A Class of PL-Homeomorphism Groups with Irrational Slopes

Jason Brown
Supervised by Dr Lawrence Reeves

Presented in partial fulfillment of the requirements of the degree
Master of Science

October 2018

School of Mathematics and Statistics
The University of Melbourne

Contents

1. Introduction

Groups of piece-wise linear and projective homeomorphisms of intervals in $\mathbb{R}$ have been the source of many rare examples and counter-examples in group theory. The prototype for these groups was Thompson's group F, and the related groups T and G. Thompson's group F can be described as:

1. The group of associative laws on binary operations [GG06].
2. The automorphism group of elements in the free monoidal category generated by a single object A and a morphism $A \otimes A \to A$ [FL10].
3. The group of piecewise linear homeomorphisms on the compact interval $[0, 1]$ with slopes that are powers of 2, and with finitely many breakpoints all lying in the dyadic rationals [CFP96].

The tendency for Thompson's groups to appear in such diverse settings has earned them the name *chameleon groups*. Definition 1 was historically the first [Bel07], given by Richard Thompson in 1965. Definition 3 has become the most widely used and, for the purposes of the generalisations we consider here, the most convenient. In this language, Thompson's group T is the group of homeomorphisms on $[0, 1]/0, 1 \cong S^1$ with slopes that are powers of 2, finitely many breakpoints all lying in the dyadic rationals and the additional requirement that dyadic rationals are mapped to dyadic rationals. G is the group of piece-wise linear right-continuous bijections on [0.1) with the same breakpoints and slopes. The group G was the first proven example of a finitely presented infinite simple group, and it was later shown that T and the commutator subgroup of F share this property. The group F is also thought to be a potential counter-example to the (now disproven) Von Neumann conjecture which states that a group is amenable if and only if it contains a non-abelian free group. While it has been shown that F does not contain a non-abelian free group, the amenability of F is still an open problem. It is, however, proven that F is not in the class EG of elementary amenable groups [CFP96]. As a finitely presented group not in EG which does not contain a non-abelian free group, it is either a counter-example to the Von Neumann conjecture for finitely presented groups (disproven in 2003 [OS03]) or the conjecture that finitely presented amenable groups are elementary amenable (disproven in 1998 [Gri98]).

The unusual properties of Thompson's groups have motivated the study of a number of generalisations. The groups $G_{n,r}$, sometimes called Higman-Thompson groups, are a generalisation of Thompson's group G introduced by Graham Higman in 1974 [Ste92]. They are the class of groups indexed by $n, r \in \mathbb{Z}$ of piecewise linear right-continuous bijections on the interval $[0, r)$ with slopes powers of n and breakpoints in $\mathbb{Z}[\frac{1}{n}]$. These groups were found to share with G the property of being a finitely presented infinite simple group for n even. Another generalisation studied, for example, by Bieri and Strebel [BS14], is groups of piecewise linear

homeomorphisms on non-compact intervals of $\mathbb{R}$, for example $\mathbb{R}_{\geq 0}$ or $\mathbb{R}$.

A generalisation for which there have been very few published results is to allow the slopes to be powers of an irrational number τ, and require breakpoints in $\mathbb{Z}[\tau]$. One exception to this is the case where $\tau = \frac{-1+\sqrt{5}}{2}$, first discussed by Sean Cleary in [Cle00] where the group was proven to have the FP_∞ property, and thus to be finitely presented, using results from [Bro87]. A much more recent work by Burillo, Nucinkis and Reeves [BNR18] has adopted a more combinatorial approach to prove a number of results linking the group, referred to as F_τ, with Thompson's group F. For example, F_τ also has a simple commutator subgroup, is finitely presented, infinite, and contains a copy of F.

The methods used in both [Cle00] and [BNR18] suggest a further generalisation to the case where $\tau = \frac{-k+\sqrt{k^2+4}}{2}$ for $k \in \mathbb{Z}_{\geq 1}$, which is the class of groups we consider here. We will refer to such a group corresponding to the integer k as $\mathcal{F}_{\tau_k}$. We will show that for each $k \in \mathbb{Z}_{\geq 1}$:

1. $\mathcal{F}_{\tau_k} \ncong \mathcal{F}_{\tau_h}$ for $h \neq k$
2. $\mathcal{F}_{\tau_k} \ncong F_h$ for $h \in \mathbb{Z}_{\geq 1}$
3. $\mathcal{F}_{\tau_k}$ is finitely presented
4. $\mathcal{F}_{\tau_k}$ contains a copy of F_k
5. $\mathcal{F}_{\tau_k}$ is not elementary amenable

Much of the important insights and proofs for these $\mathcal{F}_{\tau_k}$ groups come from studying combinatorial properties of a class of trees and the correspondence between these trees and the groups of homeomorphisms. The structure of this document reflects this. We begin with a formal definition of a broad class of groups of piece-wise linear homeomorphisms, called $F(l, A, P)$ groups, in Chapter 2. We then follow the path traced by the study of Thompson's group F and F_τ by looking in Chapter 3 at a class of subdivisions of the unit interval with the property that breakpoints of elements of $\mathcal{F}_{\tau_k}$ always lie in such a subdivision. We also introduce a notion of trees corresponding to such subdivisions, just as binary trees correspond to the subdivisions corresponding to the group F. The connection between pairs of such subdivisions is established in Chapter 4. We then establish a long list of results for the trees described in Chapter 3. This provides the tools to establish the infinite and finite presentations for $\mathcal{F}_{\tau_k}$ in Chapter 6. Chapter 7 then uses these presentations to derive a presentation for the abelianisation of $\mathcal{F}_{\tau_k}$ and a normal form closely mirroring the normal form given for F_τ in [BNR18]. Finally, using some known results for the group F_1 and general $F(l, A, P)$ groups we show that each $\mathcal{F}_{\tau_k}$ contains F, is not elementary amenable and does not contain a free non-abelian subgroup. This demonstrates a strong connection between the amenability of F_1 and $\mathcal{F}_{\tau_k}$.

2. $F(l, A, P)$ Groups

The class of $F(l, A, P)$ groups is a very broad generalisation of Thompson's group F where the slope groups, breakpoint sets and the interval are allowed to vary.

Definition ($F(l, A, P)$)**:** We construct a group of piecewise linear homeomorphism on a compact interval by choosing:

1. P, a multiplicative subgroup of $\mathbb{R}_{>0}$
2. A, a $\mathbb{Z}[P]$-submodule of $\mathbb{R}$ (where P acts on $\mathbb{R}$ by multiplication) satisfying $P \cdot A = A$
3. An element $l \in A \cap \mathbb{R}_{>0}$

$F(l, A, P)$ is then the group of piecewise linear homeomorphisms $f : [0, l] \to [0, l]$ with finitely many breakpoints, all in A, and with the slopes of all the linear segments in P. The group operation is composition. We will adopt the convention that multiplication on the left is precomposition, so $fg = g \circ f$.

We could similarly define the analogous generalisations $T(l, A, P)$ and $G(l, A, P)$ of T and G respectively, though they are not discussed in this paper.

Thompson's group F can be expressed as the $F(l, A, P)$ group $F\big([0, 1], \mathbb{Z}[\frac{1}{2}], \langle 2\rangle\big)$, where $\langle 2\rangle$ denotes the multiplicative subgroup of $\mathbb{R}_{>0}$ generated by 2. A related class of groups, are the groups $F\big([0, 1], \mathbb{Z}[\frac{1}{k}], \langle k\rangle\big)$, for $k \in \mathbb{Z}_{>1}$ which we will henceforth denote by F_{k-1}. Our reason for not using F_k to denote $F\big([0, 1], \mathbb{Z}[\frac{1}{k}], \langle k\rangle\big)$ is to highlight the correspondence we will demonstrate between the groups $\mathcal{F}_{\tau_k}$ and F_k. Note that with this definition, Thompson's group is F_1.

A comprehensive reference for this general class of groups is [BS14]. We will state two results for general $F(l, A, P)$ groups which are both found in [BS14].

Let $B = F(l, A, P)$ for some choice of l, A and P.

Proposition 1: *B is torsion free*

Proof. Assume $f \in B$ is not the identity. Let $D_x(f)$ denote the derivative at $x \in [0, 1]$ from the right (i.e. $\lim_{h\to 0^+}\left(\frac{f(a+h)-f(x)}{h}\right)$) and let a be the infimum of the set $\{x \in [0, 1] \mid f(x) \neq x\}$. Then $f(a) = a$, but $D_a(f) \neq 1$. Then $D_a(f^n) = (D_a f)^n \neq 1$ for any $n \in \mathbb{Z}$, so f is not a torsion element. We conclude that B is torsion free. □

In particular, B is either trivial or infinite.

The next theorem we state paraphrases an important result proven By Brin and Squier in [BS85].

Theorem (3.1 from [BS85])**:** *B does not contain a non-abelian free group.*

Every $F(l, A, P)$ group is therefore either amenable, or a counter-example to the Von Neumann conjecture.

We will investigate a class of $F(l, A, P)$ groups related to a class of algebraic integers we now define.

Definition (τ_k)**:** For $k \in \mathbb{Z}_{>0}$ we let τ_k denote the positive solution to the quadratic equation: $x^2 + kx = 1$ which is given by:

$$\tau_k := -\frac{k}{2} + \sqrt{\left(\frac{k}{2}\right)^2 + 1}$$

These algebraic integers are sometimes referred to as *metallic means*. We note that $\tau_k \in (0, 1)$ for all $k \in \mathbb{Z}_{>0}$.

Definition ($\mathcal{F}_{\tau_k}$)**:** For $k \in \mathbb{Z}_{\geq 1}$ we define $\mathcal{F}_{\tau_k}$ to be $F(1, \mathbb{Z}[\tau_k], \langle \tau_k \rangle)$

With this notation, the group discussed in [Cle00] and [BNR18] with slopes powers of $\frac{-1+\sqrt{5}}{2}$ is $\mathcal{F}_{\tau_1}$.

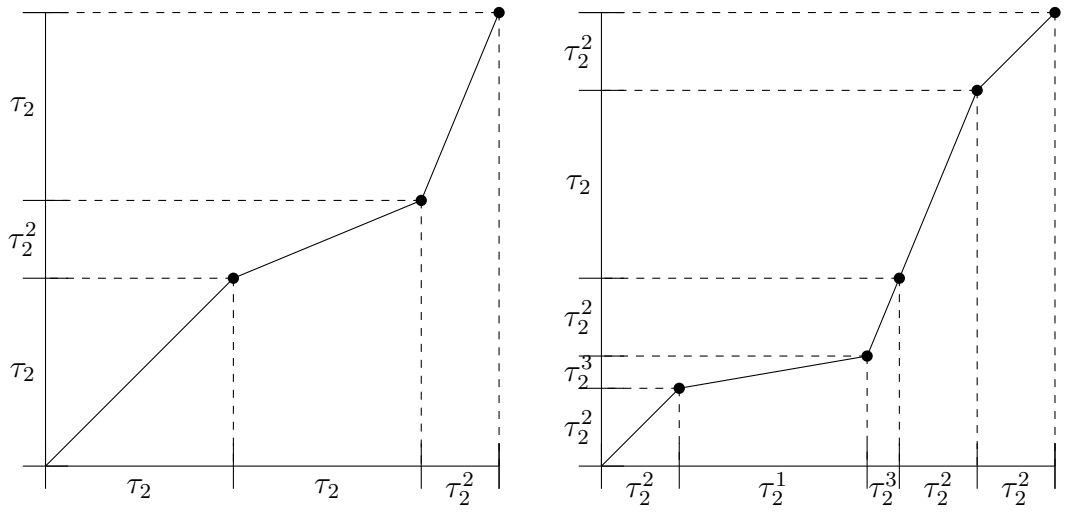


Figure 2.1: Two elements of $\mathcal{F}_{\tau_2}$

3. k-subdivisions

3.1 Definitions

For any τ_k there is a natural way to divide the unit interval into k long pieces of length τ_k and one short piece of length τ_k^2. In fact there are $k+1$ ways to do this, since such a subdivision is uniquely determined by where the short piece of length τ_k^2 appears. We can then repeat this subdivision process to each of these subintervals to split the interval further. We will call a subdivision obtained in this way a k-subdivisions, which we now define formally.

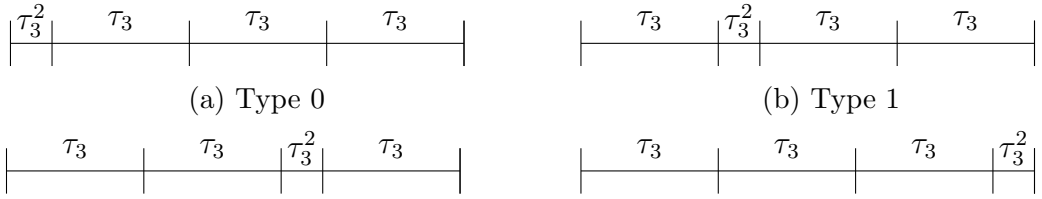
Definition (k-partition)**:** A k-partition of an the interval $[0,1]$ of type $0 \leq i \leq k$ is a partition of the form:

$$[0,\tau_k],[\tau_k,2\tau_k],\ldots,[(i-1)\tau_k,i\tau_k],[i\tau_k,i\tau_k+\tau_k^2],[i\tau_k+\tau_k^2,(i+1)\tau_k+\tau_k^2],\ldots,[(k-1)\tau_k+\tau_k^2,1]$$

Similarly, a k-partition of an a general interval $[a,b]$ of type $0 \leq i \leq k$ is defined to be the image of the intervals of the k-partition of $[0,1]$ of type i under the map

$$x \mapsto a + (b-a)x$$

An example is more illuminating than the definition given above. Figure 3.1 shows all the possible 3-partitions of the unit interval.



(a) Type 0 (b) Type 1

(c) Type 2 (d) Type 3

Figure 3.1: The four 3-partitions of the unit interval

Definition (k-subdivision)**:** A k-subdivision is either:

1. $[0,1]$
2. A collection of intervals obtained by replacing an interval in a k-subdivision with a k-partition of that interval (of any type)

For example, (*) shows a 2-subdivision obtained by a sequence of two 2-partitions.

$$
\begin{aligned}
&[0,1] \\
\longrightarrow &[0,\tau_k],[\tau_k,2\tau_k],[2\tau_k,1] \\
\longrightarrow &[0,\tau_k^2],[\tau_k^2,\tau_k(\tau_k+\tau_k^2)],[\tau_k(\tau_k+\tau_k^2),\tau_k],[\tau_k,2\tau_k],[2\tau_k,1] \qquad (*)
\end{aligned}
$$

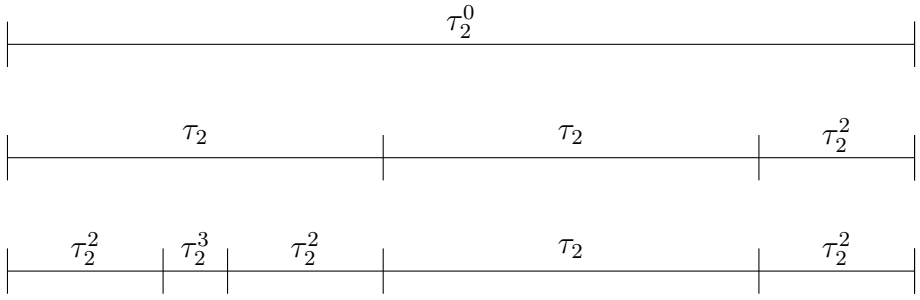


Figure 3.2: The 2-subdivision * obtained by two 2 partitions

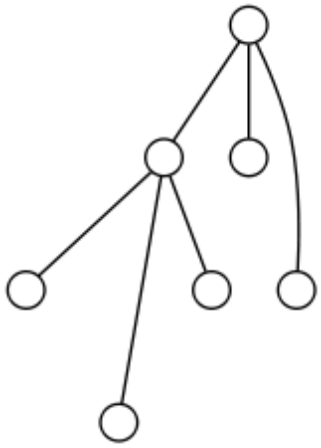
A natural way to represent a subdivision is to construct a tree, where branching in the tree corresponds to a partition of the interval in the subdivision. Studying such a representation has proven fruitful in the study of Thompson's groups [CFP96; Bur18] and the group $\mathcal{F}_{\tau_1}$ [BNR18]. With some small modifications, we can use trees in a similar way to study the general groups $\mathcal{F}_{\tau_k}$. For a given k-subdivision, we will construct a tree which represents it by starting with a single node to represent the interval $[\,0\ ,\ 1\,]$. Then each time a k-partition is performed on an interval in the k-subdivision, we will add $k+1$ children to the node representing that interval in the tree, identifying each child with an interval produced by the k-partition in left-to-right order. Since each k-partition has a distinguished short interval, we need to somehow distinguish the corresponding child of each node of our tree. We will do this by drawing it to be twice as far below its parent as the other children. This convention has the property that all nodes at a given height on the tree correspond to intervals of the same length in the subdivision.

Figure 3.3a shows an example of such a tree which corresponds to the 2-subdivision (*).

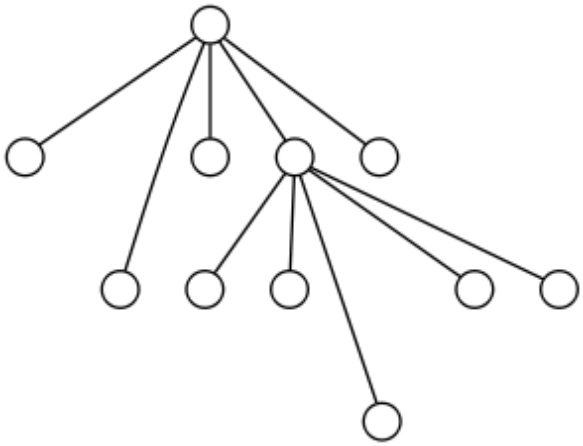
These trees differ from the usual ordered $(k+1)$-ary trees by having edges with weights: either 1 or 2. Each node of the tree is either a leaf or has $(k+1)$ children (i.e. the tree is *full*), and exactly one child is attached with an edge of weight 2. We will refer to such a tree associated to a value $k \in \mathbb{Z}_{\geq 1}$ as a k-tree.

We will make some definitions to make it easier to discuss k-trees:

Definition (height)**:** We define the *height* of a node in a k-tree to be the sum of the weights of the edges along the unique arc from that node to the root of the tree. Note that this corresponds to how far below the root the node lies with our convention of drawing weight 2 edges twice as long as weight 1 edges. We define the *height* of a k-tree to be the maximum of the heights of its leaves.



(a) A 2-tree. This is the tree representation for the subdivision given by (*) with leaf sequence $(2, 3, 2, 1, 2)$



(b) A 4-tree, with leaf sequence $(1, 2, 1, 2, 2, 3, 2, 2, 1)$

Figure 3.3: Two k-trees with their associated leaf sequences

Definition (Type of a node)**:** The *type* of a non-leaf node N in a k-tree is the number of short children of N to the left of the long child of N. Note that the type of N must be greater than or equal to 0 and less than or equal to k.

For example, the root nodes of T_1, T_2 and T_3 from Figure 3.5 have types 0, 1 and 2 respectively. Note that our definition for the type of a node corresponds to our definition of the type of a k-partition: performing a k-partition of type i to an interval is equivalent to replacing the leaf node that represented that interval with a node of type i in the k-tree.

We will use the notation $N(i)$ to indicate the $(i+1)^{\text{th}}$ child (from the left) of the node N. For a non-leaf node, N, the children of N from left to right will thus be $N(0), \ldots, N(k)$. If N has type l, then $N(l)$ is the long child of N. Henceforth, we shall also use 0-indexing for the leaves, so that the i^{th} leaf node is the leaf with i leaves to its left, and the 0^{th} leaf is the left-most leaf.

Definition (leaf sequence)**:** The *leaf sequence* of a k-tree is the tuple produced by reading the heights of the leaf nodes of the k-tree from left to right. The leaf sequence of a k-subdivision will be the leaf sequence of a corresponding k-tree.

Examples of k-trees with their corresponding leaf sequences are given in Figure 3.3.

Definition (Subtree under a node)**:** For a node N of a k-tree we will define the *subtree under* N to be the k-tree composed of all the descendants of N.

Definition (Tree refinement)**:** We will say a k-tree T is a *refinement* of another k-tree T' if T can be obtained by adding nodes to T'.

We will also introduce a more compact diagram for k-trees, which will prove much more pleasant to manipulate, but sacrifices some of the visual connection to subdivisions. The

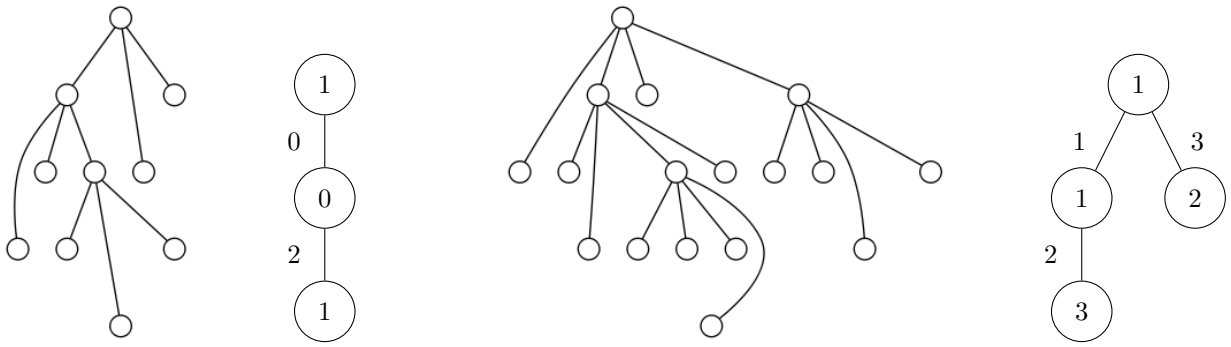


Figure 3.4: Two examples of k-trees, with $k = 2$ and 3, comparing the compact and standard notations

principle motivating the compact notation is the bijection between full k-ary trees and k-ary trees given by removing all leaf nodes. Because k-trees have a distinguished child, they carry more information than usual k-ary trees. We will keep track of this information by labelling each node of the k-tree with its type before removing the leaf nodes. We will also label the edges connecting a parent node to its children, so we can keep track of which of the $k + 1$ children each node represents without including the leaves in the diagram.

A comparison of our standard and compact notations for trees is given in Figure 3.4.

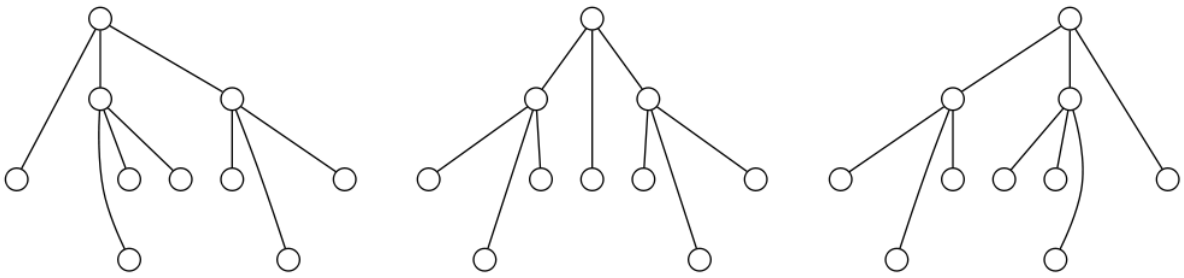
We recover a k-subdivision from a k-tree by looking at its leaf sequence. More precisely, the sequence of break points $(0 = b_0, b_1 \ldots, b_{n-1}, 1 = b_n)$ of the k-subdivision can be recovered from the leaf sequence $(l_1, \ldots, l_n)$ of the k-tree by the recursive computation:

$$b_{i+1} = b_i + \tau_k^{l_i}$$

though it will sometimes be more convenient to identify the k-subdivision with its leaf sequence, rather than its set of breakpoints.

It is not possible in general to recover a k-tree from a given k-subdivision, since there are often many k-trees corresponding to a single k-subdivision. This is a property that k-trees and k-subdivisions do not share with the binary trees and subdivisions used to describe Thompson's groups. Figure 3.5 gives an example of this: three distinct 2-trees are shown which all have the same leaf sequence, and thus represent the same k-subdivision. We will later obtain a class of relations on our group $\mathcal{F}_{\tau_k}$ that is connected to the non-uniqueness of k-trees representing a given k-subdivision.

We will say that two k-trees, T_1 and T_2, which have the same leaf sequence are *leaf equivalent*, written $T_1 \sim T_2$, and thus k-subdivisions are in bijection with leaf-equivalence classes of k-trees.



(a) T_1 (b) T_2 (c) T_3

Figure 3.5: Three distinct 2-trees with the same leaf sequence $(2, 3, 2, 2, 2, 3, 2)$

3.2 Grafting

There is another way to describe this equivalence in terms of an operation that can be performed on k-trees which will prove useful. We will call this operation *grafting* (to the left or right). Below we will describe right grafting at a node; left grafting is the inverse operation.

To perform a right graft at a node N we require that:

1. N is a non-leaf node of type $n < k$
2. $C = N(n + 1)$ is a non-leaf node of type $m < k$

We then perform a *right graft at* N in the following way:

1. The long child $L = N(n)$ is detached from N and attached to the left of C as a short child, becoming $C(0)$ (bringing any subtree with it).
2. The short child $S = C(k)$ is detached from S and attached to N as a long node, becoming $N(n + 1)$ (bringing any subtree with it).

Figure 3.6 shows a right graft performed at the root node of T_1 from Figure 3.5 to produce T_2 in standard form.

It will be important to note that performing a right graft operation at a node N with type n changes its type to $n+1$, as well as increasing the type of its child $C = N(n+1)$ by one, and moving it from $N(n + 1)$ to $N(n)$. From this we see that the conditions required to perform the inverse left graft operation at N are:

1. N is a non-leaf node of type $n > 0$
2. $C = N(n - 1)$ is a non-leaf node of type $m > 0$

Note that the result of step 1 is not a valid k tree: C would have one too many short children and N would have one too few long children. The height and order of all nodes remains unchanged however. Step 2 then fixes this imbalance in children, again without changing heights or orders of any of the nodes. We conclude that:

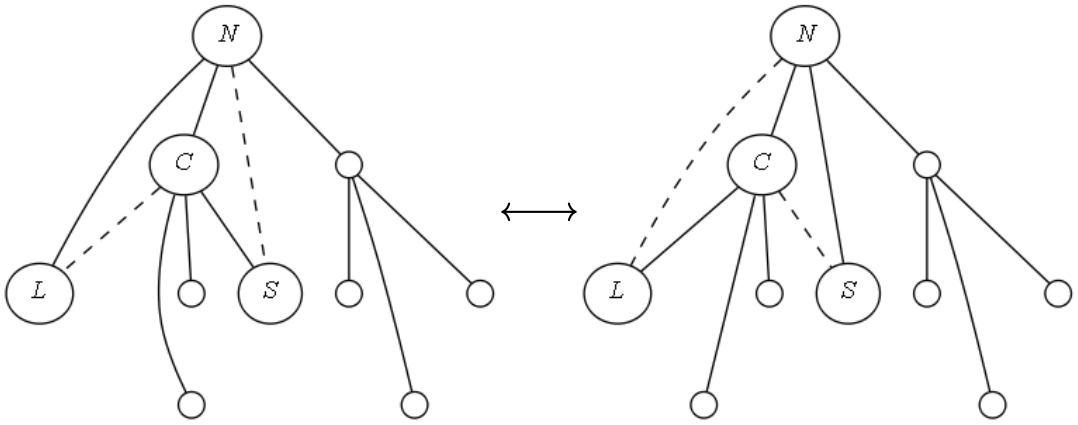


Figure 3.6: Using a right graft operation to transform T_1 from Figure 3.5 into T_2

1. The grafting operations are well-defined on k-trees: their output is always another k-tree.
2. The output of a grafting operation is always leaf equivalent to the input: the graft operation does not change the height, order, or property of being a leaf for any of the nodes and thus the leaf sequence remains unchanged.

So for any two k-trees, T_1 and T_2, if there exists a sequence of graft operations by which T_1 can be transformed to T_2 we conclude that $T_1 \sim T_2$. We will show in Chapter 5 that the converse is also true: if $T_1 \sim T_2$ then there exists a sequence of graft operations by which T_1 can be transformed to T_2.

Our compact tree notation will be the most useful for identifying potential graft operations and realising them. In these diagrams, a right graft operation will be possible whenever a node of type $n < k$ has a child of a node of type $m < k$ attached with an edge labeled $n + 1$. A left graft operation will be possible whenever a node of type $n > 0$ has a child of type $m > 0$ attached by an edge labeled $n - 1$. The realisation of a graft operation in compact notation is shown in Figure 3.7. Note that the subtrees seem to rotate around the child node C.

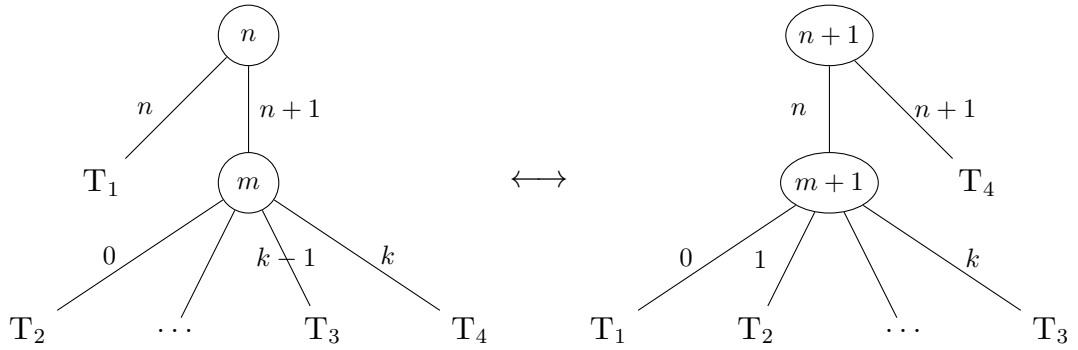


Figure 3.7: The graft operation at a node of type n shown in compact tree notation. The subtrees under the children of the nodes being grafted are indicated by T_i

Lemma 1: *Let T be a k-tree, and N a non-leaf node in T with type l. Then by expanding at most one leaf node (adding k short children and one long child) and performing graft operations, the type of N can be decreased, if $N \neq 0$, or increased, if $N \neq k$.*

Proof. We will first describe the algorithms by which this can be done, and then justify that the algorithm is well-defined and produces the desired result.

We give the description of the two algorithms for decreasing and increasing the type of a node N using pseudo-code in Alg. 3.1.

We will say DECREASETYPE or INCREASETYPE is *effective* for an appropriate input node N of type n (i.e. $n < k$ for INCREASETYPE and $n > 0$ for DECREASETYPE) if it increases (resp. decreases) the type of N without expanding more than one leaf node (i.e replacing a leaf node with a node of type i for some $0 \leq i \leq k$).

We first note that DECREASETYPE is effective if $N(n-1)$ is a non-leaf node of type greater than 0, because in this case the condition for a left graft at N is satisfied, and the left graft will decrease the type of N. Similarly, INCREASETYPE is effective when $N(n+1)$ is a non-leaf node of type less than k. No leaf nodes are expanded in these cases.

We also see that if $C = N(n-1)$ is a leaf, then the algorithm DECREASETYPE replaces C with a node of type k, and then a left graft is performable, so DECREASETYPE is effective when $N(n-1)$ is a leaf. Similarly, INCREASETYPE is effective when $C = N(n+1)$ is a leaf.

It remains to consider the cases where $C = N(n-1)$ or $C = N(n+1)$ is of type 0 or k respectively. We can prove this by induction on the height of N.

Because all nodes of height 2 have only leaves as children, it follows that both algorithms are effective whenever N is of height 2. If we assume that the algorithms are both effective for node heights less than or equal to some $p \in \mathbb{Z}_{\geq 2}$, then we can show that the algorithms are also both effective for an input N of height $p+1$. If $C = N(n-1)$ is of type 0, then C is a node of height less than or equal to p, so INCREASETYPE(C) successfully increases the type of C (by our induction hypothesis), and then a left graft is performable. Similarly, if $C = N(n+1)$ is of type k, then DECREASETYPE(C) successfully decreases the type of C, and then a right graft operation is performable. We have already shown that the algorithms are effective for other cases of $N(n-1)$ and $N(n+1)$ so we conclude that both DECREASETYPE and INCREASETYPE are effective for all input nodes of height $p+1$.

By induction, we conclude that both algorithms are effective for all inputs. □

3.3 Refinements

Thompson's groups are studied using binary subdivisions and trees. Binary subdivisions are defined recursively as either the interval $[0, 1]$ or the result of splitting an interval in a binary subdivision evenly in two. These subdivisions have the property that for any two subdivisions, there exists a third subdivision which is a common refinement. This is also

```
1: procedure DECREASETYPE(node N)
2:     n ← type of N
3:     C ← N(n − 1)
4:     if C is a leaf then
5:         add a node of type k to C
6:     else
7:         m ← type of C
8:         if m = 0 then
9:             INCREASETYPE(C)
10:        end if
11:    end if
12:    perform a left graft at N
13: end procedure
```

(a) Decreasing the type of a node

```
1: procedure INCREASETYPE(node N)
2:     n ← type of N
3:     C ← N(n + 1)
4:     if C is a leaf then
5:         add a node of type 0 to C
6:     else
7:         m ← type of C
8:         if m = k then
9:             DECREASETYPE(C)
10:        end if
11:    end if
12:    perform a right graft at N
13: end procedure
```

(b) Increasing the type of a node

Alg. 3.1: These coupled algorithms describe how the type of any given node can be changed by adding at most one node to the tree, and then performing graft operations. Figure 3.8 gives an example of the output of the INCREASETYPE algorithm

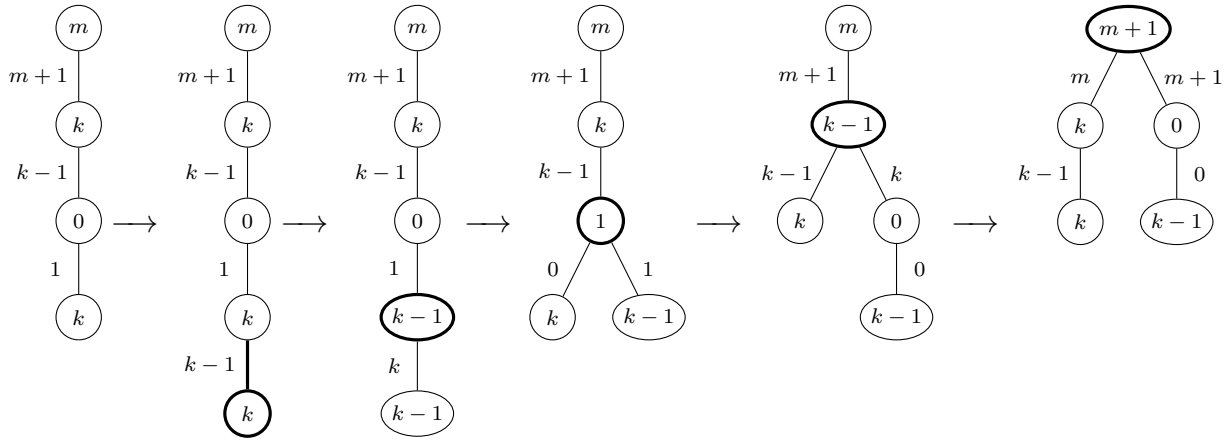


Figure 3.8: The operation of the algorithm INCREASETYPE on a sample input, where $m < k$. This is an example where a node of the tree must first be expanded to increase the type of the type m node N. Note that these diagrams are suppressing other nodes that may be attached to the tree but are not relevant to or altered by the algorithm.

true for k-subdivisions, though not obvious. To prove this fact we will first establish some technical results about k-subdivisions.

Formally, what we would like to show is that for any two k-subdivisions P_1 and P_2 there exists a k-subdivision P such that the set of breakpoints of P_1 and P_2 are both subsets of the set of breakpoints of P. We will prove a slightly more general result, after introducing a more general class of subdivisions.

Definition (improper k-subdivision)**:** An *improper k-subdivision* is a partition of the unit interval into finitely many intervals whose length is a power of τ_k.

Note that all k-subdivisions are also improper k-subdivisions, however there exist improper k-subdivisions which are not k-subdivisions, for example the improper 1-subdivision given by:

$$P = \left\{ \left[0 \ , \ \tau_1^4 \right], \left[\tau_1^4 \ , \ \tau_1^4 + \tau_1 \right], \left[\tau_1^4 + \tau_1 \ , \ 1 \right] \right\} \qquad (\dagger)$$

We note that $(1 - (\tau_1^4 + \tau_1)) = \tau_1^3$. This couldn't be a 1-subdivision because the only way to obtain an interval of length τ_1 in a k-subdivision is by performing a 1-partition on the unit interval, and when we do this the τ_1 interval contains either 0 or 1.

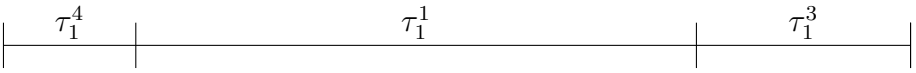


Figure 3.9: The improper 1-subdivision (†) which is not a 1-subdivision

To simplify notation we will henceforth use the leaf sequence notation; with this notation P is written as $(4, 1, 3)$.

Definition (uniform k-subdivision)**:** A *uniform k-subdivision of level l* is an improper k-subdivision satisfying that all intervals have length τ_k^l or τ_k^{l+1}. This is equivalent to all elements of the leaf sequence of the subdivision being either l or $l + 1$

An example of a uniform 2-subdivision of level 3 is given by the leaf sequence $(3, 3, 3, 3, 3, 4, 4)$. Note that this is not a 2-subdivision, whereas the uniform 2-subdivision given by $(3, 3, 4, 3, 3, 4, 3)$ is.

Definition (refinement)**:** For two improper k-subdivisions, P and P', if there exists a finite sequence $P_1, \ldots, P_n$ of improper k-subdivisions such that $P_1 = P$, $P_n = P'$ and each P_{i+1} is obtained from P_i by performing a k-partition on an interval in P_i, then we say that P' is a *refinement* of P.

We note that this notion of refinement corresponds with our notion of refinement for k-trees. If T is a k-tree representing a k-subdivision P, then for any refinement P' of P, there exists a refinement T' of T which represents P' obtained by replacing each leaf with a type i node whenever the corresponding interval in P undergoes a type i partition. Similarly, any refinement T' of T represents a k-subdivision which is a refinement of P.

We note also that any refinement of a k-subdivision remains a k-subdivision. We will see shortly that the converse is not true, some k-subdivisions can be expressed as refinements of improper k-subdivisions which are not k-subdivisions.

Now we have the language to describe a result that we will need for our discussion of the $\mathcal{F}_{\tau_k}$ groups: every two improper k-subdivisions have a common refinement. We prove this below in several steps.

Lemma 2: *Every improper k-subdivision has a uniform refinement. Moreover, if l is the maximum value in the leaf sequence of P, then P has a uniform refinement of level $l-1$*

Proof. Let P be an improper k-subdivision, let l be the maximum value in its leaf sequence and let h be the minimum value. We can prove this result by induction on $l-h$. If $l-h \leq 1$ then the improper k-subdivision is already uniform of level $l-1$. Assume for some $n \geq 1$ that in cases where $l-h \leq n$ there exists a uniform refinement of P of level $l-1$.

Then if $l-h = n+1$, we can perform a k partition of each interval with size h in the leaf sequence, removing the intervals of size h and adding intervals of size $h+1$ and $h+2$. We will call this refinement P'. Note that $h+2 \leq l$, since by hypothesis $l-h = n+1 \geq 2$. So the minimum value in the leaf sequence of P' is now $h+1$ but the maximum value is still l. Then P' has a uniform refinement by the induction hypothesis, and thus so does P. □

Lemma 3: *If P is a uniform k-subdivision of level l, and $h > l$ then P has a refinement which is a uniform k-subdivision of level h*

Proof. Assume P is a uniform k-subdivision of level l.

It suffices to show that P has a refinement that is a uniform k-subdivision of level $l+1$.

All the intervals in P have length either l or $l+1$. If there are no intervals of length l, then P is already a uniform k-subdivision of level $l+1$. Otherwise, performing a k-partition on one of the intervals of length l gives a refinement P' of P whose leaf sequence has maximum value $l+2$. By Lemma 2 there exists a refinement of P' which is uniform at level $l+1$, which is therefore also a refinement of P. □

Lemma 4: *The leaf sequence of a uniform k-subdivision at a given level is unique up to permutation.*

Proof. It suffices to show that for any given $l \in \mathbb{Z}_{\geq 0}$ there exists unique $n, m \in \mathbb{Z}$ such that $1 = n\tau_k^l + m\tau_k^{l+1}$. We prove this by contradiction. If this were not the case, then there exists a distinct pair $n', m' \in \mathbb{Z}$ such that $n\tau_k^l + m\tau_k^{l+1} = n'\tau_k^l + m'\tau_k^{l+1}$, from which it follows that:

$$\tau_k = \frac{n-n'}{m'-m}$$

But τ_k is irrational for all $k > 0$, so we have a contradiction. □

The following very technical lemma represents the crucial step in the proof of common refinements.

Lemma 5: *Let P and Q be two uniform k-subdivisions of level l such that the leaf sequence of Q is obtained by switching an $l+1$ with an l to its right in the leaf sequence of P. Let P' be a refinement of P. Then there exists a common refinement of P' and Q.*

Proof. As a refinement of P, P' is constructed by performing a k-subdivision on each interval of P. By our correspondence between k-trees and k-subdivisions, we can therefore think of

P' as a collection of k-trees, one for each interval in P. We will refer to the i^{th} interval of P as P_i and we will refer to the k-tree associated with P_i as T_i. Similarly, let Q_i denote the i^{th} interval in the partition Q. We now want to construct a refinement Q' of Q, or equivalently an association of k-trees with each Q_i such that $Q' = P'$. Assume that the l swapped with an $l+1$ interval to its left in P to obtain Q is the j^{th} element of the leaf sequence of P. Let N denote the root node of T_1.

Case 1: the type of N_j is $s < k$

Then we construct a refinement Q' of Q by associating k-trees to each interval Q_i in Q:

1. For $i \neq j, j-1$, attach T_i to Q_i.
2. To Q_j (of size $l+1$) attach the subtree of T_j under $N_j(k)$.
3. To Q_{j-1} (of size l) attach the tree T' constructed as follows:
   (a) Let the root node N' of T' be of type $s+1$.
   (b) Attach T_{j-1} to $N'(0)$.
   (c) To each other child $N'(r)$ of N', attach the subtree of T_j with root node $N_j(r-1)$

Claim: *Q' represents the same improper k-subdivision as P'.*

Proof. We need to check that the leaf sequences of P' and Q' are the same. To do this we will make use of three facts:

1. When a k-subdivision with leaf sequence $(l_1, \ldots, l_n)$ is performed on an interval of size t (i.e. of length τ_k^t) then the leaf sequence of the resulting subdivision is given by $(l_1 + t, \ldots, l_n + t)$.
2. For a k-tree T, and some node N in T, the leaf sequence of the subtree of T under N is given by subtracting the height of N from the subsequence of the leaf sequence of T corresponding to the leaf nodes which are descendants of N.
3. $P_{j-1} \cup P_j = Q_{j-1} \cup Q_j$. This is because the segments of the leaf sequences of P and Q to the left of P_{j-1} and Q_{j-1} respectively are the same, and the leaf sequences of P and Q to the right of P_j and Q_j are the same by hypothesis.

To check that Q' and P' are the same, we therefore only need to look at the leaf sequence below the intervals P_{j-1}, P_j, Q_{j-1} and Q_j since all the other intervals' subdivisions are constructed to be the same. For convenience we will use $L(T)$ to denote the leaf sequence of a k-tree T (only in this proof) and we will also use $D^t(L)$ to indicate the result of adding t to each term in the leaf sequence L.

Below the size $(l+1)$ interval P_{j-1} in P' we have the leaf sequence $D^{l+1}(L(T_{j-1}))$ and below the size l interval P_j in P' we have the leaf sequence $D^l(L(T_j))$ (see Fact 1 above). We can therefore express the leaf sequence $L_{P'}$ of P' over $P_{j-1} \cup P_j$ as the concatenation:

$$L_{P'} = D^{l+1}(L(T_{j-1}))D^l(L(T_j))$$

Let S_r denote the subtree of T_j below $N_j(r)$. From our construction of Q', we see that the leaf sequence of Q' under the size $l+1$ interval Q_j is given by $D^{l+1}L(S_k)$. The leaf sequence of Q' under the size l interval Q_{j-1} is given by:

$$D^l\big(D^1L(T_{j-1})D^1L(S_0)\dots D^2L(S_s)\dots D^1L(S_{k-1})\big)$$

Concatenating these gives the leaf sequence $L_{Q'}$ of Q' under the interval $Q_{j-1}\cup Q_j$ as:

$$\begin{aligned}
L_{Q'} &= D^l\big(D^1L(T_{j-1})D^1L(S_0)\dots D^2L(S_s)\dots D^1L(S_{k-1})\big)D^{l+1}L(S_k)\\
&= D^{l+1}L(T_{j-1})D^l\big(D^1L(S_0)\dots D^2L(S_s)\dots D^1L(S_{k-1})\big)D^{l+1}L(S_k)\\
&= D^{l+1}L(T_{j-1})D^l\big(D^1L(S_0)\dots D^2L(S_s)\dots D^1L(S_{k-1})D^1L(S_k)\big)\\
&= D^{l+1}L(T_{j-1})D^l(L(T_j))\\
&= L_{P'}
\end{aligned}$$

We conclude that P' and Q' are the the same subdivision. □

Thus $P'=Q'$ is a common refinement of P' and Q.

Case 2: $s=k$ Assume the type of N is k. By Lemma 1, we can expand at most one leaf of T_j, then perform graft operations to change the type of N_j to $k-1$. This corresponds to subdividing P' to get a further refinement P''. Then, by the above argument, we can construct a Q refinement $Q'=P''$, and thus P'' is a common refinement of Q and P'. □

We will give an example to clarify the construction of Q' in the proof above, as the details of the proof can be confusing. Let P be the uniform 2-subdivision given by the leaf sequence $(1,2,1)$ and let Q be the uniform 2-subdivision given by the leaf sequence $(1,1,2)$. Note that P and Q satisfy the conditions described in Lemma 5. We now choose a refinement of P. We do this by picking a 2-tree for each interval in P, since this represents a subdivision process for that interval. Three such trees are shown as T_1, T_2 and T_3 in Figure 3.10. Following the construction for the three corresponding trees to construct the refinement Q' gives the three trees T_1', T_2' and T_3'. Observe from the diagrams that P' and Q' have the same leaf sequence.

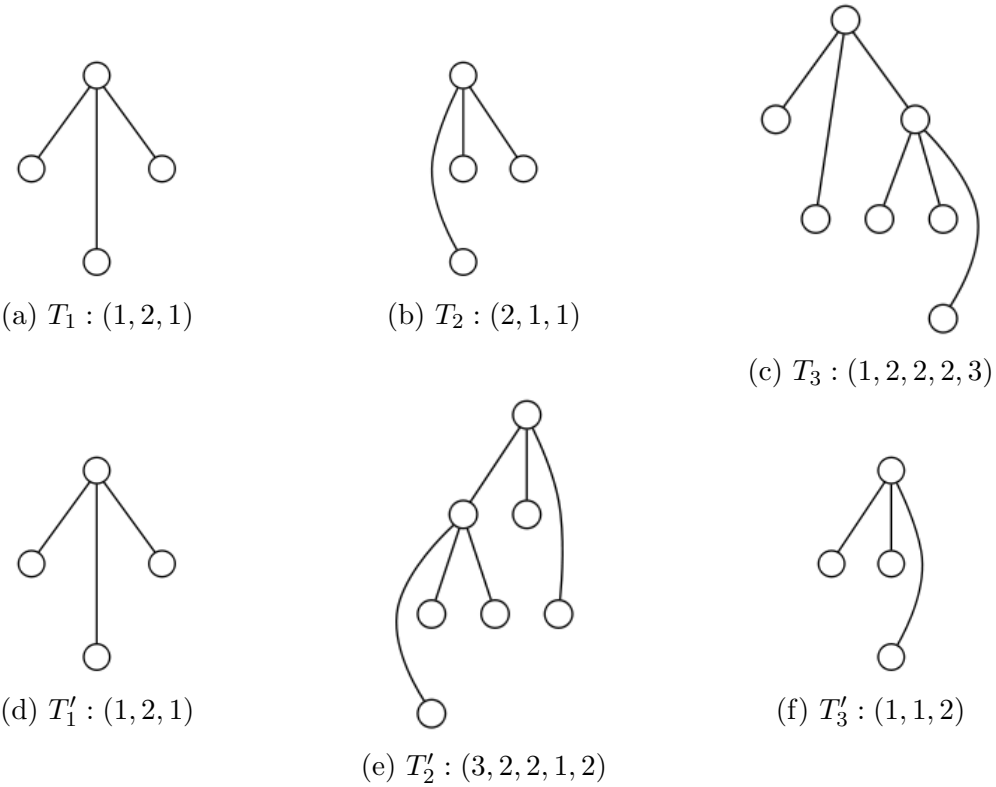
Lemma 6: *If P and Q are two uniform k-subdivisions of level l, then P and Q have a common refinement*

Proof. Assume P and Q are both uniform k-subdivisions of level l. By Lemma 4 the leaf sequence of Q is a permutation of the leaf sequence of P. We can therefore get from P to Q by performing a sequence $s_1, s_2, \dots, s_n$ of operations where we switch adjacent l and $l+1$'s in the leaf sequence of P. Let $Q_1 = s_1(P)$, $Q_2 = s_2(Q_1)$ up to $Q = Q_n = s_n(Q_{n-1})$.

We now obtain our result by induction.First, note that by Lemma 5 we have that there exists a common refinement P_1 of P and Q_1. Now assume we have a common refinement P_i for P and Q_i, for $i = d-1$. By Lemma 5 there exists a common refinement P_d of P_{d-1} and Q_d, which is thus a common refinement of P and Q_d.

By induction, there exists a common refinement of P and $Q = Q_n$. □

Lemma 7: *Any two improper k-subdivisions have a common refinement*



(a) $T_1 : (1, 2, 1)$ (b) $T_2 : (2, 1, 1)$ (c) $T_3 : (1, 2, 2, 2, 3)$

(d) $T_1' : (1, 2, 1)$ (e) $T_2' : (3, 2, 2, 1, 2)$ (f) $T_3' : (1, 1, 2)$

Figure 3.10: The three trees along the top represent P'. The leaf sequence for P' is thus given by $(2, 3, 2,\ 4, 3, 3,\ 2, 3, 3, 3, 4)$. The three trees along the bottom are the trees constructed to form Q'. Note that the leaf sequence of Q', $(2, 3, 2,\ 4, 3, 3, 2, 3,\ 3, 3, 4)$, is the same as that of P'.

Proof. Let P and Q be two improper k-subdivisions.

By Lemma 2, both P and Q have uniform refinements, which we shall denote P' and Q' respectively. Let l_P denote the level of P' and let l_Q denote the level of Q'. Without loss of generality, we assume that $l_P < l_Q$. By Lemma 3 there exists a refinement P'' of P' which is a uniform k-subdivision of level l_Q. Then by Lemma 6 there exists a common refinement of P'' and Q' which is therefore also a common refinement of P and Q. □

Corollary 1: *If T and S are k-trees, then there exists a (tree) refinement T' of T and a refinement S' of S such that S and T represent the same k-subdivision.*

Proof. Let P be the k-subdivision represented by T and let Q be the k-subdivision represented by S. By Lemma 7 there exists a common refinement R of P and Q. We can then obtain tree refinements T' of T and S' of S which both represent this refinement. □

Note that though T' and S' from the above proof may represent the same k-subdivision, they will not in general be the same k-tree.

Lemma 8: *If P is an improper k-subdivision then it has a refinement which is a k-subdivision.*

Proof. Let Q be any k-subdivision (for example the unit interval). Then by Lemma 7 there exists a common refinement P' of P and Q. Because P' is a refinement of the k-subdivision

Q, P' is also a k-subdivision. □

Another property of binary subdivisions that is important in studying Thompson's groups is that for any finite set of dyadic rationals in the unit interval, there exists a binary subdivision of the unit interval whose breakpoints are a superset of the set of dyadic rationals. This property also holds for k subdivisions and finite subsets of the ring $\mathbb{Z}[\tau_k]$. To show this we first establish some properties of elements of this ring.

Lemma 9: *If $x \in \mathbb{Z}[\tau_k]$ then there exists $m, n, N \in \mathbb{Z}$ such that*

$$x = m\tau_k^N + n\tau_k^{N+1}$$

Proof. It suffices to show that for any $r, N \in \mathbb{Z}$ with $r \leq N+1$, $\tau_k^r = m\tau_k^N + n\tau_k^{N+1}$ for some $m, n \in \mathbb{Z}$. Once we have proven this for $x = \tau_k^r$, then the result follows for any $x = \sum_{i=0}^{l} \alpha_i \tau_k^i$, since we can write each $\alpha_i \tau_k^i$ term in the form $m\tau_k^l + n\tau_k^{l+1}$.

We can prove this by induction on the difference between r and N. First note that the proposition holds trivially if $r = N+1$ or if $r = N$.

Assume that the result holds in the case where $N+1 \geq r \geq N-i$ for some $i \geq 0$.

Then if $r = N-i-1$ we can write:

$$\tau_k^r = (\tau_k^2 + k\tau_k)\tau_k^r = \tau_k^{r+2} + k\tau_k^{r+1} = \tau_k^{N-i+1} + k\tau_k^{N-i}$$

By the induction hypothesis, there exists $m, n, m', n' \in \mathbb{Z}$ satisfying:

$$\tau_k^{N-i+1} = m\tau_k^N + n\tau_k^{N+1}, \qquad \tau_k^{N-i} = m'\tau_k^N + n'\tau_k^{N+1}$$

So we have that:

$$\tau_k^r = (m + km')\tau_k^N + (n + kn')\tau_k^{N+1}$$

□

Lemma 10: *If $x = m\tau_k^N + n\tau_k^{N+1}$ then there exist $m', n' \in \mathbb{Z}$ such that $x = m'\tau_k^{N+1} + m'\tau_k^{N+2}$ with $\tau_k m' - n' = -\tau_k(\tau_k m - n)$*

Proof. Take $m' = n + km$ and $n' = m$. Then:

$$\begin{aligned} m'\tau_k^{N+1} + n'\tau_k^{N+2} &= (n + km)\tau_k^{N+1} + m\tau_k^{N+2} \\ &= n\tau_k^{N+1} + m\tau_k^N\left(k\tau_k + \tau_k^2\right) \\ &= n\tau_k^{N+1} + m\tau_k^N \\ &= x \end{aligned}$$

We also have that:

$$\begin{aligned} \tau_k m' - n' &= \tau_k(km + n) - m \\ &= \tau_k(km + n) - (\tau_k^2 + k\tau_k)m && (\tau_k^2 + k\tau_k = 1) \\ &= \tau_k km + \tau_k n - \tau_k^2 m - \tau_k km \\ &= \tau_k n - \tau_k^2 m \\ &= -\tau_k(\tau_k m - n) \end{aligned}$$

□

Lemma 11: *If $x \in \mathbb{Z}[\tau_k]$ and $x \geq 0$ then there exists $m, n, N \in \mathbb{Z}$ with $m, n \geq 0$ such that $x = m\tau_k^N + n\tau_k^N$*

Proof. The result holds trivially for $x = 0$, so we will assume for the rest of the proof that $x > 0$.

By Lemma 9 we know that there exists some $m, n, N \in \mathbb{Z}$ with m and n not necessarily positive, such that $x = m\tau_k^N + n\tau_k^{N+1}$.

By repeated use of Lemma 10 we can then find for any $l \in \mathbb{Z}_{>0}$ another pair $m', n' \in \mathbb{Z}$ such that $x = m'\tau_k^{N+l} + n'\tau_k^{N+l+1}$ with $|\tau_k m' - n'| = \tau_k^l|\tau_k m - n|$. Now chose l such that $N + l \geq 2$ and $\tau_k^l|\tau_k m - n| < \frac{x}{2}$ (recall that $0 < \tau_k < 1$). Then we have $m', n' \in \mathbb{Z}$ with $x = m'\tau_k^{N+l} + n'\tau_k^{N+l+1}$ and $|\tau_k m' - n'| < \frac{x}{2}$.

Now we show that n' must be positive:

$$
\begin{aligned}
& m'\tau_k^{N+l} + n'\tau_k^{N+l+1} = x \\
\implies & \Big(n' + \frac{x}{2}\Big)\tau_k^{N+l-1} + n'\tau_k^{N+l+1} > x && \big(|\tau_k m' - n'| < \tfrac{x}{2} \implies \tau_k m' < n' + \tfrac{x}{2}\big) \\
\implies & n'\big(\tau_k^{N+l-1} + \tau_k^{N+l+1}\big) > x - \frac{x}{2}\tau_k^{N+l-1} \\
\implies & n'\big(\tau_k^{N+l-1} + \tau_k^{N+l+1}\big) > \frac{x}{2} && \big(x - \tfrac{x}{2}\tau_k^{N+l-1} \geq x - \tfrac{x}{2} \text{ since } N + l \geq 1\big) \\
\implies & n' > \frac{x}{2} && \big(\big(\tau_k^{N+l-1} + \tau_k^{N+l+1}\big) < (\tau_k + k\tau_k^2) = 1\big)
\end{aligned}
$$

But since $(\tau_k m' - n') > -\frac{x}{2}$ we have that:

$$\tau_k m' > n' - \frac{x}{2} > 0$$

and so $m' > 0$ as well. □

Lemma 12: *If $S \subseteq \mathbb{Z}[\tau_k] \cap [0, 1]$ is finite then there exists a k-subdivision whose breakpoints are a superset of S.*

Proof. Let $S = \{s_1, \ldots, s_n\} \subseteq \mathbb{Z}[\tau_k] \cap [0, 1]$ where $0 \leq s_1$, $s_i < s_{i+1}$ for all $0 < i < n$ and $s_n \leq 1$. We will add $s_0 = 0$ and $s_{n+1} = 1$ to this set to simplify the proof, since it suffices to show a improper k-subdivision is a superset of a superset of S.

Then for each i, $s_{i+1} - s_i \in \mathbb{Z}[\tau_k] \cap [0, 1]$, and so we can write $s_{i+1} - s_i = m_i\tau_k^{N_i} + n_i\tau_k^{N_i+1}$ where $m_i, n_i, N_i \in \mathbb{Z}_{\geq 0}$ by Lemma 11.

We can therefore obtain a improper k-subdivision by partitioning each interval $[s_i, s_{i+1}]$ into m_i intervals of size $\tau_k^{N_1}$ and n_i intervals of size $\tau_k^{N_i+1}$.

Explicitly, such a improper k-subdivision is given by the leaf sequence:

$$P = \Bigg(\underbrace{N_0, \ldots, N_0}_{m_0\ N_0\text{'s}} \underbrace{N_0 + 1, \ldots, N_0 + 1}_{n_0\ (N_0+1)\text{'s}} \underbrace{N_1, \ldots, N_1}_{m_1\ N_1\text{'s}} \cdots\cdots \underbrace{N_n + 1, \ldots, N_n + 1}_{n_n\ (N_n+1)\text{'s}} \Bigg)$$

We note that P is a well-defined improper k-subdivision and that S is a subset of the breakpoints of P. By Lemma 8 there exists a refinement P' of P which is a k -subdivision whose breakpoints are a superset of S. $\square$

4. The Group $\mathcal{F}_{\tau_k}$

4.1 $\mathcal{F}_{\tau_k}$ as pairs of k-subdivisions

Thompson's group F has the property that any element of Thompson's group can be represented by a pair of subdivisions, and conversely that every pair of subdivisions with the same number of intervals represents an element of F. We are now in a position to be able to establish the same result for the groups $\mathcal{F}_{\tau_k}$ and k-subdivisions.

From a pair P, Q of k-subdivisions with breakpoint sets $(p_0 = 0 < p_1 < \cdots < p_n = 1)$ and $(q_0 = 0 < q_1 < \cdots < q_n = 1)$ of the same size we can construct a function $f_{P,Q} : [0,1] \to [0,1]$ by mapping the breakpoints of P to the breakpoints of Q and then interpolating linearly:

$$f_{P,Q}(x) := q_i + \frac{q_{i+1} - q_i}{p_{i+1} - p_i}(x - p_i) \qquad x \in [p_i, p_{i+1}]$$

Studying the graph of such a function $f_{P,Q}$ is the best way to understand this construction. An example of a pair of k-subdivisions and its corresponding function is given in Figure 4.1.

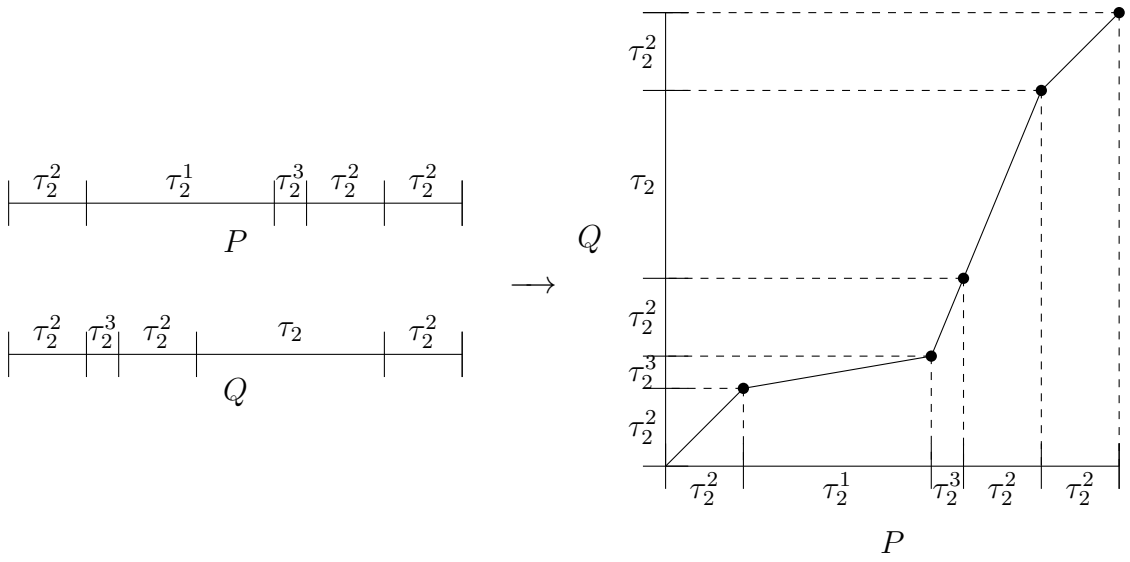


Figure 4.1: A visual depiction of the map $F : P_3 \to \mathcal{F}_{\tau_3}$

Lemma 13: *If P and Q are k-subdivisions with the same number of breakpoints, then $f_{P,Q} \in \mathcal{F}_{\tau_k}$*

Proof. We need to check each of the conditions on elements of the group $\mathcal{F}_{\tau_k}$:

1. $\boldsymbol{f_{P,Q}}$ **is piece-wise linear:** this follows directly from the definition
2. $\boldsymbol{f_{P,Q}}$ **is a homeomorphism:** as a map from a compact space to a Hausdorff space, it suffices to show that f is continuous and bijective. Continuity follows from the Pasting Lemma since the finitely many closed sets $[p_i, p_{i+1}]$ cover $[0,1]$ and the restriction of $f_{P,Q}$ to each closed set is affine, and therefore continuous. Injectivity follows from the fact that f is strictly increasing, and surjectivity follows from the Intermediate Value Theorem and the fact that 0 and 1 are in the image of f
3. **The breakpoints of** $\boldsymbol{f_{P,Q}}$ **lie in** $\mathbb{Z}[\boldsymbol{\tau_k}]$**:** because the breakpoints of $f_{P,Q}$ are the breakpoints of P and the breakpoints of a k-subdivision lie in $\mathbb{Z}[\tau_k]$.
4. **The slopes of the affine segments of** $\boldsymbol{f_{P,Q}}$ **are powers of** $\boldsymbol{\tau_k}$**:** Each k-subdivision is in particular a improper k-subdivision, each $(p_{i+1} - p_i)$ and each $(q_{i+1} - q_i)$ is a power of τ_k. The slope of each affine segment given by $\frac{q_{i+1}-q_i}{p_{i+1}-p_i}$ is therefore also a power of τ_k.

□

Let P_k denote the set of pairs of k-subdivisions with the same number of breakpoints. By Lemma 13 we can define a map $F : P_k \to \mathcal{F}_{\tau_k}$ given by $(P, Q) \mapsto f_{P,Q}$.

Lemma 14: *$F : P_k \to \mathcal{F}_{\tau_k}$ is surjective.*

Proof. We need to show that given any element $f \in \mathcal{F}_{\tau_k}$ we can construct a $(P, Q) \in P_k$ such that $f_{P,Q} = f$. We can do so by the following argument:

1. Let $S = (s_0 = 0 < s_1 < \cdots < s_l = 1) \subseteq \mathbb{Z}[\tau_k]$ be the set of breakpoint of f.
2. By Lemma 12 there exists a k-subdivision, R, whose set of breakpoints contains S as a subset.
3. Let Q be the subdivision of $[0,1]$ given by the image of R under the map f. Note that because each interval of R lies between consecutive breakpoints of f the length of each interval $[r_i, r_{i+1}]$ in R is multiplied by some power of τ_k to obtain the length of the interval $[f(r_i), f(r_{i+1})]$ in Q. It follows that Q is also an improper k-subdivision since its intervals are powers of τ_k, though it is not necessarily a k-subdivision.
4. Note that if we perform a k-partition on an interval $[r_i, r_{i+1}]$ in R to obtain:

$$\{[\, r_i \,,\, r'_1 \,], [\, r'_1 \,,\, r'_2 \,], \ldots, [\, r_k - 1' \,,\, r'_k \,], [\, r'_k \,,\, r_{i+1} \,]\}$$

   then the partition of the interval $[\, f(r_i) \,,\, f(r_{i+1}) \,]$ in Q given by:

$$\{[\, f(r_i) \,,\, f(r'_1) \,], [\, f(r'_1) \,,\, f(r'_2) \,], \ldots, [\, f(r_k - 1') \,,\, f(r'_k) \,], [\, f(r'_k) \,,\, f(r_{i+1}) \,]\}$$

   is also a k-partition of the same type as that performed on $[\, r_i \,,\, r_{i+1} \,]$ because f is affine on the interval $[r_i, r_{i+1}]$. This means that the map f respects k-partitions of R. More formally, if P is a improper k-subdivision containing the breakpoints of f, and P' is a refinement of P, then $f(P')$ is a refinement of $f(P)$. Similarly, if Q is a refinement of $f(P)$ then $f^{-1}(Q)$ is a refinement of P.
5. By Lemma 8 there exists a refinement Q' of Q that is a k-subdivision. Then $R' = f^{-1}(Q')$ is a refinement of the k-subdivision R, and thus R' is also a k-subdivision.

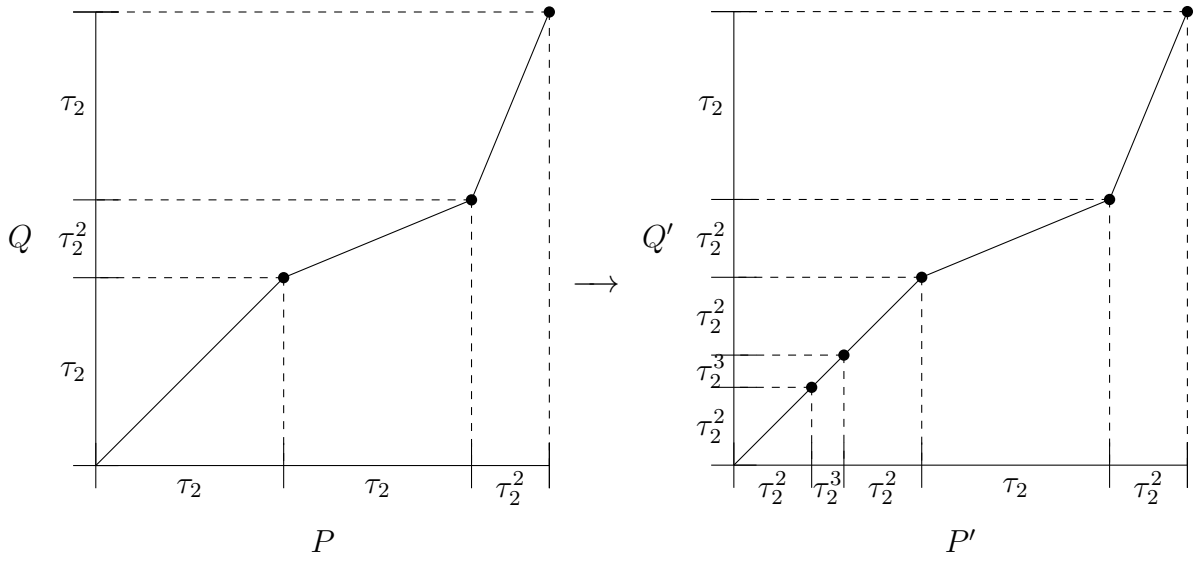


Figure 4.2: Performing the same k-partition on corresponding intervals leaves the element $f_{P,Q}$ unchanged.

$F(Q', R')$ is then an element of $\mathcal{F}_{\tau_k}$ that agrees with f on its set of breakpoints Q' and Q' contains the breakpoints of f. We conclude that $F(Q', R') = f$. □

Note that the map F is not injective: for any two k-subdivisions P and Q, $F(P, P) = F(Q, Q) = \mathrm{id}_{[0,1]}$. More generally, for $(P, Q) \in P_k$ if we perform a k-partition on an interval in P to obtain P' and a k-partition of the same type on the corresponding interval in Q to obtain Q', then $F(P', Q') = F(P, Q)$. This is illustrated in Figure 4.2.

4.2 $\mathcal{F}_{\tau_k}$ as pairs of k-trees

We can now construct a surjection from pairs of k-trees with the same number of leaf nodes to $\mathcal{F}_{\tau_k}$ via the surjective map that takes a k-tree to the k-subdivision that it represents. In this way we can also think of a pair of k-trees as representing an element of $\mathcal{F}_{\tau_k}$. This representation will prove the most useful in constructing a presentation for this group and establishing a normal form.

Definition ($\mathcal{T}_k$)**:** We define $\mathcal{T}_k$ to be the set of pairs of k-trees (T_1, T_2) such that T_1 and T_2 have the same number of leaf nodes.

Let $H : \mathcal{T}_k \to \mathcal{F}_{\tau_k}$ be the map that takes a pair of k-trees in $\mathcal{T}_k$ to the element in $\mathcal{F}_{\tau_k}$ that it represents. Explicitly, this map is given by:

$$(T_1, T_2) \mapsto F(P_1, P_2)$$

where P_1 and P_2 are the k-subdivisions represented by T_1 and T_2 respectively.

The value of our tree-pair representation of $\mathcal{F}_{\tau_k}$ will come from how it reflects composition of elements in $\mathcal{F}_{\tau_k}$. We will begin by defining a procedure by which, given two elements $x, y \in \mathcal{T}_k$, a third element $z \in \mathcal{T}_k$ can be constructed satisfying $H(z) = H(x)H(y)$.

4.2.1 Tree products

For two elements $(T_1, T_2), (S_1, S_2) \in \mathcal{T}_k$, we construct a *product*, $(T_1, T_2) \star (S_1, S_2)$ in the following way:

1. Chose two k-trees, T_2' and S_1', such that T_2' is a refinement of T_2, S_1' is a refinement of S_1, and T_2' and S_1' represent the same k-subdivision. This is possible by Lemma 1.
2. Perform the corresponding subdivision used to obtain T_2' from T_2 to obtain a k-tree T_1' from T_1. Precisely what this mean is that each time the i^{th} leaf node of T_2 is replaced with a node of type l in the process of obtaining T_2', replace the i^{th} leaf node of T_1 with a node also of type l. Note that by constructing T_1' in this way, $H(T_1', T_2') = H(T_1, T_2)$. Similarly construct an S_2' from S_2 such that $H(S_1, S_2) = H(S_1', S_2')$ (see Figure 4.2).
3. We define $(T_1, T_2) \star (S_1, S_2)$ to be (T_1', S_2').

An example of the construction of a product in $\mathcal{T}_2$ is given in Figure 4.3. Note that as a special case, if $T_2 \sim S_1$, then a tree product $(T_1, T_2) \star (S_1, S_2)$ is given by (T_1, S_2).

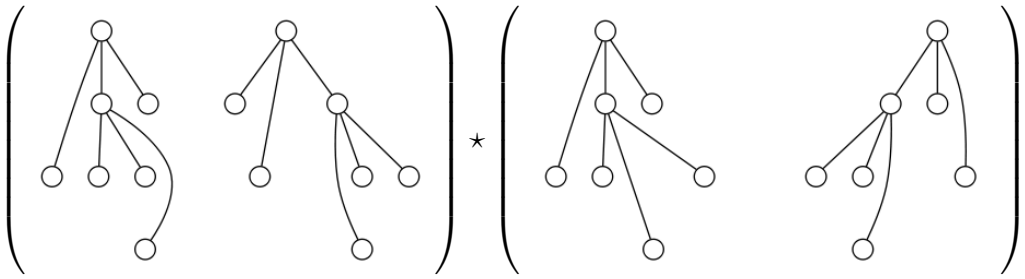
At this stage the product of two k-trees is not a well-defined k-tree. This is because we did not specify a procedure for choosing the trees T_2' and S_1' that represent the same k-subdivision. Indeed, for any such choice, we could add a node of type 0 to the first leaf node of T_2' and S_1' and obtain another pair of k-trees T_2'' and S_1'' that represent the same subdivision, with T_2'' a refinement of T_2 and S_1'' a refinement of S. We will shortly define an equivalence relation on $\mathcal{T}_k$ with respect to which any two choices of product are equivalent. For now, we will let $x \star y$ denote the set of all products of $x, y \in \mathcal{T}_k$.

Lemma 15: *If* $(X, Y) \in (T_1, T_2) \star (S_1, S_2)$ *then* $H(X, Y) = H(T_1, T_2)H(S_1, S_2)$

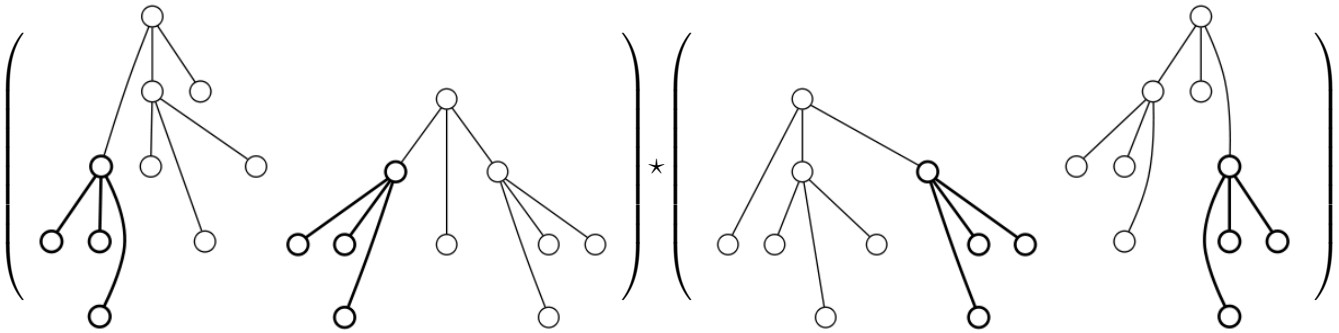
Proof. By hypothesis, there exists tree refinements T_2' of T_2 and S_1' of S_1 with $T_2' \sim S_1$ such that $H(X, T_2') = H(T_1, T_2)$ and $H(S_1', Y) = H(S_1, S_2)$. It therefore suffices to show that $H(X, Y) = H(X, T_2')H(S_1', Y)$.

Let P_1, P_2 and P_3 denote the sets of breakpoints of the subdivisions induced by X, $T_2' \sim S_1'$ and Y respectively. Then $H(X, T_2')$ is the linear interpolation of the map $P_1 \to P_2$ and $H(S_1', Y)$ is the linear interpolation of the map $P_2 \to P_3$. It follows that $H(X, T_2')H(S_1', Y)$ is given by the linear interpolation of the map $P_1 \to P_3$ which is $H(X, Y)$ (recall that, in $\mathcal{F}_{\tau_k}$, $fg = g \circ f$). □

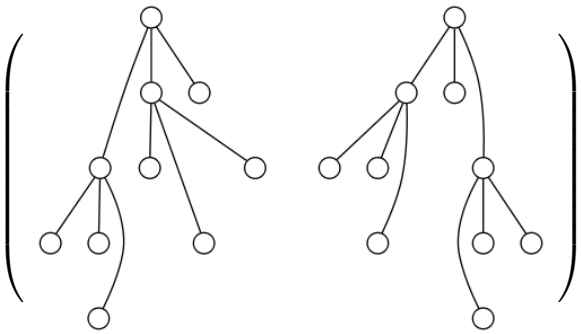
It follows that for any two tree pairs, $x, y \in \mathcal{T}_k$, and any two choices of products, $z, z' \in x \star y$, we have $H(z) = H(z')$.



(a) $(T_1, T_2) \star (S_1, S_2)$



(b) $(T_1', T_2') \star (S_1', S_2')$. Note that the leaf sequences of T_2' and S_1' are the same.



(c) (T_1', S_2')

Figure 4.3: Computing a product of two pairs of 2-trees

4.2.2 The group $\widetilde{\mathcal{T}_k}$

Definition ($\widetilde{\mathcal{T}_k}$)**:** We define an equivalence relation $\sim$ on $\mathcal{T}_k$ (not to be confused with leaf equivalence on k-trees) by:

$$x \sim y \iff H(x) = H(y)$$

We define $\widetilde{\mathcal{T}_k}$ to be the set of these equivalence classes. We define a binary operation $\star : \widetilde{\mathcal{T}_k}^2 \to \widetilde{\mathcal{T}_k}$ as:

$$[x] \star [y] := [x \star y] = H^{-1}(H(x)H(y))$$

Because $H : \mathcal{T}_k \to \mathcal{F}_{\tau_k}$ is by construction constant on the equivalence classes of $\widetilde{\mathcal{T}_k}$ and surjective, H descends to a surjective map $\widetilde{H} : \widetilde{\mathcal{T}_k} \to \mathcal{F}_{\tau_k}$. Explicitly, if $p : \mathcal{T}_k \to \widetilde{\mathcal{T}_k}$ is the map $x \mapsto [x]$, then $\widetilde{H} \circ p = H$.

Lemma 16: *$\widetilde{\mathcal{T}_k}$ with binary operation $\star$ is a group, and $\widetilde{H} : \widetilde{\mathcal{T}_k} \to \mathcal{F}_{\tau_k}$ is a group isomorphism.*

Proof. By construction, $\widetilde{H}$ is clearly a bijection. It therefore suffices to note that for any $[x], [y] \in \widetilde{\mathcal{T}_k}$, $H([x] \star [y]) = H([x])H([y])$. This follows from the definition of $\star : \widetilde{\mathcal{T}_k}^2 \to \widetilde{\mathcal{T}_k}$ and Lemma 15. □

Lemma 17: *For any k-trees T_1, T_2 and S with the same number of leaf nodes, the following are equivalent:*

1. $T_1 \sim T_2$
2. $[T_1, T_2] = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$
3. $[S, T_1] = [S, T_2]$
4. $[T_1, S] = [S, T_2]^{-1}$
5. $[T_1, S] = [T_2, S]$

Proof. $1 \iff 2$ follows from the fact that $H(T_1, T_2) = \mathrm{id}$ if and only if the images of each breakpoints of T_1 (i.e the corresponding breakpoint of T_2) is itself, since a linear interpolation of the identity on a subset of $[\,0\,,\,1\,]$ containing 0 and 1 is the identity on $[\,0\,,\,1\,]$.

For $2 \implies 3$:

$$[S, T_1] = [S, T_1] \star \mathrm{id}_{\widetilde{\mathcal{T}_k}} = [S, T_1] \star [T_1, T_2] = [S, T_2]$$

For $3 \implies 4$:

$$[T_1, S] = [S, T_1]^{-1} \star [S, T_1] \star [T_1, S] = [S, T_1]^{-1} \star [S, S] = [S, T_1]^{-1} \star \mathrm{id}_{\widetilde{\mathcal{T}_k}} = [S, T_1]^{-1}$$

For $4 \implies 5$:

$$[T_1, S] = [S, T_2]^{-1} = [S, T_2]^{-1} \star [S, S] = [S, T_2]^{-1} \star [S, T_2] \star [T_2, S] = [T_2, S]$$

For $5 \implies 2$:

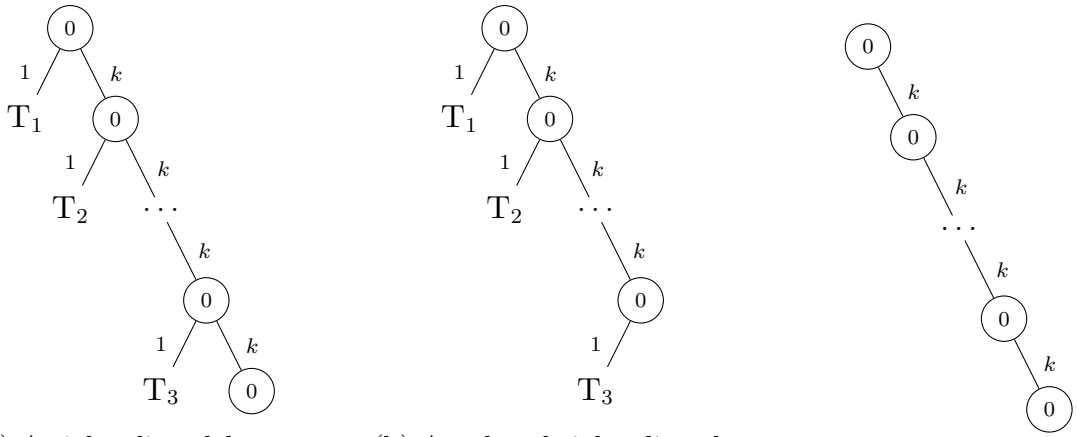
$$[T_1, T_2] = [T_1, S] \star [S, T_2] = [T_1, S] \star [S, T_2] = [T_1, S] \star [T_1, S]^{-1} = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$$

□

Definition (Right-aligned k-tree)**:** We will call a k-tree *right-aligned* if all the nodes along its right edge are of type 0. A precise definition is that a k-tree is right aligned if all the predecessors (parents, grandparent, etc.) of the right-most leaf are of type 0.

Definition (Reduced right-aligned k-tree)**:** We will say that a right-aligned k-tree is *reduced* if the type 0 node at the end of its right edge has at least one child which is not a leaf.

Definition (Spine)**:** A *spine* is a k-tree constructed by repeatedly adding nodes of type 0 to the right-most leaf. More precisely, it is a k-tree such that all nodes are of type of 0, and for each node N and each $j < k$, $N(j)$ is a leaf node.



(a) A right-aligned k-tree which is not reduced

(b) A reduced right-aligned k-tree

(c) A spine

Figure 4.4: Right-aligned k-trees. Note that T_i in the diagrams denotes a subtree

Spines are in particular right-aligned k-trees, though they are never reduced.

For a given k-tree T we will let $\mathrm{sp}(T)$ denote the spine with the same number of leaf nodes as T. This allows us to discuss a class of k-tree pairs of the form $(T, \mathrm{sp}(T))$. We will then use the notation $\widetilde{T}$ to denote the equivalence class $[T, \mathrm{sp}(T)] \in \widetilde{\mathcal{T}_k}$.

Lemma 18: *Let T_1, T_2 be two k-trees with the same number of nodes. Let S be a spine. Then:*

1. $[\mathrm{sp}(T_1), T_1] = \widetilde{T_1}^{-1}$
2. $[T_1, T_2] = \widetilde{T_1} \star \widetilde{T_2}^{-1}$
3. $\widetilde{S} = 1_{\widetilde{\mathcal{T}_k}}$
4. *If* $T_1 \sim T_2$ *then* $\widetilde{T_1} = \widetilde{T_2}$
5. *The set* $\left\{\widetilde{T} \,|\, T \text{ a } k\text{-tree}\right\}$ *generates* $\widetilde{\mathcal{T}_k}$

Proof. 1 follows directly from Lemma 17. 2 follows from 1 and the fact that $(T_1, T_2) \in (T_1, \mathrm{sp}(T_1)) \star (\mathrm{sp}(T_2), T_2)$ from our construction of tree products, since $\mathrm{sp}(T_1) = \mathrm{sp}(T_2)$. 3 follows from Lemma 17 and the fact that if S is a spine, then $\mathrm{sp}(S) = S$. 4 follows from

Lemma 17. 5 follows from 2 and the fact that every element of $\widetilde{\mathcal{T}}_k$ can be expressed as $[T', T'']$ for some pair of k-trees T' and T''. □

Lemma 19: *If T_1 and T_2 are right-aligned k-trees with T_2 reduced and $T_1 \sim T_2$ then T_1 is reduced.*

Proof. The right-most leaves of any such pair T_1, T_2 of k-trees must be of the same height h, given that T_1 and T_2 have the same leaf sequence. If T_1 were not reduced, then its lowest type 0 node along the right edge would have no children, and its leaf sequence would terminate with

$$\ldots h+1, \underbrace{h, \ldots, h}_{k\, h\text{'s}}$$

But this cannot be the end of the leaf sequence of T_2, because at least one of the children of the last type 0 node along the right edge of T_2 is not a leaf. We conclude that T_1 must be reduced. □

Lemma 20: *If T_1 and T_2 are right-aligned, reduced k-trees satisfying $\widetilde{T_1} \star \widetilde{T_2}^{-1} = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$, then $T_1 \sim T_2$.*

Proof. By Lemma 17 it suffices to show that T_1 and T_2 have the same number of nodes. Once we have established this, $\widetilde{T_1} \star \widetilde{T_2}^{-1} = [T_1, T_2]$ by Lemma 18 and then $T_1 \sim T_2$ by Lemma 17. We will that T_1 and T_2 have the same number of nodes by contradiction.

We will assume without loss of generality that T_2 has a number of nodes greater than that of T_1, and thus that the length of $\mathrm{sp}(T_2)$ is greater than that of $\mathrm{sp}(T_1)$. It follows that $\mathrm{sp}(T_2)$ is a common refinement of $\mathrm{sp}(T_1)$ and $\mathrm{sp}(T_2)$, and $\mathrm{sp}(T_2)$ is obtained from $\mathrm{sp}(T_1)$ by adding some number $d \geq 1$ of type 0 nodes to the right edge. Let T_1' be the k-tree obtained from T_1 by adding that same number of type 0 nodes to its right edge. Note that T_1' is therefore right-aligned but not reduced. It follows that $(T_1', T_2) \in (T_1, \mathrm{sp}(T_1)) \star (\mathrm{sp}(T_2), T_2)$ and so $[T_1', T_2] = \mathrm{id}_{\tau_k}$ and thus $T_1' \sim T_2$ (by Lemma 17). By Lemma 19 we conclude that T_1' is reduced, which gives us a contradiction.

It follows that T_1 and T_2 have the same number of nodes, and therefore $T_1 \sim T_2$. □

5. k-trees

5.1 k-trees and the group $\widetilde{\mathcal{T}}_k$

5.1.1 Grafting

The value of demonstrating an isomorphism between $\mathcal{F}_{\tau_k}$ and the group of classes of tree pairs $\widetilde{\mathcal{T}}_k$ is that we can now learn about $\mathcal{F}_{\tau_k}$ by investigating combinatorial properties of k-trees. We begin by giving another definition for leaf equivalence of trees, which we will eventually translate into a class of relations on $\mathcal{F}_{\tau_k}$.

First, we define a transformation of k-trees that preserves leaf sequences. We will call this transformation *grafting* (to the left or right), or a graft operation. Below we will describe right grafting at a node; left grafting is the inverse operation.

5.2 Grafting and leaf equivalence

Lemma 21: *Let T be a k-tree, and N a non-leaf node with height n and type l which represents the interval $[\,a\ ,\ b\,]$. The interval represented by $N(i)$ is given by:*

$$\begin{cases} \left[\, a + (i-1)\tau_k^{n+1} + \tau_k^{n+2}\ ,\ b - (k-i)\tau_k^{n+1} \,\right] & l < i \\ \left[\, a + (i)\tau_k^{n+1}\ ,\ b - (k-i)\tau_k^{n+1} \,\right] & l = i \\ \left[\, a + i\tau_k^{n+1}\ ,\ b - (k-1-i)\tau_k^{n+1} - \tau_k^{n+2} \,\right] & l > i \end{cases}$$

Proof. This is a matter of counting the children of N on either side of $N(i)$. There will always be i children to the left of $N(i)$ and $k-i$ children to the right. The length of a short child interval will be τ_k^{n+1} and the length of a long child interval will be τ_k^{n+2}. The lower bound of the interval represented by $N(i)$ will be a plus the length of all the children of N to the left of $N(i)$. Similarly, the upper bound of the interval for $N(i)$ is b minus the lengths of all children of N to the right of $N(i)$. We have three cases to consider:

1. If $l < i$, then the long child is to the left of $N(i)$ so the lower bound of the interval is $a + (i-1)\tau_k^{n+1} + \tau_k^{n+2}$ and the upper bound is $b - (k-i)\tau_k^{n+1}$
2. If $l = i$, then $N(i)$ is the long node, so all children of N to the left and right of $N(i)$ have length τ_k^{n+1} and the interval for $N(i)$ is therefore $\left[\, a + i\tau_k^{n+1}\ ,\ b - (k-i)\tau_k^{n+1} \,\right]$
3. If $l > i$, then the long child of N is on the right of $N(i)$ whose interval is therefore given by: $\left[\, a + i\tau_k^{n+1}\ ,\ b - (k-i-1)\tau_k^{n+1} - \tau_k^{n+2} \,\right]$

□

Lemma 22: *Let T_1 and T_2 be two k-trees with root nodes of type m and n respectively, where $m < n$. If the algorithm* increaseType *with the root node of T_1 as input adds a node to T_1, then $T_1 \not\sim T_2$.*

Proof. Let N denote the type m root node of T_1.

Looking at the algorithms increaseType and decreaseType we can determine the conditions on the tree T_1 that results in increaseType(N) adding a node. Clearly, if $C_1 = N(m+1)$ is a leaf, then a node will be added. If C_1 is a node of type k and $C_2 = C_1(k-1)$ is a leaf node then a node will be added. We can formally describe the emerging pattern. Let $\mathrm{I}(N)$ and $\mathrm{D}(N)$ be the predicates for the condition that increaseType(N) and decreaseType(N) respectively add a node to the tree. Let isLeaf(N) be the condition that N is a leaf, and let Type(N) denote the type of N. Then we can define $\mathrm{I}(N)$ and $\mathrm{D}(N)$ recursively as:

$$\begin{aligned}\mathrm{D}(N) =&(\textsc{isLeaf}(N(m-1))) \text{ or } (\textsc{Type}(N(m-1)) = 0 \text{ and } \mathrm{I}(N(m-1)))\\ \mathrm{I}(N) =&(\textsc{isLeaf}(N(m+1))) \text{ or } (\textsc{Type}(N(m+1)) = k \text{ and } \mathrm{D}(N(m+1)))\end{aligned}$$

From this definition we see that $\mathrm{I}(N)$ is equivalent to the property that we have a chain of nodes $C_1, \ldots C_h$ in T_1 for some $h \in \mathbb{Z}_{\geq 1}$ with $C_1 = N(m+1)$ of type k (where $\neq 1$), C_h a leaf node. For $1 < i < h$ odd $C_i = C_{i-1}(1)$ is of type k and for i even, $C_i = C_{i-1}(k-1)$ is of type 0. A diagram for such a chain where $h = 4$ is shown in Figure 5.1.

Claim: *C_h represents an interval containing $(m+1)\tau_k$.*

Proof. We will make repeated use of Lemma 21.

The interval represented by $C_1 = N(m+1)$ is $[m\tau_k + \tau_k^2, 1 - (k-m-1)\tau_k]$.

For $i > 1$, every time we descend to a node C_i from C_{i-1} with i odd (so $C_i = C_{i-1}(1)$ where C_{i-1} has type 0) we increase the lower bound of the interval by τ_k^{i+2} and we decrease the upper bound by $(k-1)\tau_k^{i+1}$. Every time we descend to a C_i with i even (so $C_i = C_{i-1}(k-1)$ where C_{i-1} has type k) we increase the lower bound by $(k-1)\tau_k^{i+1}$ and we decrease the upper bound by τ_k^{i+2}.

So the interval represented by C_i is given by:

$$\begin{cases}\left[\, m\tau_k + \sum_{i=1}^{p} k\tau_k^{2i} \;,\; 1 - (k-m)\tau_k - \tau_k \sum_{i=1}^{p-1} k\tau_k^{2i} \,\right] & i = 2p\\ \left[\, m\tau_k + \sum_{i=1}^{p} \tau_k^{2i} + \tau_k^{2i+1} \;,\; 1 - (k-m)\tau_k - \tau_k \sum_{i=1}^{p} k\tau_k^{2i} \,\right] & i = 2p+1\end{cases}$$

We note that all the C_i intervals contain the value $m\tau_k + \sum_{i=1}^{\infty} k\tau_k^{2i} = 1 - (k-m)\tau_k - \tau_k \sum_{i=1}^{\infty} k\tau_k^{2i}$ in their interior by the fact that the sizes of the intervals for each C_i decrease to zero as $i \to \infty$.

Claim: $\sum_{i=1}^{\infty} k\tau_k^{2i} = \tau_k$

Proof.

$$\sum_{i=1}^{\infty} k\tau_k^{2i} = \sum_{i=1}^{\infty} k\tau_k^{2i-1}\tau_k = \sum_{i=1}^{\infty} \tau_k^{2i-1}(1-\tau_k^2) = \sum_{i=1}^{\infty} \left(\tau_k^{2i-1} - \tau_k^{2i+1}\right) = \tau_k$$

□

We conclude that $(m+1)\tau_k = m\tau_k + \sum_{i=1}^{\infty} k\tau_k^{2i}$ is in the interior of all C_i intervals, and in particular in C_h. □

Because C_h is a leaf node, $(m+1)\tau_k$ is not a breakpoint of the subdivision represented by the k-tree T_1.

Let R be the root node of T_2. Then the interval represented by $R(m)$ is:

$$\left[\, m\tau_k \,,\; 1-(k-1-m)\tau_k - \tau_k^2 \,\right] = \left[\, m\tau_k \,,\; (m+1)\tau_k \,\right]$$

So $(m+1)\tau_k$ is a breakpoint of T_2 but not of T_1, and therefore T_1 is not leaf equivalent to T_2 whenever $\mathrm{I}(N)$ holds. □

Lemma 23: *If $T_1 \sim T_2$, then there exists a sequence of graft operations which transforms T_1 to T_2.*

Proof. We prove this by induction on the height of the trees.

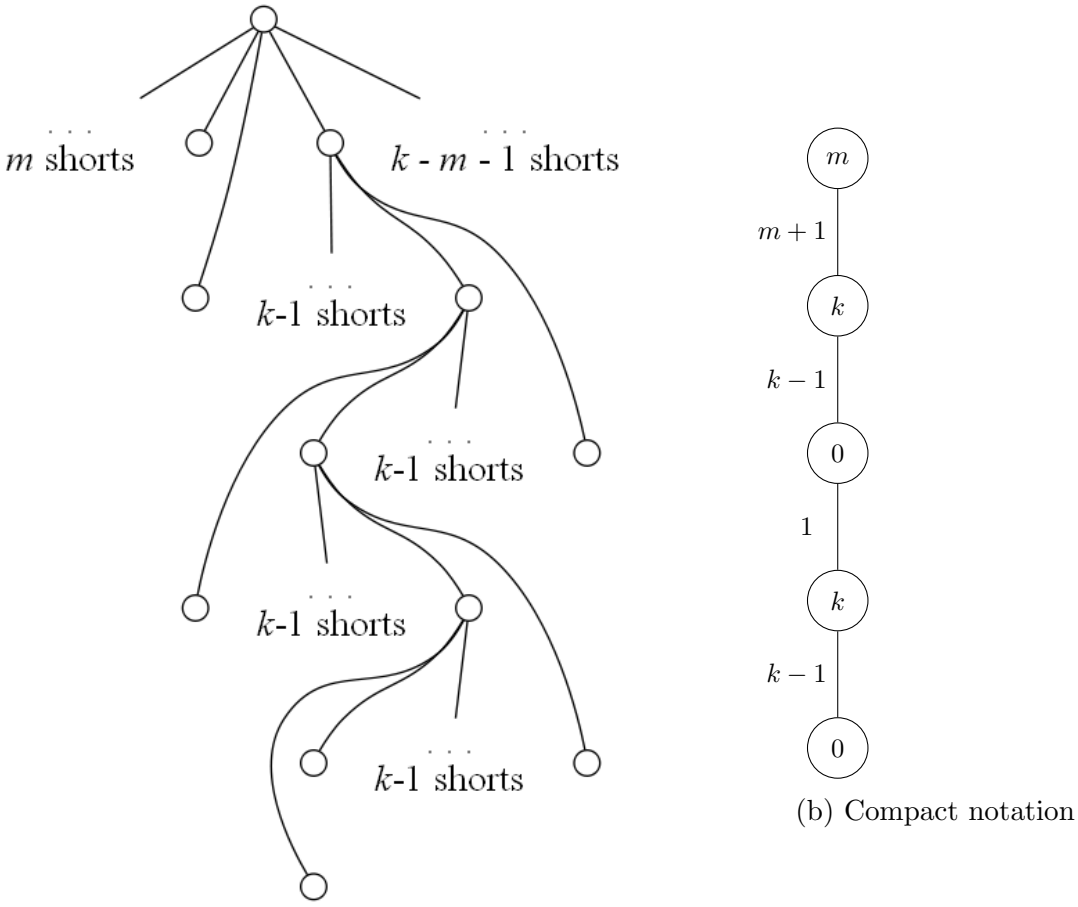
For trees of height 0 (i.e. the empty tree) the result is trivial. Now assume that the result holds for all trees of height less than or equal to n.

Assume T_1 and T_2 are k-trees of height $n+1$ with $T_1 \sim T_2$. Note that if no such trees exist (as is the case for $n = 0$) then the result is vacuously true. Let R_1 and R_2 denote the root nodes of T_1 and T_2 respectively. If the types of R_1 and R_2 are the same, then for each $i \in \{0, \ldots, k\}$ the intervals corresponding to $R_1(i)$ and $R_2(i)$ are the same, and thus the subtrees of T_1 and T_2 with root nodes $R_1(i)$ and $R_2(i)$ are leaf equivalent and have height less than or equal to n. By the induction hypothesis, there therefore exists a sequence of graft operations by which each $R_1(i)$ is transformed into $R_2(i)$, and thus T_1 can be transformed to T_2 by graft operations.

If the types of R_1 and R_2 are not the same, it suffices to show that there exists a sequence of graft operations that changes the type of R_1 to the type of R_2. Assume, without loss of generality, that R_1 is the node of lower type m, and R_2 has type $l > m$. By Lemma 22 and the fact that $T_1 \sim T_2$, there exists a sequence of graft operations by which the type of R_1 is changed to $m+1$. This process can be repeated until the types of R_1 and R_2 match.

We conclude that for any two leaf equivalent trees T_1, T_2 of any height, there exists a sequence of graft operations that transforms T_1 into T_2. □

Lemma 24: *If T_1 and T_2 are k-trees with all nodes of type less than $l+1$ and $T_1 \sim T_2$, then T_1 can be transformed into T_2 by a series of graft operations without ever increasing the type of any node above l.*



(a) Standard notation

(b) Compact notation

Figure 5.1: The form of a k-tree in which it is not possible to right-displace the long child of the root. This chain has length $h = 4$

Proof. We will prove this in the case where we allow the root nodes of T_1 and T_2 to be of any type (i.e. greater than l). Just as for Lemma 23 it suffices to show that if the root node N of T_1 has type m less than the type n of the root node of T_2, then there exists a sequence of graft operations which increases the type of T_1 by one, this time without ever increasing the type of any other node in the tree above l. The result is trivial for $l = k$, so we will henceforth only consider the case where $l < k$

The result clearly holds in the case where $C = N(m+1)$ is of type $p < l$ since we can perform a right graft operation at N to achieve the result. C cannot be a leaf node, since then no graft operation increases the type of N, which would contradict the fact that $T_1 \sim T_2$ (Lemma 22). It therefore remains only to deal with the case where C is of type l.

We will now prove our result by induction on the height of the tree T_1 (and therefore T_2). The case where T_1 has height 2 is trivial, because in this case T_1 and T_2 are both nodes of the same type. Now assume that this result holds when T_1 is a tree of height less than d for some $d > 2$, and assume that T_1 and T_2 are both trees of height d with all nodes except the root of type less than or equal to l, and $T_1 \sim T_2$. Moreover, assume that $N(m+1)$ is a node of type l.

We know that there exists some sequence of graft operations that increases the type of m to n by Lemma 23. Moreover, we know that this is achieved by implementing the algorithm INCREASETYPE(N) $n - m$ times. By hypothesis there are no nodes of type k ($k > l$) in T_1 (except perhaps the root), so INCREASETYPE(N) will simply perform a right graft at N each time it is called. Let N' denote the root of T_1 after its type has been increased to n. Then because $C = N(m+1)$ was a node of type l, $C' = N'(m)$ is a node of type $l+1$, and the subtree under $C'(l)$ is the same as the subtree under $C(l-1)$ (Figure 3.7 provides a useful reference for this).

Let R denote the root node of T_2. Because $T_1 \sim T_2$ and TYPE(N') = TYPE(R), it follows that the subtrees S_1 and S_2 under $C' = N'(m)$ and $R(m)$ respectively are also leaf equivalent. S_1 and S_2 are k-trees of height less than d, so by the induction hypothesis we can use a sequence of graft operations to transform S_1 into S_2 without ever increasing the type of a node to $l+1$ below the root C' of S_1. Because the root node of S_2 is of type less than $l+1$, this grafting process must at some point include a left graft at C', which means either that $C'(l)$ is of type greater than 0, or that $C'(l)$ is of type 0 but there exists a sequence of graft operations which increases the type of $C'(l)$ without creating any nodes of type greater than l.

Because the trees under $C(l-1)$ and $C'(l)$ are the same, it follows that either $C(l-1)$ is a node of type greater than 0, or $C(l-1)$ is of type 0 there exists a sequence of graft operations by which the type of $C(l-1)$ can be increased. This means that in the original tree T_1 we can:

1. First perform graft operations, if necessary, to change the type of $C(l-1)$ to be greater than zero, without creating any nodes of type $l+1$.
2. Perform a left graft operation at $C = N(m+1)$, changing its type to $l-1$, again without increasing the type of any node to $l+1$
3. Perform a right graft operation at N, which now does not create a node of type $l+1$

because $C = N(m+1)$ is now of type $l-1$.

These steps increment the type of N without creating any nodes of type greater than l in the process.

It follows by induction that the type of N can be incremented by one without creating any nodes of type greater than l for T_1 and T_2 of any height, which gives us our result. □

5.3 Normal k-trees

We will define a property on k-trees which will prove useful in our later construction of the normal form on $\mathcal{F}_{\tau_k}$.

Definition (normal k-tree)**:** We will say a k-tree is *normal* if all of its nodes are of type 0 or 1, and if a node N is of type 1 then $N(0)$ is a leaf node. We will say a k-tree is semi-normal if it is nice every where except perhaps at its root.

Lemma 25: *If T is a semi-normal k-tree with root node of type greater than 0, then by adding nodes of type 0 and performing graft operations we can obtain a tree T' which is also semi-normal and whose root node has type less than that of the root node of T.*

Proof. We prove this by induction on the height h of the tree T, noting that the case where $h = 0$ is trivial.

Now assume that the result is true for $h < d$ for some $d \in \mathbb{Z}_{\geq 1}$, and that T is a semi-normal k-tree of height d.

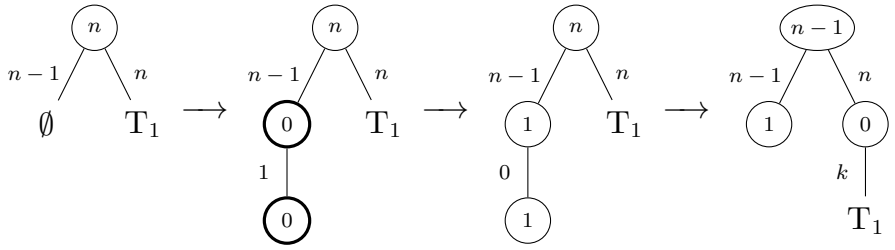
The method by which we decrease the type n of the root node N of T depends on the subtree under $C = N(n-1)$. We will consider five exhaustive cases:

1. C is a leaf node
2. C is a node of type 1
3. C is a node of type 0 and:
   (a) $C(1)$ is a leaf
   (b) $C(1)$ is a node of type 0
   (c) $C(1)$ is a node of type 1

Case 1: C is a leaf

If C is a leaf node then we can achieve the result by the adding two type 0 nodes and

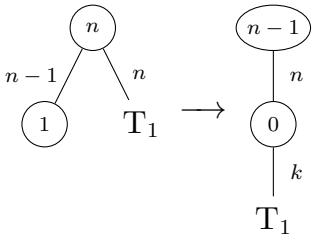
performing a left and right graft operation.



T_1 denotes an arbitrary subtree under $N(n)$. Note that the resulting tree is semi-normal because T_1 is normal by the hypothesis that T is semi-normal.

Case 2: Type$(C) = 1$

By the fact that T is semi-normal, $C(0)$ must be a leaf. We omit the other children of the type 1 node C as they are not affected by the graft operation:



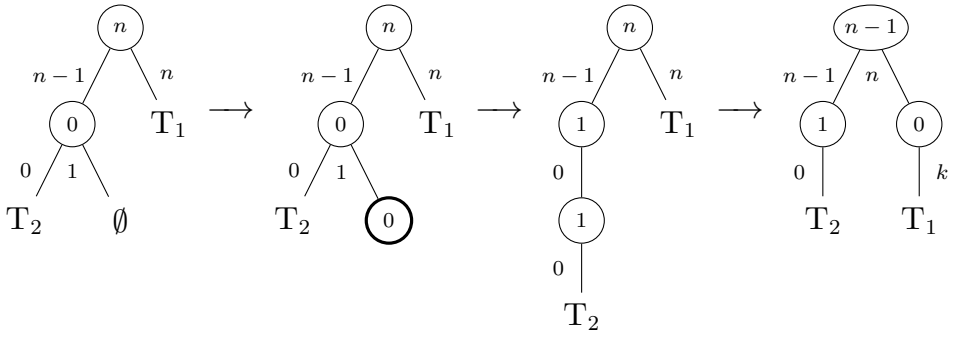
The result is a semi-normal tree because T_1 is normal.

Case 3: Type$C = 0$

Let $D := C(1)$

Case 3a: D is a leaf

We perform the following manipulation:

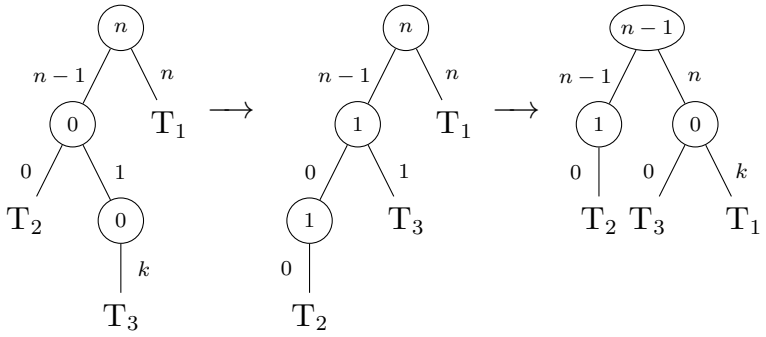


After this manipulation, the subtree S under the type one node $N(n-1)$ is not necessarily normal, and so our result is not necessarily semi-normal. S is, however, semi-normal because T_2 is normal and it has height strictly less than T. By the induction hypothesis we can then add type zero nodes perform graft operations to decrease the type of the root node to

zero and remain semi-normal. But a semi-normal tree with root node of type 0 is necessarily normal, so we have our result.

Case 3b: Type$(D) = 0$

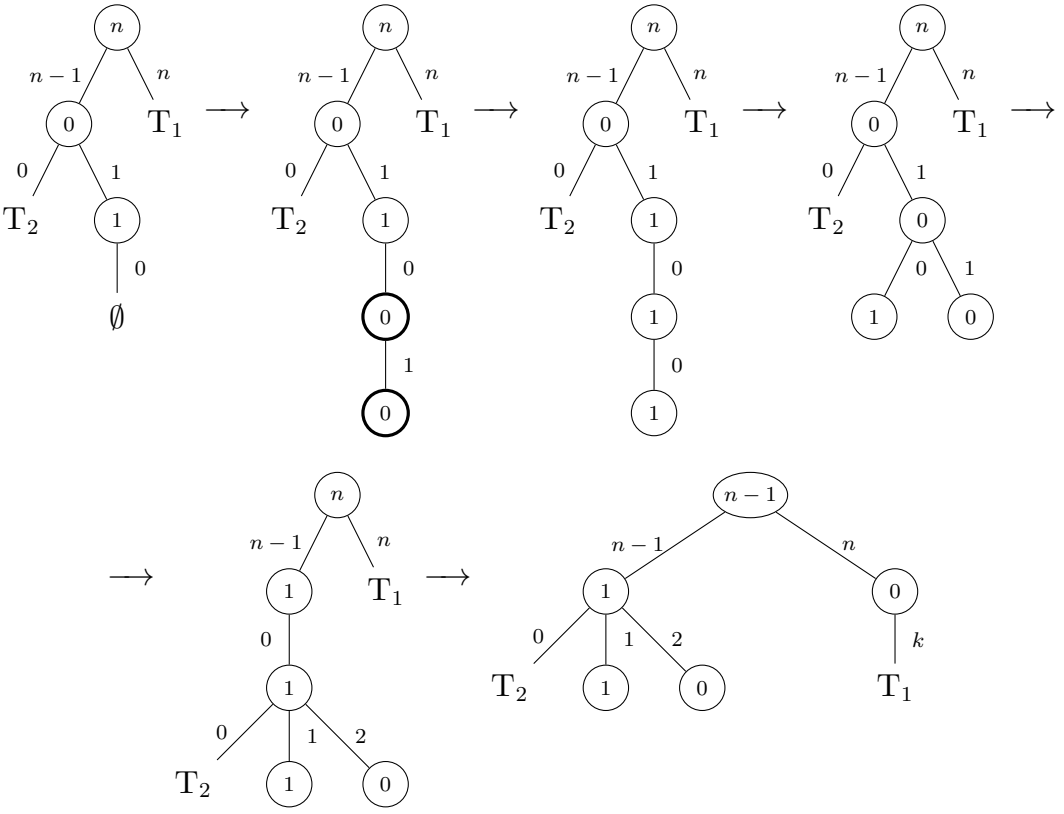
We perform the following manipulation:



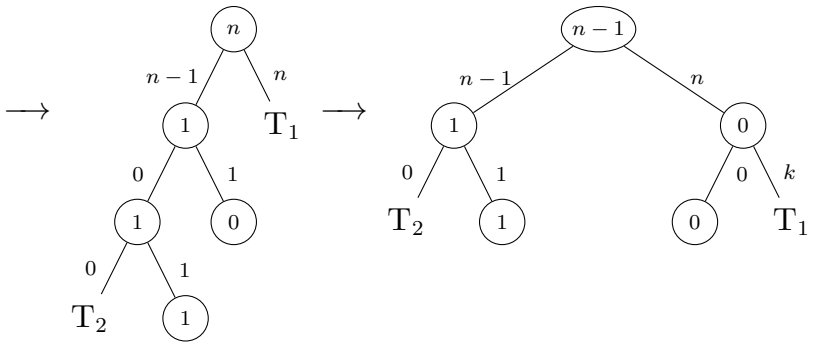
This output is similar to that for Case 3a, and by the same argument can be converted into a semi-normal tree. Note that we are not making any assumptions about the other children of D, though we have not shown them in the above manipulation as they are not relevant.

Case 3c: Type$(D) = 1$

We note again that by the assumption that T is semi-normal, the left-most child of D is a leaf.



(⋆)



(†)

Here we have two different cases depending on whether $k > 1$. The case where $k > 1$ gives $(\star)$ and the case where $k = 1$ is given by (†). Note that in both cases, by the same argument as was given for cases 3a and 3b, the resulting tree is not necessarily semi-normal, but can be converted to a semi-normal tree by the induction hypothesis. □

Corollary 2: *Every semi-normal tree can be converted to a normal tree by graft operations and the addition of nodes of type* 0*.*

Proof. Let T be a semi-normal tree. By Lemma 25 we can continue to decrease the type of the root, with the T remaining semi-normal throughout the process, and any semi-normal tree with root node of type 0 is normal. □

Lemma 26: *Every k-tree can be converted to a normal k-tree by graft operations and the addition of nodes of type* 0*.*

Proof. We prove this by induction on the height of the tree T. Every tree of height 2 is semi-normal, because it consists only of a root node. By 2 it follows that it can be converted to a normal tree. Now if the result holds for all k-trees of height $h < n$ for some $n \in \mathbb{Z}_{\geq 3}$ and T is of height h, then all of its proper subtrees are of height less than h. It follows that they can all be converted into normal trees, by graft operations and the addition of nodes of type 0, and thus T can be converted into a semi-normal tree. Then from 2, T can also be converted into a normal tree. □

5.4 Generating sets

We now return our attention to the group $\widetilde{\mathcal{T}}_k$, where we can use the results above to find representatives of a certain form in $\mathcal{T}_k$ for elements in $\widetilde{\mathcal{T}}_k$.

Lemma 27: *If $f \in \widetilde{\mathcal{T}}_k$, then there exists two right-aligned trees T_1, T_2 such that $f = [T_1, T_2]$.*

Proof. Let $(T_1, T_2) \in \mathcal{T}_k$ be a representative for f. We note that:

1. If we add a node of any type to a leaf node in T_1 to obtain T_1' and a node of the same type to the corresponding leaf node in T_2 to obtain T_2', then $H(T_1', T_2') = H(T_1, T_2)$, and so $[T_1', T_2'] = [T_1, T_2]$.
2. If $T_1 \sim T_1'$ then $[T_1, T_2] = [T_1', T_2]$ (Lemma 17).

Using these facts, it suffices to show that we can transform a k-tree T into a right-aligned k-tree by adding nodes to any but the right-most leaf of T and performing graft operations on T. If this is possible, we can perform this procedure on T_1 to obtain a right-aligned tree T_1'. Whenever we add a new node of type i to an leaf node in T_1 to obtain T_1' we will also add a node of type i to the corresponding interval in T_2, to obtain a new k-tree T_2'. By 1 above, we will then have $[T_1, T_2] = [T_1', T_2']$. Then we can perform the same process on T_2' to obtain a right-aligned k-tree T_2'' and adding corresponding nodes to T_1' gives us T_1'' such that $[T_1'', T_2''] = [T_1, T_2]$. As long as our right-aligning process doesn't involve adding a node to the right-most leaf of a tree, T_1'' will again be right-aligned, and thus we have our result.

Our right aligning algorithm is given in pseudo-code in Algorithm 1.

Algorithm 1 An algorithm which right-aligns the subtree with root node N

```
procedure RIGHTALIGN(node N)
    if N is a leaf then
        return
    else
        while type of N ≠ 0 do
            DECREASETYPE(N)
        end while
        RIGHTALIGN(N(k))
    end if
end procedure
```

If the algorithm terminates then its output is a right-aligned tree, the type of every node it has reached along the right-edge will be of type zero, and it will have reached every node along the right-edge. To check that this algorithm satisfies the condition required in the argument above, we also need to check that it always terminates, and that it never adds any new nodes to the right-most leaf of the tree.

We can see that a node is never added to the right-most leaf by looking closely at the DECREASETYPE algorithm (Alg. 3.1). In this algorithm, to decrease the type of a node N of type n, we only ever add nodes to the subtree under the node $N(n-1)$. If N is a node on the right edge of the tree, then the right-most leaf node of the tree will not be in the subtree under $N(n-1)$ for any n (it will be in the subtree under $N(k)$).

To ensure that the algorithm terminates, we need to show that the algorithm doesn't cause the right-edge to grow as fast as it can traverse down the right edge. Initially, the fact that the algorithm does not add any nodes to the right edge may appear to ensure this, but there is one other way in which this algorithm can extend the right edge of a tree. The only time that the right-most child $C = N(k)$ is affected when decreasing the type of N is when N is of type k. In this case, when a left graft is performed at N, C and the subtree under it are detached from N and attached in position k to $N(k-1)$, which becomes the k^{th} child of N. In the case where N is a node on the right edge, this means that $N(k-1)$ is inserted into the right edge of the tree, between N and $N(k)$. Figure 5.2 gives a visual depiction of this phenomenon. It is important to note that this only happens when the algorithm decreases

the type of a node of type k on the right edge. It is also important that the node that is inserted into the right edge cannot be of type k, since its type is also decreased in the left graft operation (see Figure 5.2).

The algorithm will never-the-less terminate eventually. As we have already noted, graft operations do not change the height of nodes, and no nodes are added to the right-most leaf by the algorithm, so the height of the right-most node remains unchanged. This imposes a limit on how many nodes can be in the right edge; there cannot be more nodes than the height of the right-most leaf. It follows that the algorithm will eventually reach the end of the right edge and terminate. □

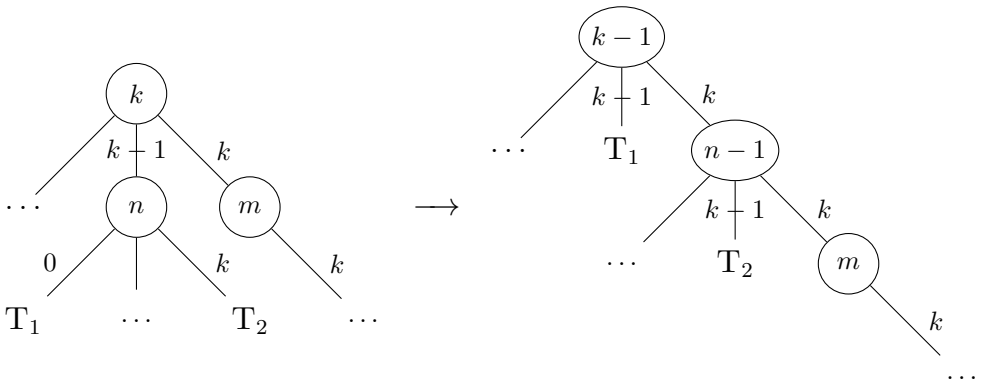


Figure 5.2: An example of a left graft operation at a type k node on the right edge which increases the length of the right edge of the tree.

An example of the construction an equivalent right-aligned tree pair from one which is not right-aligned in the case where $k = 2$ is given in Figure 5.3.

Lemma 28: *If $f \in \widetilde{\mathcal{T}}_k$, then there exists a pair of right-aligned k-trees T_1 and T_2 satisfying:*

1. *T_1 is normal*
2. *T_2 contains only nodes of type 0*

Proof. Let T_1 and T_2 be right-aligned trees such that $f = [T_1, T_2]$. We know that such k-trees exist by Lemma 27.

By Lemma 26 there we can transform T_2 into a normal tree T_2' by adding nodes of type 0 and performing graft operations. Note that in this process we never need to decrease the type of a node along the right edge of T_2, so T_2' is again a right-aligned tree. Performing the corresponding addition of type 0 nodes to T_1 gives us a right-aligned tree T_1', since we never needed to add a node to the right-most leaf of T_2 to obtain T_2'. We then have $[T_1', T_2'] = [T_1, T_2]$.

The k-tree T_2' may have nodes of type 1, but any such node has a leaf as its left-most child. For each type 1 node N in T_2', we can therefore add another type 1 node to its left-most

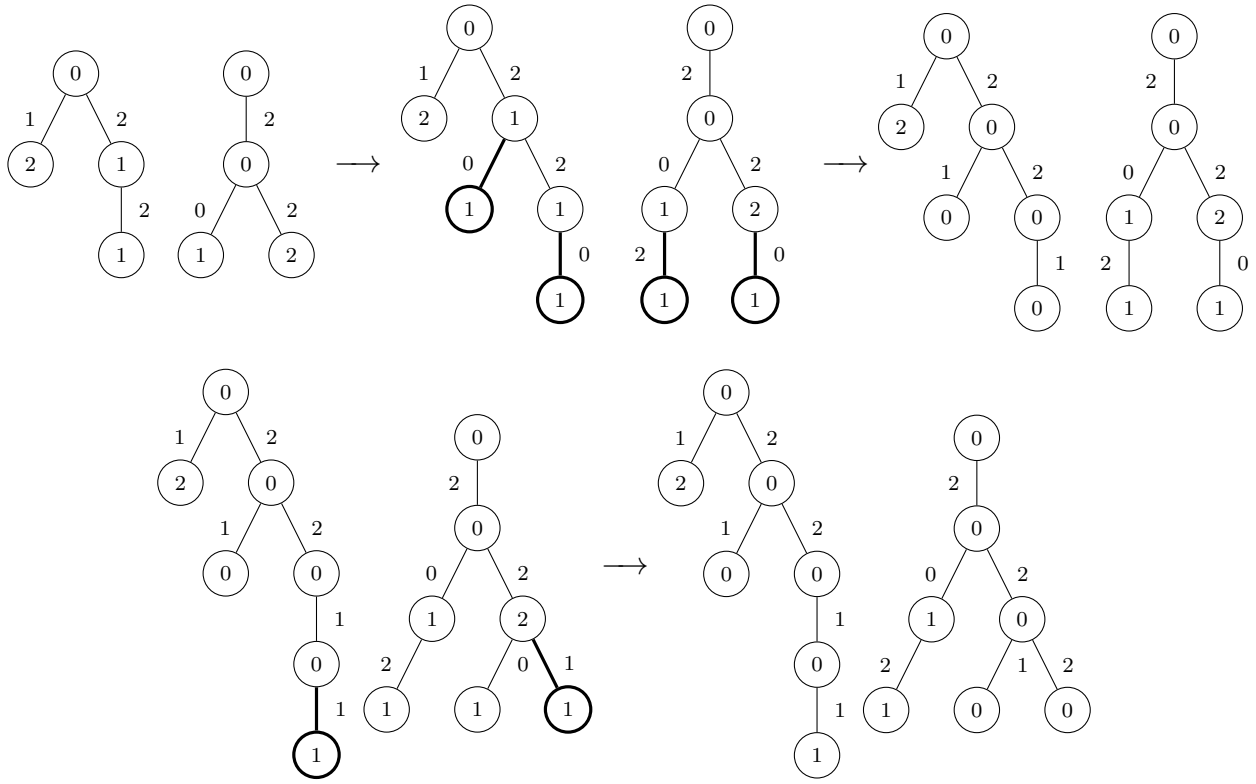
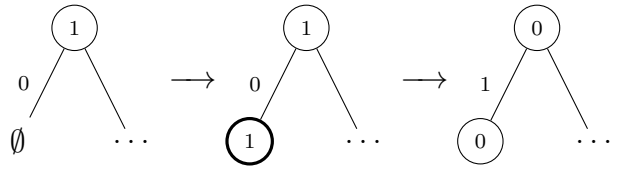


Figure 5.3: Constructing an equivalent right-aligned tree pair in the case where $k = 2$

child, and perform a left-graft operation:



Each of these operations performed at a node of type 1 in T_2' decreases the number of type 1 nodes by 1, so we can continue to do this until all type 1 nodes in T_2' are removed, obtaining a k-tree T_2'' with nodes only of type 0.

Performing the corresponding addition of type 1 nodes to T_1' gives a tree T_1'' which is still right-aligned, because no type 1 nodes were added to the rightmost leaf of T_2' to obtain T_2''. We then have $[T_1'', T_2''] = [T_1, T_2] = f$.

Finally, we can convert T_1'' into a normal tree $\widehat{T}_1$ by performing graft operations and adding only nodes of type 0. Performing the corresponding addition of type 0 nodes to T_2'' gives a k-tree $\widehat{T}$, which still contains only nodes of type zero, and $\left[\widehat{T}_1, \widehat{T}_2\right] = [T_1, T_2] = f$. This gives us our result. □

Corollary 3: *The group $\widetilde{\mathcal{T}}_k$ is generated by the set $\left\{\widetilde{T} \,|\, T \text{ is normal}\right\}$*

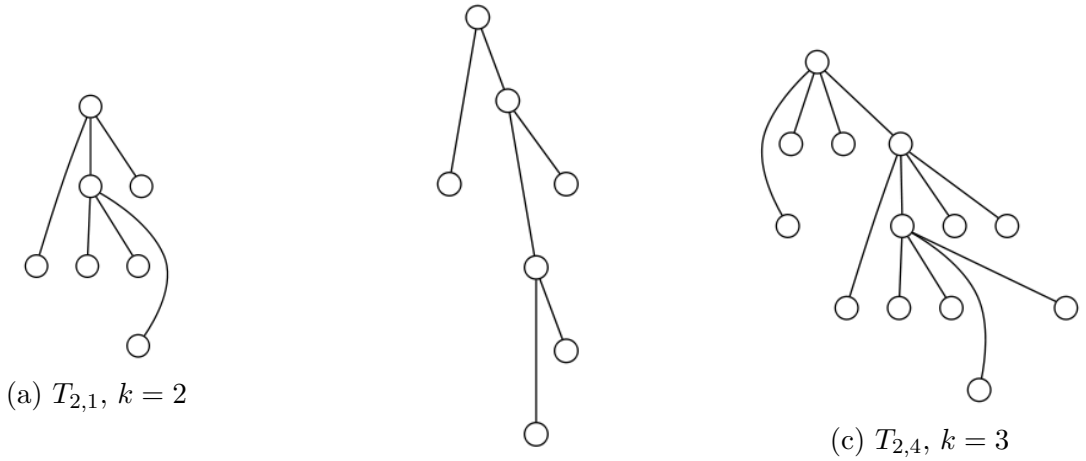
Proof. From Lemma 28 we see that every element $f \in \widetilde{\mathcal{T}}_k$ has a representation as $[T_1, T_2]$

where T_1 is normal and right-aligned, and T_2 is a tree with all nodes of type 0 (and therefore normal). From Lemma 18 we have:

$$f = \widetilde{T}_1 \star \widetilde{T}_2^{-1}$$

It follows that every element of $\widetilde{\mathcal{T}}_k$ can be expressed as the product of elements $\widetilde{T}$ and their inverses where T is normal. □

Definition (Elementary pair)**:** We define the *elementary k-tree* of type n and level i, written $T_{n,i}$, to be the k-tree constructed by starting with the smallest spine with at least $i+2$ leaves and replacing the i^{th} leaf with a node of type n. We define the *elementary pair* of type n and level i, written $[n]_i$ to be $\widetilde{T_{n,i}}$.



(a) $T_{2,1}$, $k = 2$

(b) $T_{0,1}$, $k = 1$

(c) $T_{2,4}$, $k = 3$

Figure 5.4: Three examples of elementary trees

Lemma 29: *If T is a right-aligned tree, and T' is obtained from T by adding a node of type n to the i^{th} leaf of T which is not the right-most leaf, $\widetilde{T}' = \widetilde{T} \star [n]_i$.*

Proof. It suffices to show that $(T', \mathrm{sp}(T'))$ is a leaf product of $(T, \mathrm{sp}(T))$ and $(T_{n,i}, \mathrm{sp}(T_{n,i}))$. This amounts to finding refinements T_1 and T_2 of $\mathrm{sp}(T)$ and $T_{n,i}$ respectively that represent the same k-subdivision. In this case, it is actually possible to find a single k-tree, S which is a refinement of both $\mathrm{sp}(T)$ and $T_{n,i}$.

We define the k-tree S to be the k-tree obtained by adding a node of type n to the i^{th} leaf of $\mathrm{sp}(T')$. Because the i^{th} leaf of T is not the right-most leaf, it must also be true that the i^{th} leaf of $\mathrm{sp}(T)$ is not the right-most leaf. We can therefore obtain S as a refinement of $T_{n,i}$ by adding nodes of type 0 along the right-most edge until it is as long as the spine $\mathrm{sp}(T)$.

We know need to keep track of the corresponding changes we are making to T and $\mathrm{sp}(T_{n,i})$. To T we need to add a node of type n to the i^{th} leaf node, which gives T'. To $\mathrm{sp}(T_{n,i})$ we need to add type 0 nodes along the right edge so that the spine has length one greater than the length of $\mathrm{sp}(T)$ (since $\mathrm{sp}(T_{n,i})$ has a right-edge whose length is greater by one than the length of the right edge of $T_{n,i}$). This gives us $\mathrm{sp}(T')$, because the length of $\mathrm{sp}(T')$ is greater by one than the length of $\mathrm{sp}(T)$.

We conclude that $(T', \mathrm{sp}(T')) \in (T, \mathrm{sp}(T)) \star (T_{n,i}, \mathrm{sp}(T_{n,i}))$ and therefore that $\widetilde{T'} = \widetilde{T} \star [n]_i$. □

Corollary 4: *For any right-aligned k-tree T with all nodes of type less than l for some $l \in \mathbb{Z}_{\geq 1}$ there exists a collection of elementary pairs of type less than l, $\{x_1, \ldots, x_n\}$ such that $\widetilde{T} = x_1 \star \ldots \star x_n$.*

Proof. Any right-aligned tree T can be constructed by starting with the spine S that is its right edge, and then adding nodes of type less than l to leaves other than the right-most leaf. From Lemma 29 it follows that there exists elementary pairs of type less than l, $x_1, \ldots, x_n$, such that:

$$\widetilde{T} = \widetilde{S} \star x_1 \star \ldots \star x_n = x_1 \star \ldots x_n$$

□

Corollary 5: $\Theta_k := \{[n]_i \,|\, 0 \leq n \leq 1,\ i \in \mathbb{Z}_{\geq 0}\}$ *is a generating set for* $\widetilde{\mathcal{T}_k}$

Proof. From Corollary 3, $\widetilde{\mathcal{T}_k}$ is generated by elements $\widetilde{T}$ where T is normal, and by Corollary 4 each such $\widetilde{T}$ is the product of elementary pairs of types 0 and 1. The set of such pairs is Θ. □

Corollary 6: *Let n_i denote $\widetilde{H}([n]_i)$. Then $G := \{n_i \,|\, 0 \leq n \leq 1,\ i \geq 0\}$ generates $\mathcal{F}_{\tau_k}$.*

Proof. This follows from the fact that $\widetilde{H} : \widetilde{\mathcal{T}_k} \to \mathcal{F}_{\tau_k}$ is an isomorphism. □

6. Presentations

6.1 An infinite presentation for the $\mathcal{F}_{\tau_k}$ groups

We now have a generating set for the group $\mathcal{F}_{\tau_k}$. This is not a minimal generating set; we will see later that we can can construct a generating set for each $\mathcal{F}_{\tau_k}$ with $2k+2$ elements. The generating set G is important because we can describe fairly simple infinite presentation for elements of $\mathcal{F}_{\tau_k}$ over G, analogously to the presentation given for F in, for example, [Bur18; CFP96; BS14], and for the group $\mathcal{F}_{\tau_1}$ in [BNR18]. When discussing words on the generators it will be useful to distinguish between equality of words and equality of the group elements represented by a word. We will use $u \equiv v$ to indicate that u and v are the same words, and $u = v$ will indicate merely that they represent the same element of $\mathcal{F}_{\tau_k}$.

6.1.1 Relations on the generators

There are two types of relations we need to consider. The first class of relations will be familiar from the study of the groups $F\Big([0,1], \mathbb{Z}[\frac{1}{p}], \langle p \rangle\Big)$ for $p \in \mathbb{Z}_{>1}$ and corresponds to the fact that we can construct the same k-tree in multiple ways, depending on the order in which we add the nodes. The second class of relations comes from the fact that different k-trees can have the same leaf sequence, and therefore represent the same subdivision. Each relation in the second class corresponds to a graft operation.

Let G be the free group generated by the set $\Pi := \{(n)_i \,|\, n \in \{0,1\},\ i \geq 0\}$. Let $\phi : G \to \widetilde{\mathcal{T}_k}$ be the group homomorphism extending the map $(n)_i \mapsto [n]_i$.

Relations of the first kind

Our first class of relations is of the form:

$$(n)_j (m)_i = (m)_i (n)_{j+k} \qquad \forall n, m \in \{0,1\}, i < j$$

Define $R_1 = \Big\{(n)_j (m)_i (n)_{j+k}^{-1} (m)_i^{-1} \,|\, n, m \in \{0,1\},\ i < j\Big\}$

Relations of the second kind

Our second class of relations is of the form:

$$(0)_i (0)_{i+1} = (1)_i^2 \qquad \forall i \in \mathbb{Z}_{\geq 0}$$

Define $R_2 = \left\{(0)_i(0)_{i+1}(1)_i^{-2} \,|\, i \geq 0\right\}$

Lemma 30: $R_1 \subseteq \ker(\phi)$

Proof. Assume $n, m \in \{0, \ldots, k\}$ and $i < j$. Then:

$$f := \phi\Big((n)_j(m)_i(n)_{j+k}^{-1}(m)_i^{-1}\Big) = [n]_j \star [m]_i \star [n]_{j+k}^{-1} \star [m]_i^{-1}$$

Let S be a spine with more than $j+1$ leaf nodes. Then because $\widetilde{S} = 1_{\widetilde{\mathcal{T}_k}}$ (Lemma 18 we have:

$$f = \widetilde{S} \star [n]_j \star [m]_i \star {[n]_{j+k}}^{-1} \star {[m]_i}^{-1} \star \widetilde{S}^{-1} = \Big(\widetilde{S} \star [n]_j \star [m]_i\Big) \star \Big(\widetilde{S} \star [n]_{j+k} \star [m]_i\Big)^{-1}$$

By Lemma 29, $\widetilde{S} \star [n]_j \star [m]_i = \widetilde{T_1}$ where T_1 is the tree obtained from S by first adding a node of type n to the j^{th} leaf, to obtain a new tree T_1' and then adding a node of type m to the i^{th} leaf node of T_1'.

Similarly, $\widetilde{S} \star [n]_{j+k} \star [m]_i = \widetilde{T_2}$ where T_2 is the tree obtained from adding a node of type m to the i^{th} node, obtaining T_2', and then adding a node of type n to the $j+k^{\text{th}}$ node of T_2'.

Upon inspection, $T_1 = T_2$, because the $j+k^{\text{th}}$ node of T_2' was the j^{th} node of S. We therefore have that:

$$f = \widetilde{T_1} \star {\widetilde{T_1}}^{-1} = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$$

It follows that each element of R_1 is in the kernel of ϕ. □

Lemma 31: $R_2 \subseteq \ker(\phi)$

Proof. This proof is similar to the proof for Lemma 30, but relies on the fact that performing a graft operation does not change the leaf sequence of a k-tree.

Assume $i \in \mathbb{Z}_{\geq 0}$. Then:

$$f := \phi\big((0)_i(0)_{i+1}(1)_i^{-2}\big) = \; [0]_i \star [0]_{i+1} \star {[1]_i}^{-2}$$

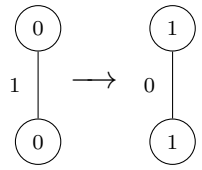
Let S be a spine with more than $j+1$ leaf nodes. Then:

$$\begin{aligned} f =& \widetilde{S} \star [0]_i \star [0]_{i+1} \star {[1]_i}^{-2} \star \widetilde{S}^{-1} \\ =& \Big(\widetilde{S} \star [0]_i \star [0]_{i+1}\Big)\Big(\widetilde{S} \star {[1]_i}^2\Big)^{-1} \end{aligned}$$

By Lemma 29, $\widetilde{S} \star [0]_i \star [0]_{i+1} = \widetilde{T_1}$ where T_1 is the tree obtained from S by first adding a node of type 0 to the i^{th} leaf, N, to obtain a new tree T_1' and then adding a node of type 0 to the $(i+1)^{\text{th}}$ leaf node of T_1'. Note that the $(i+1)^{\text{th}}$ leaf node of T_1' is $N(1)$, which is the leaf node to the right of the long child of N.

Similarly, $\widetilde{S} \star {[1]_i}^2 = \widetilde{T_2}$ where T_2 is the tree obtained from adding a node of type (1) to the i^{th} node, N, obtaining T_2', and then again adding a node of type (1) to the $(i)^{\text{th}}$ leaf node of T_2'. Note that this leaf node is $N(0)$ which is the leaf node to the left of the long child of N.

In this case, T_1 and T_2 will not be the same k-tree, but we can obtain T_2 from T_1 by performing a right graft at the node N, and so $T_1 \sim T_2$:



It follows that:

$$f = \widetilde{T}_1 \star \widetilde{T}_2{}^{-1} = \widetilde{T}_1 \star \widetilde{T}_1{}^{-1} = \mathrm{id}_{\widetilde{\mathcal{T}}_k}$$

□

By Lemmas 30 and 31 there therefore exists a unique, surjective group homomorphism:

$$\psi : \frac{G}{\langle\langle R_1 \cup R_2 \rangle\rangle} \to \mathcal{F}_{\tau_k}$$

such that $\psi \circ p = \phi$ where p is the quotient map $G \to \frac{G}{\langle\langle R_1 \cup R_2 \rangle\rangle}$. For notational simplicity we will make the definition: $\Gamma := \frac{G}{\langle\langle R_1 \cup R_2 \rangle\rangle}$.

Our next goal is to show that ψ is injective, and therefore an isomorphism. Once this is established we will have an infinite presentation for $\widetilde{\mathcal{T}}_k$, and therefore $\mathcal{F}_{\tau_k}$.

We will first make some definitions and establish some technical results.

Definition (Positive word)**:** A *positive word* w is a word in the generators $(n)_i \in \Pi$, and not in their inverses.

For example, $(0)_2^2(1)_0$ is a positive word, but $(0)_2^2(1)_0^{-1}$ is not.

Definition (Type and level of a generator)**:** We define the *type* of a generator $(n)_i$ to be n, and its *level* to be i. We will define the level of a word w to be the minimum level of the generators in w. We will use the notation $\mathcal{L}(x)$ to denote the level of a generator or word x.

Lemma 32: *For any $x \in \Gamma$ there exist positive words u and v such that $x = uv^{-1}$*

Proof. We know that there exists some word w over the generators of Γ such that $x = w$. It suffices to show that if an inverse of one of the generators $\{(n)_i \,|\, n \in \{0, \ldots, k\},\ i \geq 0\}$ occurs to the left of one of the positive generators, then we can rewrite this as a positive generator to the left of an inverse.

Assume $(n)_i^{-1}(m)_j$ is a subword of w. If $i < j$ then:

$$\begin{aligned}
(n)_i^{-1}(m)_j &= (n)_i^{-1}(m)_j(n)_i(n)_i^{-1} \\
&= (n)_i^{-1}(n)_i(m)_{j+k}(n)_i^{-1} && \text{(Using a relation of the first kind)} \\
&= (m)_{j+k}(n)_i^{-1}
\end{aligned}$$

If $i > j$:

$$\begin{aligned}(n)_i^{-1}(m)_j =& (n)_i^{-1}(m)_j(n)_{i+k}(n)_{i+k}^{-1} \\ =& (n)_i^{-1}(n)_i(m)_j(n)_{i+k}^{-1} && \text{(Using a relation of the first kind)} \\ =& (m)_j(n)_{i+k}^{-1}\end{aligned}$$

Now assume $i = j$. Here we have four cases, depending on the values for n and m. But if $n = m$, we have:

$$(n)_i^{-1}(m)_i = (n)_i^{-1}(n)_i = \mathrm{id}_\Gamma$$

and we can therefore remove this subword.

If $n = 0$ and $m = 1$ then we have then we can achieve our result by the relation of the second kind $r = (0)_i(0)_{i+1}(1)_i^{-2}$:

$$(0)_i^{-1}(1)_i = (0)_i^{-1}r(1)_i = (0)_{i+1}(1)_i^{-1}$$

.

If $n = 1$ and $m = 0$ then we have:

$$(1)_i^{-1}(0)_i = \left((0)_i^{-1}(1)_i\right)^{-1} = \left((0)_{i+1}(1)_i\right)^{-1} = (1)_i(0)_{i+1}^{-1}$$

We can therefore continue to shift each instance of an inverse of a generator towards the end of the word representing x, giving us a word of the desired form. □

For the proof of the injectivity of ψ it will be convenient to construct a function from positive words in Π to right-aligned k-trees. If $w = (n_1)_{i_1} \dots (n_t)_{i_t}$ is a word on Π, then we define $\mathcal{T}(w)$ to be the right aligned k-tree constructed by choosing a minimal spine S, and then attaching to S first a node of type n_1 to the ${i_1}^{\text{th}}$ leaf, then attaching a node of type n_2 to the ${i_2}^{\text{th}}$ leaf and so on. We retrospectively define S as minimal if it is the spine of minimal length such that none of the nodes are attached to the rightmost leaf in this process. Note that with this definition, $\mathcal{T}(w)$ will always be a reduced right-aligned k-tree. If it weren't, we could have picked a smaller spine.

Lemma 33: *For any word w over Π, $\psi(w) = \widetilde{\mathcal{T}(w)}$*

Proof. Let S be the minimal spine used to construct $\mathcal{T}(w)$. We will denote the letters of w by $w = (n_1)_{i_1} \dots (n_t)_{i_t}$. Then:

$$\phi(w) = [n_1]_{i_1} \star \dots \star [n_t]_{i_t} = S \star [n_1]_{i_1} \star \dots \star [n_t]_{i_t} = \widetilde{\mathcal{T}(w)}$$

by Lemma 29. □

Lemma 34: *If u and v are words over Π such that $\mathcal{T}(u) = \mathcal{T}(v)$, then $u = v$ as elements of Γ.*

Proof. We will prove this by induction on the length of u, noting that the result is trivial in the case where u has length 0. Now assume the result holds for u with length less than $d \in \mathbb{Z}_{>1}$, and assume u and v are words over Π with u of length d such that $\mathcal{T}(u) = \mathcal{T}(v)$.

Let $(n)_i$ be the last letter of u.

Claim: *There exists a word v' over Π such that $v = v'(n)_i$ in Γ and $\mathcal{T}(v'(n)_i) = \mathcal{T}(v)$.*

> *Proof.* Because $\mathcal{T}(u) = \mathcal{T}(v)$, we know there exists a letter in v that represents adding the node N corresponding to $(n)_i$ in u. Let $(n)_j$ denote this letter in the word v.
>
> We will prove the result by induction on the number of letters p between $(n)_j$ and the end of the word v. If $p = 0$, then $v \equiv v'(n)_j$. Because $(n)_j$ corresponds to adding the node N, it must be the case that $j = i$, and so the result holds. Now assume the result holds when $p < s$ for some $s \in \mathbb{Z}_{>0}$ and assume $(n)_j$ is distance s from the end of v. Let $(m)_r$ denote the generator following $(n)_j$ in v. The level r of this generator cannot be in $\{j, j+1, \ldots, j+k\}$ because this would correspond to adding a node to a child of N, and by the fact that N can be added last in the tree $\mathcal{T}(u) = \mathcal{T}(v)$, we conclude that all the children of N are leaves. We therefore have two cases: $r \geq j+k+1$ or $r < j$.
>
> **Case 1:** $r \geq j+k+1$
>
> In this case we can use the relation $(m)_{r-k}(n)_j = (n)_j(m)_r \in R_1$ to replace the subword $(n)_j(m)_r$ of v with $(m)_{r-k}(n)_j$ giving us a new word $\hat{v} = v$. By the argument given in Lemma 30 we note that $\mathcal{T}(\hat{v}) = \mathcal{T}(v)$ and that the generator $(n)_j$ in the word $\hat{v}$ represents the addition of N to the tree. The generator $(n)_j$ in $\hat{v}$ is less than s letters away from the end of $\hat{v}$, so the result follows by our induction hypothesis.
>
> **Case 2:** $r < j$
>
> The argument is similar to that for Case 1. Here we can use the relation $(n)_j(m)_r = (m)_r(n)_{j+k} \in R_1$ to replace the subword $(n)_j(m)_r$ of v with $(m)_r(n)_{j+k}$ giving us a new word $\hat{v} = v$. By the argument given in Lemma 30 we note that $\mathcal{T}(\hat{v}) = \mathcal{T}(v)$ and that the generator $(n)_{j+k}$ represents the addition of N to the tree. The generator $(n)_{j+k}$ $\hat{v}$ is less than s letters away from the end of $\hat{v}$, so the result follows by our induction hypothesis. □

Now we have that $u = u'(n)_i$ and $v = v'(n)_i$ satisfy $\mathcal{T}(u'(n)_i) = \mathcal{T}(v'(n)_i)$ and so $\mathcal{T}(u') = \mathcal{T}(v')$. This is simply the k-tree $\mathcal{T}(u)$ with the node N removed. The word u' has length less than d, so by our induction hypothesis it follows that $u' = v'$ in Γ, and that:

$$u \equiv u'(n)_i = v'(n)_i = v$$

as elements of Γ. □

Lemma 35: *If u and v are words over Π such that $\mathcal{T}(u)$ is obtained from $\mathcal{T}(v)$ by a single graft operation, then $u = v$ as elements of Γ.*

Proof. We first note that because $\mathcal{T}(u)$ and $\mathcal{T}(v)$ are both trees with nodes only of types 0 and 1, the only graft operations that could possibly convert $\mathcal{T}(u)$ into $\mathcal{T}(v)$ are performing a left graft at a node N of type 1, where $N(0)$ is also of type 1, or performing a right graft

at a node N of type 0 where $N(1)$ is also of type 0. Any other graft would produce nodes of types greater than 1. Because these are inverse transformations, it suffices only to check the case where $\mathcal{T}(u)$ is obtained from $\mathcal{T}(v)$ by right-graft, since otherwise $\mathcal{T}(v)$ is obtainable from $\mathcal{T}(u)$ by a right-graft, and we could swap u and v. Note also that the node at which the right graft occurs cannot be a node on the right edge, since this would produce a tree that is not right-aligned.

Assume that u and v are two words over Π that satisfy our condition. Then there exists a type 0 node N in $\mathcal{T}(u)$ such that $N(1)$ is also a type 0 node and performing a right graft at N obtains $\mathcal{T}(v)$. We could chose to construct the k-tree $\mathcal{T}(u)$ by adding the type zero node $N(1)$ directly after the node N, which would correspond to a word u' over Π of the form $u_1(0)_i(0)_{i+1}u_2$, where u_1 and u_2 are also words over Π. By Lemma 34, it follows that $u = u'$ since $\mathcal{T}(u) = \mathcal{T}(u')$ by construction.

Using a relation of the second kind $(0)_i(0)_{i+1} = (1)_i^2$, we can replace $(0)_i(0)_{i+1}$ in u' to obtain a new word:

$$\hat{u} \equiv u_1(1)_i^2 u_2$$

with $u = u' = \hat{u}$.

From our argument in Lemma 31 we see that $\mathcal{T}(\hat{u})$ is the k-tree that is obtained from $\mathcal{T}(u)$ by performing a right graft operation at N, and therefore $\mathcal{T}(\hat{u}) = \mathcal{T}(v)$. By use again of Lemma 34 we conclude that:

$$u = u' = \hat{u} = v$$

□

Lemma 36: *If u and v are words over Π such that $\mathcal{T}(u) \sim \mathcal{T}(v)$ then $u = v$.*

Proof. Because $\mathcal{T}(u)$ and $\mathcal{T}(v)$ are both trees with nodes only of types 0 and 1, the fact that $\mathcal{T}(u) \sim \mathcal{T}(v)$ implies that there exists a sequence of graft operations which never produces a node of type greater than 1 which transforms $\mathcal{T}(u)$ into $\mathcal{T}(v)$, by Lemma 24. Because grafting never needs to occur on the right edge, the output of each graft operation remains a reduced, right-aligned tree, and thus $\mathcal{T}(w)$ for some word w over Π. It follows that there is a sequence of words $u = w_0, w_1, \ldots, w_n = v$ such that each $\mathcal{T}(w_{i+1})$ is obtained from $\mathcal{T}(w_i)$ by a single graft operation. From Lemma 35, it follows that $u = w_1 = \cdots = w_n = v$ as elements of Π. □

Lemma 37: *$\psi : \Gamma \to \widetilde{\mathcal{T}_k}$ is injective.*

Proof. Assume $x \in \ker(\psi)$. By Lemma 32 there exist positive words in Π, u and v, such that

$$x = uv^{-1}$$

The condition that $x \in \ker(\psi)$ gives us that:

$$\psi(x) = \psi(u) \star \psi(v)^{-1} = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$$

We now note the following:

1. By Lemma 33, $\widetilde{\mathcal{T}(u)} \star \widetilde{\mathcal{T}(v)} = \mathrm{id}_{\widetilde{\mathcal{T}_k}}$

2. By Lemma 20, $\mathcal{T}(u) \sim \mathcal{T}(v)$
3. By Lemma 36, $u = v$ as elements of Γ

It follows that $x = uv^{-1} = uu^{-1} = \mathrm{id}_\Gamma$.

We conclude that ψ is injective.

□

6.2 A Finite Presentation for $\mathcal{F}_{\tau_k}$

From our infinite presentation we can construct a finite presentation as has been done for the groups F_k in [BS14]. For notation convenience we will use $(x \downarrow y)$ to denote $x^{-1}yx$ in this section. We will also be making frequent use of the generator $(0)_0$ of $\mathcal{F}_{\tau_k}$, so for this section only we will denote this generator by x.

Lemma 38: *For $n, m \in \{0, 1\}$ and $0 \leq i < j$, $\left((m)_i \downarrow (n)_j\right) = (n)_{j+k}$*

Proof. From the relation of the first kind $(n)_j(m)_i = (m)_i(n)_{j+k}$ we have:

$$\left((m)_i \downarrow (n)_j\right) = (m)_i^{-1}(n)_j(m)_i = (m)_i^{-1}(m)_i(n)_{j+k} = (n)_{j+k}$$

□

In particular, for any $n \in \{0, 1\}$ and $i > 0$, $(x \downarrow (n)_i) = (n)_{i+k}$.

We begin by defining a finite generating set for $\mathcal{F}_{\tau_k}$.

Lemma 39: *$\mathcal{F}_{\tau_k}$ is generated by the finite set:*

$$\Phi_k := \{(0)_i, (1)_i \,|\, 0 \leq i \leq k\}$$

Proof. Clearly each generator of level less than or equal to k is in the group generated by this set. It remains to consider generators of the form $(n)_i$ for $i > k$. In this case we can write i uniquely as $\sigma(i) + \eta(i)k$ where $1 \leq \sigma(i) \leq k$ and $\eta(i) \in \mathbb{Z}_{\geq 1}$. From Lemma 38, we therefore have:

$$(n)_i = \left(x^{\eta(i)} \downarrow (n)_{\sigma(i)}\right) \in \langle \Phi_k \rangle$$

which gives us this result. □

By rewriting all of our relators in R_1 and R_2 on the generating set Π in terms of the generators in Φ_k we will obtain another infinite presentation for $\mathcal{F}_{\tau_k}$. We will then show that we can replace these relators with a finite set of relators and thus obtain a finite presentation.

6.2.1 Removing Relations of the Second Kind

We will see that we can replace our relations of the the second kind with the finite set:

$$\mathcal{R}_2 := \left\{ (0)_i (0)_{i+1} = (1)_i^2 \,|\, 0 \leq i < k \right\} \cup \left\{ (0)_k x^{-1} (0)_1 x = (1)_k \right\}$$

Each relation of the second kind is of the form $(0)_i (0)_{i+1} = (1)_i^2$. Written in terms of Φ_k we get:

$$\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)}\right)\left(x^{\eta(i+1)} \downarrow (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \tag{‡}$$

For $i \neq nk - 1$ for $n \in \mathbb{Z}_{\geq 1}$, $\eta(i) = \eta(i+1)$ so from (‡) we have:

$$\begin{aligned}
&\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)}\right)\left(x^{\eta(i+1)} \downarrow (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \\
\iff &\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)}\right)\left(x^{\eta(i)} \downarrow (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \\
\iff &\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)} (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \\
\iff &(0)_{\sigma(i)} (0)_{\sigma(i+1)} = (1)_{\sigma(i)} \ \in \mathcal{R}_2
\end{aligned}$$

For $i = nk$ for $n \in \mathbb{Z}_{\geq 1}$, $\eta(i+1) = \eta(i) + 1$, $\sigma(i) = k$ and $\sigma(i+1) = 1$, so we have:

$$\begin{aligned}
&\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)}\right)\left(x^{\eta(i+1)} \downarrow (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \\
\iff &\left(x^{\eta(i)} \downarrow (0)_k\right)\left(x^{\eta(i)} \downarrow x^{-1}(0)_1 x\right) = \left(x^{\eta(i)} \downarrow (1)_k\right) \\
\iff &\left(x^{\eta(i)} \downarrow (0)_k x^{-1}(0)_1 x\right) = \left(x^{\eta(i)} \downarrow (1)_k\right) \\
\iff &(0)_k x^{-1}(0)_1 x = (1)_k \ \in \mathcal{R}_2
\end{aligned}$$

And for $i = 0$, $\sigma(i) = \eta(i) = 0$, so we have:

$$\begin{aligned}
&\left(x^{\eta(i)} \downarrow (0)_{\sigma(i)}\right)\left(x^{\eta(i+1)} \downarrow (0)_{\sigma(i+1)}\right) = \left(x^{\eta(i)} \downarrow (1)_{\sigma(i)}\right) \\
\iff &(0)_0 (0)_1 = (1)_0 \ \in \mathcal{R}_2
\end{aligned}$$

If follows that the relations of the second kind can be replaced by the relations in $\mathcal{R}_2$.

6.2.2 Removing Relations of the First Kind

We now consider the relations of the first kind, all of the form $(m)_j (n)_i = (n)_i (m)_{j+k}$ for $i < j$. We can rewrite these in terms of generators in Φ_k as:

$$\left(x^{\eta(j)} \downarrow (n)_{\sigma(j)}\right)\left(x^{\eta(i)} \downarrow (m)_{\sigma(i)}\right) = \left(x^{\eta(i)} \downarrow (m)_{\sigma(j)}\right)\left(x^{\eta(j+k)} \downarrow (n)_{\sigma(j+k)}\right)$$

Because $j > 1$ (since $j > i \geq 0$), $\sigma(j+k) = \sigma(j)$ and $\eta(j+k) = \eta(j)+1$, so we have:

$$\begin{aligned}
&\Big(x^{\eta(j)} \downarrow (n)_{\sigma(j)}\Big)\Big(x^{\eta(i)} \downarrow (m)_{\sigma(i)}\Big) = \Big(x^{\eta(i)} \downarrow (m)_{\sigma(j)}\Big)\Big(x^{\eta(j+k)} \downarrow (n)_{\sigma(j+k)}\Big) \\
\iff &\Big(x^{\eta(j)} \downarrow (n)_{\sigma(j)}\Big)\Big(x^{\eta(i)} \downarrow (m)_{\sigma(i)}\Big) = \Big(x^{\eta(i)} \downarrow (m)_{\sigma(j)}\Big)\Big(x^{\eta(j)+1} \downarrow (n)_{\sigma(j)}\Big) \\
\iff &\Big(x^{-\eta(i)}(m)^{-1}_{\sigma(j)}x^{\eta(i)}\Big)\Big(x^{-\eta(j)}(n)_{\sigma(j)}x^{\eta(j)}\Big)\Big(x^{-\eta(i)}(m)_{\sigma(i)}x^{\eta(i)}\Big) = \Big(x^{-(\eta(j)+1)}(n)_{\sigma(j)}x^{(\eta(j)+1)}\Big) \\
\iff &(m)^{-1}_{\sigma(j)}\Big(x^{-(\eta(j)-\eta(i))}(n)_{\sigma(j)}x^{(\eta(j)-\eta(i))}\Big)(m)_{\sigma(i)} = \Big(x^{-(\eta(j)-\eta(i)+1)}(n)_{\sigma(j)}x^{(\eta(j)-\eta(i)+1)}\Big) \\
\iff &\Big(x^{s}(m)_{\sigma(i)} \downarrow (n)_{\sigma(j)}\Big) = \Big(x^{s+1} \downarrow (n)_{\sigma(j)}\Big)
\end{aligned}$$

where $s = \eta(j) - \eta(i)$. The condition that $j > i$ translates to the condition that $\eta(j) > \eta(i)$ (i.e $s > 0$) or $\sigma(j) > \sigma(i)$.

We can therefore replace the relations from R_1 over Π by the set of relations over Φ_k given by:

$$\mathcal{R}_1 := \Big\{\Big(x^s(m)_i \downarrow (n)_j\Big) = \Big(x^{s+1} \downarrow (n)_j\Big) \,|\, 0 \leq i,j \leq k,\ s > 0 \text{ or } j > i\Big\}$$

We can partition this set of relations as $\mathcal{R}_1 = \bigsqcup_{s=0}^{\infty} \mathcal{R}_{1,s}$ where:

$$\begin{aligned}
\mathcal{R}_{1,0} &:= \Big\{\Big((m)_i \downarrow (n)_j\Big) = \Big(x \downarrow (n)_j\Big) \,|\, 0 \leq i,j \leq k,\ j > i\Big\} \\
\mathcal{R}_{1,s} &:= \Big\{\Big(x^s(m)_i \downarrow (n)_j\Big) = \Big(x^{s+1} \downarrow (n)_j\Big) \,|\, 0 \leq i,j \leq k\Big\} \qquad (s > 0)
\end{aligned}$$

We will show that we only need a finite subset of these relations, given by the set:

$$\mathcal{R}'_1 := \mathcal{R}_{1,0} \sqcup \mathcal{R}_{1,1} \sqcup \mathcal{R}_{1,2}$$

Lemma 40: *The relations in $\mathcal{R}'_1$ are equivalent to the relations in $\mathcal{R}_1$.*

Proof. We need to show that each relation in $\mathcal{R}_1$ is implied by the relations in $\mathcal{R}'_1$. We will do this by induction, showing that for $s \in \mathbb{Z}_{\geq 0}$, the relations in $\mathcal{R}_{1,s}$ are implied by those in $\mathcal{R}'_1$.

The result is clearly true for $s = 0$, 1 and 2. Now assume that $d \geq 2$ and that the relations in $\mathcal{R}_{1,d-1}$ and $\mathcal{R}_{1,d-2}$ are implied by those in $\mathcal{R}'_1$. We will show that with this hypothesis, $\Big(x^d \downarrow (m)_i\Big) = \Big(x^{d+1} \downarrow (n)_j\Big)$ for arbitrary $0 \leq i,j \leq k$, which means that the relations in $\mathcal{R}_{1,d}$ are also implied by those in $\mathcal{R}'_1$. In this computation we will make repeated use of the fact that $(a \downarrow \cdot)$ defines a right group action. In particular, $(ab \downarrow c) = (b \downarrow (a \downarrow c))$. We will use the following substitution:

$$\begin{aligned}
x^{d+1}(m)_i &= x^2\Big((n)_j^{-1}x^{d-1}(m)_i\Big)\Big((n)_j^{-1}x^{d-1}(m)_i\Big)^{-1}x^{d-1}(m)_i \\
&= x^2(n)_j^{-1}x^{d-1}(m)_i\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \qquad (**)
\end{aligned}$$

Then we have:

$$
\begin{aligned}
&\Big(x^{d+1}(m)_i \downarrow (n)_j\Big) \\
&= \Big(x^2(n)_j^{-1}x^{d-1}(m)_i\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow (n)_j\Big) && \text{(by } **\text{)} \\
&= \Big(\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow \Big(x^2(n)_j^{-1}x^{d-1}(m)_i \downarrow (n)_j\Big)\Big) \\
&= \Big(\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow \Big((n)_j^{-1}x^{d-1}(m)_i \downarrow \Big(x^2 \downarrow (n)_j\Big)\Big)\Big) \\
&= \Big(\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow \Big((n)_j^{-1}x^{d-1}(m)_i \downarrow \Big(x(n)_j \downarrow (n)_j\Big)\Big)\Big) && \text{(from } \mathcal{R}_{1,1}\text{)} \\
&= \Big(\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow \Big(x(n)_j(n)_j^{-1}x^{d-1}(m)_i \downarrow (n)_j\Big)\Big) \\
&= \Big(\Big(x^{d-1}(m)_i \downarrow (n)_j\Big) \downarrow \Big(x^d(m)_i \downarrow (n)_j\Big)\Big) \\
&= \Big(\Big(x^d \downarrow (n)_j\Big) \downarrow \Big(x^{d+1} \downarrow (n)_j\Big)\Big) && \text{(Induction hypothesis)} \\
&= \Big(x^{-d}(n)_j x^d \downarrow \Big(x^{d+1} \downarrow (n)_j\Big)\Big) \\
&= \Big(x(n)_j x^d \downarrow (n)_j\Big) \\
&= \Big(x^d \downarrow \Big(x(n)_j \downarrow (n)_j\Big)\Big) \\
&= \Big(x^d \downarrow \Big(x^2 \downarrow (n)_j\Big)\Big) && \text{(from } \mathcal{R}_{1,1}\text{)} \\
&= \Big(x^{d+2} \downarrow (n)_j\Big)
\end{aligned}
$$

which was the desired result.

We conclude that the relations in $\mathcal{R}_1$ are equivalent to the relations in $\mathcal{R}_1'$, and therefore that $\mathcal{R}_1'$ gives a finite presentation of $\mathcal{F}_{\tau_k}$ over Φ_k. □

It follows that the finite set of relations $\mathcal{R}_1' \sqcup \mathcal{R}_2$ over the generating set Φ_k gives a finite presentation for $\mathcal{F}_{\tau_k}$.

7. Further Properties

7.1 Abelianisations of the groups $\mathcal{F}_{\tau_k}$

We will use the infinite presentation of the groups $\mathcal{F}_{\tau_k}$ to study the abelianisations of these groups. We will find that the abelianisation of $\mathcal{F}_{\tau_k}$ is isomorphic to $\mathbb{Z}_2 \oplus \mathbb{Z}^{k+1}$. Knowledge of the finite presentation provides a shortcut for this process, since most of the relations are expressed in terms of conjugation and disappear in the abelianisation. We use the infinite presentation because it is simpler.

We produce a presentation for the abelianisation $\mathcal{A}_{\tau_k}$ of $\mathcal{F}_{\tau_k}$ by taking our infinite presentation for $\mathcal{F}_{\tau_k}$ in the previous section and adding the relations $xyx^{-1}y^{-1}$ for each pair of generators s and t in our presentation. Rather than dealing with these relations explicitly, we will implicitly implement these relations by using additive notation for our group operation and assuming that it is commutative. We transforming our $\mathcal{F}_{\tau_k}$ relations accordingly to get a presentation of $\mathcal{A}_{\tau_k}$ as an abelian group:

$$G_0 := \{(n)_i \,|\, 0 \leq n \leq 1,\ i \geq 0\}$$

and relators given by $L_0 = \overline{R_1} \cup \overline{R_2}$ where:

$$\overline{R_1} = \left\{(n)_j + (m)_i - (m)_i - (n)_{j+k} \,|\, i < j\right\}$$
$$\overline{R_2} = \left\{(0)_i + (0)_{i+1} - 2(1)_i \,|\, i \geq 0\right\}$$

Note that $\overline{R_1}$ and $\overline{R_2}$ were produced by "abelianising" the relations in R_1 and R_2 on $\mathcal{F}_{\tau_k}$.

This gives us an infinite presentation for the abelian group $\mathcal{A}_{\tau_k}$, but we will see that we can obtain a much simpler finite presentation. We will achieve this by a sequence of Tietze transformations.

7.1.1 Step 1: Eliminating the relations of the first kind

One simplification of this set of relators that we can immediately perform is to replace each of the relators $\left((n)_j + (m)_i - (m)_i - (n)_{j+k}\right) \in \overline{R_1}$ with the relator $(n)_j - (n)_{j+k}$, as it is clearly equivalent. We can therefore replace our set L_0 of relations on G_0 with $L_0' = \overline{R_1}' \cup \overline{R_2}$ where:

$$\overline{R_1}' := \left\{(n)_j - (n)_{j+k} \,|\, j \geq 1\right\}$$

We will now use the relations in $\overline{R_1}'$ to eliminate most of our generators.

Lemma: *$\mathcal{A}_{\tau_k}$ is generated by the set:*

$$G_1 := \{(n)_i \,|\, 0 \leq n \leq 1,\ 0 \leq i \leq k\}$$

Proof. We need to show that each generator $(n)_i$ where $i > k$, can be expressed as a $\mathbb{Z}$-linear combination of the generators in G_1 using relations in L_0'.

Let $(n)_i$ be such a generator with $i > k$. We therefore have $(n)_{i-k} - (n)_i$ as a relator in $\overline{R_1}' \subseteq L_0'$. Using this relator, we get:

$$(n)_i = (n)_i + (n)_{i-k} - (n)_i = (n)_{i-k}$$

Then if $i - k$ is greater than k, we can repeat the process until we reduce the lower index of $(n)_i$ within the range $\{0, \ldots, k\}$. This means that we can write any generator $(n)_i$ for $i > k$ as a generator $(n)_{\sigma(i)}$ where $0 \leq \sigma(i) \leq k$ using relations of the first kind in $\mathcal{A}_{\tau_k}$. □

We now need to modify our relators in L_0' by replacing each appearance of $(n)_i \in G_0 \backslash G_1$ with its equivalent generator in G_1 to get a presentation for $\mathcal{A}_{\tau_k}$ over the generators in G_1. In doing so, we will want a notation for the value between 0 and k that we obtain by repeatedly subtracting k from i while i is greater than k. We will use the notation $\sigma(i)$ for such a value associated to an $i \in \mathbb{Z}_{\geq 0}$. Note that $\sigma(i)$ is not quite $(i \mod k)$, since $(n)_k \neq (n)_0$.

Lemma: *The relators given by the set:*

$$L_1 = \left\{(0)_i + (0)_{\sigma((i+1))} - 2(1)_i \,|\, 0 \leq i \leq k\right\}$$

define a presentation of $\mathcal{A}_{\tau_k}$ over the generating set G_1.

Proof. We first show that all the relators in $\overline{R_1}$ become trivial when we replace each $(n)_j$ with $(n)_{\sigma(j)}$. This is to be expected, as when we use a relator to eliminate a generator, the relator becomes trivial in the remaining generators.

We observe that if $j \geq 1$, then $j + k > k$, and so $\sigma(j) = \sigma((j + sk))$ for any $s \in \mathbb{Z}_{\geq 0}$. Any relator $(n)_j - (n)_{j+k} \in \overline{R_1}'$ when rewritten in terms of the generators in G_1 therefore becomes the trivial relator $\left((n)_{\sigma(j)} - (n)_{\sigma(j)}\right)$, which we may discard.

The only relators we care about, then, are those obtained by rewriting the relators in $\overline{R_2}$. Such a relator, $(0)_i + (0)_{i+1} - (1)_i^2$ written in terms of the generators in G_1 is:

$$(0)_{\sigma(i)} + (0)_{\sigma((i+1))} - (1)^2_{\sigma(i)} \tag{‡}$$

If $i \leq k$, then $\sigma(i) = i$, and the above relator is a relator in L_1. If $i > k$, then $\sigma(i) > 1$ and $i = \sigma(i) + sk$ for some $s \in \mathbb{Z}_{\geq 1}$, so

$$\sigma(i + 1) = \sigma((\sigma(i) + sk + 1)) = \sigma((\sigma(i) + 1))$$

so our relator (‡) is also given by:

$$(0)_{\sigma(i)} + (0)_{\sigma((\sigma(i)+1))} - (1)^2_{\sigma(i)}$$

which is in L_1. Thus every relator in L_0 on G_0 rewritten in G_1 generators is a relator in L_1, and clearly every relator in L_1 is the image of a relator in L_0.

□

7.1.2 Step 2: Eliminating relations of the second kind

In the previous section we used relations of the first kind to reduce the number of generators, removing the relation in the process. We now do the same thing for some of the relations of the second kind.

Lemma: *$\mathcal{A}_{\tau_k}$ is generated by the set*

$$G_2 := \{(0)_0\} \cup \{(1)_i \,|\, 0 \leq i \leq k\}$$

and the single relator:

$$r = 2\left(\sum_{s=1}^{k}(-1)^{k-s+1}(1)_s\right) + (0)_k + (-1)^{k-1}(0)_k$$

over G_2 gives a presentation for $\mathcal{A}_{\tau_k}$.

Proof. We will use k of our $k+1$ relators in L_1 to eliminate k generators from G_1, and then we will see that when we rewrite our last relator in L_1 using generators in G_2 we get r.

We have one relator for each $0 \leq i \leq k$ in L_1 of the form:

$$(0)_i + (0)_{\sigma((i+1))} - 2(1)_i$$

We note that for $i < k$, $\sigma(i+1) = i+1$, and for $i = k$, $\sigma((i+1)) = 1$ so we can write our relators as:

$$\begin{aligned} r_i =& (0)_i + (0)_{i+1} - 2(1)_i \quad i < k \\ r_k =& (0)_k + (0)_1 - 2(1)_k \end{aligned}$$

We first use r_{k-1} to replace the generator $(0)_{k-1}$ with $2(1)_{k-1} - (0)_k$. In doing so, we can remove the relator r_1 from our set of relators.

We can then use r_{k-2} to eliminate $(0)_{k-2}$:

$$(0)_{k-2} = (0)_{k-2} - r_{k-2} = 2(1)_{k-2} - (0)_{k-1} = 2(1)_{k-2} - 2(1)_{k-1} + (0)_k$$

We can proceed by induction to eliminate $(0)_i$ using r_i for each $i < k$. We see that:

$$(0)_i = 2\left(\sum_{s=i}^{k-1}(-1)^{k-s+1}(1)_s\right) + (-1)^{k-i+1}(0)_k$$

This leaves us with the generating set G_2. The only relator we have not exhausted from L_2 is r_k. Translating r_k into the G_2 generators therefore gives our single relator over G_2:

$$\begin{aligned} r_k &= (0)_k + (0)_1 - 2(1)_k \\ &= (0)_k + \left(2\left(\sum_{s=1}^{k-1}(-1)^{k-s+1}(1)_s\right) + (-1)^{k-1}(0)_k\right) - 2(1)_k \\ &= 2\left(\sum_{s=1}^{k}(-1)^{k-s+1}(1)_s\right) + (-1)^{k-1}(0)_k + (0)_k \end{aligned}$$

□

Our final step is to adjust the generators slightly, to make the relator r simpler. Let $\epsilon(k)$ be 1 if k is odd and 0 if k is even. Define:

$$x := \sum_{s=1}^{k} (-1)^{k-s+1} (1)_s + \epsilon(k)(0)_k$$

Lemma: *The relator $r' = 2x$ over the generators:*

$$G_2 := \{(0)_0\} \cup \{(1)_i \,|\, 0 \leq i \leq k-1\} \cup \{x\}$$

gives a presentation for $\mathcal{A}_{\tau_k}$.

Proof. The fact that G_2' is a generating set for $\mathcal{A}_{\tau_k}$ follows from the fact that we can recover the missing generator $(1)_k$ from G_2 as:

$$(1)_k = \sum_{s=1}^{k-1} (-1)^{k-s+1} (1)_s + \epsilon(k)(0)_k - x$$

Rewriting the relator r over G_2 in terms of the generators of G_2' gives us:

$$\begin{aligned}
r &= 2\left(\sum_{s=1}^{k} (-1)^{k-s+1} (1)_s\right) + (-1)^{k-1}(0)_k + (0)_k \\
&= \ = -2(1)_k + 2\left(\sum_{s=1}^{k-1} (-1)^{k-s+1} (1)_s\right) + (-1)^{k-1}(0)_k + (0)_k \\
&= \ = -2\left(\sum_{s=1}^{k-1} (-1)^{k-s+1} (1)_s + \epsilon(k)(0)_k - x\right) \\
&\qquad + 2\left(\sum_{s=1}^{k-1} (-1)^{k-s+1} (1)_s\right) + (-1)^{k-1}(0)_k + (0)_k \\
&= 2x - \epsilon(k) + (-1)^{k-1}(0)_k + (0)_k \\
&= 2x
\end{aligned}$$

We conclude that the relator $r' = 2x$ over G_2' gives a presentation for $\mathcal{A}_{\tau_k}$. □

Corollary 7: $\mathcal{A}_{\tau_k} \cong \mathbb{Z}_2 \oplus \mathbb{Z}^{k+1}$

Corollary 8: *For $n \neq m$, $\mathcal{F}_{\tau_n} \ncong \mathcal{F}_{\tau_m}$.*

7.2 A normal form for the groups $\mathcal{F}_{\tau_k}$

Normal forms for F_1 and $\mathcal{F}_{\tau_1}$ have been constructed in [Bur18] and [BNR18] in a way that can be naturally extended to the groups $\mathcal{F}_{\tau_k}$ for arbitrary k. Most of the work required to

establish this normal form for $\mathcal{F}_{\tau_k}$ has already been accomplished in establishing that k-tree pair has a normal representative in Lemma 28. In [Bur18] and [BNR18], the normal forms are used to give estimates for the length of words in the group. Our main application of the normal form we develop will be to the construct embeddings from F_k to $\mathcal{F}_{\tau_k}$.

Definition (Monotone Positive word)**:** A *monotone positive word* w is a positive word satisfying that if $(n)_i$ appears before $(m)_j$ in w, then $i \leq j$.

Definition (Type and level of a generator)**:** We define the *type* of a generator $(n)_i$ to be n, and its *level* to be i. We will define the level of a word w to be the minimum level of the generators in w. We will use the notation $\mathcal{L}(x)$ to denote the level of a generator or word x.

Definition (D)**:** If w is a word over the generators of $\mathcal{F}_{\tau_k}$ and their inverses of level greater than or equal to k, then we define $\mathcal{D}(w)$ to be the word obtained by decreasing the level of all generators in w by k. Similarly we define $\mathcal{D}^{-1}(w)$ to be the the word obtained by increasing the level of all the generators in w by k.

Definition (Seminormal form)**:** We will say that a word w in the generators of $\mathcal{F}_{\tau_k}$ is in *seminormal form* if it can written as $w \equiv uv^{-1}$ where:

1. u and v are both monotone positive words.
2. All the generators in u are of type 0 or 1.
3. If a type one generator $(1)_i$ occurs in u, the following generator in u must have level greater than i.
4. All the generators in v are of type 0

Proposition 2: *Each element $f \in \mathcal{F}_{\tau_k}$ can be represented as a word in seminormal form.*

Proof. From Lemma 28 we know that we have $f = \psi^{-1}([T_1, T_2]) = \psi^{-1}\left(\widetilde{T_1} \star \widetilde{T_2}^{-1}\right)$ where:

1. T_1 and T_2 are right-aligned
2. T_1 only has nodes of type 1 and 0, and nodes of type 1 have a leaf as their left-most child
3. T_2 has only nodes of type 0

We can construct T_1 by starting with a right-aligned spine and adding nodes from left to right. First, we continue adding nodes of T_1 to the left-most leaf of the tree until we reach the left-most leaf of T_1. Then, to this tree, continue to add nodes of T_1 to the leaf second from the left (if any) and so on. When constructed in this way, if a node is added to an i^{th} leaf after a node is added to an j^{th} leaf, then $i \geq j$. We can then recover a positive word u corresponding to this construction process such that $\widetilde{\mathcal{T}(u)} = \widetilde{T_1}$, and by the fact that we always added nodes from left to right, u is monotone. Moreover, because we never add a node to the left-most child of a type 1 node, any type 1 generator $(1)_i$ in u can only be followed by a generator whose level is strictly greater than i. Similarly, constructing T_2 from left to right gives a corresponding word v such that $\widetilde{\mathcal{T}(v)} = \widetilde{T_2}$, and v is therefore a monotone positive

word with all generators of type 0. It follows that:

$$f = \psi^{-1}\left(\widetilde{\mathcal{T}(u)} \star \widetilde{\mathcal{T}(v)}^{-1}\right) = \psi^{-1}\big(\psi\big(uv^{-1}\big)\big) = uv^{-1}$$

and the monotone positive words u and v satisfy the requirements of the seminormal form. □

Definition (Normal form)**:** We will say a word w over the generators of $\mathcal{F}_{\tau_k}$ is in *normal form* if it is in seminormal form and additionally satisfies:

1. If w contains a subword of the form $(0)_i u (0)_i^{-1}$ for some word u, then $\mathcal{L}(u) \leq i + k$.
2. If w contains a subword of the form $(0)_i (1)_i (0)_{i+k+1} u (0)_{i+1}^{-1} (0)_i^{-1}$ then $\mathcal{L}(u) \leq i + 2k$

Before we give a proof for the existence of a normal form for each element of $\mathcal{F}_{\tau_k}$, we will establish a technical result:

Proposition 3: *If w is a monotone positive word in the generators of $\mathcal{F}_{\tau_k}$ of the form $w \equiv (n)_i u$ where $\mathcal{L}(u) \geq i + k + 1$, then the word $w' = \mathcal{D}(u)(n)_i$ represents the same element as w in $\mathcal{F}_{\tau_k}$.*

Proof. We first note that if $j > i + k + 1$, and thus $i < j - l$ then for any $0 \leq m, n \leq k$ we get the following equality using a relator of the first kind:

$$(m)_i (n)_j = \Big((n)_{j-k} (m)_i (n)_j^{-1} (m)_i^{-1}\Big)(m)_i (n)_j = (n)_{j-k} (m)_i$$

This means we can commute $(n)_i$ with any of the generators to its right in w, decreasing the level of the generator in the process. Doing this for every generator in the word u gives us our result. □

Corollary 9: *If w is a monotone positive word in the generators of $\mathcal{F}_{\tau_k}$ with $\mathcal{L}(w) = i + 1$, then $(n)_i^{-1} w = \mathcal{D}^{-1}(w)(n)_i^{-1}$ for any $0 \leq n \leq k$.*

Proof. We have:

$$(n)_i^{-1} w = (n)_i \mathcal{D}\big(\mathcal{D}^{-1}(w)\big)(n)_i (n)_i^{-1} = (n)_i^{-1} (n)_i \mathcal{D}^{-1}(w)(n)_i^{-1} = \mathcal{D}^{-1}(w)(n)_i^{-1}$$

□

Proposition 4: *Each element $f \in \mathcal{F}_{\tau_k}$ can be represented as a word in normal form*

Proof. We have already shown in Proposition 2 that any element $f \in \mathcal{F}_{\tau_k}$ can be represented by a word in seminormal form. We will show that if w is a seminormal form word for f that is not in normal form, then there exists another word w' that also represents f, is in seminormal form, and satisfies that the maximum level of generators in w' is strictly less than the maximum level of generators in w. It follows that f must be representable by a word in normal form.

Assume that w is a word in seminormal form representing f that is not in normal form. There are two cases, depending on which condition of the normal form w violates.

Case 1: w contains a subword of the form $(0)_i u (0)_i^{-1}$ with $\mathcal{L}(u) \geq i+k+1$.

In this case, w must be of the form $w \equiv (r_0(0)_i r_1)(l_0(0)_i l_1)^{-1}$ where r_0, r_1, l_0, l_1 are all monotone positive words, and $\mathcal{L}(r_1),\ \mathcal{L}(l_1) \geq i+k+1$.

By Proposition 3 this element is also represented by the word:

$$w' \equiv (r_0 \mathcal{D}(r_1)(0)_i)(l_0 \mathcal{D}(l_1)(0)_i)^{-1}$$

Then cancelling the type zero generators at the end we get:

$$w'' \equiv (r_0 \mathcal{D}(r_1))(l_0 \mathcal{D}(l_1))^{-1}$$

w'' is again in seminormal form, and the maximum level of its generators is k less than the maximum level of the generators in w.

Case 2: w contains a subword of the form $(0)_i (1)_i (0)_{i+k+1} u (0)_{i+1}^{-1} (0)_i^{-1}$ with $\mathcal{L}(u) \geq i+2k+1$.

In this case, w must be of the form $w \equiv \big(r_0(0)_i(1)_i(0)_{i+k+1} r_1\big)\big(l_0(0)_i(0)_{i+1} l_1\big)$ where r_0, l_0, r_1 and l_1 are all monotone positive words, and $\mathcal{L}(r_1)$, $\mathcal{L}(l_1) \geq i+2k+1$.

Our first step is to use a relation of the first kind to commute $(1)_i$ and $(0)_{i+k+1}$, giving:

$$w_1 \equiv \big(r_0(0)_i(0)_{i+1}(1)_i r_1\big)\big(l_0(0)_i(0)_{i+1} l_1\big)$$

We can then use a relation of the second kind: $(0)_i(0)_{i+1} = (1)_i^2$, twice, giving:

$$w_2 \equiv \big(r_0(1)_i^3 r_1\big)\big(l_0(0)_i(0)_{i+1} l_1\big) \qquad w_3 \equiv \big(r_0(1)_i(0)_i(0)_{i+1} r_1\big)\big(l_0(0)_i(0)_{i+1} l_1\big)$$

Using Proposition 3 (twice) then gives us:

$$w_4 \equiv \big(r_0(1)_i \mathcal{D}^2(r_1)(0)_i(0)_{i+1}\big)\big(l_0 \mathcal{D}^2(l_1)(0)_i(0)_{i+1}\big)$$

Finally, cancelling the type 0 generators at the ends gives us:

$$w_5 \equiv \big(r_0(1)_i \mathcal{D}^2(r_1)\big)\big(l_0 \mathcal{D}^2(l_1)\big)$$

This is a new word in seminormal form representing the same element as w with maximum generator level $2k$ less than that of w. □

Proposition 5: *Every element of $\mathcal{F}_{\tau_k}$ is represented by a unique normal form word over the generators.*

Proof. We have shown the existence of a normal form word for each element in Proposition 4, so all that remains to show is uniqueness.

To do so, we follow the approach taken in [Bur18] to produce a normal form for F and in [BNR18] to produce a normal form for $\mathcal{F}_{\tau_1}$.

We will prove uniqueness by contradiction. Assume that there exists some element which is represented by more than one normal form word. Then we can chose, among all the pairs of distinct normal form words representing the same element in $\mathcal{F}_{\tau_k}$, a pair (u, v) such that

sum of the lengths of u and v is minimal. We will assume for notational simplicity that the level of u is zero. The case for u of arbitrary level d can be proved by increasing the level of all generators by d. Our two words u and v are therefore of the form:

$$u \equiv ((0)_0^{a_0}(1)_0^{\epsilon_0} r_1)\Big((0)_0^{b_0} l_1\Big)^{-1} \qquad v \equiv ((0)_0^{c_0}(1)_0^{\eta_0} \rho_1)\Big((0)_0^{d_0} \lambda_1\Big)^{-1}$$

We can read the gradient at zero of the group element represented by these words. We know that the interval represented by a leaf of a k-tree has length given by τ_k^i where i is the height of the leaf. From the definition of our map H from pairs of k-trees to $\mathcal{F}_{\tau_k}$ we know that the word u is the image of a tree pair (T_1, T_2) where the left-most leaf of T_1 is hanging under $a_0 + 1$ type 0 nodes (the "+1" comes from the root node) and ϵ_0 type 1 nodes. Because the left-most child of a type 0 node has length 2, it follow that the height of this leaf node is given by $2(a_0 + 1) + \epsilon_0$. Similarly, the height of the left-most leaf of T_2 is given by $2(b_0 + 1) + \eta_0$. The element f represented by u therefore maps the interval $\left[0\ ,\ \tau_k^{2(a_0+1)+\epsilon_0}\right]$ to the interval $\left[0\ ,\ \tau_k^{2(b_0+1)}\right]$ and thus has gradient $\tau_k^{2(b_0-a_0)-\epsilon_0}$. The gradient of the element of $\mathcal{F}_{\tau_k}$ represented by v is then $\tau_k^{2(d_0-c_0)-\eta_0}$. The condition that these words represent the same element of $\mathcal{F}_{\tau_k}$ gives us the equation:

$$2(b_0 - a_0) - \epsilon_0 = 2(d_0 - c_0) - \eta_0$$

Because ϵ_0, $\eta_0 \in \{0, 1\}$, we must have that $\epsilon_0 = \eta_0$. It follows that $b_0 - a_0 = d_0 - c_0$, and also that we cannot have all of a_0, b_0, c_0 and d_0 equal zero, since it would then follow, by the fact that one of u, v is a word of level 0, that $\epsilon_0 = \eta_0 = 1$. This would mean we could cancel $(1)_0$ from the front of both u and v, obtaining a pair of distinct normal form words representing the same element in $\mathcal{F}_{\tau_k}$ with shorter total length than u and v.

We also know that one of a_0, c_0 must be zero, and one of b_0, d_0 must be zero, since otherwise we would be able to cancel a type 0 generator from the start or end of both u and v, and thus obtain a distinct pair of normal form words still representing the same element of shorter total length. We will assume that $c_0 = 0$ without loss of generality. It then follows that d_0 must also be zero, since if $b_0 = 0$ we have $-a_0 = d_0$ and a_0, d_0 are both positive and can't both be zero if c_0 and b_0 are already zero as well. It follows that $a_0 = b_0$. We can then rewrite our words u and v as:

$$u \equiv ((0)_0^{a_0}(1)_0^{\epsilon_0} r_1)((0)_0^{a_0} l_1)^{-1} \qquad v \equiv ((1)_0^{\epsilon_0} \rho_1)(\lambda_1)^{-1}$$

We will get our contradiction by considering the two cases for the value of ϵ_0 independently.

Case 1: $\epsilon_0 = 0$

In this case we can write u and v as:

$$u \equiv ((0)_0^{a_0} r_1)((0)_0^{a_0} l_1)^{-1} \qquad v \equiv (\rho_1)(\lambda_1)^{-1}$$

Conjugating both u and v by $(0)_0^{-a_0}$ gives another pair of words which represent the same element:

$$u' \equiv (0)_0^{-a_0} u (0)_0^{a_0} = r_1 l_1^{-1} \equiv u'', \qquad v' \equiv (0)_0^{-a_0} v (0)_0^{a_0} \equiv \big((0)_0^{-a_0}\rho_1\big)\big((0)_0^{-a_0}\lambda_1\big)^{-1}$$

Using Corollary 9 we can simplify the word v' as:

$$\begin{aligned} v' \equiv \big((0)_0^{-a_0}\rho_1\big)\big((0)_0^{-a_0}\lambda_1\big)^{-1} =& \big(\mathcal{D}^{-a_0}(\rho_1)(0)_0^{-a_0}\big)\big(\mathcal{D}^{-a_0}(\lambda_1)(0)_0^{-a_0}\big)^{-1} \\ =& \mathcal{D}^{-a_0}(\rho_1)\big(\mathcal{D}^{-a_0}(\lambda_1)\big)^{-1} \equiv v'' \end{aligned}$$

It follows that the normal form words u'' and v'' both represent the same element in $\mathcal{F}_{\tau_k}$ and have total length less than u and v. By our hypothesis, u'' and v'' must therefore be the same word. But clearly $\mathcal{L}(v'') \geq 1 + a_0 k \geq 1 + k$, and so $\mathcal{L}(u') = \mathcal{L}\big(r_1 l_1^{-1}\big) \geq 1 + k$, which violates condition one of the normal form for the word u. This gives us a contradiction.

Case 2: $\epsilon_0 = 1$

We proceed by a similar argument, we conjugate both u and v by $(0)_0^{-a_0}$ to get two new words representing the same element:

$$(0)_0^{-a_0} u (0)_0^{a_0} = (1)_0 r_1 l_1^{-1} \equiv u', \qquad v' \equiv (0)_0^{-a_0} v (0)_0^{a_0} \equiv \big((0)_0^{-a_0}(1)_0\rho_1\big)\big((0)_0^{-a_0}\lambda_1\big)^{-1}$$

We now make the observation that when we conjugate a word in normal form $w = (1)_0 r l^{-1}$ by $(0)_0^{-1}$ we can represent the resulting element of the group $\mathcal{F}_{\tau_k}$ by the normal form word $w' \equiv \big((1)_0(0)_{i+k}\mathcal{D}^2(r)\big)\big((0)_1\mathcal{D}^2(l)\big)^{-1}$ which is a word of the same length as $(0)_0^{-1} w (0)_0$. We do this by the following manipulation:

$$\begin{aligned} (0)_0^{-1} w (0)_0 &= \big((0)_0^{-1}(1)_0 r\big)\big((0)_0^{-1} l\big)^{-1} \\ &= \big((0)_0^{-1}(1)_0\big((0)_0(0)_1(0)_1^{-1}(0)_0^{-1}\big)r\big)\big((0)_0^{-1}\big((0)_0(0)_1(0)_1^{-1}(0)_0^{-1}\big)l\big)^{-1} \\ &= \big((0)_0^{-1}(1)_0(0)_0(0)_1\mathcal{D}^{-2}(r)(0)_1^{-1}(0)_0^{-1}\big)\big((0)_0^{-1}(0)_0(0)_1\mathcal{D}^{-2}(l)(0)_1^{-1}(0)_0^{-1}\big)^{-1} \\ &= \big((0)_0^{-1}(1)_0(0)_0(0)_1\mathcal{D}^{-2}(r)\big)\big((0)_1\mathcal{D}^{-2}(l)\big)^{-1} \\ &= \big((0)_0^{-1}(1)_0^3\mathcal{D}^{-2}(r)\big)\big((0)_1\mathcal{D}^{-2}(l)\big)^{-1} \\ &= \big((0)_0^{-1}(0)_0(0)_1(1)_0\mathcal{D}^{-2}(r)\big)\big((0)_1\mathcal{D}^{-2}(l)\big)^{-1} \\ &= \big((0)_1(1)_0\mathcal{D}^{-2}(r)\big)\big((0)_1\mathcal{D}^{-2}(l)\big)^{-1} \\ &= \big((1)_0(0)_{1+k}\mathcal{D}^{-2}(r)\big)\big((0)_1\mathcal{D}^{-2}(l)\big)^{-1} \end{aligned}$$

In particular, we have that $(0)_0^{-1} w (0)_0$ can be expressed as $\big((1)_0(0)_{1+k}r'\big)\big((0)_1 l'\big)^{-1}$ where $\mathcal{L}(r'), \mathcal{L}(l') \geq 2k + 1$.

We can repeatedly apply this rule to the word v' (which is just v conjugated $a_0 \neq 0$ times by $(0)_0^{-1}$) to see that v' has an expression as a normal word of the form:

$$v'' = \big((1)_0(0)_{1+k}\rho'\big)\big((0)_1\lambda'\big)^{-1}$$

where the length of v'' is equal to that of v', and therefore the sum of the lengths of u' and v'' is equal to the sum of the lengths of the words u and v.

But now we can cancel the $(1)_0$ from the front of both u' and v'' to get two normal form words:

$$\hat{u} \equiv r_1 l_1^{-1}, \qquad \hat{v} \equiv \big((0)_{1+k}\rho'\big)(0)_1{\lambda'}^{-1}$$

representing the same element in $\mathcal{F}_{\tau_k}$ with lower total length than the pair u, v, and which must therefore be the same word. It follows that the word u was of the form:

$$\big((0)_0^{a_0}(1)_0(0)_{1+k}\rho'\big)\big((0)_0^{a_0}(0)_1\lambda'\big)^{-1}$$

where $\mathcal{L}(\rho')$, $\mathcal{L}(\lambda') \geq 2k+1$. The word u was therefore not in normal form, since it violates condition 2, so we get a contradiction.

We conclude that there can be no two distinct normal form words representing the same element of $\mathcal{F}_{\tau_k}$, and thus such a normal form representation is unique. □

8. Connections between $\mathcal{F}_{\tau_k}$ and F_k

8.1 Embeddings and Amenability

We first state some facts about the groups F_k.

Lemma 41 ([BS14])**:** *Each group F_k has an infinite presentation given by:*

$$\left\langle x_i \; i \geq 0 \,|\, x_j x_i x_{j+k}^{-1} x_i^{-1} \; i < j \right\rangle$$

There are many proof of this result for the case F_1, for example in [Bur18] and [CFP96]. These proofs use representations of elements of Thompson's group by pairs of binary trees, and the result for F_k can be proven analogously considering instead pairs of $(k+1)$-ary trees.

Lemma 42: *The abelianisation of F_k is isomorphic to $\mathbb{Z}^{k+1}$*

There are many ways to show this. The most direct is to consider the finite presentations for the groups F_k given by Bieri and Strebel in [BS14] which consists of $k+1$ generators $\{x_0, \ldots, x_k\}$ and the relations:

$$\begin{aligned} (x_i \downarrow x_j) &= (x_0 \downarrow x_j) && (1 \leq i < j \leq p) \\ (x_0 x_i \downarrow x_j) &= \left(x_0^2 \downarrow x_j\right) && (1 \leq i, j \leq p,\ j \leq i+1) \\ \left(x_0^2 x_k \downarrow x_j\right) &= \left(x_0^3 \downarrow x_j\right) \end{aligned}$$

These are expressed in terms of our conjugation notation developed in 6. All these relations become trivial when the commutator relations are added, so the abelianisation of F_k is isomorphic to the free abelian group on $k+1$ generators: $\mathbb{Z}^k$.

Lemma 43: *Each element f of F_k has a unique representation as a word w over the generators $\{x_i \; i \geq 0\}$ of F_k satisfying:*

1. *$w = uv^{-1}$, where u and v are monotone positive words*
2. *If $x_i w' x_i^{-1}$ is a subword of w, then w' has level less than or equal to $i+k$*

This result is obtained by the proof given in [Bur18] for the normal form of Thompson's group.

Lemma 44: *Each group F_k contains a subgroup isomorphic to Thompson's group F_1.*

Proof. We define a group homomorphism $\phi : F_1 \to F_k$ by mapping each generator x_i of F_2 to the generator $x_{ki} \in F_k$. To check that this is a well-defined homomorphism, we need to check that the image of the relators in F_1 is the identity in F_k. The image of a relator $x_j x_i x_{j+1}^{-1} x_i^{-1}$ in F_1 for $i < j$ under ϕ is:

$$\phi\big(x_j x_i x_{j+1}^{-1} x_i^{-1}\big) = x_{kj} x_{ki} x_{kj+k}^{-1} x_{ki}^{-1}$$

which is a relator in F_k, and thus equal to the identity. To see that this homomorphism is injective, we use the fact that the map on the generators of F_1 induces a map on words over the generators which sends normal form words in F_1 to normal form words in F_k. If w is a monotone positive word in F_1, then $\phi(w)$ is clearly a monotone positive word in F_k, so if $w \equiv uv^{-1}$ is a normal form word in F_1, then $\phi(w) \equiv \phi(u)\phi(v)^{-1} \equiv \hat{u}\hat{v}^{-1}$ where $\hat{u}$ and $\hat{v}$ are monotone positive words. By the hypothesis that w is a normal form word in F_1, any subword $x_i w' x_i^{-1}$ of w for some $i \in \mathbb{Z}_{\geq 0}$ satisfies that $L(w') \leq i + 1$. It follows that the word $\hat{w} = \phi(w)$ satisfies that for any subword $x_{ki}\hat{w}'x_{ki}^{-1}$, $L(\hat{w}') \leq ki + k$, and $\hat{w}$ is therefore a normal form word in F_k.

The unique normal form word in F_k representing the identity is the empty word, so if $\phi(w) = \mathrm{id}_{F_k}$ for a normal form word w then w is the empty word in F_1, and therefore id_{F_1}.

It follows that $\phi : F_1 \to F_k$ is injective, and therefore its image is a subgroup of F_k isomorphic to F_1. □

Lemma 45: *Each group $\mathcal{F}_{\tau_k}$ contains a subgroup isomorphic to F_k.*

Proof. We can again resort to an argument based on the normal forms for each group. We define a map $\xi : F_k \to \mathcal{F}_{\tau_k}$ by sending a generator x_i of F_k to the generator $(0)_i$ of $\mathcal{F}_{\tau_k}$. This map is a well-defined homomorphism by the fact that each relator $x_j x_i x_{j+k}^{-1} x_i^{-1}$ on F_k is mapped by ξ to the relator $(0)_j (0)_i (0)_{j+k}^{-1} (0)_i^{-1}$ on $\mathcal{F}_{\tau_k}$. As was the case with our map ϕ, we see that the induced map on words over generators in F_k sends normal form words to normal form words over the generators in $\mathcal{F}_{\tau_k}$. As noted for ϕ, ξ clearly sends a word of the form uv^{-1} in F_k with u and v positive and monotone, to a word $\hat{w} = \hat{u}\hat{v}^{-1}$ over the type 0 generators of $\mathcal{F}_{\tau_k}$, where $\hat{u}$ and $\hat{v}$ are monotone and positive. Note that such a word $\hat{w}$ is therefore a word in seminormal form.

If we assume that w is a normal word in F_k, then it also satisfies the condition that is contains no subword of the form $x_i w' x_i^{-1}$, where $L(w') \leq i + k$. It follows that the image $\xi(w)$ is a word that contains no subword of the form $(0)_i \hat{w}' (0)_i^{-1}$ with $L(\hat{w}') \leq i + k$, which is precisely condition 1 from our definition of the normal form on $\mathcal{F}_{\tau_k}$. Condition 2 is automatically satisfied by the fact that the image $\hat{w}$ of w contains no generators of type 1.

Because the unique normal form for $\mathrm{id}_{\mathcal{F}_{\tau_k}}$ is the empty word, it follows that if $\xi(w) = \mathrm{id}_{\mathcal{F}_{\tau_k}}$, then w is the empty word, and so represents id_{F_k}. We conclude that $\xi : F_k \to \mathcal{F}_{\tau_k}$ is an injective group homomorphism, and therefore its image is a subgroup of $\mathcal{F}_{\tau_k}$ isomorphic to F_k. □

Corollary 10: *Each group $\mathcal{F}_{\tau_k}$ contains a subgroup isomorphic to Thompson's group F_1.*

Theorem ([CFP96])**:** *Thompson's group F_1 is not elementary amenable.*

A proof for this result was given in [CFP96] based on previous work by [Cho+80].

Corollary 11: *Each group $\mathcal{F}_{\tau_k}$ is not elementary amenable.*

Proof. This follows from the fact that subgroups of elementary amenable groups are elementary amenable. Thus, if $\mathcal{F}_{\tau_k}$ were elementary amenable, this would imply F_1 is elementary amenable, which is false. □

Corollary 12: *If $\mathcal{F}_{\tau_k}$ is amenable for any $k \in \mathbb{Z}_{\geq 1}$, then so is F_1.*

Proof. This follows from the fact that subgroups of amenable (discrete) groups are amenable. □

8.2 Unanswered Questions

One important property of Thompson's group F_1 that we have not investigated here is whether the commutator subgroup of each F_k is simple. This result was proven in [BNR18] for the group $\mathcal{F}_{\tau_1}$ by a method exploiting a property called transitivity which $\mathcal{F}_{\tau_1}$ shares with F_1 but not with F_k for $k > 1$.

Another question prompted by the embedding of F_k in $\mathcal{F}_{\tau_k}$ is when $\mathcal{F}_{\tau_n}$ is embeddable in $\mathcal{F}_{\tau_m}$ of F_m for $n \neq m$. In particular, the question of whether $\mathcal{F}_{\tau_1}$ is embedded in F_1 has generated some interest and remains open.

Two natural extensions of an investigation into the groups $\mathcal{F}_{\tau_k}$ are the investigation of the related groups $T(1, \mathbb{Z}[\tau_k], \langle \tau_k \rangle)$ and $G(1, \mathbb{Z}[\tau_k], \langle \tau_k \rangle)$ and an investigation into the groups $F(1, \mathbb{Z}[\tau], \langle \tau \rangle)$ for other algebraic integers.

Bibliography

[Bel07] James Belk. "Thompson's group F". In: *arXiv preprint arXiv:0708.3609* (2007).

[BNR18] José Burillo, Brita Nucinkis, and Lawrence Reeves. "An Irrational-slope Thompson's Group". In: *arXiv preprint arXiv:1806.00108* (2018).

[Bro87] Kenneth S. Brown. "Finiteness properties of groups". In: *Journal of Pure and Applied Algebra* 44.1 (Feb. 1, 1987), pp. 45–75. ISSN: 0022-4049. DOI: 10.1016/0022-4049(87)90015-6. URL: http://www.sciencedirect.com/science/article/pii/0022404987900156 (visited on 09/12/2017).

[BS14] Robert Bieri and Ralph Strebel. "On groups of PL-homeomorphisms of the real line". In: *arXiv preprint arXiv:1411.2868* (2014).

[BS85] Matthew G Brin and Craig C Squier. "Groups of piecewise linear homeomorphisms of the real line". In: *Inventiones mathematicae* 79.3 (1985), pp. 485–498.

[Bur18] José Burillo. *Introduction to Thompson's Group F*. 2018. URL: https://mat-web.upc.edu/people/pep.burillo/book_en.php.

[CFP96] James W Cannon, William J Floyd, and Walter R Parry. "Introductory notes on Richard Thompson's groups". In: *Enseignement Mathématique* 42 (1996), pp. 215–256.

[Cho+80] Ching Chou et al. "Elementary amenable groups". In: *Illinois Journal of Mathematics* 24.3 (1980), pp. 396–407.

[Cle00] Sean Cleary. "Regular subdivision in $\mathbf{Z}\left(\frac{1+\sqrt{5}}{2}\right)$". In: *Illinois Journal of Mathematics* 44.3 (2000), pp. 453–464.

[FL10] Marcelo Fiore and Tom Leinster. "An abstract characterization of Thompsons group F". In: *Semigroup Forum*. Vol. 80. 2. Springer. 2010, pp. 325–340.

[GG06] Ross Geoghegan and Fernando Guzmán. "Associativity and Thompson's group". In: *Contemporary Mathematics* 394 (2006), pp. 113–136.

[Gri98] Rostislav I Grigorchuk. "An example of a finitely presented amenable group not belonging to the class EG". In: *Sbornik: Mathematics* 189.1 (1998), p. 75.

[OS03] Alexander Yu Olshanskii and Mark V Sapir. "Non-amenable finitely presented torsion-by-cyclic groups". In: *Publications Mathématiques de l'Institut des Hautes Études Scientifiques* 96.1 (2003), pp. 43–169.

[Ste92] Melanie Stein. "Groups of Piecewise Linear Homeomorphisms". In: *Transactions of the American Mathematical Society* 332 (1992).