620	1.993_{-1}	1,074,392
A_9	9,496	2.180_{0}	13,531,414
E_7	10,208	4.842_{2}	7,785,186
A_{10}	35,696	2.906_{1}	185,791,174
A_{11}	140,152	8.366_{2}	2,778,111,763
A_{12}	568,504	1.616_{4}	44,575,586,260

Table B.1: Benchmark results for TWOA1

W	$\lvert Wk(W, \mathrm{id})\rvert$	Time in seconds	Element comparisons
A_1	2	1.965_{-5}	1
A_2	4	5.572_{-5}	6
BC_2	6	6.161_{-4}	9
A_3	10	2.173_{-4}	29
BC_3	20	3.497_{-3}	57
A_4	26	8.811_{-4}	120
H_3	32	4.183_{-3}	93
D_4	44	1.292_{-2}	193
A_5	76	3.891_{-3}	501
BC_4	76	2.478_{-2}	344
F_4	140	5.020_{-2}	640
D_5	156	4.857_{-2}	975
A_6	232	1.724_{-2}	2,043
BC_5	312	9.745_{-2}	2,009
H_4	572	1.913_{-1}	2,578
D_6	752	3.493_{-1}	6,206
A_7	764	8.154_{-2}	8,569
E_6	892	3.720_{-1}	7,210
BC_6	1,384	6.780_{-1}	11,794
A_8	2,620	3.533_{-1}	36,218
A_9	9,496	1.645_{0}	157,611
E_7	10,208	7.904_{0}	100,996
A_{10}	35,696	8.005_{0}	697,613
A_{11}	140,152	4.155_{1}	3,172,316
E_8	199,952	3.501_{2}	2,399,476
A_{12}	568,504	2.148_{2}	14,711,015
A_{13}	2,390,480	1.192_{3}	69,917,802

Table B.2: Benchmark results for TWOA2

W	$\lvert Wk(W, \mathrm{id})\rvert$	Time in seconds	Element comparisons
A_1	2	2.085_{-5}	1
A_2	4	7.068_{-5}	3
BC_2	6	4.163_{-4}	5
A_3	10	3.275_{-4}	11
BC_3	20	2.273_{-3}	22
A_4	26	1.385_{-3}	40
H_3	32	2.758_{-3}	37
D_4	44	6.944_{-3}	62
A_5	76	6.903_{-3}	164
BC_4	76	1.594_{-2}	116
F_4	140	3.704_{-2}	219
D_5	156	2.778_{-2}	307
A_6	232	2.564_{-2}	691
BC_5	312	6.325_{-2}	655
H_4	572	1.076_{-1}	916
D_6	752	1.973_{-1}	1,989
A_7	764	1.887_{-1}	3,048
E_6	892	2.240_{-1}	2,347
BC_6	1,384	3.947_{-1}	3,942
A_8	2,620	5.340_{-1}	13,635
A_9	9,496	3.592_{0}	62,630
E_7	10,208	4.128_{0}	33,468
A_{10}	35,696	1.105_{1}	291,699
A_{11}	140,152	5.668_{1}	1,388,533
E_8	199,952	2.405_{2}	844,805
A_{12}	568,504	3.104_{2}	6,712,656
A_{13}	2,390,480	1.650_{3}	33,109,919

Table B.3: Benchmark results for TWOA3

C References

[1] Kathryn Brenneman, Ruth Haas, and Aloysius G. Helminck. Implementing an algorithm for the twisted involution poset for weyl groups. 2006.

[2] Francis Buekenhout and Arjeh M. Cohen. *Diagram geometry - related to classical groups and buildings*. Springer-Verlag, 2012. Draft from `http://www.win.tue.nl/~amc/buek/book1n2.pdf` (04.10.2012, 18:22).

[3] Tom Denton. Lifting property and poset structure of finite coxeter groups. 2009.

[4] Vinay V. Deodhar. Some characterizations of bruhat ordering on a coxeter group and determination of the relative möbius function. *Invent. Math. 39*, pages 187–198, 1977. MR0435249.

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

[6] Max Horn. *Involutions of Kac-Moody Groups*. PhD thesis, 2009.

[7] Axel Hultman. Fixed points of involutive automorphisms of the bruhat order. *Adv. Math. 195*, pages 283–296, 2005. MR2145798.

[8] Axel Hultman. The combinatorics of twisted involutions in coxeter groups. *Transactions of the American Mathematical Society, Volume 359*, pages 2787–2798, 2007. MR2286056.

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