# AN ALGORITHM TO OBTAIN REVEALING PAIRS OF ELEMENTS OF THOMPSON'S GROUP $V$

Contents



## 1. Introduction

## 2. Alphabets, Words, and Prefixes

Let $n \in \mathbb{Z}_{>1}$ be given, and let $X_n$ be the set $X_n := \{0, 1, \ldots, n-1\}$. We will refer to $X_n$ as *the (n-ary) alphabet*.

Let $X_n^*$ denote the set of all finite strings of elements in $X_n$, including the empty string $\varepsilon$, called *words*. We also use $X_n^\omega$ to represent the set of infinite strings over $X_n$ (each such string can be thought of as a function from the limit ordinal $\omega$ to $X_n$).

For any words $w \in X_n^*$ and $v \in X^\omega \cup X_n^*$, write $wv$ as the *concatenation* of two strings (the string created by starting with the string $w$ and then continuing with the string $v$). We say that $w \in X_n^*$ is a *prefix* of $v \in X^\omega \cup X_n^*$ if there exists some $y \in X^w \cup X_n^*$ such that $v = wy$, denoted $w \preceq v$ (or $w \prec v$ if it is known that $y \neq \varepsilon$). We refer to $\preceq$ as the *prefix relation (on $X_n^*$)*. We will also denote by $\mid w \mid$ the length of any given word $w$ (e.g. $\mid \varepsilon \mid = 0$ and $\mid 0110 \mid = 4$).

If we give $X_n$ the discrete topology, then $X^\omega$ is an infinite product space and this is a standard definition of a Cantor space $\mathfrak{C}_n$. If $w$ is a word in $X_n^*$ then $w\mathfrak{C}_n$ shall denote all the elements in $\mathfrak{C}_n$ which have prefix $w$, and we refer to the set $w\mathfrak{C}_n$ as the *cone (of $\mathfrak{C}_n$) at $w$*. As is well known, for each $w \in X_n^*$ the set $w\mathfrak{C}_n$ is clopen in $\mathfrak{C}_n$, and the set

$\mathscr{B}_n := \{w\mathfrak{C}_n \mid w \in X_n^*\}$ is a clopen basis for the topology of the space $\mathfrak{C}_n$.

If two words $w, v \in X_n^*$ satisfy both $w \not\preceq v$ and $v \not\preceq w$, then they are *incomparable (with respect to the prefix relation $\preceq$ on $X_n^*$)*, and we write $w \perp v$. A set of words which are all pairwise incomparable is called an *antichain*. If an antichain $A$ is finite and maximal, it is said to be *complete* (i.e. $B$ is not an antichain for all $A \subsetneq B$). From now on we will assume that complete antichains are ordered lexicographically.

## 3. Rooted $n$-ary Trees

We now introduce the use of trees to visualise the concepts introduced in the previous section. Throughout this paper, we will use upper case calligraphic letters to denote trees.

**Definition 3.1** (The tree $\mathcal{T}_n$)**.** We shall denote by $\mathcal{T}_n$ the infinite rooted $n$-ary tree with vertex set $V(\mathcal{T}_n) := X_n$ and edge set $E(\mathcal{T}_n) := \{\{v, w\} \mid v, w \in X^*, x \in X, w = vx\}$.

It is immediate that the graph $\mathcal{T}_n$ is a tree, as if $w$ is a word appearing in an edge $\{v, w\}$ where $|v| < |w|$, then $v$ is uniquely determined by $w$. Also, as $V(\mathcal{T}_n) = X_n^*$, we might refer to vertices of $\mathcal{T}_n$ as words, below.

If we visualise $\mathcal{T}_n$ as "descending downward", with a "root" $\varepsilon$ drawn at the top. For example, we may visualise $\mathcal{T}_2$ as in Figure 1 below.

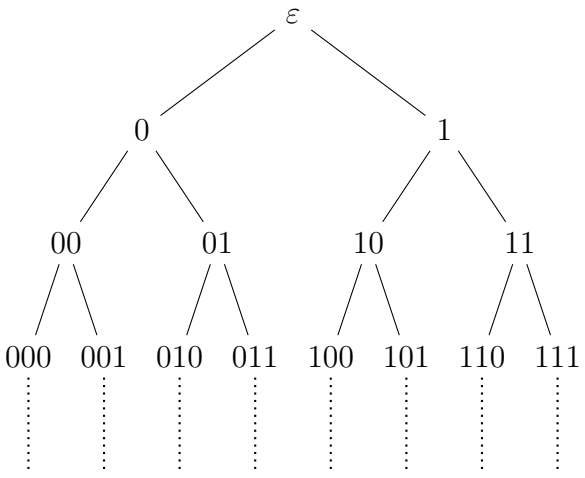


Figure 1. The infinite tree $\mathcal{T}$

Note that with this visualisation, the relation $\preceq$ determines an *ancestor–descendant* relation on the vertices $X_n^*$. Suppose $t$, $u$, $v$, and $w \in X_n^*$. If $v \preceq w$ we say that $v$ is an *ancestor of* $w$ and that $w$ *is a descendant*

*of* $v$. Further, if $v$ is an ancestor of $w$ and $|w| = |v| - 1$ we say that $w$ is a *child of* $v$ and that $v$ *is the parent of* $w$ (note that any vertex has at most one parent). If $u$ and $v$ are both children of $t \in X_n^*$, then we say that $u$ and $v$ are *siblings*.

**Definition 3.2** (Caret)**.** For any word $w$ in $\mathcal{T}_n$, define $\widehat{w}$ to be the sub-tree of $\mathcal{T}_n$ spanned by the set of vertices $\{w\} \cup \{wx \mid x \in X_n\}$. We call $\widehat{w}$ the *(n-ary) caret rooted at* $w$.

When we refer to a sub-tree in $\mathcal{T}_n$, we will always mean a tree which admits a decomposition as a union of $n$-ary carets. We define such sub-trees below.

**Definition 3.3** (Full subtree)**.** We say that a sub-tree $\mathcal{D}$ of $\mathcal{T}_n$ is a *full* sub-tree of $\mathcal{T}_n$ if there is a set $C(\mathcal{D}) \subset X_n^*$ so that for any edge $e$ of $\mathcal{D}$ there is $w_e \in C(\mathcal{D})$ so that $e$ is an edge of $\widehat{w_e}$, and, for each $w \in C(\mathcal{D})$ the caret $\widehat{w}$ is contained in $\mathcal{D}$.

**Definition 3.4** (Root of sub-tree)**.** Let $\mathcal{D}$ be any full non-empty sub-tree of $\mathcal{T}_n$. Assign the *root* of $\mathcal{D}$ be the vertex in $\mathcal{D}$ with the shortest length as a word in $X_n^*$.

The above definition is well-defined since a sub-tree of $\mathcal{T}_n$ must be connected, and having two distinct shortest words would imply some common descendant of these, but each vertex has at most one parent.

**Definition 3.5** (Leaf set)**.** For any full non-empty sub-tree $\mathcal{D}$ of $\mathcal{T}_n$, define the *leaf set* $L_{\mathcal{D}}$ to be the set of all vertices of $\mathcal{D}$ with degree one.

Note that for $w \in X_2^*$ the root of $\widehat{w}$ is $w$ and we have $L_{\widehat{w}} = \{w0, w1\}$. We may visualise this as in Figure 2.

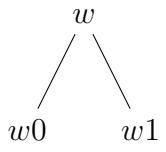


FIGURE 2. A caret $\widehat{w}$ within $\mathcal{T}_2$

**Definition 3.6** (Lower sub-tree)**.** Let $w$ be any vertex of $\mathcal{T}_n$. Define its *lower sub-tree* $\mathcal{T}_{[w]}$ to be the sub-tree of $\mathcal{T}_n$ spanned by the vertex set:

$$V(\mathcal{T}_{[w]}) := \{v \mid v \in \mathcal{T} \text{ and } w \preceq v\}.$$

**Remark 3.7.** It follows from this definition that for two incomparable vertices $w, v$ in $\mathcal{T}_n$, we have that $V(\mathcal{T}_{[w]}) \cap V(\mathcal{T}_{[v]}) = \varnothing$.

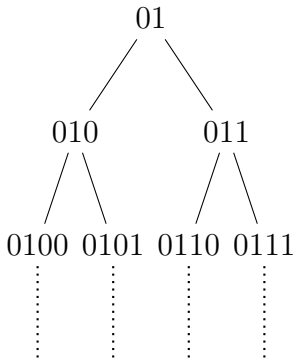


FIGURE 3. The infinite lower sub-tree $\mathcal{T}_{[01]}$

**Definition 3.8.** Let $w \in X_n^*$. We set the *boundary of* $\mathcal{T}_{[w]}$ to be the set $w\mathfrak{C}_n \subset \mathfrak{C}_n$.

**Definition 3.9** (Unions and intersections)**.** Let $\mathcal{R}$ and $\mathcal{S}$ be full sub-trees of $\mathcal{T}_n$. We define the following as subgraphs of $\mathcal{T}_n$ spanned by these vertex sets:

- $\mathcal{R} \cup \mathcal{S}$ has vertex set $V(\mathcal{R}) \cup V(\mathcal{S})$
- $\mathcal{R} \cap \mathcal{S}$ has vertex set $V(\mathcal{R}) \cap V(\mathcal{S})$

**Definition 3.10** (Subtraction)**.** Let $\mathcal{R}$ and $\mathcal{S}$ be any full sub-trees of $\mathcal{T}_n$, and let $w$ be the root of $\mathcal{S}$. If $\mathcal{R}$ and $\mathcal{S}$ satisfy the condition that $L_{\mathcal{S}} \subseteq L_{\mathcal{R}}$, then define $\mathcal{R} - \mathcal{S}$ to be the sub-tree of $\mathcal{T}$ spanned by the vertex set:

$$V(\mathcal{R} - \mathcal{S}) := (V(\mathcal{R}) \setminus V(\mathcal{S})) \cup \{w\}.$$

**Remark 3.11.** The subtraction of full sub-trees can be thought of as taking a *difference of carets* of the two trees. In other words, $\mathcal{R} - \mathcal{S}$ contains all the carets in $\mathcal{R}$ which are not present in $\mathcal{S}$.

**Lemma 3.12.** $\mathcal{R}$ and $\mathcal{S}$ be any full sub-trees of $\mathcal{T}_n$ so that $L_{\mathcal{S}} \subseteq L_{\mathcal{R}}$. Then, $\mathcal{R} - \mathcal{S}$ is the full subtree so that $C(\mathcal{R} - \mathcal{S}) = C(\mathcal{R}) \backslash C(\mathcal{S})$.

**Definition 3.13** (Corresponding sub-tree)**.** Let $D = \{d_1, d_2, ..., d_k\}$ be a complete antichain of $X_n^*$, for some $k \in \mathbb{Z}_{>0}$. We call $\mathcal{D}$ the *corresponding sub-tree* of $D$, and define it by the following:

$$\mathcal{D} := \mathcal{T} - \bigcup_{1 \leq i \leq k} \mathcal{T}_{d_i}.$$

Note that there is a natural bijection between complete antichains of $X_n^*$ and their corresponding trees.

## 4. Definitions of $F$, $T$, and $V$

In this section we will show how pairs of complete antichains of equal order can be used to define homeomorphisms of Cantor space. We will then use this to define Thompson's Group $V$, $T$, and $F$.

**Definition 4.1** (Prefix exchange map)**.** Let $D$ be a complete antichain, $R \subseteq \mathfrak{C}$, and $\sigma : D \to R$ a map. For each element $\rho \in \mathfrak{C}$ there exists unique $d \in D$ and $w \in \mathfrak{C}$ such that $\rho = dw$. Let $\varphi : \mathfrak{C} \to \mathfrak{C}$ be a function such that

$$\rho\varphi = dw\varphi = d\sigma w.$$

We call $\varphi$ the *prefix exchange map* with respect to $\sigma$.

We will justify the use of *map* in this section, as at this point we only know that $\varphi$ is a set function and not a continous function between topological spaces.

**Theorem 4.2.** *Prefix exchange maps are continuous.*

*Proof.* Let $D$ be a complete antichain, $R$ an antichain, $\sigma : D \to R$ a map, and $\varphi$ the prefix exchange map with respect to $\sigma$. We will show that $\varphi$ is continuous by showing that the preimage of every open set in $\mathfrak{C}$ is open. We will do this by showing the pre-image of any cone is a union of cones.

Let $w \in \{0, 1, \ldots, n-1\}^*$. We will show that $w\mathfrak{C}_n\varphi^{-1}$ is a union of cones.

For each $d \in D$, we will consider $d\sigma$. If $d\sigma$ is incomparable to $w$, then the cone under $d$ will not be in $w\mathfrak{C}_n\varphi^{-1}$. If $w \preceq d\sigma$, then $d\mathfrak{C}$ is in $w\mathfrak{C}_n\varphi^{-1}$. Finally, if $d\sigma \prec w$, then we can find a $\Gamma \in X_n^*$ such that $d\sigma\Gamma = w$, and then $d\mathfrak{C} \cap w\mathfrak{C}_n\varphi^{-1} = d\Gamma\mathfrak{C}$. In each of these cases, let the resulting open set be called $U_{w,d}$.

Now, it follows by construction that $w\mathfrak{C}_n\varphi^{-1} = \bigcup_{d\in D} U_{w,d}$, which is the union of open sets and hence open.

□

Let $\sigma$ be a bijection between any two complete antichains $D$ and $R$. We'll denote by $\Phi_\sigma$ the prefix exchange map with respect to $\sigma$.

**Lemma 4.3.** If $\sigma : D \to R$ is a bijection between two complete antichains, then $\Phi_\sigma$ is a bijection.

*Proof.* Let $\sigma : D \to R$ be a bijection between two complete antichains. To prove $\Phi_\sigma$ is bijective we will show that $\Phi_\sigma$ is surjective and injective.

Let $\rho \in \mathfrak{C}_n$. Since $R$ is a complete antichain, there exists $r \in R$ and a $\omega \in \mathfrak{C}_n$ such that $\rho = r\omega$. As $\sigma$ is a bijection, there exists $d \in D$ such

that $d\sigma = r$. Hence $\rho = r\omega = d\sigma\omega = d\omega\Phi_\sigma$. As $\rho$ was arbitrary, $\Phi_\sigma$ is surjective.

Now, let $\mu, \nu \in \mathfrak{C}_n$ be two distinct elements in the domain. There exists $d_1, d_2 \in D$ and $\omega_1, \omega_2 \in \mathfrak{C}_n$ such that $\mu = d_1\omega_1$ and $\nu = d_2\omega_2$. If $d_1 = d_2$, then $\omega_1 \neq \omega_2$ and $d_1\sigma = d_2\sigma$, so $d_1\sigma\omega_1 \neq d_2\sigma\omega_2 \Rightarrow \mu\Phi_\sigma \neq \nu\Phi_\sigma$. If $d_1 \neq d_2$, then $d_1\sigma \neq d_2\sigma$ and hence $d_1\sigma\omega_1 \neq d_2\sigma\omega_2 \Rightarrow \mu\Phi_\sigma \neq \nu\Phi_\sigma$. Therefore $\Phi_\sigma$ is also injective.

□

Since $\Phi_\sigma$ is a bijection, it has a well-defined inverse $\Phi_\sigma^{-1}$.

**Lemma 4.4.** If $\sigma : D \to R$ is a bijection between two complete antichains, then:

- $\Phi_\sigma^{-1}$ is a prefix exchange map;
- $\Phi_\sigma$ is a homeomorphism; and
- $\Phi_\sigma^{-1} = \Phi_{\sigma^{-1}}$.

*Proof.* Let $\rho \in \mathfrak{C}_n$ with $\rho = r\omega$ where $r \in R$ and $\omega \in \mathfrak{C}_n$. Then

$$\rho\Phi_\sigma^{-1} = r\omega\Phi_\sigma^{-1} = r\sigma^{-1}\omega.$$

Therefore $\Phi_\sigma^{-1} = \Phi_{\sigma^{-1}}$ is a prefix exchange map. All three points follow from this. □

Let $V$ be the set of all prefix exchange maps with respect to bijections between two complete antichains. We will show that $V$ forms a group.

The following lemma is a useful tool to prove that $V$ is closed under composition of maps.

**Lemma 4.5.** Let $n \in \mathbb{N}$ be given. Every open set $U$ of $\mathfrak{C}$ can be written as a union of clopen cones $w_i\mathfrak{C}$ such that $\forall i \in I$, $|w_i| > n$ where $w_i \in X_n^*$ and $I$ is some index set.

*Proof.* Recall that $\mathscr{B} = \{w\mathfrak{C} : w \in X^*\}$ is a basis for Cantor space. Hence we can find an index set $I$ and words $w_i \in X_n^*$ such that $U = \bigcup_{i\in I} w_i\mathfrak{C}$. If $|w_i| \leq n$, let $A_i = \{w_i v : v \in X^* \text{ and } |v| = n - |w_i| + 1\}$ or $A_i = \{w_i\}$ otherwise. For each $i$, $w_i\mathfrak{C} = \bigcup_{a\in A_i} a\mathfrak{C}$. Therefore

$$U = \bigcup_{i\in I} \bigcup_{a\in A_i} a\mathfrak{C}.$$

This proves the lemma. □

**Lemma 4.6.** Composition under $V$ is closed.

We will prove this using an algorithm.

**Algorithm 4.7.** Let $\Omega$ and $\Psi$ be two prefix exchange maps with respect to bijections $\pi : D \to R$ and $\tau : D' \to R'$, where $D$, $R$, $D'$, and $R'$ are all complete antichains. If $R = D'$, then $\Omega\Psi$ is a prefix exchange map with respect to the bijection $\pi\tau$, and hence is in $V$. Assume that $R \neq D'$. Let $\lambda$ be a map which we will define in the algorithm.

Step 1: If $R \setminus D' = \emptyset$, go to Step 4:. Otherwise, choose a $r \in R \setminus D'$. If $r \prec d'$ for some $d \in D'$, continue to Step 2:. Otherwise, $d' \prec r$ for a unique $d' \in D'$, and continue to Step 3:.

Step 2: Let $A = \{d' \in D' : r \prec d'\}$ and $R'' = A \cup R \setminus \{r\}$. Let $B = \{r\pi^{-1}w : d' = rw \text{ for some } d' \in A \text{ and } w \in X^*\}$ and $D'' = D \cup B \setminus \{r\pi^{-1}\}$. For each $b \in B$, we can find a $w_b \in X_n^*$ such that $r\pi^{-1}w_b = b$. Define $b\lambda = rw_b\tau$ for each $b \in B$.

Set $R'' = R$ and $D'' = D'$. Return to Step 1.

Step 3: Let $A = \{r \in R : d' \prec r\}$ and $D'' = D' \cup A \setminus \{d'\}$. Let $B = \{d'\tau w : r = d'w \text{ for some } w \in X^*\}$. Let $R'' = R' \cup B \setminus \{d'\tau\}$. For each $a \in A$, we can find a $w_a \in X_n^*$ such that $r = d'w_a$. Define $a\pi^{-1}\lambda = d'\tau w_a$ for each $a \in A$.

Set $R'' = R'$ and $D'' = D'$. Return to Step 1.

Step 4: If $D' \setminus R = \emptyset$, go to Step 7:. Otherwise, choose a $d' \in D' \setminus R$. If $d' \prec r$ for some $r \in R$, continue to Step 5:. Otherwise, $r \prec d'$ for a unique $r \in R$, and continue to Step 6:.

Step 5: Let $A = \{r \in R : d' \prec r\}$ and $D'' = D' \cup A \setminus \{d'\}$. Let $B = \{d'\tau w : r = d'w \text{ for some } w \in X^*\}$. Let $R'' = R' \cup B \setminus \{d'\tau\}$. For each $a \in A$, we can find a $w_a \in X_n^*$ such that $r = d'w_a$. Define $a\pi^{-1}\lambda = d'\tau w_a$ for each $a \in A$.

Set $R'' = R'$ and $D'' = D'$. Return to Step 1.

Step 6: Let $A = \{d' \in D : r \prec d'\}$ and $R'' = A \cup R \setminus \{r\}$. Let $B = \{r\pi{-}1w : d' = rw \text{ for some } d' \in A \text{ and } w \in X^*\}$ and $D'' = D \cup B \setminus \{r\pi^{-1}\}$. For each $b \in B$, we can find a $w_b \in X_n^*$ such that $r\pi^{-1}w_b = b$. Define $b\lambda rw_b\tau$ for each $b \in B$.

Set $R'' = R$ and $D'' = D'$. Return to Step 1.

Step 7: For each $x \in R \cap D'$, define $x\pi^{-1}\lambda = x\tau$. This is the end of the algorithm.

**Lemma 4.8.** Let $\Omega$, $\Psi$, $D$, $R$, $D'$, $R'$, $\pi$, and $\tau$ be like they are in algorithm 4.7. The sets generated in this algorithm are complete antichains and the map $\lambda$ is a bijection. Furthermore, the prefix exchange map with respect to $\lambda$ is equivalent to the homeomorphism $\Omega\Psi$.

*Proof.* Later... □

We can now prove lemma 4.6.

*Proof of lemma 4.6.* From Algorithm 4.7 and lemma 4.8, for every prefix exchange map $\Omega$ and $\Psi$ with respect to bijections $\pi : D \to R$ and $\tau : D' \to R'$ respectively, we can find complete antichains $Q$ and $W$ and a bijection $\lambda : Q \to W$ such that $\Omega\Psi$ is a prefix exchange map with respect to $\lambda$. □

**Lemma 4.9.** The set $V$ is a group under composition of functions.

*Proof.* From lemma 4.6, $V$ is closed under composition of functions. The prefix exchange map with respect to the identity mapping of a complete antichain is the identity homeomorphism. The composition of functions is associative.

Let $\Theta$ be a prefix exchange map in $V$. An identical argument as in lemma 4.4 shows that that $\Theta^{-1}$ is a prefix exchange map. Therefore every element in $V$ has an inverse.

This concludes the proof that $V$ is a group.

□

We are now in a position to define the Thompson's Groups.

**Definition 4.10** (Thompson's Groups)**.** Let $\Psi$ be a prefix exchange map with respect to the bijection $\pi : D \to R$, where $D$ and $R$ are complete antichains. Suppose that $D = \{d_1, d_2, ..., d_n\}$ and $R = \{r_1, r_2, ..., r_n\}$ are in lexicographic order. Let $F$ and $T$ be subsets of $V$, defined below.

- The group $V$ is called *Thompson's Group V*.
- If $\forall\ 1 \leq i \leq n$, $d_i\pi = r_i$, then let $\Psi \in F$. All elements in $F$ are of this form. We call $F$ *Thompson's Group F*.
- If $\forall\ 1 \leq i \leq n$, $d_i\pi = r_{i \mod n}$, then let $\Psi \in T$. All elements in $T$ are of this form. We call $T$ *Thompson's Group T*.

From the definition we can see that $F \subseteq T \subseteq V$. As expected from their names, it can be shown that $F$ and $T$ are subgroups of $V$.

## 5. Tree Pairs

We will now give a way to represent elements of $V$ using objects called tree pairs.

**Definition 5.1** (Tree Pair)**.** Let $D$ and $R$ be two complete antichains of equal order and $\sigma$ a bijection between them. We define the triple $(\mathcal{D}, \mathcal{R}, \sigma)$ to be a *tree pair* where $\mathcal{D}$ and $\mathcal{R}$ are the corresponding trees of $D$ and $R$ respectively.

For a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$, the sets $L_{\mathcal{D}}$ and $L_{\mathcal{R}}$ are complete antichains. Since $\sigma$ is a bijection between them, this defines a prefix exchange map with respect to $\sigma$. In Corollary **??**, we showed that this prefix exchange

map is a homeomorphism of Cantor space and from Definition 4.10, an element of Thompson's Group $V$. Therefore, every tree pair represents an element of $V$. Conversely, if an element of $V$, say $\Phi$, is a prefix exchange map with respect to $\sigma : D \to R$, then the tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ is a representation of this element.

**Remark 5.2.** There is not a unique tree pair for each element in $V$, but every tree pair represents a unique element of $V$, as we will show in the next subsection.

5.3. **Altering tree pairs.** In this subsection, we will show different ways of altering tree pairs so they represent the same element of $V$.

**Definition 5.4** (Pair of dangling carets)**.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing some element $\alpha \in V$. Consider some carets $\widehat{a}$ and $\widehat{b}$ such that $L_{\widehat{a}} \subseteq L_{\mathcal{D}}$ and $L_{\widehat{b}} \subseteq L_{\mathcal{R}}$. We call $(\widehat{a}, \widehat{b})$ a *pair of dangling carets* of $(\mathcal{D}, \mathcal{R}, \sigma)$ if $(a0)\sigma = b0$ and $(a1)\sigma = b1$.

**Lemma 5.5.** Consider a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha \in V$ with a pair of dangling carets $(\widehat{a}, \widehat{b})$. Then the tree pair $(\mathcal{D} - \widehat{a}, \mathcal{R} - \widehat{b}, \sigma')$, where

$$d\sigma' = \begin{cases} d\sigma & \text{if } d \neq a \\ b & \text{if } d = a \end{cases}$$

for all $d \in L_{\mathcal{D}}$, represents the same element $\alpha$.

*Proof.* Let $\Psi$ be the prefix exchange map with respect to $\sigma : L_{\mathcal{D}} \to L_{\mathcal{R}}$ and $\Omega$ be the prefix exchange map with respect to $\sigma : L_{\mathcal{D}'} \to L_{\mathcal{R}'}$. Consider an element $\rho \in \mathfrak{C}$. If $a \not\prec \rho$, then $\rho\Psi = \rho\Omega$. Otherwise, either $a0 \prec \rho$ or $a1 \prec \rho$. Suppose $a0 \prec \rho$, and let $\omega \in \mathfrak{C}$ such that $\rho = a0\omega$. Then

$$\rho\Psi = a0\omega\Psi = a0\sigma\omega = b0\omega = a\sigma'0\omega = \rho\Omega$$

and a similar argument shows the same if $a1 \prec \rho$. Hence $\Psi = \Omega$. □

**Algorithm 5.6** (Minimisation)**.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing an element $\alpha$ in $V$.

Step 1: Let $\mathcal{D}' := \mathcal{D}$, $\mathcal{R}' := \mathcal{R}$, and $\sigma := \sigma'$.

Step 2: If the tree pair $(\mathcal{D}', \mathcal{R}', \sigma')$ has no dangling carets, then terminate. Otherwise, choose dangling carets $(\widehat{a}, \widehat{b})$ and go to Step 2.

Step 3: Let $\mathcal{D}'' := \mathcal{D}' - \widehat{a}$, $\mathcal{R}'' := \mathcal{R}' - \widehat{b}$. Define $\sigma'' : \mathcal{D}' \to \mathcal{R}'$ by:

$$d\sigma'' = \begin{cases} d\sigma & \text{if } d \neq a \\ b & \text{if } d = a \end{cases}$$

for all $d \in \mathcal{D}''$. Let $\mathcal{D}' = \mathcal{D}''$, $\mathcal{R}' = \mathcal{R}''$, and $\sigma' = \sigma''$ and return to Step 2.

We call the tree pair $(\mathcal{D}', \mathcal{R}', \sigma')$ a *reduced tree pair* of $(\mathcal{D}, \mathcal{R}, \sigma)$.

Since there are finite edges in a any tree pair, Algorithm 5.6 must terminate.

**Remark 5.7.** It can be shown that there is a unique reduced tree pair for every tree pair, and hence the choice of dangling carets in Step 2 of Algorithm 5.6 does not matter. CITE EWA.

**Lemma 5.8.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing an element $\alpha \in V$. Then the reduced tree pair $(\mathcal{D}', \mathcal{R}', \sigma')$ of $(\mathcal{D}, \mathcal{R}, \sigma)$ represents the same element $\alpha$.

*Proof.* This follows from repeated application of Lemma 5.5. □

We can also add dangling carets to tree pairs, which will be important later.

**Definition 5.9** (Adding a caret)**.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing an element $\alpha \in V$ and consider $d \in L_{\mathcal{D}}$. Let $\mathcal{D}' = \mathcal{D} \cup \widehat{d}$, $\mathcal{R}' = \mathcal{R} \cup \widehat{d\sigma}$, and define $\sigma'$ to be the map

$$l\sigma' = \begin{cases} l\sigma & \text{if } l \notin \{d0, d1\} \\ r0 & \text{if } l = d0 \\ r1 & \text{if } l = d1 \end{cases}$$

for all $l \in L_{\mathcal{D}'}$. We say that we've *added a caret to* $d$.

**Lemma 5.10.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing an element $\alpha \in V$ and consider $d \in L_{\mathcal{D}}$. The tree pair $(\mathcal{D}', \mathcal{R}', \sigma')$ resulting from adding a caret to $d$ as in Definition 5.9 represents the same element $\alpha$.

*Proof.* From the definition, $(\widehat{d}, \widehat{d\sigma'})$ is a dangling caret of $(\mathcal{D}', \mathcal{R}', \sigma')$. From Lemma 5.5, this represents the same element as $(\mathcal{D}, \mathcal{R}, \sigma)$. □

All of the above justifies Remark 5.2.

## 6. Chains

In this section we will define chains which will allow us to partition all the leaves in a tree pair.

**Definition 6.1** (Chain)**.** Consider $w \in X^*$, $\alpha \in V$, and a continuous subinterval $L \subseteq \mathbb{Z}$. Then the $L$ *chain at* $w$ is the sequence $(w\alpha^i)_{i \in L}$.

**Definition 6.2** (Leaf chain)**.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair representing an element $\alpha \in V$. Consider $l \in L_{\mathcal{D}}$. There exists a maximal $k \in \mathbb{Z}_{>0}$ such that

- for $0 \leq i < k$, $l\alpha^i \in L_{\mathcal{D}}$;
- for $0 < j \leq k$, $l\alpha^j \in L_{\mathcal{R}}$;
- for $0 \leq s, t < k$ and $s \neq t$, $l\alpha^s \neq l\alpha^t$.

Let $L = \{0, 1, ..., k\}$. We define the *leaf chain at* $l$ to be the $L$ chain at $l$.

**Definition 6.3** (Complete leaf chain)**.** Consider the leaf chain $(l\alpha^i)_{i=0}^k$ for some tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$. We call this chain a *complete leaf chain* if either $l \in L_{\mathcal{D}} \setminus L_{\mathcal{R}}$ and $l \in L_{\mathcal{R}} \setminus L_{\mathcal{D}}$, or $l = l\alpha^k$.

**Remark 6.4.** We will refer to leaf chains which are not complete as *partial leaf chains*.

We will apply an algorithm to classify complete leaf chains into exactly one of five types.

**Algorithm 6.5** (Classifying complete leaf chains)**.** Consider a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha$ in $V$ and let $(l\alpha^i)_{i=0}^k$ be a complete leaf chain for this tree pair.

Step 1: If $l = l\alpha^k$ then we call it a *P-chain*, for *"periodic"* and terminate. Otherwise, continue to Step 2.

Step 2: If $l \prec l\alpha^k$ then we call it an *A-chain*, for *"attractor"* and terminate. Otherwise, continue to Step 3.

Step 3: If $l\alpha^k \prec l$ then we call it an *R-chain* for *"repeller"* and terminate. Otherwise, continue to Step 4.

Step 4: If $l \prec r$ for $r \in L_{\mathcal{R}} \setminus L_{\mathcal{D}}$ or $l\alpha^k \prec d$ for $d \in L_{\mathcal{D}} \setminus L_{\mathcal{R}}$, then we call it an *F-chain* for *"fragmenter"* and terminate. Otherwise, continue to Step 5.

Step 5: Call $(l\alpha)_{i=0}^k$ a *S-chain* for *"source-sink"*.

From this algorithm, every complete leaf chain is classified as exactly one chain type. We give names to the elements in a complete leaf chain:

**Definition 6.6** (Leaves of complete leaf chains)**.** Let $(l\alpha^i)_{i=0}^k$ be a complete leaf chain for a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$. We call

- $l$ a *repeller* of period $k$ and $l\alpha^k$ a *range of repulsion* if $(l\alpha^i)_{i=0}^k$ is an $R$-chain;
- $l\alpha^k$ an *attractor* of period $k$ and $l$ a *domain of attraction* if $(l\alpha^i)_{i=0}^k$ is an $A$-chain;
- $l$ a *source* and $l\alpha^k$ is a *sink* if $(l\alpha^i)_{i=0}^k$ is a $S$-chain;
- $l$ a *start-fragmenter* if $(l\alpha^i)_{i=0}^k$ is a $F$-chain and $l \prec r$ for $r \in L_{\mathcal{R}} \setminus L_{\mathcal{D}}$;

- $l\alpha^k$ a *end-fragmenter* if $(l\alpha^i)_{i=0}^k$ is a $F$-chain and $l\alpha^k \prec d$ for $d \in L_{\mathcal{D}} \setminus L_{\mathcal{R}}$;
- leaves in $L_{\mathcal{D}} \cap L_{\mathcal{R}}$ *neutral leaves*.

## 7. Revealing Pairs

It is possible to alter a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ to $(\mathcal{D}', \mathcal{R}', \sigma')$ such that there are no complete leaf chains of type $F$ for $(\mathcal{D}', \mathcal{R}', \sigma')$. We give these types of altered trees a name:

**Definition 7.1** (Revealing pair)**.** A tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ is said to be a *revealing pair* if all complete leaf chains are of type $P$, $R$, $A$, or $S$.

Our algorithm in Section 8 outputs a revealing pair for any given tree pair. To understand the algorithm and form a termination algorithm for it, we will give a visual representation for the types of leaves defined in Definition 6.6, using tree pairs and connected components.

**Definition 7.2** (Connected components)**.** Let $S$ be a full subtree of $\mathcal{T}_n$. We define an equivalence relation $\sim$ on $S$, such that for all $a, b \in X^*$ $a \sim b$ if and only if there exists a path from $a$ to $b$ in $S$. This is the standard definition of connectedness in graphs.

We can define the connected components of $S$ as the subtrees of $\mathcal{T}_n$ spanned by set of words in each equivalence class.

**Definition 7.3** (Complementary Components)**.** Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair of an element of V.

A complementary component of $(\mathcal{D}, \mathcal{R}, \sigma)$ is a connected component of $\mathcal{D} - \mathcal{R}$ or $\mathcal{R} - \mathcal{D}$.

**Lemma 7.4.** Each connected component of a revealing pair contains exactly one attractor or repeller.

*Proof.* Let $(\mathcal{D}, \mathcal{R}, \sigma)$ be a tree pair of an element of V represented by $\alpha$.

Proceeding by contradiction, assume that $\mathcal{D} - \mathcal{R}$ contains a connected component with atleast two distinct repellers. We shall name these $l_1, l_2$ such that $(l_1\alpha^i)_{i=0}^l$ and $(l_1\alpha^i)_{i=0}^l$ are R-chains. This is the case if and only if $l_1 \prec l_1\alpha^k$ and $l_2 \prec l_2\alpha^k$, but $l_1\alpha^k$ and $l_2\alpha^k$ must be leaves in $\mathcal{R}$, and the unique leaf in $\mathcal{R}$ such that it is the prefix of the leaves in our connected component is the root of the connected component. This implies $l_1\alpha^k = l_2\alpha^k$ which gives us $l_1 = l_2$ by invertibility of $\alpha^k$. By symmetry, as a repeller for $\alpha$ is an attractor for $\alpha^{-1}$, a connected component of a revealing tree pair cannot contain more than one repeller or attractor.

Assuming that a connected component in $\mathcal{D} - \mathcal{R}$ does not contain a repeller. The root of this connected component is a leaf in $\mathcal{R}$, we name this $l$, which means there exists a leaf $l'$ and chain $(l'\alpha^i)_{i=0}^{k}$ such that $l$ is in this chain. $l$ is the root of a connected component, so it is not a sink. Therefore, this chain must be an $F - chain$, which contradicts that the pair is revealing.

□

## 8. Revealing Pair Algorithm

In this section, we present an algorithm to construct a revealing tree pair for an element of $V$ ($T$ or $F$), given a tree pair for the element.

**Definition 8.1** (Expansion along a leaf chain)**.** Consider a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha$ in $V$, and let $(l\alpha^i)_{i=0}^k$ be a leaf chain in the tree pair. Define the *expansion* of the tree along this leaf chain to be the result of adding carets (by Algorithm 5.9) to the tree at $l\alpha^i$ for all $0 \leq i < k$.

**Remark 8.2** (Result of expansion on a leaf chain.)**.** Consider a leaf chain, $(l\alpha^i)_{i=0}^k$ for a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha$ in $V$.

On expansion of this leaf chain. We can construct two leaf chains $(l0\alpha^i)_{i=0}^k$ and $(l1\alpha^i)_{i=0}^k$. As these are a result of adding carets to leaves in $L_{\mathcal{D}}$ and $L_{\mathcal{R}}$, these two chains will still be leaf chains.

Note that if the original leaf chain is complete, the two chains obtained after expanding along this chain may not be complete.

**Algorithm 8.3** (Slowest rollings)**.** Consider a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha$ in $V$. Maintain a stack of words.

Step 1: For every element of $L_{\mathcal{D}} \setminus L_{\mathcal{R}}$, generate a leaf chain starting at that element and ending when it reaches a non-neutral leaf.

Classify all of these chains following Algorithm 6.5.

If there are no $F$-chains then the tree pair is already revealing by Definition 7.1 and we are done. Otherwise go to Step 2.

Step 2: Apply Algorithm 5.6 to the tree pair to obtain a reduced tree pair, then go to Step 3.

Step 3: For every element of $L_{\mathcal{D}} \setminus L_{\mathcal{R}}$, generate a leaf chain starting at that element and ending when it reaches a non-neutral leaf.

Classify all of these chains following Algorithm 6.5.

Pick the first $F$-chain in lexicographical order of the first leaf, add this chain to the stack, then go to Step 4.

If there are none, then the tree pair is already revealing by Definition 7.1 and we are done.

Step 4: If the stack is empty, go to Step 3. If the stack is not empty, remove the last leaf chain added to the stack, call this leaf chain $(l\alpha^i)_{i=0}^k$ Expand along this chain using Algorithm 8.1.

Consider the leaf chains $(l0\alpha^i)_{i=0}^k$ and $(l1\alpha^i)_{i=0}^k$. Note that these leaf chains may not be complete.

If they are complete, we classify them by Algorithm 6.5 and if they are $F$-chains, we add them to the stack in lexicographical order. Repeat this step until the stack is empty.

**Remark 8.4** (Fragmentation and complimentary components)**.** Consider an $F$-chain, $(l\alpha^i)_{i=0}^k$.

Observe that the start-fragmenter or end-fragmenter is the root of a connected component in $\mathcal{R}-\mathcal{D}$ or $\mathcal{D}-\mathcal{R}$ respectively. This follows from the definitions $F$-chains and connected components.

We say that a complimentary component fragments, if the root of that complimentary component is a start-fragmenter or end-fragmenter.

Our algorithm, in expanding $F$-chains, acts towards removing this fragmentation, by adding carets under the root of the fragmenting complimentary component, thus removing these carets from $\mathcal{R}-\mathcal{D}$ or $\mathcal{D}-\mathcal{R}$ respectively.

In Step 3 of the algorithm, we choose a fragmenting complimentary component to resolve. Expanding along this chain may result in additional fragmenting complimentary components but they are added to the stack for resolution.

We only return to this Step 3 if the stack is empty, which means that the carets of the fragmenting complimentary component are entirely removed from complimentary components of the tree. Thus a fragmenting complimentary component is removed from the complimentary components of the tree.

**Definition 8.5** (Complementary Carets)**.** A complementary caret is a caret which is a sub-tree of $\mathcal{D}-\mathcal{R}$ or $\mathcal{R}-\mathcal{D}$.

**Lemma 8.6** (Each loop of Step 4 of Algorithm 4 either finds an attractor/repeller or decreases number of complementary carets)**.** UNDER CONSTRUCTION: Expanding along an **F** chain in $(\mathcal{D},\mathcal{R},\sigma)$, $(w\sigma^i)_{i=0}^k$ can only possibly add complementary carets at $w\in D$ or $w\sigma^k\in R$. Expanding at $w\sigma^i$ corresponds to adding a caret to $w\sigma^i\in\mathcal{D}$ and $w\sigma^{i+1}\in R$, but for all $w\sigma^i\neq w\sigma^k$ we also expand at $w\sigma^{i+1}\in D$, and so the added caret at $w\sigma^{i+1}\in\mathcal{D}$ is not complementary, for $i=0,\dots,k-1$. Similarly, the added caret at $w\sigma^i\in\mathcal{R}$ is not complementary for $i=1,\dots,k$. Thus, the only possible complementary carets added are the caret added at $w$ and the caret added at $w\sigma^{k+1}$ (which comes from expanding at $w\sigma^k$).

As $(w\sigma^i)_{i=0}^k$ is an **F** chain $w\sigma^k\prec d$, where $d\in L_{\mathcal{D}}\setminus w$, or $w\prec r$, where $d\in L_{\mathcal{R}}\setminus w\sigma^k$. Suppose first that $w\prec r$, where $r\in L_{\mathcal{D}}\setminus w\sigma^k$. As $w\prec r$, the caret rooted at $w$ including the words $w0$ and $w1$ is in $\mathcal{R}$, and this caret is complementary. However, adding a caret at $w\in D$ means this new $\mathcal{D}'$ includes $w0$ and $w1$, and so the caret at $w$ in $\mathcal{R}$ is no longer complementary. Thus even if the caret added at $w\sigma^k$ is complementary, the number of complementary carets cannot increase. If $w\sigma^k\prec d$, where $d\in L_{\mathcal{D}}\setminus w$, a similar argument holds.

Remarkish Thing? Each loop of step 4 of the algorithm corresponds to

Each loop of Step 4 continues until the stack is empty, meaning there are no **F** chains

**Lemma 8.7** (The length of each Step 4 from Algorithm 8.3 is bounded)**.** Consider a tree pair $(\mathcal{D}, \mathcal{R}, \sigma)$ representing an element $\alpha$ in $V$.

Suppose Algorithm 8.3 is being applied to this tree pair, and suppose that the chain $(l_0, ..., l_k)$ is the **F** chain chosen in Step 3.

- Let $a :=$ number of elements $d \in L_{\mathcal{D}} \setminus l_0$ which satisfy $l_k \preceq d$.
- Let $b :=$ number of elements $r \in L_{\mathcal{R}} \setminus l_k$ which satisfy $l_0 \preceq r$.

Then the number of times Step 4 is applied before returning to Step 3 is bounded by $a + b$.