Simple Rigs Hold Fast

T.R.I.E.
Kris Coward*, D.R. Toliver*

1 Introduction

An important use of computational systems is updating the state of an object while preserving some set of invariants. That object might be a file, a row in a database, or perhaps an entry in a distributed system. Its invariants may place limits on its relationships with other objects, and generally include maintaining a unique identity across updates.

The system maintaining these invariants is responsible for not *equivocating*[1], or presenting conflicting versions of an object. We call this absence of equivocation *integrity*.

Today the system providing integrity to an object also manages its state. Indeed, it has long been assumed that this must be the case, and that only the system managing an object's state can provide it with integrity.

We show that this assumption is wrong. Objects can maintain the integrity of one system while having their state fully managed by another.

We begin by examining prior work, and then define the basic structures of rigging: twists, lines, and hitches. We use these to construct rigs.

A rig is a cryptographic data structure that provides integrity-at-a-distance. Expressing an object in this structure allows its source of truth to be transferred while retaining its source of integrity.

This guarantee holds despite the integrity source being completely oblivious about the object, and can be verified locally without network calls. The result is that an object whose state is managed on one machine, like an untrusted laptop, can have the same integrity as a highly trusted server.

We define the rig property of holding fast, which is a strong form of causal connection between an object (i.e. a line) and an integrity provider (also a line).

We present guilds, which are sets of rigs, and introduce the property of support that some guilds have. Supportiveness is a uniqueness property: a rig

*TODAQ

[1] An equivocation in an object management system is much like an inconsistency in a logic system. Has this object upheld its invariants? The question is effectively meaningless in the presence of equivocation, because the answer can be yes, and also no.

from a supportive guild is proof that no other rig in that guild holds a conflicting version of that object fast to that integrity provider's line.

We introduce a very restrictive guild, G_H, containing only single hitch rigs, and show that G_H is supportive.

We present two basic operations on rigs, splicing and lashing, and prove that rigs are closed under them. We introduce a guild transformer, $\Psi(G)$, and prove that it preserves supportiveness.

We define $G_\uparrow$, the simplest guild that is closed under the splicing and lashing operations.

Finally, we show that $G_\uparrow$ is supportive.

2 Prior Work

The data structures described in this paper utilize idealized one-way functions, which provide a deterministic, fixed-size output from a given input, and are easy to compute, but impossible to invert, collide, or correlate.

Although these ideal one-way functions do not exist, their desired properties are sufficiently closely approximated by cryptographic hash functions [1] for our purposes. We acknowledge this by referring to these one-way functions, and their outputs, as hashes, but use the ideal functions for the sake of simplifying statements such as saying that a structure is unique, rather than unique up to hash collisions.

There is a causal/chronological relation between the input and output of a hash. Taking h to be a hash function, if y is some message incorporating $h(x)$, and $h(y)$ is the hash of that message, then $h(x)$ must be known before $h(y)$ is computed. This relation is used heavily in log management [4], and forms a strict partial order [5] (i.e. transitive, antisymmetric, and non-reflexive). We call this relationship *causal precedence*, and denote it as $x \prec\!\!\!\prec y$.

The trie data structure described by René de la Briandais [2] and Edward Fredkin [3], when prepared with hashes in the manner described by Ralph Merkle [7], provides an elegant way to fingerprint key-value data structures, with concise proofs of membership that uniquely associate a key and its value.

Rigging implementations employ these concise proofs for operational efficacy, and when we speak of *tries* it is this form to which we refer. Further, when we speak of taking the hash of a trie, or including it in another structure, the value can be taken to be its root node when this is sufficient to verify the proofs just described.

We do not directly consider the nature of cryptographic devices for restricting updates, but note that there are multiple mechanisms by which this can be accomplished, such as signatures based on public key ciphers like RSA [8], and one-time signatures like those designed by Leslie Lamport [6].

3 Twists

Rigs are made of *twists*. A twist $\mathtt{y}$ is a structure containing the following hash references:

- $\mathtt{p}(\mathtt{y})$, its *previous* twist, arranges twists sequentially into lines;
- $\mathtt{t}(\mathtt{y})$, its *tether* twist, establishes relations between lines; and
- $r(\mathtt{y})$, its *rigging* trie, holds key-value pairs that complete relations between lines.

As a concrete data structure y is merely a concatenation of hashes[2]. However the selector functions above return twists, not hashes, allowing statements like $\mathtt{x} = \mathtt{p}(\mathtt{y})$ to be made. This is standard object/pointer ambiguity, which we can disambiguate when needed by saying $h(\mathtt{x}) = h(\mathtt{p}(\mathtt{y}))$.

Similarly, we treat $\mathtt{p}$, $\mathtt{t}$, and r as functions on the space of twists, even though the non-injectivity of hash functions technically makes them relations rather than functions[3].

A twist $\mathtt{x}$ is a *predecessor* of another twist $\mathtt{y}$ (denoted $\mathtt{x} \prec \mathtt{y}$) if and only if:

- $\mathtt{x} = \mathtt{p}(\mathtt{y})$ (the hash $h(\mathtt{x})$ is equal to $\mathtt{y}$'s previous twist hash), or
- $\exists \mathtt{z} : \mathtt{z} \prec \mathtt{y}, \mathtt{x} = \mathtt{p}(\mathtt{z})$ (there is a twist $\mathtt{z}$ which is a predecessor of $\mathtt{y}$, and which has $\mathtt{x}$ as its previous).

Note that $\mathtt{x} \prec \mathtt{y}$ implies $\mathtt{x} \prec\!\!\prec \mathtt{y}$, due to the inclusion of each twist's direct predecessor's hash in its own structure and the recursive construction of the predecessor relation.

A twist $\mathtt{y}$ is a *successor* of $\mathtt{x}$ (denoted $\mathtt{y} \succ \mathtt{x}$) if and only if $\mathtt{x}$ is a predecessor of $\mathtt{y}$.

When $\mathtt{x} = \mathtt{p}(\mathtt{y})$, we may also say that $\mathtt{x}$ is the *direct predecessor* of $\mathtt{y}$, or that $\mathtt{y}$ is a *direct successor* of $\mathtt{x}$. In reference to the previous field in each twist, we also refer to a twist's direct predecessor as its *previous*.

Further, if $\mathtt{x} \prec \mathtt{y}$ or $\mathtt{x} = \mathtt{y}$, we denote this relation as $\mathtt{x} \preccurlyeq \mathtt{y}$ (alternately $\mathtt{y} \succcurlyeq \mathtt{x}$).

It is worth noting that the relation $\prec$ is a partial order on the set of twists. Moreover, because each twist has only one previous, this partial order has the structure of a tree.

We classify twists based whether their tether field is null[4].

A *fast twist* is tethered to the value of its tether field. The twist is fastened, or made fast, by this tethering.

[2] Implementation details can be found in "Rigging Specifications".

[3] This simplification can also be treated as an assumption that our ideal one-way function h is perfectly collision resistant and therefore injective.

[4] Mathematically we can think of this as being the empty hash. Implementations use an out-of-band signal, the null hash algorithm byte, as described in Rigging Specifications.

A *loose twist* has a null tether, and is not fastened to anything.

To work with fast twists efficiently we introduce a number of functions that take twists and produce fast twists.

We take ${}^*\mathtt{p}(\mathtt{x})$, the *fast previous* of $\mathtt{x}$, to be the fast twist which most directly precedes $\mathtt{x}$.

$${}^*\mathtt{p}(\mathtt{x}) = \begin{cases} \mathtt{p}(\mathtt{x}), & \text{if } \mathtt{p}(\mathtt{x}) \text{ is fast} \\ {}^*\mathtt{p}(\mathtt{p}(\mathtt{x})) & \text{otherwise} \end{cases}$$

It is trivial to see that $\mathtt{y} = {}^*\mathtt{p}(\mathtt{x})$ iff $\mathtt{y}$ is fast, and $\mathtt{y} \prec \mathtt{x}$, and $\mathtt{y} \prec \mathtt{z} \prec \mathtt{x}$ implies $\mathtt{z}$ is loose.

Similarly, we take the *fast tether* ${}^*\mathtt{t}(\mathtt{x})$ to be the most directly preceding or equal fast twist to the tether of $\mathtt{x}$.

$${}^*\mathtt{t}(\mathtt{x}) = \begin{cases} \mathtt{t}(\mathtt{x}), & \text{if } \mathtt{t}(\mathtt{x}) \text{ is fast} \\ {}^*\mathtt{p}(\mathtt{t}(\mathtt{x})) & \text{otherwise} \end{cases}$$

4 Lines

Twists link together into lines.

Definition. *A* line *is a sequence of consecutive twists* $[\mathtt{c}_0, ..., \mathtt{c}_n]$*, where* $\mathtt{c}_i = \mathtt{p}(\mathtt{c}_{i+1})$ *for each* i*.*

Between twists in a line, the relation $\mathtt{c} \preccurlyeq \mathtt{d}$ is a linear order[5]. Lines contain all their *sublines*, in the obvious way.

Definition. *A line* $\mathtt{A}$ *is called an* enveloping line *for a collection of lines* $\{\mathtt{A}_i\}_{i=0}^n$ *if* $\mathtt{A}$ *contains every twist in each* $\mathtt{A}_i$*.*

Definition. *The lines in a collection with an enveloping line are called* aligned*, denoted* $\asymp \{\mathtt{A}_i\}_{i=0}^n$ *or* $\mathtt{A} \asymp \mathtt{B}$*.*

If a pair of twists $\mathtt{x}$ and $\mathtt{y}$ are comparable by the predecessor relation (i.e $\mathtt{x} \preccurlyeq \mathtt{y}$ or $\mathtt{y} \preccurlyeq \mathtt{x}$), then the single-twist lines $[\mathtt{x}]$ and $[\mathtt{y}]$ are necessarily aligned. Consequently, we use the same notation $\mathtt{x} \asymp \mathtt{y}$ to denote when twists are aligned.

This allows us to describe the tree order structure on the predecessor relation as providing that if $\mathtt{x} \preccurlyeq \mathtt{z}$ and $\mathtt{y} \preccurlyeq \mathtt{z}$, then $\mathtt{x} \asymp \mathtt{y}$.

On the other hand, given for example twists $\mathtt{x}$, $\mathtt{y}$, and $\mathtt{z}$ with $\mathtt{z} = \mathtt{p}(\mathtt{x})$ and $\mathtt{z} = \mathtt{p}(\mathtt{y})$, then $\mathtt{z} \preccurlyeq \mathtt{x}$ together with $\mathtt{z} \preccurlyeq \mathtt{y}$ is clearly not sufficient to imply $\mathtt{x} \asymp \mathtt{y}$. When two lines share a common twist but are not aligned, we say that they are *misaligned*.

Another consequence of the concept of alignment is that if $\mathtt{x} \asymp \mathtt{y}$ and $\mathtt{x} \not\prec \mathtt{y}$, then it follows (from asymmetry and non-reflexivity of $\not\prec$) that $\mathtt{x} \prec \mathtt{y}$.

[5]This linearity is part of the reason consecutive twists are referred to as a "line". Rigs provide a way to maintain that linearity in the forward direction.

We introduce a restricted notion of the length of a line, where the line being measured begins on a fast twist, ends on a fast twist, and the measurement of interest is only concerned with fast twists.

Definition. *A* fast line *is a line that begins and ends with fast twists.*

Definition. *The* length *of a fast line is one less than the number of fast twists it contains.*

Roughly speaking, fast twists are treated as fenceposts, and the length of the line is the number of sections of fence. Equivalently, we can say that a fast line $\mathtt{X} = [({}^*\mathtt{p})^n(\mathtt{x}), ..., \mathtt{x}]$ has a length $l(\mathtt{X}) = n$.

Length is undefined on lines that are not fast. A line can be expanded or contracted to make it fast.

5 Hitches

The basic unit of rigging is called a *hitch*. The operation that a hitch performs between a pair of lines is called *hitching*.

A hitch H involves 5 twists

- $\mathtt{f}(H)$, the *fastener*
- $\mathtt{l}(H)$, the *lead*
- $\mathtt{m}(H)$, the *meet*
- $\mathtt{h}(H)$, the *hoist*
- $\mathtt{n}(H)$, the *post*[6]

The relations between the twists in a hitch are as follows

- $\mathtt{f}(H) = \mathtt{t}(\mathtt{l}(H))$ (the fastener is the tether of the lead)
- $\mathtt{l}(H) = {}^*\mathtt{p}(\mathtt{m}(H))$ (the lead is the fast predecessor of the meet)
- $\mathtt{m}(H) = {}^*\mathtt{p}(\mathtt{n}(H))$ (the meet is the fast predecessor of the post)
- $r(\mathtt{h}(H)) : h(\mathtt{l}(H)) \mapsto h(\mathtt{m}(H))$ (the hoist's rigging trie associates lead:meet as key:value)
- the hoist is *the first* successor of the fastener to associate a value with the lead in its rigging trie
- $r(\mathtt{n}(H)) : h(\mathtt{l}(H)) \mapsto h(\mathtt{h}(H))$ (the post's rigging trie associates lead:hoist as key:value)

[6]While the other twists are identified by their first letter, `p` is already in use, so we use `n` because it comes sequentially after `l` and `m`. The logic of this sequencing comes from the definition of footline below.

Note that while a direct inclusion of $h(\mathtt{l}(H)) : h(\mathtt{m}(H))$ in the hoist's rigging trie is specified here, the only required property of the inclusion mechanism is that it can prove the unique value associated with a given key[7].

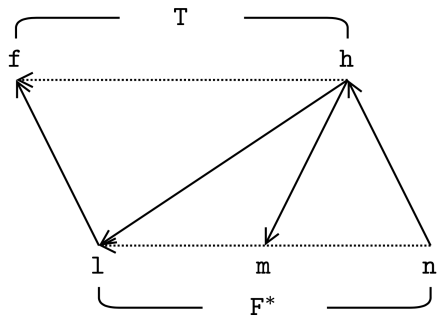
We also define a *half-hitch*, which contains all the twists of a hitch except the post, and all relations of a hitch except the two involving the post.

With the mechanics of the hitch described, we can now turn to the lines that are naturally required to exist in a hitch H based on the required relations

- $\mathtt{f}(H) \prec \mathtt{h}(H)$
- $\mathtt{l}(H) = {}^*\mathtt{p}(\mathtt{m}(H))$ (implying $\mathtt{l}(H) \prec \mathtt{m}(H)$)
- $\mathtt{m}(H) = {}^*\mathtt{p}(\mathtt{n}(H))$ (implying $\mathtt{l}(H) \prec \mathtt{m}(H) \prec \mathtt{n}(H)$)

Thus the lines that necessarily exist in every hitch are

- the *topline* $\mathtt{T}(H) = [\mathtt{f}(H), ..., \mathtt{h}(H)]$
- the *footline* $\mathtt{F}(H) = [\mathtt{l}(H), ..., \mathtt{m}(H)]$
- *extended footline* $\mathtt{F}^*(H) = [\mathtt{l}(H), ..., \mathtt{m}(H), ..., \mathtt{n}(H)]$



A basic hitch. Solid arrows represent direct reference: `l` *includes the hash of* `f`*. Dotted arrows indicate predecessors:* $\mathtt{l} \prec \mathtt{m} \prec \mathtt{n}$*. Time flows from left to right*[8]*.*

We say that `A` is *hitched to* `B` iff there is a hitch H in which $\mathtt{A} = \mathtt{F}(H)$ and $\mathtt{B} = \mathtt{T}(H)$. That is, `A` is the footline of H and `B` is the topline of H.

There is a strict chronological order imposed on the twists of a hitch through hash inclusion:

$$\mathtt{f}(H) \prec\!\!\prec \mathtt{l}(H) \prec\!\!\prec \mathtt{m}(H) \prec\!\!\prec \mathtt{h}(H) \prec\!\!\prec \mathtt{n}(H) \tag{1}$$

[7]This unique value might be null. Implementations employ a more complex inclusion mechanism, which has additional desirable properties.

[8]Unlike an arrow.

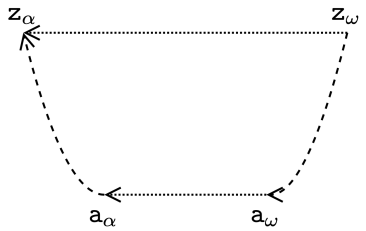
6 Rigs and Guilds

Hitches serve as the fundamental unit of rigging, and also make up the smallest rigs. Rigs generalize hitches and allow us to provide the same guarantees over larger structures.

First we introduce a causal relation between lines.

Definition. *We say that* $\mathtt{A}$ holds fast *to* $\mathtt{Z}$ *if* $\mathtt{A} = [\mathtt{a}_\alpha, ..., \mathtt{a}_\omega]$ *and* $\mathtt{Z} = [\mathtt{z}_\alpha, ..., \mathtt{z}_\omega]$ *are lines satisfying the causal relation* $\mathtt{z}_\alpha \prec\!\!\prec \mathtt{a}_\alpha \prec\!\!\prec \mathtt{a}_\omega \prec\!\!\prec \mathtt{z}_\omega$.

It is worth noting that because $\prec\!\!\prec$ is transitive, and implied by $\preccurlyeq$, the relation of holding fast is transitive.



A holding fast to Z. The dashed arrows represent chained references, and are curved to reinforce their ascent up the tethers on the past side and descent down the rigging tries on the future side.

Definition. *A* rig $\mathfrak{R}$ *is a collection of twists, consisting of a* corkline $\mathtt{C}(\mathfrak{R})$*, a* leadline $\mathtt{L}(\mathfrak{R})$*, and additional twists sufficient to demonstrate that* $\mathtt{L}(\mathfrak{R})$ *holds fast to* $\mathtt{C}(\mathfrak{R})$.

A rig contains the twists for all lines within it, including its corkline and leadline[9].

A *guild* is a set of rigs. Guilds provide constraints on acceptable rigs, and can therefore place limits on the leadlines that can be rigged to a corkline. Particularly constrained guilds have uniqueness properties which allow for useful constructions.

Our first guild, G_H, as described in Section 8, is the set of rigs which consist of a single half-hitch, where the footline of the half-hitch is the leadline of the rig, and the topline of the half-hitch is the corkline of the rig.

A half-hitch has the particular constraint that a given corkline can only support a single version of a given leadline. In other words, the uniqueness of successors on the corkline guarantees the uniqueness of successors of the lead on the leadline. This is proven below as Theorem 1.

[9]Astute readers may be wondering why this paper is called Simple Rigs Hold Fast, when in fact all rigs hold fast by definition. A linguistic shift occurred over the course of developing this work: the title came about at a time when the rigs in $G_\uparrow$ were called simple rigs, and what is now called support was referred to as holding fast. As a result the meaning of the title is a bit anticlimactic compared to "$G_\uparrow$ is Supportive", but it does have a nice ring to it.

Note that the uniqueness of predecessors enables a rig's corkline to be completely reconstructed from its last twist. As such this last twist of the corkline guarantees the uniqueness of the succession of twists from the lead forward along the leadline.

We say that a line $\mathtt{A}$ is *supported by* a twist $\mathtt{z}$ if there is a rig $\mathfrak{R}$ whose leadline is $\mathtt{A}$ and whose corkline has $\mathtt{z}$ as its last twist. In this situation, we may also say that $\mathtt{A}$ is $\mathtt{z}$-supported, and that $\mathtt{Z}$ supports $\mathtt{A}$, and that this support comes *via* or *through* $\mathfrak{R}$.

7 Support

We generalize this just-described uniqueness relation by elevating it to a property of guilds. The supportiveness of a guild allows its rigs to provide meaningful support.

We begin by defining a pair of relations between rigs. The first relation is alignment, which rigs inherit from their leadlines.

Definition. *Two rigs* $\mathfrak{R}$ *and* $\mathfrak{R}'$ *are* aligned *iff* $\mathtt{L}(\mathfrak{R}) \asymp \mathtt{L}(\mathfrak{R}')$.

The second relation is disjointness.

Definition. *Two rigs* $\mathfrak{R}_0$ *and* $\mathfrak{R}_1$ *are* disjoint *if either of the following hold:*

- $\mathtt{C}(\mathfrak{R}) \not\asymp \mathtt{C}(\mathfrak{R}')$ *(the corklines are not aligned), or*
- $\mathtt{L}(\mathfrak{R}) \cap \mathtt{L}(\mathfrak{R}') = \varnothing$ *(the leadlines have no twists in common).*

Given a pair of rigs $\mathfrak{R}$ and $\mathfrak{R}'$ we adopt the following notation conventions in this paper

- $\mathtt{L}(\mathfrak{R}) = \mathtt{A} = [\mathtt{a}_\alpha, ..., \mathtt{a}_\omega]$
- $\mathtt{L}(\mathfrak{R}') = \mathtt{A}' = [\mathtt{a}'_\alpha, ..., \mathtt{a}'_\omega]$
- $\mathtt{C}(\mathfrak{R}) = \mathtt{Z} = [\mathtt{z}_\alpha, ..., \mathtt{z}_\omega]$
- $\mathtt{C}(\mathfrak{R}') = \mathtt{Z}' = [\mathtt{z}'_\alpha, ..., \mathtt{z}'_\omega]$

Additionally, when $\mathtt{Z}$ and $\mathtt{Z}'$ are aligned, we take $\mathtt{z}$ to denote the more recent of $\mathtt{z}_\omega$ and $\mathtt{z}'_\omega$.

Extending this notation we describe $\mathfrak{R}$ and $\mathfrak{R}'$ as being *non-disjoint* iff $\mathtt{Z} \asymp \mathtt{Z}'$ and $\exists \mathtt{a} : \mathtt{a}_\alpha \preccurlyeq \mathtt{a} \preccurlyeq \mathtt{a}_\omega$ and $\mathtt{a}'_\alpha \preccurlyeq \mathtt{a} \preccurlyeq \mathtt{a}'_\omega$. Given non-disjoint $\mathfrak{R}$ and $\mathfrak{R}'$ we take $\mathtt{a}$ to be the common twist on their leadlines.

The relation needed can now be stated simply: when a pair of rigs are non-disjoint, they are also aligned. Equivalently, those rigs are either disjoint or aligned.

Definition. *A pair of rigs that is neither disjoint nor aligned is* misaligned.

Definition. *A guild G is* supportive *if it does not contain any misaligned pairs of rigs.*

Showing a guild is supportive provides a unique succession property: if G is a supportive guild, and $\mathtt{Z}$ is a line, then each twist $\mathtt{a}_0$ has at most one successor $\mathtt{a}_1$ such that the line $[\mathtt{a}_0, \mathtt{a}_1]$ is held fast to $\mathtt{Z}$ by rigs in G.

Proposition 1. *Given twists $\mathtt{a}_0$, $\mathtt{a}_1$, and $\mathtt{a}'_1$ with $\mathtt{p}(\mathtt{a}_1) = \mathtt{p}(\mathtt{a}'_1) = \mathtt{a}_0$ and non-disjoint rigs $\mathfrak{R}, \mathfrak{R}'$ in a supportive guild G with $[\mathtt{a}_0, \mathtt{a}_1]$ a subline of $\mathtt{A}$ and $[\mathtt{a}_0, \mathtt{a}'_1]$ a subline of $\mathtt{A}'$; then $\mathtt{a}_1 = \mathtt{a}'_1$.*

Proof. Because G is supportive $\mathfrak{R}$ and $\mathfrak{R}'$ cannot be misaligned. Because $\mathfrak{R}$ and $\mathfrak{R}'$ are neither disjoint nor misaligned, they must be aligned. Therefore $\mathtt{A} \succ\!\!< \mathtt{A}'$.

Thus $\mathtt{a}_1$ and $\mathtt{a}'_1$ are both twists in the enveloping line of $\mathtt{A}$ and $\mathtt{A}'$, and having the same previous they must be therefore be the same twist. □

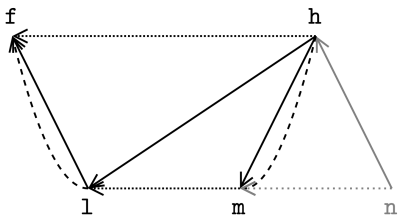
A supportive guild guarantees that the leadline of a rig has the same integrity as its corkline.

8 G_H is supportive

We previously noted that a half-hitch meets the requirements of a rig: it has a corkline bounded by its fastener and hoist; a leadline bounded by its lead and meet; and it holds fast on account of relation 1, specifically the fragment

$$\mathtt{f}(H) \prec\!\!< \mathtt{l}(H) \prec\!\!< \mathtt{m}(H) \prec\!\!< \mathtt{h}(H)$$

.



Definition. *The guild G_H is the guild consisting of all rigs $\mathfrak{R}$ for which there is a half-hitch H satisfying:*

- $\mathtt{F}(H) = \mathtt{L}(\mathfrak{R})$, *and*
- $\mathtt{T}(H)$ *is a subline of* $\mathtt{C}(\mathfrak{R})$.

In other words, G_H is the guild of rigs each consisting of a single half-hitch.

Theorem 1. *G_H is supportive.*

Proof. Let $\mathfrak{R}$ and $\mathfrak{R}'$ be rigs in G_H. Disjoint rigs can't be misaligned, therefore we take $\mathfrak{R}$ and $\mathfrak{R}'$ to be non-disjoint and show they are aligned.

Recall that the construction of a hitch requires $\mathtt{a}_\alpha$, $\mathtt{a}'_\alpha$, $\mathtt{a}_\omega$, and $\mathtt{a}'_\omega$ to be fast, and all the other twists in $\mathtt{A}$ and $\mathtt{A}'$ to be loose. Given $\mathtt{a}$ as the common twist between $\mathtt{A}$ and $\mathtt{A}'$, we note that because $\mathtt{a}_\alpha \preccurlyeq \mathtt{a}$, and $\mathtt{a}'_\alpha \preccurlyeq \mathtt{a}$, we have $\mathtt{a}_\alpha \asymp \mathtt{a}'_\alpha$, and can assume without loss of generality that $\mathtt{a}_\alpha \preccurlyeq \mathtt{a}'_\alpha$.

If $\mathtt{a}$ is fast, then one of the following must be true:

- $\mathtt{a} = \mathtt{a}_\omega = \mathtt{a}'_\omega$, in which case $\mathtt{a}_\alpha = {}^*\mathtt{p}(\mathtt{a}_\omega) = {}^*\mathtt{p}(\mathtt{a}'_\omega) = \mathtt{a}'_\alpha$, giving that $\mathtt{A} = \mathtt{A}'$, and alignment is trivial;
- $\mathtt{a} = \mathtt{a}_\omega = \mathtt{a}'_\alpha$, in which case $[\mathtt{a}_\alpha, ..., \mathtt{a}_\omega = \mathtt{a}'_\alpha, ..., \mathtt{a}'_\omega]$ is an enveloping line for $\mathtt{A}$ and $\mathtt{A}'$, providing alignment; or
- $\mathtt{a} = \mathtt{a}_\alpha = \mathtt{a}'_\alpha$, which is examined below.

If $\mathtt{a}$ is loose, then we get $\mathtt{a}_\alpha = {}^*\mathtt{p}(\mathtt{a}) = \mathtt{a}'_\alpha$, so we need only demonstrate alignment when $\mathtt{a}_\alpha = \mathtt{a}'_\alpha$.

Because $\mathfrak{R}$ is in G_H, there is a half-hitch H with

- $\mathtt{l}(H) = \mathtt{a}_\alpha$
- $\mathtt{m}(H) = \mathtt{a}_\omega$
- $\mathtt{z}_\alpha \preccurlyeq \mathtt{f}(H) \prec \mathtt{h}(H) \preccurlyeq \mathtt{z}_\omega \preccurlyeq \mathtt{z}$

Similarly, we get H' from $\mathfrak{R}'$ with

- $\mathtt{l}(H') = \mathtt{a}'_\alpha$
- $\mathtt{m}(H') = \mathtt{a}'_\omega$
- $\mathtt{z}'_\alpha \preccurlyeq \mathtt{f}(H') \prec \mathtt{h}(H') \preccurlyeq \mathtt{z}'_\omega \preccurlyeq \mathtt{z}$

Because the fastener of a hitch is equal to the tether of the hitch's lead, we have $\mathtt{f}(H) = \mathtt{t}(\mathtt{a}_\alpha) = \mathtt{t}(\mathtt{a}'_\alpha) = \mathtt{f}(H')$. Thus, the hoists $\mathtt{h}(H)$ and $\mathtt{h}(H')$ are both defined to be the earliest twist $\mathtt{z}_\mathtt{h}$ on the line $[\mathtt{f}(H), ..., \mathtt{z}]$ for which the rigging trie has $\mathtt{a}_\alpha$ as a key. Therefore they are the same twist, with the same rigging trie, which associates to $\mathtt{a}_\alpha$ the same value $\mathtt{m}(H) = \mathtt{m}(H')$.

Consequently we have that $\mathtt{A} = [\mathtt{l}(H), ..., \mathtt{m}(H)] = [\mathtt{l}(H'), ..., \mathtt{m}(H')] = \mathtt{A}'$, from which it follows trivially that $\mathtt{A} \asymp \mathtt{A}'$. □

9 Splicing

To construct rigs that span longer lines requires a composition operation that allows a rig's leadline to be lengthened by one hitch[10]. In particular, the meet

[10] In order to more easily describe the relations between the consecutive hitches, we define our splice to perform its lengthening on the past side rather than the future side. It may be more practical for implementations to perform the extensions on the other side, as new hitches become available.

of a hitch is a fast twist, and therefore has its own tether and may be used as the lead in a consecutive hitch.

We define such an operation, called *splicing*, and use it to construct rigs whose leadline $\mathtt{L}$ has a length $l(\mathtt{L}) > 1$. To simplify descriptions of splicing we define the length of a rig to be the length of its leadline, i.e. $l(\mathfrak{R}) = l(\mathtt{L}(\mathfrak{R}))$.

Definition. *A pair of rigs* $(\mathfrak{R}_0, \mathfrak{R}_1)$ *is said to be* spliceable *iff*

- $l(\mathfrak{R}_0) = 1$
- $\mathtt{L}(\mathfrak{R}_0) = [\mathtt{a}_0, ..., \mathtt{a}_1]$
- *there is a half-hitch* H_0 *in* $\mathfrak{R}_0$ *with* $\mathtt{l}(H_0) = \mathtt{a}_0$ *and* $\mathtt{m}(H_0) = \mathtt{a}_1$
- $l(\mathfrak{R}_1) = n \geq 1$
- $\mathtt{L}(\mathfrak{R}_1) = [\mathtt{a}_1, \mathtt{a}_2, ..., \mathtt{a}_{n+1}]$
- *there is a half-hitch* H_1 *in* $\mathfrak{R}_1$ *with* $\mathtt{l}(H_1) = \mathtt{a}_1$ *and* $\mathtt{m}(H_1) = \mathtt{a}_2$
- *there is a hitch* H_0^* *which adds post* $\mathtt{p}(H_0^*) = \mathtt{a}_2$ *to* H_0
- $\mathtt{C}(\mathfrak{R}_0) \asymp \mathtt{C}(\mathfrak{R}_1)$

Definition. *The* splice *of a spliceable pair of rigs* $(\mathfrak{R}_0, \mathfrak{R}_1)$*, denoted* $\mathfrak{R}_0 \Psi \mathfrak{R}_1$*, is the structure consisting of the union of each rig's collection of twists. Its corkline is the minimal enveloping line of* $\mathtt{C}(\mathfrak{R}_0)$ *and* $\mathtt{C}(\mathfrak{R}_1)$*, and its leadline is the concatenation of their leadlines.*

Proposition 2. *If (*$\mathfrak{R}_0$*,* $\mathfrak{R}_1$*) is a splicable pair of rigs, then their splice* $\mathfrak{S} = \mathfrak{R}_0 \Psi \mathfrak{R}_1$ *is also a rig.*

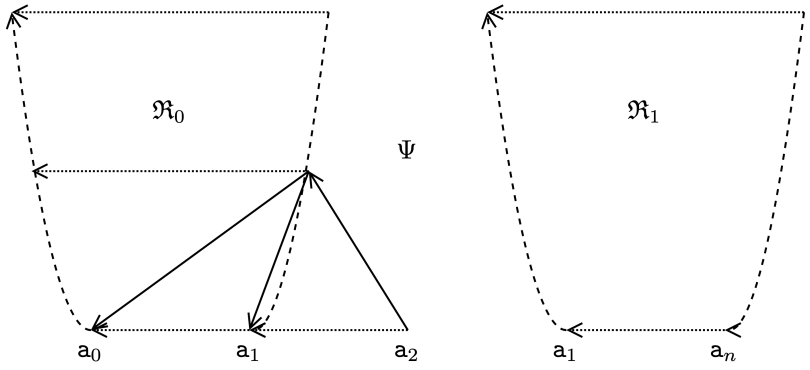
Proof. Taking $\mathfrak{S} = \mathfrak{R}_0 \Psi \mathfrak{R}_1$, $\mathtt{z}_\alpha$ to be the earliest twist in $\mathtt{C}(\mathfrak{S})$, and $\mathtt{z}_0$ to be the earliest twist in $\mathtt{C}(\mathfrak{R}_0)$, it follows from $\mathfrak{R}_0$ being a rig that $\mathtt{z}_\alpha \prec\!\!\!\prec \mathtt{z}_0 \prec\!\!\!\prec \mathtt{a}_0$.

Similarly taking $\mathtt{z}_\omega$ to be the latest twist in $\mathtt{C}(\mathfrak{S})$, and $\mathtt{z}_{n+1}$ to be the latest twist in $\mathtt{C}(\mathfrak{R}_1)$, we get $\mathtt{a}_{n+1} \prec\!\!\!\prec \mathtt{z}_{n+1} \prec\!\!\!\prec \mathtt{z}_\omega$.

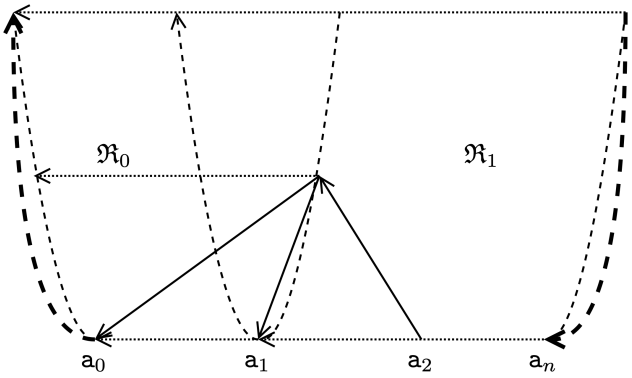
Using the fact that $\mathtt{L}(\mathfrak{S})$ is a line to combine and reduce these yields $\mathtt{z}_\alpha \prec\!\!\!\prec \mathtt{a}_0 \prec\!\!\!\prec \mathtt{a}_{n+1} \prec\!\!\!\prec \mathtt{z}_\omega$.

Thus the leadline $\mathtt{L}(\mathfrak{S})$ is held fast to the corkline $\mathtt{C}(\mathfrak{S})$ by the splice $\mathfrak{S} = \mathfrak{R}_0 \Psi \mathfrak{R}_1$, which is a rig. □

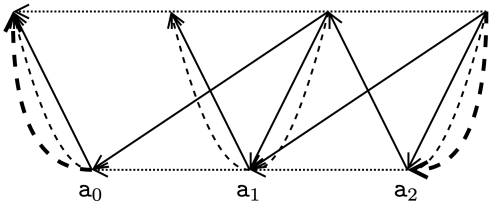
To illustrate this, below is a diagram of a pair of rigs being spliced, which also shows the hash inclusions of the half-hitch forming the bottom of $\mathfrak{R}_0$.



And this is the rig resulting from splicing them.



As a more concrete example, here is a hitch spliced to a half-hitch.



10 $\Psi(G)$ is supportive

We introduce a guild transformation operator, $\Psi(G)$, which forms the closure of a guild under splicing.

Definition. *Given a guild G where all rigs have length 1, let $\Psi(G)$ be the guild consisting of any rig that is either:*

- *a rig from G, or*
- *obtained by splicing a rig $\mathfrak{R}_0$ from G to a rig $\mathfrak{R}_1$ from $\Psi(G)$.*

To prove that this operation preserves the property of supportiveness, we start with the more restrictive case where the leadlines begin with the same twist.

Lemma 1. *Given a supportive guild G, and rigs $\mathfrak{R}$ and $\mathfrak{R}'$ from $\Psi(G)$ with $\mathtt{a}_\alpha = \mathtt{a}'_\alpha$, then $\mathfrak{R}$ and $\mathfrak{R}'$ are not misaligned.*

Proof. Since disjoint rigs cannot be misaligned, we need only consider the non-disjoint case.

Letting l and l' be the lengths of $\mathfrak{R}$ and $\mathfrak{R}'$ respectively, we assume without loss of generality that $l' \geq l$.

We prove by induction on l that $\mathtt{A} \asymp \mathtt{A}'$ for all $l \geq 1$.

Base case ($l = 1$):

In this case $\mathfrak{R}$, being of length 1, is a rig from the supportive guild G, and by definition cannot be misaligned with any other rig from G.

If $l' = 1$, then $\mathfrak{R}'$ is also a rig from G, so $\mathtt{A} \asymp \mathtt{A}'$.

If $l' > 1$, then $\mathfrak{R}'$ is obtained by splicing the rig $\mathfrak{R}'_0$ to the rig $\mathfrak{R}'_1$ at the common twist $\mathtt{a}'_1$ in their leadlines. Further, $\mathfrak{R}'_0$ is a rig from G with $\mathtt{A}'_0 = L(\mathfrak{R}'_0)$.

Since $\mathfrak{R}$ and $\mathfrak{R}'_0$ are both rigs in G (and G is supportive), they cannot be misaligned.

Further, both $\mathtt{A}'_0$ and $\mathtt{A}$ begin with $\mathtt{a}_\alpha$ and have length 1, so not only are they aligned, but in fact equal.

Then because $\mathtt{A}'_0$ is a subline of $\mathtt{A}'$ it follows that $\mathtt{A}' \asymp \mathtt{A}$.

Induction step ($l > 1$):

In this case, $\mathfrak{R}$ is obtained by splicing the rig $\mathfrak{R}_0 \in G$ to the rig $\mathfrak{R}_1$ at their common twist $\mathtt{a}_1$. Likewise, $\mathfrak{R}'$ is obtained by splicing the rig $\mathfrak{R}'_0 \in G$ to the rig $\mathfrak{R}'_1$ at their common twist $\mathtt{a}'_1$.

Now, both $\mathfrak{R}_0$ and $\mathfrak{R}'_0$ are rigs from G, with $\mathtt{A}_0 = \mathtt{L}(\mathfrak{R}_0)$ and $\mathtt{A}'_0 = \mathtt{L}(\mathfrak{R}'_0)$ both beginning at $\mathtt{a}_\alpha$. Recalling that $\mathtt{z}$ denotes the most recent of $\mathtt{z}_\omega$ and $\mathtt{z}'_\omega$, both $\mathtt{C}(\mathfrak{R}_0)$ and $\mathtt{C}(\mathfrak{R}'_0)$ end at predecessors of $\mathtt{z}$ and are therefore aligned with each other. Thus $\mathfrak{R}_0$ and $\mathfrak{R}'_0$ are non-disjoint.

G being supportive then gives that $\mathtt{A}_0 \asymp \mathtt{A}'_0$.

Because $\mathtt{A}_0$ and $\mathtt{A}'_0$ are aligned, both begin at $\mathtt{a}_\alpha$, and both have a length of 1, it follows that $\mathtt{A}_0 = \mathtt{A}'_0$ and by extension that $\mathtt{a}_1 = \mathtt{a}'_1$.

Thus we have that the leadlines $\mathtt{A}_1 = \mathtt{L}(\mathfrak{R}_1)$ and $\mathtt{A}'_1 = \mathtt{L}(\mathfrak{R}'_1)$ both begin with $\mathtt{a}_1$.

The same argument used to show alignment of $\mathtt{C}(\mathfrak{R}_0)$ and $\mathtt{C}(\mathfrak{R}'_0)$ can also be used on $\mathtt{C}(\mathfrak{R}_1)$ and $\mathtt{C}(\mathfrak{R}'_1)$ to get that $\mathfrak{R}_1$ and $\mathfrak{R}'_1$ are non-disjoint.

So our induction hypothesis provides that $\mathfrak{R}_1$ and $\mathfrak{R}'_1$, being non-disjoint rigs from $\Psi(G)$ with length less than l, must be aligned, along with their leadlines $\mathtt{A}_1 \asymp \mathtt{A}'_1$. Further, because $\mathtt{A}$ and $\mathtt{A}'$ can be obtained by tracing $\mathtt{A}_1$ and $\mathtt{A}'_1$ back to $\mathtt{a}_\alpha$, it follows that $\mathtt{A} \asymp \mathtt{A}'$. □

This more restrictive case being proven, we can now move to the general case.

Theorem 2. *Given a supportive guild G where all rigs have length 1, then $\Psi(G)$ is also supportive.*

Proof. Since disjoint rigs are trivially not misaligned, we take the rigs $\mathfrak{R}$ and $\mathfrak{R}'$ in $\Psi(G)$ to be non-disjoint and show $\mathtt{A} \asymp \mathtt{A}'$.

We assume without loss of generality that $\mathtt{a}_\alpha \preccurlyeq \mathtt{a}'_\alpha$, and because leadlines in $\Psi(G)$ must begin with fast twists, we note that this can be expressed as $\mathtt{a}_\alpha = ({}^*\mathtt{p})^n(\mathtt{a}'_\alpha)$ for some $n \geq 0$, and proceed by induction on n.

Base case ($n = 0$):
$\mathtt{a}_\alpha = ({}^*\mathtt{p})^0(\mathtt{a}'_\alpha) = \mathtt{a}'_\alpha$, so the base case is simply Lemma 1.

Induction step ($n > 0$):
If $l(\mathtt{A}) = 1$, then fastness requires that $\mathtt{a}'_\alpha$ must be either $\mathtt{a}_\alpha$ or $\mathtt{a}_\omega$, but $n > 0$ implies $\mathtt{a}'_\alpha \neq \mathtt{a}_\alpha$, so $\mathtt{a}'_\alpha = \mathtt{a}_\omega$, and we have an enveloping line

$$[\mathtt{a}_\alpha, ..., \mathtt{a}_\omega = \mathtt{a}'_\alpha, ..., \mathtt{a}'_\omega]$$

which provides $\mathtt{A} \asymp \mathtt{A}'$.

When $l(\mathtt{A}) > 1$, then $\mathfrak{R}$ is the result of splicing a rig $\mathfrak{R}_0$ from G, with $\mathtt{L}(\mathfrak{R}_0) = [\mathtt{a}_\alpha, ..., \mathtt{a}_1]$, onto a rig $\mathfrak{R}_1$ from $\Psi(G)$, with $\mathtt{L}(\mathfrak{R}_1) = [\mathtt{a}_1, ..., \mathtt{a}_\omega] = \mathtt{A}_1$.

Our induction precondition that $n > 0$ provides that $\mathtt{a}_1 = ({}^*\mathtt{p})^{n-1}(\mathtt{a}'_\alpha) \preccurlyeq \mathtt{a}'_\alpha$.

Additionally, from our induction hypothesis, we have that $\mathtt{A}_1 \asymp \mathtt{A}'$; i.e. that there is an enveloping line $[\mathtt{a}_1, ..., \mathtt{a}]$, where $\mathtt{a}$ is the more recent of $\mathtt{a}_\omega$ and $\mathtt{a}'_\omega$. Extending this enveloping line by $\mathtt{L}(\mathfrak{R}_0)$ to $[\mathtt{a}_\alpha, ..., \mathtt{a}_1, ..., \mathtt{a}]$, we get an enveloping line for $\mathtt{A}$ and $\mathtt{A}'$.

Thus $\mathtt{A} \asymp \mathtt{A}'$, and by induction we have that the Ψ operation preserves supportiveness. □

11 Lashings

The splicing operation provides the ability to form rigs that provide support for arbitrarily long lines. The lashing operation is its compliment: it provides the ability to construct rigs with intermediate lines between the corkline and the leadline.

Definition. *A pair of rigs $(\mathfrak{H}_0, \mathfrak{R}_1)$ is* lashable *when it has the following properties*

- $\mathfrak{H}_0$ *is a rig in G_H (i.e. a half-hitch)*
- $\mathfrak{R}_1$ *is an arbitrary rig*
- $\mathtt{C}(\mathfrak{H}_0) = [\mathtt{b}_0, ..., \mathtt{b}_1]$

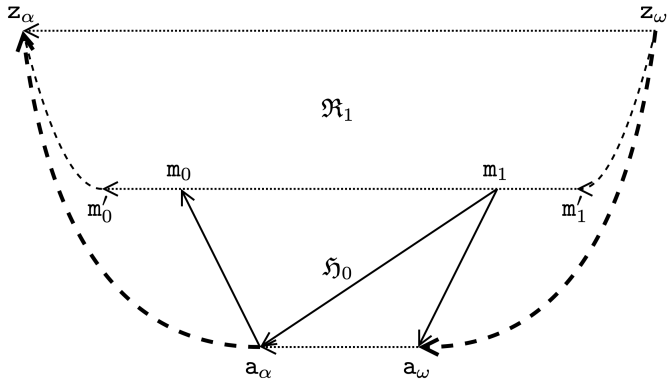
- $\mathtt{L}(\mathfrak{R}_1) = [\mathtt{b}'_0, ..., \mathtt{b}_0, ..., \mathtt{b}_1, ..., \mathtt{b}'_1]$
- *none of the twists from the corkline* $\mathtt{C}(\mathfrak{R}_1)$ *are present in* $\mathfrak{H}_0$

Definition. *When* $(\mathfrak{H}_0, \mathfrak{R}_1)$ *is a lashable pair of rigs, the lashing* $\mathfrak{H}_0 \Xi \mathfrak{R}_1$ *is taken to be the structure consisting of the union of the collections of twists from* $\mathfrak{H}_0$ *and* $\mathfrak{R}_1$*, and is assigned a corkline* $\mathtt{C}(\mathfrak{L}) = \mathtt{C}(\mathfrak{R}_1)$ *and a leadline* $\mathtt{L}(\mathfrak{L}) = \mathtt{L}(\mathfrak{H}_0)$*.*

Proposition 3. *If* $(\mathfrak{H}_0, \mathfrak{R}_1)$ *is a lashable pair of rigs, then* $\mathfrak{L} = \mathfrak{H}_0 \Xi \mathfrak{R}_1$ *is a rig.*

Proof. Taking $\mathtt{C}(\mathfrak{R}_1) = [\mathtt{z}_\alpha, ..., \mathtt{z}_\omega]$, and $\mathtt{L}(\mathfrak{H}_0) = [\mathtt{a}_\alpha, ..., \mathtt{a}_\omega]$, it follows from $\mathfrak{H}_0$ and $\mathfrak{R}_1$ holding fast that $\mathtt{z}_\alpha \prec\!\!\prec \mathtt{m}'_0 \prec\!\!\prec \mathtt{m}_0 \prec\!\!\prec \mathtt{a}_\alpha \prec\!\!\prec \mathtt{a}_\omega \prec\!\!\prec \mathtt{m}_1 \prec\!\!\prec \mathtt{m}'_1 \prec\!\!\prec \mathtt{z}_\omega$.

So $\mathfrak{L}$ holds $\mathtt{L}(\mathfrak{L})$ fast to $\mathtt{C}(\mathfrak{L})$, and is therefore a rig. □



It's worth noting that the non-duplication of corkline twists guarantees[11] that twists in a rig's corkline are not duplicated in other lines in the rig, for rigs obtained starting from hitches and applications of the splicing and lashing operations.

This lashing operation allows the construction of a rig whose leadline is not directly tethered to its corkline. To account for this we introduce the notion of a tetherline and its height.

Definition. *The* height *of a rig* $\mathfrak{R}$ *is the least* n *so that* $({}^*\mathtt{t})^n(\mathtt{a}_\alpha) \in \mathtt{C}(\mathfrak{R})$*.*

Definition. *The* tetherline *of a rig* $\mathfrak{R}$ *is the collection of twists connecting* $\mathtt{a}_\alpha$ *to the twist* $({}^*\mathtt{t})^n(\mathtt{a}_\alpha) \in \mathtt{C}(\mathfrak{R})$*.*

In other words, the tetherline of a rig is the collection of twists which contains: the first twist in the rig's leadline, and for each twist $\mathtt{x}$ in the tetherline, which is not also in the rig's corkline, that twist's tether $\mathtt{t}(\mathtt{x})$, that twist's fast tether ${}^*\mathtt{t}(\mathtt{x})$, and all intervening twists $\mathtt{y}$ with $\mathtt{t}(\mathtt{x}) \preccurlyeq \mathtt{y} \preccurlyeq {}^*\mathtt{t}(\mathtt{x})$.

[11] Except in the degenerate case of a line tethered to itself, and a rig consisting of those self-tethering hitches spliced together.

Note that the height of a rig is also the height of its tetherline.

The splicing and lashing operations can be used to construct rigs that have an arbitrary (leadline) length and an arbitrary (tetherline) height. Since the splicing operation only requires that the corklines and leadlines of the rigs being spliced be aligned, without placing any such requirements on intervening lines, these rigs can be quite flexible.

In particular, the leadline can change the intermediate line it is fastened to partway through a rig. As long as those intermediate lines are ultimately rigged up to the corkline, such moves maintain the corkline's support for that leadline.

This is a central feature of TODA files, where rigging provides integrity-at-a-distance while allowing dynamic choice of custodianship: the source of truth of a file can move from server to laptop to phone to a different server, preserving the integrity of the original server regardless of location.

12 $G_\uparrow$ is supportive

We are now ready to introduce $G_\uparrow$, the simplest guild that is closed under the splicing and lashing operations.

Definition. *$G_\uparrow$ is the guild consisting of all rigs that are either*

- *a single half-hitch*
- *a length 1 rig from $G_\uparrow$ spliced to another rig from $G_\uparrow$*
- *a single half-hitch lashed up to another rig from $G_\uparrow$*

Before we show that $G_\uparrow$ is supportive we need a small result to constrain rig height.

Proposition 4. *If for a twist* $\mathtt{x}$ *and some* $n > 0$, $({}^*\mathtt{t})^n(\mathtt{x}) \succ\!\!\prec \mathtt{x}$, *then* $({}^*\mathtt{t})^n(\mathtt{x}) \prec \mathtt{x}$.

Proof. A twist $\mathtt{x}$ includes its previous and tether hashes $\mathtt{t}(\mathtt{x})$ and $\mathtt{p}(\mathtt{x})$, so by hash inclusion we have that $\mathtt{t}(\mathtt{x}) \prec\!\!\!\prec \mathtt{x}$ and $\mathtt{p}(\mathtt{x}) \prec\!\!\!\prec \mathtt{x}$.

Because the fast tether function ${}^*\mathtt{t}$ is just a composition of $\mathtt{p}$ and $\mathtt{t}$, it follows that $({}^*\mathtt{t})^n(\mathtt{x}) \prec\!\!\!\prec \mathtt{x}$.

We have already assumed that $({}^*\mathtt{t})^n(\mathtt{x}) \succ\!\!\prec \mathtt{x}$, so $({}^*\mathtt{t})^n(\mathtt{x}) \prec\!\!\!\prec \mathtt{x}$ implies $({}^*\mathtt{t})^n(\mathtt{x}) \prec \mathtt{x}$, which is our desired result. □

We are now ready to introduce our main result.

Theorem 3. *$G_\uparrow$ is supportive.*

Proof. Once again, we take non-disjoint rigs $\mathfrak{R}$ and $\mathfrak{R}'$ in $G_\uparrow$, and seek to prove that $\mathtt{A} \succ\!\!\prec \mathtt{A}'$.

We also take l and l' to be the (leadline) lengths and h and h' to be the heights of $\mathfrak{R}$ and $\mathfrak{R}'$ respectively, and proceed by structural induction.

Base case ($l = l' = h = h' = 1$):

In this case, both $\mathfrak{R}$ and $\mathfrak{R}'$ consist of single half-hitches, putting them in G_H, so by Theorem 1 it follows that $\mathtt{A} \asymp \mathtt{A}'$.

Inductive case ($l = l' = 1$, and either $h > 1$ or $h' > 1$):

Assuming without loss of generality that $\mathtt{a}_\alpha \preccurlyeq \mathtt{a}'_\alpha$, then having $l = 1$ and non-disjoint rigs gives that we either have that $\mathtt{a}'_\alpha = \mathtt{a}_\omega$ or that $\mathtt{a}'_\alpha = \mathtt{a}_\alpha$.

If $\mathtt{a}'_\alpha = \mathtt{a}_\omega$ we simply prepend $\mathtt{A}$ to $\mathtt{A}'$ to get an enveloping line, proving that $\mathtt{A} \asymp \mathtt{A}'$.

This leaves the case where $\mathtt{a}'_\alpha = \mathtt{a}_\alpha$.

We begin by assuming that $h > h'$ and deriving a contradiction. The definition of h provides that:

- $({}^*\mathtt{t})^h(\mathtt{a}_\alpha) = \mathtt{z}_\alpha$ and
- $({}^*\mathtt{t})^{h'}(\mathtt{a}_\alpha) = \mathtt{z}'_\alpha$.

Thus it follows that $({}^*\mathtt{t})^{h-h'}(\mathtt{z}'_\alpha) \asymp \mathtt{z}_\alpha$.

Our preconditions provide that $\mathtt{z}_\alpha \asymp \mathtt{z}'_\alpha$, and the construction of lashing disallows $\mathtt{z}_\alpha \preccurlyeq \mathtt{z}'_\alpha$, because that would place $\mathtt{z}'_\alpha$ both in $\mathtt{C}(\mathfrak{R})$ and also elsewhere in the rig, so it must be the case that $\mathtt{z}'_\alpha \prec \mathtt{z}_\alpha$.

However Proposition 4 gives that because $\mathtt{z}_\alpha = ({}^*\mathtt{t})^{h-h'}(\mathtt{z}'_\alpha) \asymp \mathtt{z}'_\alpha$, it must be the case that $\mathtt{z}_\alpha \prec \mathtt{z}'_\alpha$.

This is a contradiction, so it cannot be the case that $h > h'$. A similar argument excludes the case where $h' > h$, so the heights must be equal.

In the $h = h'$ case, we know that a single half-hitch has $h = 1$, but $h > 1$ so $\mathfrak{R}$ is not a single half-hitch. Further, a rig constructed by splicing together two rigs has a length that is the sum of the lengths of the spliced rigs; since every rig has a length of at least one, a spliced rig must have a length of at least two. Thus with $l = 1$, $\mathfrak{R}$ cannot be constructed by splicing two rigs together. So we must be in the remaining case, where $\mathfrak{R}$ consists of a half-hitch H lashed up to a rig $\mathfrak{S}$, with a height of $h - 1$.

Similarly, $\mathfrak{R}'$ must consist of a half-hitch H' lashed up to a rig $\mathfrak{S}'$, with a height of $h' - 1$.

By the definition of lashing, we have that the leadline $\mathtt{B}$ of $\mathfrak{S}$ (and similarly the leadline $\mathtt{B}'$ of $\mathfrak{S}'$) must begin with ${}^*\mathtt{t}(\mathtt{a}_0)$, and with $\mathfrak{S}$ and $\mathfrak{S}'$ having the same corklines as $\mathfrak{R}$ and $\mathfrak{R}'$ respectively, we know them to be non-disjoint. Thus our induction hypothesis provides that $\mathfrak{S}$ and $\mathfrak{S}'$ cannot be misaligned, so it follows that $\mathtt{B} \asymp \mathtt{B}'$.

This alignment and the existence of a common twist between $\mathtt{A}$ and $\mathtt{A}'$ gives that $\mathfrak{H}$ and $\mathfrak{H}'$ are non-disjoint, and since they both belong to the supportive guild G_H, they must therefore be aligned. Thus we have alignment between their leadlines, i.e. $\mathtt{A} \asymp \mathtt{A}'$.

Inductive case ($l, l' \geq 1$):

Like with h in the previous case, we assume without loss of generality that $l > 1$. Because a single half-hitch has length $l = 1$, and lashings also yield rigs with length $l = 1$, it must be the case that $\mathfrak{R}$ is obtained by splicing.

The induction hypothesis provides that the spliced rigs satisfy the conditions of Theorem 2, which provides that $\mathtt{A} \asymp \mathtt{A}'$.

Thus by structural induction, we have that $G_\uparrow$ is supportive. □

Theorem 3 shows that the guild $G_\uparrow$ is supportive. Proposition 1 shows that a supportive guild guarantees unique canonical succession in a supported line.

To put this in the vernacular:

Corollary 1. *Rigs in $G_\uparrow$ prevent double spend.*

13 Glossary

We introduce and use a number of data structures, as well as functions on those structures. To maintain legibility we adhere to the following conventions

- twists are represented by lower-case monospace; e.g. `t` is a twist
- lines are represented by uppercase monospace; e.g. `A` is a line
- lines can also be represented as square brackets containing a list of twists (in chronological order); e.g. $[\mathtt{a}_0, \mathtt{a}_1, \mathtt{a}_2, \mathtt{a}_3]$
- hitches are represented by upper-case italics; e.g. H is a hitch
- guilds are also represented by upper-case italics; e.g. G is a guild
- rigs are represented by uppercase fraktur; e.g. $\mathfrak{R}$ is a rig
- other values are represented by lowercase italics (e.g. simple numbers, hash outputs, arbitrary inputs, tries); e.g. n

Further, the symbols for functions follow the same convention that is used for their images; e.g $\mathtt{f}(\mathfrak{R})$ is a twist-valued function on a rig.

Common functions are

- $\mathtt{t}(\mathtt{x})$; the tether of `x`
- ${}^*\mathtt{t}(\mathtt{x})$; the fast tether of `x`
- $\mathtt{p}(\mathtt{x})$; the previous of `x`
- ${}^*\mathtt{p}(\mathtt{x})$; the previous of `x`
- $r(\mathtt{x})$; the rigging trie of `x`
- $\mathtt{f}(H)$; the fastener of H
- $\mathtt{h}(H)$; the hoist of H
- $\mathtt{l}(H)$; the lead of H
- $\mathtt{m}(H)$; the meet of H
- $\mathtt{n}(H)$; the post of H
- $h(x)$; the hash of x

Common relations and operations are:

- $\mathtt{x} \prec \mathtt{y}$ is read "x is a predecessor of y" or "x precedes y", and means twist `x` is on the line defined by twist `y`;
- $\mathtt{x} \asymp \mathtt{y}$ is read "x is aligned with y" or "x and y are aligned" and means that $\mathtt{x} \preccurlyeq \mathtt{y}$ or $\mathtt{y} \preccurlyeq \mathtt{x}$;

- $\mathtt{A} \asymp \mathtt{B}$ is read "A is aligned with B" or "A and B aligned", and means line $\mathtt{A}$ and line $\mathtt{B}$ are both contained within a line $\mathtt{C}$;
- $\mathfrak{R} \asymp \mathfrak{R}'$ is read "R and R prime are aligned", and means $\mathtt{L}(\mathfrak{R}) \asymp \mathtt{L}(\mathfrak{R}')$;
- $x \prec\!\!\prec y$ is read "x causally precedes y", and means that y includes x through one-way functions, for instance by containing the cryptographic hash of x.

When comparing two rigs $\mathfrak{R}$ and $\mathfrak{R}'$, $\mathtt{A}/\mathtt{a}$ and $\mathtt{Z}/\mathtt{z}$ are used for their leadlines and corklines

- $\mathtt{A} = \mathtt{L}(\mathfrak{R}) = [\mathtt{a}_\alpha, ..., \mathtt{a}_\omega]$
- $\mathtt{Z} = \mathtt{C}(\mathfrak{R}) = [\mathtt{z}_\alpha, ..., \mathtt{z}_\omega]$
- $\mathtt{A}' = \mathtt{L}(\mathfrak{R}') = [\mathtt{a}'_\alpha, ..., \mathtt{a}'_\omega]$
- $\mathtt{Z}' = \mathtt{C}(\mathfrak{R}') = [\mathtt{z}'_\alpha, ..., \mathtt{z}'_\omega]$

Additionally, when $\mathfrak{R}$ and $\mathfrak{R}'$ are non-disjoint, $\mathtt{z}$ is the most recent of $\mathtt{z}_\omega$ and $\mathtt{z}'_\omega$ and $\mathtt{a}$ is the common twist on their leadlines.

14 Acknowledgements

We wish to particularly thank Toufi Saliba for cofounding the TODA vision, Hassan Khan for relentlessly championing it, and Adam Gravitis for engineering insight during the long gestation of this work. Thank you to the following reviewers for their valuable feedback: Cory Sulpizi, Mike Everson, Alexander Fertman, and Robert Moir.

The authors are grateful to TODAQ for their generous financial support of this work.

References

[1] Ivan Bjerre Damgård, *Collision free hash functions and public key signature schemes*, Advances in Cryptology — EUROCRYPT' 87 (Berlin, Heidelberg) (David Chaum and Wyn L. Price, eds.), Springer Berlin Heidelberg, 1988, pp. 203–216.

[2] Rene De La Briandais, *File searching using variable length keys*, Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference (New York, NY, USA), IRE-AIEE-ACM '59 (Western), Association for Computing Machinery, 1959, p. 295–298.

[3] Edward Fredkin, *Trie memory*, Commun. ACM **3** (1960), no. 9, 490–499.

[4] Stuart Haber and W. Scott Stornetta, *How to time-stamp a digital document*, Advances in Cryptology-CRYPTO' 90 (Berlin, Heidelberg) (Alfred J. Menezes and Scott A. Vanstone, eds.), Springer Berlin Heidelberg, 1991, pp. 437–455.

[5] Leslie Lamport, *Time, clocks, and the ordering of events in a distributed system*, Communications (1978).

[6] ———, *Constructing digital signatures from a one way function*, (1979).

[7] Ralph C. Merkle, *Protocols for public key cryptosystems*, 1980 IEEE Symposium on Security and Privacy, 1980, pp. 122–122.

[8] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM **21** (1978), no. 2, 120–126.