

# More ties than we thought

DAN HIRSCH  
Upstanding Hackers LLC  
thequux@upstandinghackers.com

MEREDITH L. PATTERSON  
Upstanding Hackers LLC  
mlp@upstandinghackers.com

ANDERS SANDBERG  
Oxford University  
anders.sandberg@philosophy.ox.ac.uk

MIKAEL VEJDEMO-JOHANSSON  
KTH Royal Institute of Technology, Stockholm  
Jožef Štefan Institute, Ljubljana  
Institute for Mathematics and its Applications, Minneapolis  
mvj@kth.se

## Abstract

We extend the existing enumeration of neck tie knots to include tie knots with a textured front, tied with the narrow end of a tie. These tie knots have gained popularity in recent years, based on reconstructions of a costume detail from *The Matrix Reloaded*, and are explicitly ruled out in the enumeration by Fink and Mao (2000).

We show that the relaxed tie knot description language that comprehensively describes these extended tie knot classes is either context sensitive or context free. It has a sub-language that covers all the knots that inspired the work, and that is regular. From this regular sub-language we enumerate 177 147 distinct tie knots that seem tieable with a normal necktie. These are found through an enumeration of 2 046 winding patterns that can be varied by tucking the tie under itself at various points along the winding.

## I. INTRODUCTION

There are several different ways to tie a necktie. Classically, knots such as the four-in-hand, the half windsor and the full windsor have been commonly taught to new tie-wearers. In a sequence of papers and a book, Fink and Mao [1–3] defined a formal language for describing tie knots, encoding the topology and geometry of the knot tying process into the formal language, and then used this language to enumerate all tie knots that could reasonably be tied with a normal-sized necktie.

The enumeration of Fink and Mao crucially depends on dictating a particular finishing sequence for tie knots: a finishing sequence that forces the front of the knot – the façade – to be a flat stretch of fabric. With this assumption in place, Fink and Mao produce a list of 85 distinct tie knots, and determine several novel knots that extend the previously commonly known list of tie knots.

In recent years, however, interest has been growing for a new approach to tie knots. In *The Matrix Reloaded* [12], the character of “The Merovingian” has a sequence of particularly fancy tie knots. Attempts by fans of the movie to recreate the tie knots from the Merovingian have led to a collection of new tie knot inventions, all of which rely on tying the tie with the thin end of the tie – the thin blade. Doing this allows for a knot with textures or stylings of the front of the knot, producing symmetric and pleasing patterns.

Knorr [4] tells the story of the main participants and their additions to the conversation:

*On 21 June 2003 Luke edeity Housego invents the inverse tie-knots. The day before he had seen ‘Matrix Reloaded’ at the cinema and wanted to have a tie-knot as cool as the one the character ‘Merovingian’ sports in the movie.*

*On 28 September 2003 Luke publishes a .pdf-tutorial for his knot on the Internet. He calls his invention ‘edeity’s knot.’*

*On 03 February 2006 Victor Allen Lord Whimsy Crawford III publishes a .pdf-tutorial for a tie-knot he calls ‘The Merovingian.’ In fact it is edeity’s sequence, but rendered much more clearly than in edeity’s original .pdf. Whimsy had the idea from said .pdf, but was not sure, if he had matched the sequence.*

*On 16 February 2007 Henry SimplyJustHen Hu publishes a video on YouTube wherein he shows how to tie a knot he calls the ‘Hen Tie.’ In the video Henry makes clear that he has the idea from edeity’s .pdf-tutorial, but that he was not sure if he had matched the sequence. In fact Henry’s sequence slightly differs from edeity’s.*

*On 18 February 2007 the knot called ‘Merovingian’ appears in the German version of the Wikipedia, linking to Lord Whimsy’s tutorial.*

*On 04 May 2008 Jeffrey cwtrain Eldredge publishes a video on YouTube, demonstrating how to tie an even larger inverse tie-knot he calls the ‘Eldredge.’ Luke edeity Housego gave the world the inverse tie-knots, and Jeffrey Eldredge invented a subterfuge in tie-knotting not to be found in the literature so far: He simply tucks away the rest of the tie’s narrow end under the collar, thereby making possible the largest tie-knot known. This move rightfully can be called ‘the Eldredge tuckaway.’ But there is a problem with Jeffrey’s knot: It’s not a knot, but more a ‘wrapping.’*

*On 19 October 2008 Alexander zephyrin\_xirdal Knorr publishes the description and sequence of the ‘Eldredge Variant’ in his weblog, making the ‘Eldredge’ into a true knot.*

*On 19 June 2010 Jeffrey Eldredge publishes the video ‘The Eldredge Knot: Revisited’ on YouTube, demonstrating how to tie the sequence of the ‘Eldredge Variant.’*

In this paper, we present a radical simplification of the formal language proposed by Fink and Mao, together with an analysis of the asymptotic complexity class of the tie knots language. We produce a novel enumeration of necktie knots tied with the thin blade, and compare it to the results of Fink and Mao.

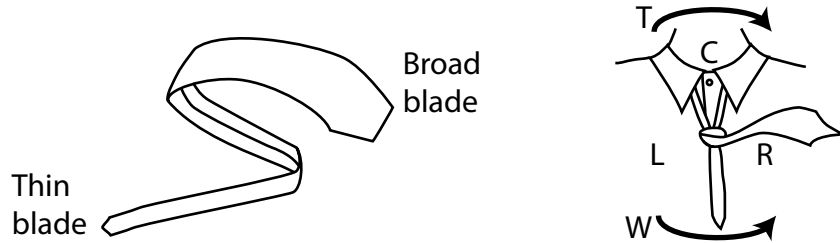


Fig. 1. The parts of a necktie, and the division of the wearer’s torso with the regions (Left, Center Right) and the winding directions (Turnwise, Widdershins) marked out for reference.

### A. Formal languages

The work in this paper relies heavily on the language of formal languages, as used in theoretical computer science and in mathematical linguistics. For a comprehensive reference, we recommend the textbook by Sipser [11].

Recall that given a finite set  $\mathcal{L}$  called an *alphabet*, the set of all sequences of any length of items drawn (with replacement) from  $\mathcal{L}$  is denoted by  $\mathcal{L}^*$ . A *formal language* on the alphabet  $\mathcal{L}$  is some subset  $\mathcal{A}$  of  $\mathcal{L}^*$ . Depending on how difficult it is to accurately determine membership in a formal language  $\mathcal{A}$ , it places in one of several complexity classes. Languages that are described by finite state automata are *regular*; languages that require a pushdown automaton are *context free*; languages that require a linear bounded automaton are *context sensitive* and languages that require a full Turing machine to determine are called *recursively enumerable*. This sequence builds an increasing hierarchy of expressibility and computational complexity for syntactic rules for strings of some arbitrary sort of tokens.

One way to describe a language is to give a *grammar* – a set of production rules that decompose some form of abstract tokens into sequences of abstract or concrete tokens, ending with a sequence of elements in some alphabet. The standard notation for such grammars is the Backus-Naur form, which uses  $::=$  to denote the production rules and  $\langle \text{some name} \rangle$  to denote the abstract tokens. Further common symbols are  $*$  – the Kleene star, that denotes an arbitrary number of repetitions of the previous token (or group in brackets), and  $|$  denoting a choice of one of the adjoining options.

## II. THE ANATOMY OF A NECKTIE

In the following, we will be referring quite a lot to various parts and constructions with a necktie. We call the ends of a necktie *blades*, and distinguish between the *broad blade* and the *thin blade*<sup>1</sup> – see Figure 1 for these names. The tie knot can be divided up into a *body*, consisting of all the twists and turns that are not directly visible in the final knot, and a *façade*, consisting of the parts of the tie actually visible in the end. In Figure 2 we demonstrate this distinction. The body builds up the overall shape of the tie knot, while the façade gives texture to the front of the knot. The enumeration of Fink and Mao only considers knots with trivial façades, while these later inventions all consider more interesting façades. As a knot is in place around a wearer, the Y-shape of the tie divides the torso into 3 regions: Left, Center and Right – as shown to the right in Figure 1.

A tie knot has to be tied by winding and tucking one of the two blades around the other: if both blades are active, then the tie can no longer be adjusted in place for a comfortable fit. We shall refer to the blade used in tying the knot as the *leading blade* or the *active blade*. Each time the active blade is moved across the tie knot – in front or in back – we call the part of the tie laid on top of the knot a *bow*.

<sup>1</sup>There are neckties without a width difference between the ends. We ignore this distinction for this paper.

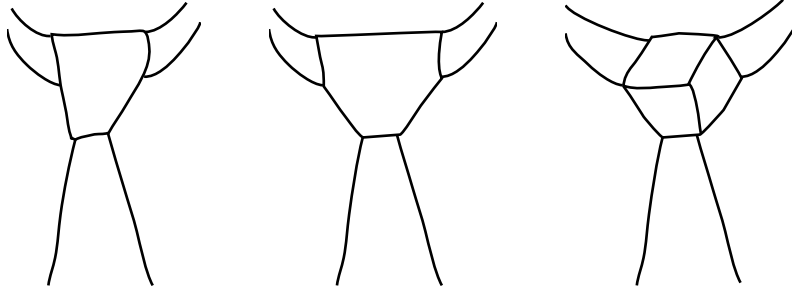


Fig. 2. Different examples of tie knots. Left, a 4-in-hand; middle, a double windsor; right a trinity. The 4-in-hand and double windsor share the flat façade but have different bodies producing different shapes. The trinity has a completely different façade, produced by a different wind and tuck pattern.

### III. A LANGUAGE FOR TIE KNOTS

Fink and Mao [2] observe that once the first crossing has been made, the wrapping sequence of a classical tie knot is completely decided by the sequence of regions into which the broad blade is moved. Adorning the region specifications with a direction – is the tie moving away from the wearer or towards the wearer – they establish a formal alphabet for describing tie knots with 7 symbols. We reproduce their construction here, using  $U$  for the move to tuck the blade *Under* the tie itself<sup>2</sup>. The notation proposed by Fink and Mao [2] interprets repetitions  $U^k$  of  $U$  as tucking the blade  $k$  bows under the top. It turns out that the complexity analysis is far simpler if we instead write  $U^k$  for tucking the blade under the bow that was produced  $2k$  windings ago. This produces a language on the alphabet:

$$\{L_{\otimes}, L_{\odot}, C_{\otimes}, C_{\odot}, R_{\otimes}, R_{\odot}, U\}$$

They then introduce relations and restrictions on these symbols:

*Tie*<sub>1</sub> No region ( $L, C, R$ ) shall repeat.  $U$  moves do not influence this.

*Tie*<sub>2</sub> No direction ( $\odot, \otimes$ ) shall repeat.  $U$  moves do not influence this.

*Tie*<sub>3</sub> Tucks ( $U$ ) are valid after an outward move.

*Tie*<sub>4</sub> A tie knot can end only on one of  $C_{\otimes}, C_{\odot}$  or  $U$ . In fact, almost all classical knots end on  $U$ .<sup>3</sup>

*Tie*<sub>5</sub> A  $k$ -fold tuck  $U^k$  is only valid after at least  $2k$  preceding moves. Fink and Mao [2] do not pay much attention to the conditions on  $k$ -fold tucks, since these show up in their enumeration as stylistic variations, exclusively at the end of a knot.

This collection of rules allow us to drastically shrink the tie language, both in alphabet and axioms. Fink and Mao are careful to annotate whether tie knot moves go outwards or inwards at any given point. We note that the inwards/outwards distinction follows as a direct consequence of axioms *Tie*<sub>2</sub>, *Tie*<sub>3</sub> and *Tie*<sub>4</sub>. Since non-tuck moves must alternate between inwards and outwards, and the last non-tuck move must be outwards, the orientation of any sequence of moves follows by backtracking from the end of the string.

Hence, when faced with a non-annotated string like

$$RCLCRCLCRCLRURCLU$$

we can immediately trace from the tail of the knot string: the last move before the final tuck must be outwards, so that  $L$  must be a  $L_{\odot}$ . So it must be preceded by  $R_{\odot}C_{\otimes}$ . Tracing backwards, we can specify

<sup>2</sup>Fink and Mao used  $T$  for Tuck

<sup>3</sup>The exemption here being the Onassis style knot, favored by the eponymous shipping magnate, where after a classical knot the broad blade is brought up with a  $C_{\odot}$  move to fall in front of the knot, hiding the knot completely.

the entire string above to

$$R_{\otimes}C_{\circ}L_{\otimes}C_{\circ}R_{\otimes}C_{\circ}L_{\otimes}C_{\circ}R_{\otimes}C_{\circ}L_{\otimes}R_{\circ}UC_{\otimes}R_{\circ}C_{\otimes}L_{\circ}U$$

Next, the axiom  $Tie_1$  means that a sequence will not contain either of  $LU^*L, CU^*C, RU^*R$  as subsequences<sup>4</sup>. Hence, the listing of regions is less important than the direction of transition: any valid transition is going to go either clockwise or counterclockwise. Writing  $T$  for clockwise<sup>5</sup> and  $W$  for counterclockwise<sup>6</sup>, we can give a strongly reduced tie language on the alphabet  $T, W, U$ . To completely determine a tie knot, the sequence needs a starting state: an annotation on whether the first crossing of a tie knot goes across to the right or to the left. In such a sequence, a  $U$  instruction must be followed by either  $T$  or  $W$  dictating which direction the winding continues after the tuck, unless it is the last move of the tie: in this case, the blade is assumed to continue straight ahead – down in front for most broad-blade tie knots, tucked in under the collar for most thin-blade knots.

Position of the leading blade after a sequence of  $W/T$  windings is a direct result of  $\#W - \#T \pmod{3}$ . This observation allows us to gain control over several conditions determining whether a distribution of  $U$  symbols over a sequence of  $W/T$  produces a physically viable tie knot.

*Theorem 1:* A position in a winding sequence is valid for a  $k$ -fold tuck if the position is preceded by  $2k$  winding symbols (W or T) that either

- 1) starts with W and satisfies  $\#W - \#T = 2 \pmod{3}$
- 2) starts with T and satisfies  $\#T - \#W = 2 \pmod{3}$

*Proof:* The initial symbol produces the bow under which the tuck will go. If the initial symbol goes, say, from R to L, then the tuck move needs to come from C in order to go under the bow. In general, a tuck needs to come from the one region not involved in the covering bow. Every other bow goes in front of the knot, and the others go behind the knot. Hence, there are  $2k - 1$  additional winding symbols until the active blade returns to the right side of the knot. During these  $2k - 1$  symbols, we need to transition one more step around the sequence of regions. The transitions W and T are generator and inverse for the cyclic group of order 3, concluding the proof. ■

We may notice that with the usual physical constraints on a tie – where we have experimentally established that broad blade ties tend to be bounded by 9 moves, and thin blade ties by 15 moves, we can expect that no meaningful tuck deeper than 7 will ever be relevant; 4 for the broad blade ties. The bound of 4 is achieved in the enumeration by Fink and Mao [3].

#### IV. LANGUAGE COMPLEXITY

In this section, we examine the complexity features of the tie knot language. Due to the constraints we have already observed on the cardinality of  $W$  and  $T$ , we will define a grammar for this language using the *attribute grammar* formalism of Knuth [5]. We will write this grammar with an annotated Backus-Naur form – symbols may have a finite number of numeric or boolean attributes that can be used to validate a potentially correct string in the grammar. Although in practice it is only possible to realise finite strings in the tie knot language due to the physical properties of fabric, we assume an arbitrarily long (but finite), infinitely thin tie.

##### A. Single-depth tucks

The following regular grammar describes all valid winding sequences for knots in which tucks only pass the active blade under the most recent bow made over the knot (which we will call *depth-1-tuckable*).

<sup>4</sup>Recall that the Kleene star  $F^*$  is used to denote sequences of 0 or more repetitions of the string  $F$ .

<sup>5</sup> $T$  for Turnwise

<sup>6</sup> $W$  for Widdershins

This subclass can be described by a simple grammar; if we want to allow deeper tucks – going under earlier bows – then an attribute grammar as in Section IV-B is called for.

$$\begin{aligned}
\langle \text{prefix} \rangle &::= 'T' | 'W' | \epsilon \\
\langle \text{tuck} \rangle &::= 'TTU' | 'WWU' \\
\langle \text{pair} \rangle &::= 'WT' | 'WW' | 'TT' | 'TW' \\
\langle \text{tie} \rangle &::= \langle \text{prefix} \rangle [\langle \text{pair} \rangle | \langle \text{tuck} \rangle] * \langle \text{tuck} \rangle
\end{aligned}$$

As section V elaborates, the distribution of  $T$  and  $W$  varies by type of knot: for classical knots,  $\#W - \#T = 2 \pmod{3}$ ; for modern knots that tuck to the right,  $\#W - \#T = 1 \pmod{3}$ ; and for modern knots that tuck to the left,  $\#W - \#T = 0 \pmod{3}$ . This grammar does not discriminate between these three subclasses.

### B. Recursive tucks

We allow for greater tuck depth by making the  $\langle \text{tuck} \rangle$  rule recursive:

$$\langle \text{tuck} \rangle ::= \langle \text{pair} \rangle \langle \text{tuck} \rangle' \cup' | \epsilon$$

We then add attributes to the  $\langle \text{pair} \rangle$  and  $\langle \text{tuck} \rangle$  rules as follows:

$$\begin{array}{ll}
\langle \text{pair} \rangle ::= 'TT' | & [\mathbf{pair.t} = 2, \mathbf{pair.w} = 0] \\
& 'TW' | & [\mathbf{pair.t} = 1, \mathbf{pair.w} = 1] \\
& 'WT' | & [\mathbf{pair.t} = 1, \mathbf{pair.w} = 1] \\
& 'WW' & [\mathbf{pair.t} = 0, \mathbf{pair.w} = 2] \\
\langle \text{tuck} \rangle ::= \langle \text{pair} \rangle \langle \text{tuckable} \rangle' \cup' & [\mathbf{tuck.t} = \mathbf{pair.t} + \mathbf{tuckable.t} \\
& & \mathbf{tuck.w} = \mathbf{pair.w} + \mathbf{tuckable.w} \\
& & \mathbf{tuck.valid} = \mathbf{tuck.t} - \mathbf{tuck.w} \pmod{3} \text{ if } \mathbf{tuck}[0] = 'W' \\
& & \mathbf{tuck.w} - \mathbf{tuck.t} \pmod{3} \text{ if } \mathbf{tuck}[0] = 'T'] \\
\langle \text{tuckable} \rangle ::= \epsilon | & [\mathbf{tuckable.t} = 0, \mathbf{tuckable.w} = 0] \\
& \langle \text{tuck} \rangle & [\mathbf{tuckable.t} = \mathbf{tuck.t}, \mathbf{tuckable.w} = \mathbf{tuck.w}]
\end{array}$$

Note that the validity of a tuck depends only on the count of  $T$  and  $W$  in the entire sequence comprising the tuck, and not the validity of any tucks recursively embedded into it. For instance,  $TWTTUU$  is a valid depth-2-tuckable sequence, as is its embedded depth-1-tuckable sequence  $TTU$ . However,  $TTWTUU$  is also a valid depth-2-tuckable sequence, even though  $WTU$  is not a valid depth-1-tuckable sequence.

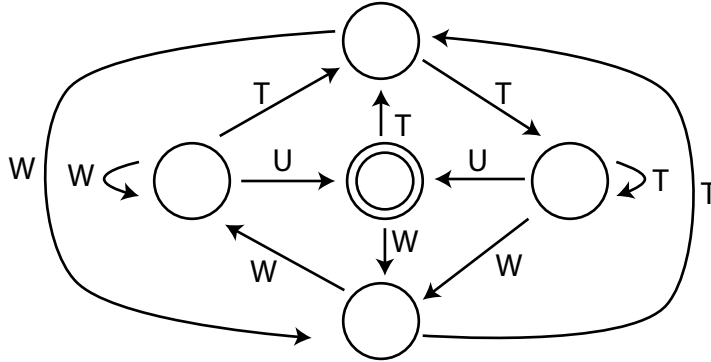
Finally, we add one last attribute to the top-level  $\langle \text{tie} \rangle$  rule:

$$\langle \text{tie} \rangle ::= \langle \text{prefix} \rangle [\langle \text{pair} \rangle | \langle \text{tuck} \rangle] * \langle \text{tuck} \rangle \quad [\mathbf{tie.valid} = \text{True if all } \mathbf{tuck.valid} = 1]$$

Since all the semantic information passes up the parse tree, the grammar uses only synthesized attributes and is therefore an S-attributed grammar, amenable to both top-down and bottom-up parsing approaches.

### C. Classification of the tie-knot language

If we limit our attention to only the single depth tie knots described in Section IV-A, then the grammar is regular, proving that this tie language is a regular language and can be described by a finite automaton. In fact, an automaton accepting these tie knots is given by:



Execution starts at the middle node, but has to go outside and return before the machine will accept input.

As for the deeper tucked language in Section IV-B, classification is significantly harder: the presence of recursion means it is at least context-free, whereas the expressibility with an attribute grammar implies at most context-sensitive. The language does not seem immediately amenable to pumping lemma arguments, nor to treatment with Ogden's lemma or Parikh's theorem or interchange lemmas[8–10, 13].

## V. ENUMERATION

We can cut down the enumeration work by using some apparent symmetries. Without loss of generality we can assume that a tie knot starts by putting the active blade in region R: any knot starting in the region L is the mirror image of a knot that starts in R and swaps all W to T and vice versa.

### A. The classical case

Knots finishing in region C include all classical knots as classified by Fink and Mao [2]: these are all tied with the broad blade and finish with C. Hence, they are classified completely by W/T sequences with  $\#W - \#T = 1 \pmod{3}$ , by not counting the initial crossing as one of the moves –

For a tie knot with  $k$  moves, the W and T moves need to divide so that  $\#W - \#T = 1 \pmod{3}$ . This is possible in  $\lceil k/3 \rceil$  ways. For each allotment of W and T moves, then, a classical tie knot must end with at least a single order tuck. If it weren't for this condition, we could count the number of ties for each allotment as  $\binom{k}{\#T}$ , by choosing the positions that receive a T symbol. However, we need to disqualify the choices that place different symbols at the two last positions of the string.

These disqualified versions end with either WT or TW and are preceded by an arbitrary sequence of  $k - 2$  W and T symbols. They are only possible if there are at least 1 of each symbol available to begin with.

*Theorem 2:* The number of classical ties with exactly  $k$  winding steps is

$$\sum_{\substack{t+w=k \\ t-w=1 \pmod{3}}} \binom{k}{t} - 2 \binom{k-2}{t-1}$$

*Proof:* Out of the  $k$  winding steps, we need 1 more T than W to reach the C position for the finishing move. For any choice of  $\#T, \#W$ , there are  $\binom{k}{\#T}$  of these. Out of all ways to place the requisite number of T in the winding sequence, we remove those that use up one each of T and W for the last two moves.

Such windings are counted by  $\binom{k-2}{\#T_{-1}}$ , and for each such sequence, both TW and WT are forbidden finishing sequences.

Hence, for each candidate sequence of  $k - 2$  moves we need to exclude two sequences. ■

In particular, the cases enumerated by Fink and Mao [2] are

Winding length	2	3	4	5	6	7	8	total
# tie knots	1	1	3	5	11	21	43	85

### B. The modern case

A knot with the thick blade active will cover up the entire knot with each new bow. As such, all thick blade active tie knots will fall within the classification by Fink and Mao [2].

The modern case, thus, deals with thin blade active knots. As evidenced by the Trinity and the Eldredge knots, thin blade knots have a wider range of interesting façades and of interesting tuck patterns. Where for thick blade knots, it was enough to assume that the tuck happens last, and from the C region, the thin blade knots have a far wider variety.

The case remains that unless the last move is a tuck – or possibly finishes in the C region – the knot will unravel from gravity. We can thus expect this to be a valid requirement for the enumeration. There are often more valid tuck sites than the final position in a knot, and the tuck need no longer come from the C region: R and L are at least as valid.

To enumerate windings that end with a single depth tuck move, the formula from Theorem 2 will serve our needs, with some slight modification. More interesting, however, is to enumerate all tie knots with all their choices of possible tuck sites.

Winding sequences can be generated with the same building blocks we used for the classical case above.

*Theorem 3:* The number of tie winding sequences ending with a single depth tuck from the left after  $k$  winding steps is

$$\sum_{\substack{t+w=k \\ t-w=0 \pmod{3}}} \binom{k}{t} - 2 \binom{k-2}{t-1}$$

The number of winding sequences ending with a single depth tuck from the right after  $k$  winding steps is

$$\sum_{\substack{t+w=k \\ t-w=2 \pmod{3}}} \binom{k}{t} - 2 \binom{k-2}{t-1}$$

*Proof:* The proof closely follows that of Theorem 2. In each case, we need to enforce repetition of the last winding symbol to allow for a single depth tuck at the end. In each case, we also need to end our winding sequence in the correct region for the tuck to take place. The region placement is handled by the mod 3 residue. ■

While a good closed form expression for the total number of tie knots, especially including all the tuck combinations of different depths, seems out of reach at the current time, the descriptions we have given so far provide accessible routes to algorithmically enumerating possible tie knots. In the appendix, we include a listing of single depth tuck knots, as well as python code that will list and print knots and potential tuck sites for any depth and winding length.

To extend the table above, we can use the formulae in Theorems 2 and 3 to establish the following:

Winding length	2	3	4	5	6	7	8	9	10	11	total
# left single tuck windings	0	2	2	6	10	22	42	86	170	324	682
# right single tuck windings	1	1	3	5	11	21	43	85	171	341	682
# center single tuck windings	1	1	3	5	11	21	43	85	171	341	682



The reason for the similarity between the right and the center counts is that if  $t - w = 1 \pmod{3}$ , then  $w - t = 2 \pmod{3}$ . Hence, a winding sequence for a center tuck can be mirrored to a winding sequence for a right tuck.

By counting the number of detected tuck sites, we have calculated the total number of tie knots using only single depth tucks to be 177 147. Of these, 59 016 each end with a right or a center tuck, and 59 115 end with a left tuck.

## VI. AESTHETICS

Fink and Mao [2] propose several measures to quantify the aesthetic qualities of a necktie knot; notably *symmetry* and *balance*, corresponding to the quantities  $\#R - \#L$  and the number of transitions from a streak of W to a streak of T or vice versa.

By considering the popular thin-blade neck tie knots: the Eldredge and the Trinity, as described in [6, 7], we can immediately note that balance no longer seems to be as important for the look of a tie knot as is the shape of its façade. Symmetry still plays an important role in knots, and is easy to calculate using the CLR notation for tie knots.

Knot	TW-string	CLR-string	Balance	Symmetry
<b>Eldredge</b>	TTTWTTTUTTWU	LCRLRCRLUCRCLU	3	0
<b>Trinity</b>	TWWTTTUTTU	LCLRCRLCURLU	2	1

We do not in this paper attempt to optimize any numeric measures of aesthetics, as this would require us to have a formal and quantifiable measure of the knot façades. This seems difficult with our currently available tools.

## VII. CONCLUSION

In this paper, we have extended the enumeration methods originally used by Fink and Mao [2] to provide a larger enumeration of necktie knots, including those knots tied with the thin blade of a necktie to produce ornate patterns in the knot façade.

We have found 2 046 winding patterns that take up to 11 moves to tie and are anchored by a final single depth tuck, and thus are reasonable candidates for use with a normal necktie. We chose the number of moves by examining popular thin-blade tie knots – the Eldredge tie knot uses 11 moves. Most of these winding patterns allow several possible tuck patterns, and thus the 2 046 winding sequences generate 177 147 tie knots with single depth tucks.

We have further shown that in the limit, the language describing neck tie knots is either context sensitive or context free, with a regular sub-language describing the 177 147 knots above.

Questions that remain open to our mind include:

- Settle the language complexity class of the full tie knot language.
- Find a way to algorithmically divide a knot description string into a body/façade distinction.
- Using such a distinction, classify all possible knot façades with reasonably short necktie lengths.

## REFERENCES

- [1] T. Fink and Y. Mao. *The 85 ways to tie a tie*. Fourth Estate, 2001.
- [2] T. Fink and Y. Mao. “Tie knots, random walks and topology”. In: *Physica A: Statistical Mechanics and its Applications* 276.1 (2000), pp. 109–121.
- [3] T.M. Fink and Y. Mao. “Designing tie knots by random walks”. In: *Nature* 398.6722 (1999), pp. 31–32.
- [4] A. Knorr. *eldredge reloaded*. Blog post on <http://xirdalium.net>. Accessed on 26 December 2012. June 2010. URL: <http://xirdalium.net/2010/06/20/eldredge-reloaded/>.
- [5] D. Knuth. “Semantics of context-free languages”. In: *Mathematical Systems Theory* 2.2 (1967), pp. 127–145.
- [6] A. Krasny. *Eldredge Tie Knot - How to Tie a Eldredge Necktie Knot*. Blog post on <http://agreeordie.com>. Accessed on 26 December 2012. Nov. 2012. URL: <http://agreeordie.com/blog/musings/545-how-to-tie-a-necktie-eldredge-knot>.
- [7] A. Krasny. *Trinity Tie Knot - How to Tie a Trinity Necktie Knot*. Blog post on <http://agreeordie.com>. Accessed on 26 December 2012. Dec. 2012. URL: <http://agreeordie.com/blog/musings/553-how-to-tie-a-necktie-trinity-knot>.
- [8] William Ogden. “A helpful result for proving inherent ambiguity”. In: *Theory of Computing Systems* 2.3 (1968), pp. 191–194.
- [9] William Ogden, Rockford J Ross, and Karl Winklmann. “An “interchange lemma” for context-free languages”. In: *SIAM Journal on Computing* 14.2 (1985), pp. 410–415.
- [10] Rohit J Parikh. “On context-free languages”. In: *Journal of the ACM (JACM)* 13.4 (1966), pp. 570–581.
- [11] Michael Sipser. *Introduction to the Theory of Computation*. Vol. 2. Thomson Course Technology Boston, 2006.
- [12] A. Wachowski et al. *The matrix reloaded*. Warner Bros. Pictures, 2003.
- [13] David S Wise. “A strong pumping lemma for context-free languages”. In: *Theoretical Computer Science* 3.3 (1976), pp. 359–369.

## APPENDIX

**Classical ties in WTU notation**

The 85 ties enumerated by Fink and Mao [2] are given by the following winding strings in our WTU notation for tie knots:

Index	Tie string	Index	Tie string
1	WWU	43	WTWTWTTU
2	WTTU	44	WTWWWTTU
3	WTWU	45	WTTTWTU
4	TTTTU	46	WTTTTTU
5	TWWU	47	WWWTTU
6	WTWTTU	48	WTWTTWU
7	WWWWU	49	WTWTTWU
8	WTTTTU	50	WTWTTWU
9	TTTWU	51	WTTTWTU
10	TWTTU	52	TTTWTU
11	WTWTWU	53	TWTTWU
12	WTTTTU	54	WWWWWWWU
13	WWWTTU	55	WTTTTTTU
14	WTTWWU	56	WWWTTTU
15	WTTWU	57	TTTWWWWU
16	TTTWTU	58	TTTTTWWU
17	TWTTWU	59	WTTTWWU
18	TTWWWWU	60	TTTWTTU
19	TTWTTU	61	WTTWWWWU
20	TWTTTTU	62	WWWTTWU
21	TWTWWU	63	WTTTWTU
22	WTWTWTTU	64	WTTWWTTU
23	WTTTTWU	65	TWTTWU
24	WTWWWWWU	66	TTWTTWU
25	WWWTTWU	67	TTTWTU
26	WTWTTTU	68	TWTTTTTU
27	WTTTWTU	69	TWWWWWTTU
28	TTTWTWU	70	TTTTWTTU
29	WTTWTTU	71	TWTTWWU
30	TWTTWTTU	72	WTTTTTTU
31	TTTTTTU	73	WWWTTWTTU
32	TWWWWWU	74	WTTTWWU
33	TTTTWWWU	75	WTTTWWU
34	WWWTTTTU	76	TWTTWTTU
35	TWTTTTU	77	TWTTWU
36	WTTWWWWU	78	TTWTTTTU
37	WWWTTWWU	79	TWTTWWWWU
38	TTWTTTU	80	TTTWWWWU
39	WTTTWTU	81	TTWTTWWU
40	TWTTTWU	82	TWTTTTU
41	TTTWTWU	83	TTTWTU
42	TWTTWTTU	84	TWTTTTU
		85	TWTTWWU

### Thin blade active tie knots

The knots that started this project were the Eldredge and the Trinity. Our transcriptions of these ties as demonstrated by Krasny [6, 7] are:

Eldredge	TTTWWTTUTTWWU
Trinity	TWWTTTUTTU

Note that these show up in the enumerations below as L-373 (uuUu) and L-110 (uU).

In the next several pages of this appendix, we shall list enumerations for up to 12 winding moves – including the Eldredge and the Trinity in our enumeration. Throughout, we write u for a potential tuck site, where a single depth tuck is allowed, and U for the final tuck that keeps the tie knot in place. We split the enumeration based on where the final tuck takes place: from L, from R or from C.

We write the TWU strings with a lowercase u for any additionally feasible front tuck site. First ties knots that tuck from the left.

Index	Tie string	Index	Tie string	Index	Tie string
L-1	TTTU	L-41	WWWuTTuWWU	L-81	WWuWTWWuWWU
L-2	WWWU	L-42	WWWuWWuTTU	L-82	WWuTWWuWWU
L-3	TTuWWU	L-43	TTuTTuWTTTU	L-83	WWuWWuWTWWU
L-4	WWuTTU	L-44	TTuTTuTWTTU	L-84	WWuWWuTWWU
L-5	TWTTTU	L-45	TTuWTTTuTTU	L-85	TTTuTTuTTuTTU
L-6	TTWTTU	L-46	TTuTWTTuTTU	L-86	TTTuTTuWTWWU
L-7	WTTuTTU	L-47	WTTTuTTuTTU	L-87	TTTuTTuTWWU
L-8	TWWuWWU	L-48	TWTTuTTuTTU	L-88	TTTuWTTTuWWU
L-9	WWTWWU	L-49	TTuTTuWWuWWU	L-89	TTTuWTWWuTTU
L-10	WTWWU	L-50	TTuWTWWU	L-90	TTTuTWTTuWWU
L-11	TTuTTuTTU	L-51	TTuWTTWWU	L-91	TTTuTWWuTTU
L-12	TTuWTWWU	L-52	TTuTWWTWWU	L-92	TTTuWWuWTTTU
L-13	TTuTWWU	L-53	TTuTWTWWU	L-93	TTTuWWuTWTTU
L-14	WTTTuWWU	L-54	TTuWWuTTuWWU	L-94	TWTTTuTTuWWU
L-15	WTWWuTTU	L-55	TTuWWuWWuTTU	L-95	TWTTTuWWuTTU
L-16	TWTTuWWU	L-56	WTTTuWTWWU	L-96	TWTWTWTTU
L-17	TWWuTTU	L-57	WTTTuTWWU	L-97	TWTWTWTTU
L-18	WWuWTTTU	L-58	WTWTTTuWWU	L-98	TWTTWTTTU
L-19	WWuTWTTU	L-59	WTWTWWuTTU	L-99	TWTTWTWTTU
L-20	WWuWWuWWU	L-60	WTTWTuWWU	L-100	TWTWWuTTuTTU
L-21	TTTuTTuWWU	L-61	WTTWWuTTU	L-101	TTWTTuTTuWWU
L-22	TTTuWWuTTU	L-62	WTWWuWTTTU	L-102	TTWTTuWWuTTU
L-23	TWTWTTTU	L-63	WTWWuTWTTU	L-103	TTWWTWTTTU
L-24	TWTTWTTU	L-64	TWTTuWTWWU	L-104	TTWWTWTTU
L-25	TTWTTTU	L-65	TWTTuTWWU	L-105	TTWWTWTTU
L-26	TTWTWTTU	L-66	TWTTTuWWU	L-106	TTWWTWTTU
L-27	TWWuTTuTTU	L-67	TWWTWWuTTU	L-107	TTWwwuTTuTTU
L-28	WTTuWTTTU	L-68	TWTTTuWWU	L-108	TWWuTTuWTTTU
L-29	WTTuTWTTU	L-69	TWTWWuTTU	L-109	TWWuTTuTWTTU
L-30	WTTTuTTU	L-70	TWWuWTTTU	L-110	TWWuWTTTuTTU
L-31	WTWTTuTTU	L-71	TWWuTWTTU	L-111	TWWuTWTTuTTU
L-32	TWTWWuWWU	L-72	WWuTTuTTuWWU	L-112	WTTuTTuTTuWWU
L-33	TTWWuWWU	L-73	WWuTTuWWuTTU	L-113	WTTuTTuWWuTTU
L-34	TWWuWTWWU	L-74	WWuWTWTTTU	L-114	WTTuWTWTTTU
L-35	TWWuTWWU	L-75	WWuWTTWTTU	L-115	WTTuWTTWTTU
L-36	WTTuWWuWWU	L-76	WWuTWWTTTU	L-116	WTTuTWWTTTU
L-37	WWTWTWWU	L-77	WWuTWTTTU	L-117	WTTuTWTTTU
L-38	WTTTWWU	L-78	WWuWWuTTuTTU	L-118	WTTuWWuTTuTTU
L-39	WTWTTWWU	L-79	WTWWuWWuWWU	L-119	WTTTuWTTTU
L-40	WTWTWWU	L-80	TWWuWWuWWU	L-120	WTTTuTWTTU

Index	Tie string	Index	Tie string	Index	Tie string
L-121	WWTWTTTuTTU	L-171	TTuTTuTTuTTuWWU	L-221	TTuWWuWTWWuTTU
L-122	WWTWTuTTU	L-172	TTuTTuTTuWWuTTU	L-222	TTuWWuTWTuWWU
L-123	WTWTuWTTU	L-173	TTuTTuWTWTTU	L-223	TTuWWuTWWWuTTU
L-124	WTWTuTWTTU	L-174	TTuTTuWTTWTTU	L-224	TTuWWuWWuWTTTU
L-125	WTWWTTTuTTU	L-175	TTuTTuTWWTuTTU	L-225	TTuWWuWWuTWTTU
L-126	WTWTuTTuTTU	L-176	TTuTTuTWTWTTU	L-226	WTTTuTTuWWuWWU
L-127	WWWuTTuTTuTTU	L-177	TTuTTuWWuTTuTTU	L-227	WTTTuWTWTWWU
L-128	TTuWWuWWuWWU	L-178	TTuWTTTuWTTTU	L-228	WTTTuWTTWWWU
L-129	TWTWTWWuWWU	L-179	TTuWTTTuTWTTU	L-229	WTTTuTWWTWWU
L-130	TWTWWWuWWU	L-180	TTuWTWTTTuTTU	L-230	WTTTuTWTWWWU
L-131	TWTWWuWTWWU	L-181	TTuWTTWTTuTTU	L-231	WTTTuWWuTTuWWU
L-132	TWTWWuTWWWU	L-182	TTuTWTTuWTTTU	L-232	WTTTuWWuWWuTTU
L-133	TTWWTWWuWWU	L-183	TTuTWTTuTWTTU	L-233	WTWTTTuWTWWU
L-134	TTWTWWWuWWU	L-184	TTuTWWTTuTTU	L-234	WTWTTTuTWWWU
L-135	TTWWuWTWWU	L-185	TTuTWTWTTuTTU	L-235	WTWTWTTTuWWU
L-136	TTWWuTWWWU	L-186	TTuWWuTTuTTuTTU	L-236	WTWTWTTuTTU
L-137	TWWuTTuWWuWWU	L-187	WTTTuTTuWTTTU	L-237	WTWTTWTTuWWU
L-138	TWWuWTWTWWU	L-188	WTTTuTTuTWTTU	L-238	WTWTTWWWuTTU
L-139	TWWuTWTWWWU	L-189	WTTTuWTTTuTTU	L-239	WTWTWWuWTTTU
L-140	TWWuTWWTWWU	L-190	WTTTuTWTTuTTU	L-240	WTWTWWuTWTTU
L-141	TWWuTWTWWWU	L-191	WTWTTTuTTuTTU	L-241	WTTWTTuWTWWU
L-142	TWWuWWuTTuWWU	L-192	WTTWTTuTTuTTU	L-242	WTTWTTuTWWWU
L-143	TWWuWWuWWuTTU	L-193	TWTTuTTuWTTTU	L-243	WTTWTTTuWWU
L-144	WTuWTWWuWWU	L-194	TWTTuTTuTWTTU	L-244	WTTWWTWWuTTU
L-145	WTuTWWWuWWU	L-195	TWTTuWTTTuTTU	L-245	WTTWTWTTuWWU
L-146	WTuWWuWTWWU	L-196	TWTTuTWTTuTTU	L-246	WTTWTWWWuTTU
L-147	WTuWWuTWWWU	L-197	TWTTTuTTuTTU	L-247	WTTWWWuWTTTU
L-148	WTTTuWWuWWU	L-198	TWTWTTuTTuTTU	L-248	WTTWWWuTWTTU
L-149	WWTWTWTTU	L-199	WWuTTuTTuTTuTTU	L-249	WTWWuTTuTTuWWU
L-150	WWTWTWWWU	L-200	TTuTTuWTWWuWWU	L-250	WTWWuTTuWWuTTU
L-151	WWTTWWTWWU	L-201	TTuTTuTWWWuWWU	L-251	WTWWuWTWTTTU
L-152	WWTWTWWWU	L-202	TTuTTuWWuWTWWU	L-252	WTWWuWTTWTTU
L-153	WWTWWuTTuWWU	L-203	TTuTTuWWuTWWWU	L-253	WTWWuTWTWTTU
L-154	WWTWWuWWuTTU	L-204	TTuWTTTuWWuWWU	L-254	WTWWuTWTWTTU
L-155	WTWTuWWuWWU	L-205	TTuWTWTWTTWWU	L-255	WTWWuWWuTTuTTU
L-156	WTWWTWTWWU	L-206	TTuWTWTWTTWWWU	L-256	TWTTuTTuWWuWWU
L-157	WTWWTTWWWU	L-207	TTuWTWTWTTWWU	L-257	TWTTuWTWTWWU
L-158	WTWTWTTWWU	L-208	TTuWTTWTWTTWWWU	L-258	TWTTuWTTWTTWWWU
L-159	WTWTWTTWWWU	L-209	TTuWTWTTuTTuWWU	L-259	TWTTuTWWTWWU
L-160	WTWWuTTuWWU	L-210	TTuWTWTTuWWuTTU	L-260	TWTTuTWTWTTWWWU
L-161	WTWWuWWuTTU	L-211	TTuTWTTuWWuWWU	L-261	TWTTuWWuTTuWWU
L-162	WWWuTTuWTWWU	L-212	TTuTWTWTTWTTWWU	L-262	TWTTuWWuWWuTTU
L-163	WWWuTTuTWWWU	L-213	TTuTWTWTTWTTWWWU	L-263	TWTTTuWTWTTWWU
L-164	WWWuWTTTuWWU	L-214	TTuTWTWTTWTTWWU	L-264	TWTTTTuTWWWU
L-165	WWWuWTWTTuTTU	L-215	TTuTWTWTTWTTWWWU	L-265	TWTTWTTTuWWU
L-166	WWWuTWTTuWWU	L-216	TTuTWWWuTTuWWU	L-266	TWTTWTTWWuTTU
L-167	WWWuTWWWuTTU	L-217	TTuTWWWuWWuTTU	L-267	TWTTWTTTuWWU
L-168	WWWuWWuWTTTU	L-218	TTuWWuTTuWTWTTWWU	L-268	TWTTWTTWWWuTTU
L-169	WWWuWWuTWTTU	L-219	TTuWWuTTuTWWWU	L-269	TWTTWTTWTTTU
L-170	WWWuWWuWWuWWU	L-220	TTuWWuWTTTuWWU	L-270	TWTTWTTWTTTU

Index	Tie string	Index	Tie string
L-271	TWTWTTuWTWWU	L-321	TWWWuWTWWuWWWU
L-272	TWTWTTuTWWWU	L-322	TWWWuTWWWuWWWU
L-273	TWTWWTTTuWWWU	L-323	TWWWuWWuWTWWU
L-274	TWTWWTWWuTTU	L-324	TWWWuWWuTWWWU
L-275	TWTWTTuWWWU	L-325	WWuTTuWWuWWuWWWU
L-276	TWTWTTWWuTTU	L-326	WWuWTWTTWWuWWWU
L-277	TWTWWWuTTTTU	L-327	WWuWTTWWWuWWWU
L-278	TWTWWWuTWTU	L-328	WWuWTWWuWTWWU
L-279	TWWWuTTuTTuWWWU	L-329	WWuWTWWuTWWWU
L-280	TWWWuTTuWWuTTU	L-330	WWuTWTWWuWWWU
L-281	TWWWuWTWTTU	L-331	WWuTWTWWWuWWWU
L-282	TWWWuWTTWTTU	L-332	WWuTWWWuWTWWU
L-283	TWWWuTWWTTU	L-333	WWuTWWWuTWWWU
L-284	TWWWuTWTWTTU	L-334	WWuWWuTTuWWuWWWU
L-285	TWWWuWWuTTuTTU	L-335	WWuWWuWTWTTWWU
L-286	WWuTTuTTuWTWWU	L-336	WWuWWuWTTWWWU
L-287	WWuTTuTTuTWWWU	L-337	WWuWWuTWTWWU
L-288	WWuTTuWTTTuWWWU	L-338	WWuWWuTWTWWWU
L-289	WWuTTuWTWWuTTU	L-339	WWuWWuWWuTTuWWWU
L-290	WWuTTuTWTTuWWWU	L-340	WWuWWuWWuWWuTTU
L-291	WWuTTuTWWWuTTU	L-341	TTTuTTuTTuWTTTTU
L-292	WWuTTuWWuWTTU	L-342	TTTuTTuTTuTWTU
L-293	WWuTTuWWuTWTU	L-343	TTTuTTuWTTTuTTU
L-294	WWuWTTTuTTuWWWU	L-344	TTTuTTuTWTTuTTU
L-295	WWuWTTTuWWuTTU	L-345	TTTuWTTTuTTuTTU
L-296	WWuWTWTTU	L-346	TTTuTWTTuTTuTTU
L-297	WWuWTWTTU	L-347	TWTTTuTTuTTuTTU
L-298	WWuWTTWTTU	L-348	TTWTTuTTuTTuTTU
L-299	WWuWTTWTTU	L-349	WTTuTTuTTuTTuTTU
L-300	WWuWTWWuTTuTTU	L-350	TTTuTTuTTuWWuWWWU
L-301	WWuTWTTuTTuWWWU	L-351	TTTuTTuWTWTTWWU
L-302	WWuTWTTuWWuTTU	L-352	TTTuTTuWTTWWWU
L-303	WWuTWWTWTU	L-353	TTTuTTuTWWTTWWU
L-304	WWuTWWTTWTTU	L-354	TTTuTTuTWTWWWU
L-305	WWuTWTWTTU	L-355	TTTuTTuWWuTTuWWWU
L-306	WWuTWTWTTU	L-356	TTTuTTuWWuWWuTTU
L-307	WWuTWWWuTTuTTU	L-357	TTTuWTTTuWTWWU
L-308	WWuWWuTTuWTTU	L-358	TTTuWTTTuTWWWU
L-309	WWuWWuTTuTWTU	L-359	TTTuWTWTTTuWWWU
L-310	WWuWWuWTTTuTTU	L-360	TTTuWTWTTWWuTTU
L-311	WWuWWuTWTTuTTU	L-361	TTTuWTTWTTuWWWU
L-312	TTuWWuWWuWWuWWWU	L-362	TTTuWTTWWWuTTU
L-313	WTWTTWWuWWuWWWU	L-363	TTTuWTWWuWTTTTU
L-314	WTTWWWuWWuWWWU	L-364	TTTuWTWWuTWTU
L-315	WTWWuWTWWuWWWU	L-365	TTTuTWTTuWTWWU
L-316	WTWWuTWWWuWWWU	L-366	TTTuTWTTuTWWWU
L-317	WTWWuWWuWTWWU	L-367	TTTuTWWTTTuWWWU
L-318	WTWWuWWuTWWWU	L-368	TTTuTWWTTWWuTTU
L-319	TWWTWWuWWuWWWU	L-369	TTTuTWTWTTuWWWU
L-320	TWTWWWuWWuWWWU	L-370	TTTuTWTWWWuTTU

Index	Tie string	Index	Tie string
L-371	TTTuTWWWuWTTTTU	L-421	TTWTWTTuTTuWWU
L-372	TTTuTWWWuTWTTU	L-422	TTWTWTTuWWuTTU
L-373	TTTuWWuTTuTTuWWU	L-423	TTWTWWTWTTTTU
L-374	TTTuWWuTTuWWuTTU	L-424	TTWTWWTWTTU
L-375	TTTuWWuWTWTTTTU	L-425	TTWTWWTWTTTTU
L-376	TTTuWWuWTTWTTU	L-426	TTWTWWTWTTU
L-377	TTTuWWuTWTTTTU	L-427	TTWTWWWuTTuTTU
L-378	TTTuWWuTWTWTTU	L-428	TTWWWuTTuWTTTTU
L-379	TTTuWWuWWuTTuTTU	L-429	TTWWWuTTuTWTTU
L-380	TWTTTuTTuWTWWU	L-430	TTWWWuWTTTTuTTU
L-381	TWTTTuTTuTWWWU	L-431	TTWWWuTWTTuTTU
L-382	TWTTTuWTTTuWWU	L-432	TWWuTTuTTuTTuWWU
L-383	TWTTTuWTWWuTTU	L-433	TWWuTTuTTuWWuTTU
L-384	TWTTTuTWTTuWWU	L-434	TWWuTTuWTWTTTTU
L-385	TWTTTuTWWWuTTU	L-435	TWWuTTuWTTWTTU
L-386	TWTTTuWWuWTTTTU	L-436	TWWuTTuTWWTTTTU
L-387	TWTTTuWWuTWTU	L-437	TWWuTTuTWTWTTU
L-388	TWTWTTTuTTuWWU	L-438	TWWuTTuWWuTTuTTU
L-389	TWTWTTTuWWuTTU	L-439	TWWuWTTTTuWTTTTU
L-390	TWTWWTWTTTTU	L-440	TWWuWTTTuTWTTU
L-391	TWTWWTWTTTU	L-441	TWWuWTTWTTTTuTTU
L-392	TWTWTTWTTTTU	L-442	TWWuWTTWTTuTTU
L-393	TWTWTTWTWTTU	L-443	TWWuTWTTuWTTTTU
L-394	TWTWTTWuTTuTTU	L-444	TWWuTWTTuTWTTU
L-395	TWTTWTTuTTuWWU	L-445	TWWuTWWTTTTuTTU
L-396	TWTTWTTuWWuTTU	L-446	TWWuTWTWTTuTTU
L-397	TWTTWWTWTTTTU	L-447	TWWuWWuTTuTTuTTU
L-398	TWTTWWTWTTU	L-448	WTTuTTuTTuWTWWU
L-399	TWTTWTWWTUU	L-449	WTTuTTuTTuTWWWU
L-400	TWTTWTWTTTU	L-450	WTTuTTuWTTTTuWWU
L-401	TWTTWWWuTTuTTU	L-451	WTTuTTuWTWWuTTU
L-402	TWTWwuTTuWTTTTU	L-452	WTTuTTuTWTTuWWU
L-403	TWTWwuTTuTWTTU	L-453	WTTuTTuTWWWuTTU
L-404	TWTWwuWTTTTuTTU	L-454	WTTuTTuWWuWTTTTU
L-405	TWTWwuTWTTuTTU	L-455	WTTuTTuWWuTWTTU
L-406	TTWTTuTTuWTWWU	L-456	WTTuWTTTTuTTuWWU
L-407	TTWTTuTTuTWWWU	L-457	WTTuWTTTTuWWuTTU
L-408	TTWTTuWTTTuWWU	L-458	WTTuWTWWTWTTTTU
L-409	TTWTTuWTWWuTTU	L-459	WTTuWTWTTWTTU
L-410	TTWTTuTWTTuWWU	L-460	WTTuWTTWWTUU
L-411	TTWTTuTWWWuTTU	L-461	WTTuWTTWWTWTTU
L-412	TTWTTuWWuWTTTTU	L-462	WTTuWTTWuTTuTTU
L-413	TTWTTuWWuTWTTU	L-463	WTTuTWTTuTTuWWU
L-414	TTWTTTTuTTuWWU	L-464	WTTuTWTTuWWuTTU
L-415	TTWTTTTuWWuTTU	L-465	WTTuTWWTWTTTTU
L-416	TTWTTWTTTTU	L-466	WTTuTWWTTWTTU
L-417	TTWTTWTTWTTU	L-467	WTTuTWTTWTTTTU
L-418	TTWTTWTTTTU	L-468	WTTuTWTTWTTU
L-419	TTWTTWTTWTTU	L-469	WTTuTWWWuTTuTTU
L-420	TTWTTWwuTTuTTU	L-470	WTTuWWuTTuWTTTTU



Index	Tie string	Index	Tie string
L-471	WTTuWWuTTuTWTTU	L-521	TWTWTWwuWTWWU
L-472	WTTuWWuWTTTuTTU	L-522	TWTWTWwuTWWWU
L-473	WTTuWWuTWTTuTTU	L-523	TWTTWWTWwuWWU
L-474	WWTTTuTTuTTuWWU	L-524	TWTTWTWwWuWWU
L-475	WWTTTuTTuWWuTTU	L-525	TWTTWwWuWTWWU
L-476	WWTTTuWTWTTTU	L-526	TWTTWwWuTWWWU
L-477	WWTTTuWTTWTTU	L-527	TWTWwuTTuWwWuWWU
L-478	WWTTTuTWTTTU	L-528	TWTWwuWTWTWwU
L-479	WWTTTuTWTTTU	L-529	TWTWwuWTTWWWU
L-480	WWTTTuWwuTTuTTU	L-530	TWTWwuTWWTWWU
L-481	WWTWTTTuWTTTU	L-531	TWTWwuTWTWWWU
L-482	WWTWTTTuTWTTU	L-532	TWTWwuWwuTTuWwU
L-483	WWTWTTTuTTU	L-533	TWTWwuWwWuWwU
L-484	WWTWTTWTTuTTU	L-534	TTWTTuWwWuWwWuWWU
L-485	WWTTWTTuWTTTU	L-535	TTWWTWTWwWuWWU
L-486	WWTTWTTuTWTTU	L-536	TTWWTWWWuWwU
L-487	WWTTWTTTuTTU	L-537	TTWWTWwWuWTWwU
L-488	WWTTWTWTTuTTU	L-538	TTWWTWwWuTWWWU
L-489	WWTWwuTTuTTuTTU	L-539	TTWTWWTWwWuWWU
L-490	WTWTTuTTuTTuWwU	L-540	TTWTWTWwWuWWU
L-491	WTWTTuTTuWwWuTTU	L-541	TTWTWWWuWTWwU
L-492	WTWTTuWTWTTTU	L-542	TTWTWWWuTWWWU
L-493	WTWTTuWTTWTTU	L-543	TTWwWuTTuWwWuWWU
L-494	WTWTTuTWTTTU	L-544	TTWwWuWTWTWwU
L-495	WTWTTuTWTTTU	L-545	TTWwWuWTTWWWU
L-496	WTWTTuWwWuTTuTTU	L-546	TTWwWuTWWTWWU
L-497	WTWTTTuWTTTU	L-547	TTWwWuTWTWWWU
L-498	WTWTTTuTWTTU	L-548	TTWwWuWwWuTTuWwU
L-499	WTWWTWTTTuTTU	L-549	TTWwWuWwWuWwU
L-500	WTWWTWTTuTTU	L-550	TWwWuTTuWTWwWuWWU
L-501	WTWTWTTuWTTTU	L-551	TWwWuTTuTWWwWuWWU
L-502	WTWTWTTuTWTTU	L-552	TWwWuTTuWwWuWTWwU
L-503	WTWTWWTTTuTTU	L-553	TWwWuTTuWwWuTWWWU
L-504	WTWTWTWTTuTTU	L-554	TWwWuTTTuWwWuWWU
L-505	WTWwWuTTuTTuTTU	L-555	TWwWuTWTWTWwU
L-506	WwWuTTuTTuWTTTU	L-556	TWwWuTWTWWWU
L-507	WwWuTTuTTuTWTTU	L-557	TWwWuTWTWWTWWU
L-508	WwWuTTuWTTTuTTU	L-558	TWwWuTWTWWWU
L-509	WwWuTTuTWTTuTTU	L-559	TWwWuTWWuTTuWwU
L-510	WwWuWTTTuTTuTTU	L-560	TWwWuTWWuWwWuTTU
L-511	WwWuTWTTuTTuTTU	L-561	TWwWuTWTuWwWuWWU
L-512	TTTuWTTWwWuWwWuWWU	L-562	TWwWuTWWTWTWwU
L-513	TTTuTWWWuWwWuWWU	L-563	TWwWuTWWTWWWU
L-514	TTTuWwWuWTWwWuWWU	L-564	TWwWuTWTWWTWWU
L-515	TTTuWwWuTWWWuWWU	L-565	TWwWuTWTWWWU
L-516	TTTuWwWuWwWuWTWwU	L-566	TWwWuTWWWuTTuWwU
L-517	TTTuWwWuWwWuTWWWU	L-567	TWwWuTWWWuWwWuTTU
L-518	TWTTTuWwWuWwWuWWU	L-568	TWwWuWwWuTTuWTWwU
L-519	TWTWTWTWwWuWWU	L-569	TWwWuWwWuTTuTWWWU
L-520	TWTWTWWWuWwWu	L-570	TWwWuWwWuWTTTuWwU

Index	Tie string	Index	Tie string
L-571	TWWuWWuWTWWuTTU	L-621	WTWTTuWWuTWWWU
L-572	TWWuWWuTWTTuWWU	L-622	WTWTTTtuWWuWWU
L-573	TWWuWWuTWWWuTTU	L-623	WTWWTWTWWWU
L-574	TWWuWWuWWuWTTTU	L-624	WTWWTWTWWWU
L-575	TWWuWWuWWuTWTTU	L-625	WTWWTWWWWWWU
L-576	WTTuTTuWWuWWuWWU	L-626	WTWWTWTWWWU
L-577	WTTuWTWTWWWuWWU	L-627	WTWWTWWuTTuWWU
L-578	WTTuWTTWWWuWWU	L-628	WTWWTWWuWWuTTU
L-579	WTTuWTWWuWTWWU	L-629	WTWTWTTuWWuWWU
L-580	WTTuWTWWuTWWWU	L-630	WTWTWWTWWWU
L-581	WTTuTWWTWWWuWWU	L-631	WTWTWWTWWWU
L-582	WTTuTWTWWWuWWU	L-632	WTWTWWTWWWU
L-583	WTTuTWWWuWTWWU	L-633	WTWTWWTWWWU
L-584	WTTuTWWWuTWWWU	L-634	WTWTWWWuTTuWWU
L-585	WTTuWWuTTuWWuWWU	L-635	WTWTWWWuWWuTTU
L-586	WTTuWWuWTWTWWWU	L-636	WTWWWuTTuWTWWWU
L-587	WTTuWWuWTTWWWU	L-637	WTWWWuTTuTWWWU
L-588	WTTuWWuTWWTWWWU	L-638	WTWWWuWTTTtuWWU
L-589	WTTuWWuTWTWWWU	L-639	WTWWWuWTWWuTTU
L-590	WTTuWWuWWuTTuWWU	L-640	WTWWWuTWTTuWWU
L-591	WTTuWWuWWuWWuTTU	L-641	WTWWWuTWWWuTTU
L-592	WWTTTuWTWWuWWU	L-642	WTWWWuWWuWTTTU
L-593	WWTTTuTWWWuWWU	L-643	WTWWWuWWuTWTTU
L-594	WWTTTuWWuWTWWU	L-644	WWWuTTuTTuWWuWWU
L-595	WWTTTuWWuTWWWU	L-645	WWWuTTuWTWTWWWU
L-596	WWTWTTTuWWuWWU	L-646	WWWuTTuWTTWWWU
L-597	WWTWTWTWWWU	L-647	WWWuTTuTWWTWWWU
L-598	WWTWTWTWWWU	L-648	WWWuTTuTWTWWWU
L-599	WWTWTWTWWWU	L-649	WWWuTTuWWuTTuWWU
L-600	WWTWTWTWWWU	L-650	WWWuTTuWWuWWuTTU
L-601	WWTWTWWuTTuWWU	L-651	WWWuWTTTuWTWWWU
L-602	WWTWTWWuWWuTTU	L-652	WWWuWTTTuTWWWU
L-603	WWTWTWTuWWuWWU	L-653	WWWuWTWTTTuWWU
L-604	WWTWTWTWWWU	L-654	WWWuWTWTWWuTTU
L-605	WWTWTWTWWWU	L-655	WWWuWTTWTTuWWU
L-606	WWTWTWTWWWU	L-656	WWWuWTTWWWuTTU
L-607	WWTWTWTWWWU	L-657	WWWuWTWWuWTTTU
L-608	WWTTWWWuTTuWWU	L-658	WWWuWTWWuTWTTU
L-609	WWTTWWWuWWuTTU	L-659	WWWuTWTTuWTWWWU
L-610	WWTWWuTTuWTWWWU	L-660	WWWuTWTTuTWWWU
L-611	WWTWWuTTuTWWWU	L-661	WWWuTWWTTTuWWU
L-612	WWTWWuWTTTuWWU	L-662	WWWuTWWTWWuTTU
L-613	WWTWWuWTWWuTTU	L-663	WWWuTWTTTuWWU
L-614	WWTWWuTWTTuWWU	L-664	WWWuTWTTWWWuTTU
L-615	WWTWWuTWWWuTTU	L-665	WWWuTWWWuWTTTU
L-616	WWTWWuWWuWTTTU	L-666	WWWuTWWWuTWTTU
L-617	WWTWWuWWuTWTTU	L-667	WWWuWWuTTuTTuWWU
L-618	WTWTTuWTWWuWWU	L-668	WWWuWWuTTuWWuTTU
L-619	WTWTTuTWWWuWWU	L-669	WWWuWWuWTWTTTU
L-620	WTWTTuWWuWTWWWU	L-670	WWWuWWuWTTWTTU

Index	Tie string
L-671	WWWuWWuTWWTTTU
L-672	WWWuWWuTWTWTTU
L-673	WWWuWWuWWuTTuTTU
L-674	TWWuWWuWWuWWuWWU
L-675	WWTWWuWWuWWuWWU
L-676	WTWWWuWWuWWuWWU
L-677	WWWuWTWWuWWuWWU
L-678	WWWuTWWWuWWuWWU
L-679	WWWuWWuWTWWuWWU
L-680	WWWuWWuTWWWuWWU
L-681	WWWuWWuWWuWTWWU
L-682	WWWuWWuWWuTWWWU

Next, we list the knots that tuck from the right. Again, with all optional single depth front tuck sites marked with a lowercase u.

Index	Tie string	Index	Tie string	Index	Tie string
R-1	TTU	R-41	WWWuTWTTU	R-81	WWuWTTWWWU
R-2	TWU	R-42	WWWuWWuWWU	R-82	WWuTWWTWWU
R-3	WTTTTU	R-43	TTuTTuTTuTTU	R-83	WWuTWTWWWU
R-4	TWTTU	R-44	TTuTTuWTWWU	R-84	WWuWWuTTuWWU
R-5	WWuWWU	R-45	TTuTTuTWWWU	R-85	WWuWWuWWuTTU
R-6	TTTuTTU	R-46	TTuWTTTuWWU	R-86	TTTuTTuTTuWWU
R-7	TWTWWU	R-47	TTuWTTWuTTU	R-87	TTTuTTuWWuTTU
R-8	TTWWWU	R-48	TTuTWTTuWWU	R-88	TTTuWTTWTTU
R-9	WTTuWWU	R-49	TTuTWWWuTTU	R-89	TTTuWTTWTTU
R-10	WWWuTTU	R-50	TTuWWuWTTTU	R-90	TTTuTWWTTTU
R-11	TTuTTuWWU	R-51	TTuWWuTWTTU	R-91	TTTuTWWTTTU
R-12	TTuWWuTTU	R-52	WTTTuTTuWWU	R-92	TTTuWWuTTuTTU
R-13	WTWTTTU	R-53	WTTTuWWuTTU	R-93	TWTTTuWTTTU
R-14	WTTWTTU	R-54	WTWTWTTTU	R-94	TWTTTuTWTTU
R-15	TWWTTTU	R-55	WTWTWTTU	R-95	TWTWTTTuTTU
R-16	TWTWTTU	R-56	WTTWTTTU	R-96	TWTWTTTuTTU
R-17	WWuTTuTTU	R-57	WTTWTTTU	R-97	TTWTTuWTTTU
R-18	WTWWuWWU	R-58	WTWWuTTuTTU	R-98	TTWTTuTWTTU
R-19	TWWWuWWU	R-59	TWTTuTTuWWU	R-99	TTWTTTuTTU
R-20	WWuWTWWU	R-60	TWTTuWWuTTU	R-100	TTWTTTuTTU
R-21	WWuTWWWU	R-61	TWWTWTTTU	R-101	TWuTTuTTuTTU
R-22	TTTuWTTTU	R-62	TWWTWTTU	R-102	WTTuTTuWTTTU
R-23	TTTuTWTTU	R-63	TWTWTTTU	R-103	WTTuTTuTWTTU
R-24	TWTTTuTTU	R-64	TWTWTTTU	R-104	WTTuWTTTuTTU
R-25	TTWTTuTTU	R-65	TWWWuTTuTTU	R-105	WTTuTWTTuTTU
R-26	WTTuTTuTTU	R-66	WWuTTuWTTTU	R-106	WTTTuTTuTTU
R-27	TTTuWWuWWU	R-67	WWuTTuTWTTU	R-107	WTWTTuTTuTTU
R-28	TWTWTTU	R-68	WWuWTTTuTTU	R-108	TTTuWTTWuWWU
R-29	TWTTWWWU	R-69	WWuTWTTuTTU	R-109	TTTuTWWWuWWU
R-30	TTWTTWWU	R-70	TTuWWuWWuWWU	R-110	TTTuWWuWTWWU
R-31	TTWTWWWU	R-71	WTWTWuWWU	R-111	TTTuWWuTWWWU
R-32	TWuTTuWWU	R-72	WTTWWWuWWU	R-112	TWTTTuWWuWWU
R-33	TWuWWuTTU	R-73	WTWWuWTWWU	R-113	TWTWTTWU
R-34	WTTuWTWWU	R-74	WTWWuTWWWU	R-114	TWTWTTWWWU
R-35	WTTuTWWWU	R-75	TWWTWuWWU	R-115	TWTTWTTWU
R-36	WTTTuWWU	R-76	TWTWWWuWWU	R-116	TWTTWTTWWWU
R-37	WTTWuTTU	R-77	TWWWuWTWWU	R-117	TWTWuTTuWWU
R-38	WTWTTuWWU	R-78	TWWWuTWWWU	R-118	TWTWuWWuTTU
R-39	WTWWWuTTU	R-79	WWuTTuWWuWWU	R-119	TTWTTuWWuWWU
R-40	WWWuWTTTU	R-80	WWuWTTWU	R-120	TTWTTWTTWU

Index	Tie string	Index	Tie string	Index	Tie string
R-121	TTWWTWWU	R-171	TTuTTuTTuWTTTU	R-221	WTWTTWTTTU
R-122	TTWTWTTWU	R-172	TTuTTuTTuTWTTU	R-222	WTWTTWTTTU
R-123	TTWTWTTWU	R-173	TTuTTuWTTTU	R-223	WTWTTWTTTU
R-124	TTWWTuTTuWU	R-174	TTuTTuTWTTTU	R-224	WTTWTTuTTuWU
R-125	TTWWTuWU	R-175	TTuWTTTU	R-225	WTTWTTuWU
R-126	TWWTuTTuWU	R-176	TTuTWTTTU	R-226	WTTWTTWTTTU
R-127	TWWTuTTuWU	R-177	WTTTU	R-227	WTTWTTWTTTU
R-128	TWWTuTTuWU	R-178	TWTTuTTuTTU	R-228	WTTWTTWTTTU
R-129	TWWTuTTuWU	R-179	TTuTTuTTuWU	R-229	WTTWTTWTTTU
R-130	TWWTuTTuWU	R-180	TTuTTuWTTWU	R-230	WTTWTTuTTuTTU
R-131	TWWTuTTuWU	R-181	TTuTTuWTTWU	R-231	WTWWTuTTuTTU
R-132	TWWTuTTuWU	R-182	TTuTTuWTTWU	R-232	WTWWTuTTuTTU
R-133	TWWTuTTuWU	R-183	TTuTTuWTTWU	R-233	WTWWTuTTuTTU
R-134	WTTuTTuWU	R-184	TTuTTuWU	R-234	WTWWTuTTuTTU
R-135	WTTuTTuWU	R-185	TTuTTuWU	R-235	TWTTuTTuWU
R-136	WTTuTTuWU	R-186	TTuWTTTU	R-236	TWTTuTTuWU
R-137	WTTuTTuWU	R-187	TTuWTTTU	R-237	TWTTuTTuWU
R-138	WTTuTTuWU	R-188	TTuWTTTU	R-238	TWTTuTTuWU
R-139	WTTuTTuWU	R-189	TTuWTTTU	R-239	TWTTuTTuWU
R-140	WTTuTTuWU	R-190	TTuWTTTU	R-240	TWTTuTTuWU
R-141	WTTuTTuWU	R-191	TTuWTTTU	R-241	TWTTuTTuWU
R-142	WTTuTTuWU	R-192	TTuWTTTU	R-242	TWTTuTTuWU
R-143	WTTuTTuWU	R-193	TTuWTTTU	R-243	TWTTuTTuWU
R-144	WTTuTTuWU	R-194	TTuWTTTU	R-244	TWTTuTTuWU
R-145	WTTuTTuWU	R-195	TTuWTTTU	R-245	TWTTuTTuWU
R-146	WTTuTTuWU	R-196	TTuWTTTU	R-246	TWTTuTTuWU
R-147	WTTuTTuWU	R-197	TTuWTTTU	R-247	TWTTuTTuWU
R-148	WTTuTTuWU	R-198	TTuWTTTU	R-248	TWTTuTTuWU
R-149	WTTuTTuWU	R-199	TTuWTTTU	R-249	TWTTuTTuWU
R-150	WTTuTTuWU	R-200	TTuWTTTU	R-250	TWTTuTTuWU
R-151	WTTuTTuWU	R-201	TTuWTTTU	R-251	TWTTuTTuWU
R-152	WTTuTTuWU	R-202	TTuWTTTU	R-252	TWTTuTTuWU
R-153	WTTuTTuWU	R-203	TTuWTTTU	R-253	TWTTuTTuWU
R-154	WTTuTTuWU	R-204	TTuWTTTU	R-254	TWTTuTTuWU
R-155	WTTuTTuWU	R-205	TTuWTTTU	R-255	TWTTuTTuWU
R-156	WTTuTTuWU	R-206	TTuWTTTU	R-256	TWTTuTTuWU
R-157	WTTuTTuWU	R-207	TTuWTTTU	R-257	TWTTuTTuWU
R-158	WTTuTTuWU	R-208	TTuWTTTU	R-258	TWTTuTTuWU
R-159	WTTuTTuWU	R-209	WTTTU	R-259	TWTTuTTuWU
R-160	WTTuTTuWU	R-210	WTTTU	R-260	TWTTuTTuWU
R-161	WTTuTTuWU	R-211	WTTTU	R-261	WTTuTTuTTuWU
R-162	WTTuTTuWU	R-212	WTTTU	R-262	WTTuTTuTTuWU
R-163	WTTuTTuWU	R-213	WTTTU	R-263	WTTuTTuTTuWU
R-164	WTTuTTuWU	R-214	WTTTU	R-264	WTTuTTuTTuWU
R-165	WTTuTTuWU	R-215	WTTTU	R-265	WTTuTTuTTuWU
R-166	WTTuTTuWU	R-216	WTTTU	R-266	WTTuTTuTTuWU
R-167	WTTuTTuWU	R-217	WTTTU	R-267	WTTuTTuTTuWU
R-168	WTTuTTuWU	R-218	WTTTU	R-268	WTTuTTuTTuWU
R-169	WTTuTTuWU	R-219	WTTTU	R-269	WTTuTTuTTuWU
R-170	WTTuTTuWU	R-220	WTTTU	R-270	WTTuTTuTTuWU

Index	Tie string	Index	Tie string
R-271	WWuWTTWTTuTTU	R-321	WWuWTWTTWWU
R-272	WWuTWTTuWTTTU	R-322	WWuWTTWTTWWU
R-273	WWuTWTTuTWTTU	R-323	WWuWTTWTWWU
R-274	WWuTWWTTTuTTU	R-324	WWuTWWuTTuWWU
R-275	WWuTWTWTTuTTU	R-325	WWuTWWuWWuTTU
R-276	WWuWWuTTuTTuTTU	R-326	WWuTWTTuWWuWWU
R-277	TTuWTWuWWuWWU	R-327	WWuTWWTTWWU
R-278	TTuTWWuWWuWWU	R-328	WWuTWWTTWWU
R-279	TTuWWuTWuWWuWWU	R-329	WWuTWTTWWU
R-280	TTuWWuTWWuWWU	R-330	WWuTWTWTWWU
R-281	TTuWWuWWuTWuWWU	R-331	WWuTWWuTTuWWU
R-282	TTuWWuWWuTWWU	R-332	WWuTWWuWWuTTU
R-283	WTTTuWWuWWuWWU	R-333	WWuWWuTTuTWuWWU
R-284	WTWTWTTWuWWU	R-334	WWuWWuTTuTWWU
R-285	WTWTWuWWuWWU	R-335	WWuWWuWTTTuWWU
R-286	WTWTWuTWuWWU	R-336	WWuWWuTWuWWuTTU
R-287	WTWTWuTWWU	R-337	WWuWWuTWTTuWWU
R-288	WTTWTTWuWWU	R-338	WWuWWuTWWuTTU
R-289	WTTWTTWuWWU	R-339	WWuWWuWWuWTTTU
R-290	WTTWuTWuWWU	R-340	WWuWWuWWuTWTTU
R-291	WTTWuTWWU	R-341	WWuWWuWWuWWuWWU
R-292	WTWuTTuWWuWWU	R-342	TTTuTTuTTuTTuTTU
R-293	WTWuWTWTTWU	R-343	TTTuTTuTTuTWuWWU
R-294	WTWuWTTWuWWU	R-344	TTTuTTuTTuTWWU
R-295	WTWuTWWTTWU	R-345	TTTuTTuWTTTuWWU
R-296	WTWuTWTTWuWWU	R-346	TTTuTTuTWuWWuTTU
R-297	WTWuWWuTTuWWU	R-347	TTTuTTuTWTTuWWU
R-298	WTWuWWuWWuTTU	R-348	TTTuTTuTWWuTTU
R-299	TWTTuWWuWWuWWU	R-349	TTTuTTuWWuWTTTU
R-300	TWTTWTTWuWWU	R-350	TTTuTTuWWuTWTTU
R-301	TWTTWuWWuWWU	R-351	TTTuWTTTuTTuWWU
R-302	TWTTWuTWuWWU	R-352	TTTuWTTTuWWuTTU
R-303	TWTTWuTWWU	R-353	TTTuWTWTWTTTU
R-304	TWTTWTTWuWWU	R-354	TTTuWTWTWTTU
R-305	TWTTWuWWuWWU	R-355	TTTuWTTWTTTU
R-306	TWTTWuTWuWWU	R-356	TTTuWTTWTTU
R-307	TWTTWuTWWU	R-357	TTTuWTWuTTuTTU
R-308	TWTTuTTuWWuWWU	R-358	TTTuTWTTuTTuWWU
R-309	TWTTWTTWuWWU	R-359	TTTuTWTTuWWuTTU
R-310	TWTTuWTTWuWWU	R-360	TTTuTWWTTWTTU
R-311	TWTTuTWWTTWuWWU	R-361	TTTuTWWTTWTTU
R-312	TWTTuTWTTWuWWU	R-362	TTTuTWTTWTTTU
R-313	TWTTuWWuTTuWWU	R-363	TTTuTWTTWTTU
R-314	TWTTuWWuWWuTTU	R-364	TTTuTWWuTTuTTU
R-315	WWuTTuTWuWWuWWU	R-365	TTTuWWuTTuWTTTU
R-316	WWuTTuTWWuWWU	R-366	TTTuWWuTTuTWTTU
R-317	WWuTTuWWuTWuWWU	R-367	TTTuWWuWTTTuTTU
R-318	WWuTTuWWuTWWU	R-368	TTTuWWuTWTTuTTU
R-319	WWuWTTTuWWuWWU	R-369	TWTTTuTTuTTuWWU
R-320	WWuWTWTWTTWU	R-370	TWTTTuTTuWWuTTU

Index	Tie string	Index	Tie string
R-371	TWTTTuWTWTTTU	R-421	WTTuTWTWTTuTTU
R-372	TWTTTuWTTWTTU	R-422	WTTuWWuTTuTTuTTU
R-373	TWTTTuTWWTITU	R-423	WWTTTuTTuWTTTU
R-374	TWTTTuTWTWTTU	R-424	WWTTTuTTuTWTU
R-375	TWTTTuWWuTTuTTU	R-425	WWTTTuWTTTuTTU
R-376	TWTWTTTuWTTTU	R-426	WWTTTuTWTTuTTU
R-377	TWTWTTTuTWTTU	R-427	WWTTTuTTuTTuTTU
R-378	TWTWTTTuTTU	R-428	WWTTTuTTuTTuTTU
R-379	TWTWTTTuTTU	R-429	WTWTTuTTuWTTTU
R-380	TWTTWTTuWTTTU	R-430	WTWTTuTTuTWTU
R-381	TWTTWTTuTWTTU	R-431	WTWTTuWTTTuTTU
R-382	TWTTWTTTuTTU	R-432	WTWTTuTWTTuTTU
R-383	TWTTWTTTuTTU	R-433	WTWTTTuTTuTTU
R-384	TWTWWuTTuTTuTTU	R-434	WTWTTTuTTuTTU
R-385	TTWTTuTTuTTuWWU	R-435	WWWuTTuTTuTTuTTU
R-386	TTWTTuTTuWWuTTU	R-436	TTTuTTuWWuWWuWWU
R-387	TTWTTuWTWTTTU	R-437	TTTuWTWTTWWuWWU
R-388	TTWTTuWTTWTTU	R-438	TTTuWTTWWuWWU
R-389	TTWTTuTWWTITU	R-439	TTTuWTWuWTWU
R-390	TTWTTuTWTWTTU	R-440	TTTuWTWuTWWWU
R-391	TTWTTuWWuTTuTTU	R-441	TTTuTWWTTWWuWWU
R-392	TTWTTTuWTTTU	R-442	TTTuTWTWWuWWU
R-393	TTWTTTuTWTTU	R-443	TTTuTWWuWTWU
R-394	TTWTTWTTTuTTU	R-444	TTTuTWWuTWWWU
R-395	TTWTTWTTTuTTU	R-445	TTTuWWuTTuWWuWWU
R-396	TTWTTTuWTTTU	R-446	TTTuWWuWTWTTWU
R-397	TTWTTTuTWTTU	R-447	TTTuWWuWTTWU
R-398	TTWTTWTTTuTTU	R-448	TTTuWWuTWTWU
R-399	TTWTTWTTTuTTU	R-449	TTTuWWuTWTWWU
R-400	TTWWuTTuTTuTTU	R-450	TTTuWWuWWuTTuWWU
R-401	TWWuTTuTTuWTTTU	R-451	TTTuWWuWWuWWuTTU
R-402	TWWuTTuTTuTWTTU	R-452	TWTTTuWTWuWWU
R-403	TWWuTTuWTTTuTTU	R-453	TWTTTuTWuWWU
R-404	TWWuTTuTWTTuTTU	R-454	TWTTTuWWuWTWU
R-405	TWWuWTTTuTTuTTU	R-455	TWTTTuWWuTWU
R-406	TWWuWTTTuTTuTTU	R-456	TWTWTTTuWWuWWU
R-407	WTTuTTuTTuTTuWWU	R-457	TWTWTTWTTWU
R-408	WTTuTTuTTuWWuTTU	R-458	TWTWTTTWU
R-409	WTTuTTuWTWTTTU	R-459	TWTWTTTWU
R-410	WTTuTTuWTTWTTU	R-460	TWTWTTTWU
R-411	WTTuTTuTWTTTU	R-461	TWTWTTWuTTuWWU
R-412	WTTuTTuTWTWTTU	R-462	TWTWTTWuWWuTTU
R-413	WTTuTTuWWuTTuTTU	R-463	TWTWTTuWWuWWU
R-414	WTTuWTTTuWTTTU	R-464	TWTWTTWTTWU
R-415	WTTuWTTTuTWTTU	R-465	TWTWTTTWU
R-416	WTTuWTTTuTTU	R-466	TWTWTTWTTWU
R-417	WTTuWTTTuTTU	R-467	TWTWTTTWU
R-418	WTTuWTTTuWTTTU	R-468	TWTWTTWuTTuWWU
R-419	WTTuWTTTuTWTTU	R-469	TWTWTTWuWWuTTU
R-420	WTTuWTTTuTTU	R-470	TWTWTTuWTWU

Index	Tie string	Index	Tie string
R-471	TWTWwuTTuTWWwU	R-521	TWwUtwWtTTuWwU
R-472	TWTWwUWTTTuWwU	R-522	TWwUtwWtWwUttU
R-473	TWTWwUWTWwUttU	R-523	TWwUtwTwtTuWwU
R-474	TWTWwUtwTtuWwU	R-524	TWwUtwTwwWuttU
R-475	TWTWwUtwWwUttU	R-525	TWwUtwWwUwTTTU
R-476	TWTWwUwwUwTTTU	R-526	TWwUtwWwUtwTTU
R-477	TWTWwUwwUtwTTU	R-527	TWwUwwUttUttUWwU
R-478	TTWTTuWTWwUWwU	R-528	TWwUwwUttUwwUttU
R-479	TTWTTuTWWwUWwU	R-529	TWwUwwUwTwtTTU
R-480	TTWTTuWwUWTWwU	R-530	TWwUwwUwTTWTTU
R-481	TTWTTuWwUTWWwU	R-531	TWwUwwUtwWTTTU
R-482	TTWWTTTuWwUWwU	R-532	TWwUwwUtwTWTU
R-483	TTWWTWTWwU	R-533	TWwUwwUwwUttUttU
R-484	TTWWTWTtWwU	R-534	WttUttUwTwwUwwU
R-485	TTWWTtWwU	R-535	WttUttUtwWwUwwU
R-486	TTWWTtWtWwU	R-536	WttUttUwwUwTwwU
R-487	TTWWTWwUttUWwU	R-537	WttUttUwwUtwWwU
R-488	TTWWTWwUwwUttU	R-538	WttUwTTTuwwUwwU
R-489	TTWTWTTuWwUWwU	R-539	WttUwTwtWtWwU
R-490	TTWTWWTWwU	R-540	WttUwTwtTwwU
R-491	TTWTWWTtWwU	R-541	WttUwTTWtWwU
R-492	TTWTWtWtWwU	R-542	WttUwTTWtWwU
R-493	TTWTWtWtWwU	R-543	WttUwTwwUttUwwU
R-494	TTWTWwUttUWwU	R-544	WttUwTwwUwwUttU
R-495	TTWTWwUwwUttU	R-545	WttUwTttUwwUwwU
R-496	TTWwUttUwTwwU	R-546	WttUtwWtWtWwU
R-497	TTWwUttUtWwwU	R-547	WttUtwWtTwwU
R-498	TTWwUwTTTuWwU	R-548	WttUtwTwtWwU
R-499	TTWwUwTwwUttU	R-549	WttUtwTwtWwU
R-500	TTWwUwTwtTuWwU	R-550	WttUtwWwUttUwwU
R-501	TTWwUwTwwUttU	R-551	WttUtwWwUwwUttU
R-502	TTWwUwWwUwTTTU	R-552	WttUwwUttUwTwwU
R-503	TTWwUwWwUtwTTU	R-553	WttUwwUttUtWwwU
R-504	TWwUttUttUwwUwwU	R-554	WttUwwUwTTTuWwU
R-505	TWwUttUwTwtWwU	R-555	WttUwwUwTwwUttU
R-506	TWwUttUwTtWwU	R-556	WttUwwUwTttUwwU
R-507	TWwUttUtWwtWwU	R-557	WttUwwUtWwwUttU
R-508	TWwUttUtWtWwU	R-558	WttUwwUwwUwTTTU
R-509	TWwUttUwwUttUwwU	R-559	WttUwwUwwUwTttU
R-510	TWwUttUwwUwwUttU	R-560	WtTTTuTTUwwUwwU
R-511	TWwUwTTTuWtWwU	R-561	WtTTTuWtWtWwU
R-512	TWwUwTTTuTwwU	R-562	WtTTTuWtTwwU
R-513	TWwUwTwtTTUwwU	R-563	WtTTTuTwwTwwU
R-514	TWwUwTwtWwUttU	R-564	WtTTTuTwtWwU
R-515	TWwUwTtWttUwwU	R-565	WtTTTuwwUttUwwU
R-516	TWwUwTtWwwUttU	R-566	WtTTTuwwUwwUttU
R-517	TWwUwTwwUwTTTU	R-567	WtWtTTTuWtWwU
R-518	TWwUwTwwUtwTTU	R-568	WtWtTTTuTwwU
R-519	TWwUwTwtTuWtWwU	R-569	WtWtWtTTTuWwU
R-520	TWwUwTwtTuTwwU	R-570	WtWtWtWtWwUttU



Index	Tie string	Index	Tie string
R-571	WWTWTTWTTuWWU	R-621	WWWuTTuTTuTWWWU
R-572	WWTWTTWWWuTTU	R-622	WWWuTTuWTTTtuWWU
R-573	WWTWTTWuWTTTU	R-623	WWWuTTuWTWwuTTU
R-574	WWTWTTWuTWTTU	R-624	WWWuTTuTWTTuWWU
R-575	WWTWTTTuTWWWU	R-625	WWWuTTuTWWWuTTU
R-576	WWTWTTTuTWWWU	R-626	WWWuTTuWWuWTTTU
R-577	WWTTWTTTtuWWU	R-627	WWWuTTuWWuTWTTU
R-578	WWTTWTTWwuTTU	R-628	WWWuWTTTtuTTuWWU
R-579	WWTTWTTWTTuWWU	R-629	WWWuWTTTtuWWuTTU
R-580	WWTTWTTWWWuTTU	R-630	WWWuWTWTTWTTU
R-581	WWTTWWWuWTTTU	R-631	WWWuWTWTTWTTU
R-582	WWTTWWWuTWTTU	R-632	WWWuWTTWTTTU
R-583	WWTWwuTTuTTuWWU	R-633	WWWuWTTWTTTU
R-584	WWTWwuTTuWWuTTU	R-634	WWWuWTWwuTTuTTU
R-585	WWTWwuWTWTTTU	R-635	WWWuTWTTtuTTuWWU
R-586	WWTWwuWTTWTTU	R-636	WWWuTWTTuWWuTTU
R-587	WWTWwuTWTTTU	R-637	WWWuTWWTWTTTU
R-588	WWTWwuTWTWTTU	R-638	WWWuTWWTWTTU
R-589	WWTWwuWWuTTuTTU	R-639	WWWuTWTWTTTU
R-590	WTWTTuTTuWWuWWU	R-640	WWWuTWTWTTTU
R-591	WTWTTuWTWTTWU	R-641	WWWuTWWWuTTuTTU
R-592	WTWTTuWTTWWWU	R-642	WWWuWWuTTuWTTTU
R-593	WTWTTuTWWTTWU	R-643	WWWuWWuTTuTWTTU
R-594	WTWTTuTWTWWWU	R-644	WWWuWWuWTTTtuTTU
R-595	WTWTTuWWuTTuWWU	R-645	WWWuWWuTWTTtuTTU
R-596	WTWTTuWWuWWuTTU	R-646	TWTWwuWWuWWuWWU
R-597	WTWTTTtuWTTWU	R-647	TTWWWuWWuWWuWWU
R-598	WTWTTTtuTWWWU	R-648	TWwuWTWwuWWuWWU
R-599	WTWTTWTTtuWWU	R-649	TWwuTWWWuWWuWWU
R-600	WTWTTWTTWuTTU	R-650	TWwuWWuWTWwuWWU
R-601	WTWTTWTTtuWWU	R-651	TWwuWWuTWWWuWWU
R-602	WTWTTTWWWuTTU	R-652	TWwuWWuWWuWTWwu
R-603	WTWTTWwuWTTTU	R-653	TWwuWWuWWuTWWWU
R-604	WTWTTWwuTWTTU	R-654	WTTuWWuWWuWWuWWU
R-605	WTWTTWTTuWTTWU	R-655	WWTWTTWwuWWuWWU
R-606	WTWTTWTTuTWWWU	R-656	WWTTWWWuWWuWWU
R-607	WTWTTWTTTtuWWU	R-657	WWTWwuWTWwuWWU
R-608	WTWTTWTTWuTTU	R-658	WWTWwuTWWWuWWU
R-609	WTWTTWTTTuWWU	R-659	WWTWwuWWuWTWwu
R-610	WTWTTWTTWwuTTU	R-660	WWTWwuWWuTWWWU
R-611	WTWTTWwuWTTTU	R-661	WTWTTWwuWWuWWU
R-612	WTWTTWwuTWTTU	R-662	WTWTTWwuWWuWWU
R-613	WTWWWuTTuTTuWWU	R-663	WTWWWuWTWwuWWU
R-614	WTWWWuTTuWWuTTU	R-664	WTWWWuTWWWuWWU
R-615	WTWWWuWTTWTTU	R-665	WTWWWuWWuWTWwu
R-616	WTWWWuWTTWTTU	R-666	WTWWWuWWuTWWWU
R-617	WTWWWuTWTTTU	R-667	WWWuTTuWWuWWuWWU
R-618	WTWWWuTWTTTU	R-668	WWWuWTWTTWwuWWU
R-619	WTWWWuWWuTTuTTU	R-669	WWWuWTTWWWuWWU
R-620	WWWuTTuTTuWTTWU	R-670	WWWuWTWwuWTWwu

Index	Tie string
R-671	WWWuWTWWuTWWWU
R-672	WWWuTWWTWWuWWU
R-673	WWWuTWTWWWuWWU
R-674	WWWuTWWWuWTWWU
R-675	WWWuTWWWuTWWWU
R-676	WWWuWWuTTuWWuWWU
R-677	WWWuWWuWTWTWWU
R-678	WWWuWWuWTWWWU
R-679	WWWuWWuTWWTWWU
R-680	WWWuWWuTWTWWWU
R-681	WWWuWWuWWuTTuWWU
R-682	WWWuWWuWWuWWuTTU

Next, we list the knots that tuck from the center. Again, with all optional single depth front tuck sites marked with a lowercase u. All these tie knots will have the thin blade sitting on top of the broad blade, which for most choices will be an unusual look, even for a modern tie knot. By comparing entries we may see that the 85 first knots in this enumeration are – with medial tucks ignored and with some permutation of assigned indices – the same as the 85 tie knots enumerated by Fink and Mao [2].

Index	Tie string	Index	Tie string	Index	Tie string
C-1	WWU	C-41	WWWuWTWWU	C-81	WWuTWTuWWU
C-2	WTTU	C-42	WWWuTWWWU	C-82	WWuTWWWuTTU
C-3	TTuTTU	C-43	TTuTTuTTuWWU	C-83	WWuWWuWTTTU
C-4	WTWWU	C-44	TTuTTuWWuTTU	C-84	WWuWWuTWTTU
C-5	TWWWU	C-45	TTuWTWTTTU	C-85	WWuWWuWWuWWU
C-6	TTTuWWU	C-46	TTuWTTWTTU	C-86	TTTuTTuWTTTU
C-7	TWWuTTU	C-47	TTuTWWTTTU	C-87	TTTuTTuTWTTU
C-8	WTTTTU	C-48	TTuTWTWTTU	C-88	TTTuWTTTuTTU
C-9	WTWTTU	C-49	TTuWWuTTuTTU	C-89	TTTuTWTTuTTU
C-10	WWWuWWU	C-50	WTTTuWTTTU	C-90	TWTTTuTTuTTU
C-11	TTuWTTTU	C-51	WTTTuTWTTU	C-91	TTWTTuTTuTTU
C-12	TTuTWTTU	C-52	WTWTTTuTTU	C-92	WTTuTTuTTuTTU
C-13	WTTTuTTU	C-53	WTTWTTuTTU	C-93	TTTuTTuWWuWWU
C-14	TWTTuTTU	C-54	TWTTuWTTTU	C-94	TTTuWTWTTWU
C-15	TTuWWuWWU	C-55	TWTTuTWTTU	C-95	TTTuWTTWWWU
C-16	WTWTTWU	C-56	TWTTTuTTU	C-96	TTTuTWWTTWU
C-17	WTTWWWU	C-57	TWTWTTuTTU	C-97	TTTuTWTWWWU
C-18	TWWTWWU	C-58	WWuTTuTTuTTU	C-98	TTTuWWuTTuWWU
C-19	TWTWWWU	C-59	TTuWTWuWWU	C-99	TTTuWWuWWuTTU
C-20	WWuTTuWWU	C-60	TTuTWWuWWU	C-100	TWTTTuWTTWU
C-21	WWuWWuTTU	C-61	TTuWWuWTWWU	C-101	TWTTTuTWWWU
C-22	TTTuTTuTTU	C-62	TTuWWuTWWWU	C-102	TWTWTTTuWWU
C-23	TTTuWTWWU	C-63	WTTTuWWuWWU	C-103	TWTWTTWuTTU
C-24	TTTuTWWWU	C-64	WTWTTWU	C-104	TWTTWTTuWWU
C-25	TWTTTuWWU	C-65	WTWTTWWWU	C-105	TWTTWWWuTTU
C-26	TWTWuTTU	C-66	WTTWTTWU	C-106	TWTWuWTTTU
C-27	TTWTTuWWU	C-67	WTTWTTWWWU	C-107	TWTWuTWTTU
C-28	TTWwWuTTU	C-68	WTWuTTuWWU	C-108	TTWTTuWTTWU
C-29	TWWuWTTTU	C-69	WTWuWWuTTU	C-109	TTWTTuTWWWU
C-30	TWWuTWTTU	C-70	TWTTuWWuWWU	C-110	TTWTTTuWWU
C-31	WTTuTTuWWU	C-71	TWWTWTTWU	C-111	TTWWTWuTTU
C-32	WTTuWWuTTU	C-72	TWWTWWWU	C-112	TTWTWTTuWWU
C-33	WWTWTTTU	C-73	TWTWTTWU	C-113	TTWTWWWuTTU
C-34	WWTTWTTU	C-74	TWTWTTWWWU	C-114	TTWWWuWTTTU
C-35	WTWTTTU	C-75	TWWWuTTuWWU	C-115	TTWWWuTWTTU
C-36	WTWTWTTU	C-76	TWWWuWWuTTU	C-116	TWuTTuTTuWWU
C-37	WWWuTTuTTU	C-77	WWuTTuWTWWU	C-117	TWuTTuWWuTTU
C-38	TWWuWWuWWU	C-78	WWuTTuTWWWU	C-118	TWuWTTWTTU
C-39	WWTWuWWU	C-79	WWuWTTTuWWU	C-119	TWuWTTWTTU
C-40	WTWWWuWWU	C-80	WWuWTWuTTU	C-120	TWuTWWTTTU

Index	Tie string	Index	Tie string	Index	Tie string
C-121	TWwUuTWTWTTU	C-171	TTuTTuTTuTTuTTU	C-221	TWwTTTuWTTTU
C-122	TWwUuWwUuTTuTTU	C-172	TTuTTuTTuWTWwU	C-222	TWwTTTuTWTTU
C-123	WTTuTTuWTWwU	C-173	TTuTTuTTuTWWwU	C-223	TWwTWTTTuTTU
C-124	WTTuTTuTWWwU	C-174	TTuTTuWTTTuWwU	C-224	TWwTWTTTuTTU
C-125	WTTuWTTTuWwU	C-175	TTuTTuWTWwUttU	C-225	TWTWTTuWTTTU
C-126	WTTuWTWwUttU	C-176	TTuTTuTWTTuWwU	C-226	TWTWTTuTWTTU
C-127	WTTuTWTTuWwU	C-177	TTuTTuTWWwUttU	C-227	TWTWTTTuTTU
C-128	WTTuTWWwUttU	C-178	TTuTTuWwUWTTTU	C-228	TWTWTTTuTTU
C-129	WTTuWwUWTTTU	C-179	TTuTTuWwUWTTU	C-229	TWWwUttUttUttU
C-130	WTTuWwUWTTU	C-180	TTuWTTTuTTuWwU	C-230	WwUttUttUWTTTU
C-131	WwTTTuTTuWwU	C-181	TTuWTTTuWwUttU	C-231	WwUttUttUWTTU
C-132	WwTTTuWwUttU	C-182	TTuWTWTTTU	C-232	WwUttUWTTTuTTU
C-133	WwTWTTTU	C-183	TTuWTWTTTU	C-233	WwUttUWTTTuTTU
C-134	WwTWTTTU	C-184	TTuWTTWTTTU	C-234	WwUWTTTuTTuTTU
C-135	WwTTWTTTU	C-185	TTuWTTWTTTU	C-235	WwUWTTTuTTuTTU
C-136	WwTTWTTTU	C-186	TTuWTWwUttUttU	C-236	TTuTTuWwUWwUWwU
C-137	WwTWwUttUttU	C-187	TTuTWTTuTTuWwU	C-237	TTuWTWTTWwUWwU
C-138	WTWTTuTTuWwU	C-188	TTuTWTTuWwUttU	C-238	TTuWTTWwUWwU
C-139	WTWTTuWwUttU	C-189	TTuTWTTWTTTU	C-239	TTuWTWwUWTTWwU
C-140	WTWTTWTTTU	C-190	TTuTWTTWTTTU	C-240	TTuWTWwUWTTWwU
C-141	WTWTTWTTTU	C-191	TTuTWTTWTTTU	C-241	TTuTWTTWwUWTTWwU
C-142	WTWTTWTTTU	C-192	TTuTWTTWTTTU	C-242	TTuTWTTWwUWTTWwU
C-143	WTWTTWTTTU	C-193	TTuTWwUttUttU	C-243	TTuTWwUttUttU
C-144	WTWwUttUttU	C-194	TTuWwUttUWTTTU	C-244	TTuTWwUttUttU
C-145	WwWuTTuWTTTU	C-195	TTuWwUttUWTTU	C-245	TTuWwUttUWwUWwU
C-146	WwWuTTuWTTU	C-196	TTuWwUWTTTuTTU	C-246	TTuWwUWTTWTTWwU
C-147	WwWuWTTTuTTU	C-197	TTuWwUWTTTuTTU	C-247	TTuWwUWTTWTTWwU
C-148	WwWuTWTTuTTU	C-198	WTTTuTTuTTuWwU	C-248	TTuWwUWTTWTTWwU
C-149	TWTWwUWwUWwU	C-199	WTTTuTTuWwUttU	C-249	TTuWwUWTTWTTWwU
C-150	TTWwUWwUWwU	C-200	WTTTuWTTTU	C-250	TTuWwUWwUttUWwU
C-151	TWwUuWTWwUWwU	C-201	WTTTuWTTTU	C-251	TTuWwUWwUWwUttU
C-152	TWwUuTWWwUWwU	C-202	WTTTuTWTTTU	C-252	WTTTuWTTWwUWwU
C-153	TWwUuWwUuWTWwU	C-203	WTTTuTWTTTU	C-253	WTTTuTWWwUWwU
C-154	TWwUuWwUuTWWwU	C-204	WTTTuWwUttUttU	C-254	WTTTuWwUWTTWwU
C-155	WTTuWwUWwUWwU	C-205	WTWTTTuWTTTU	C-255	WTTTuWwUWTTWwU
C-156	WwTWTTWwUWwU	C-206	WTWTTTuWTTU	C-256	WTWTTTuWwUWwU
C-157	WwTTWwUWwU	C-207	WTWTTTuTTU	C-257	WTWTTWTTWwU
C-158	WwTWwUWTTWwU	C-208	WTWTTTuTTU	C-258	WTWTTWTTWwU
C-159	WwTWwUWTTWwU	C-209	WTTWTTuWTTTU	C-259	WTTWTTWTTWwU
C-160	WTWTTWwUWwU	C-210	WTTWTTuWTTU	C-260	WTWTTWTTWwU
C-161	WTWTTWwUWwU	C-211	WTTWTTTuTTU	C-261	WTWTTWwUttUWwU
C-162	WTWwUuWTTWwU	C-212	WTTWTTTuTTU	C-262	WTWTTWwUWwUttU
C-163	WTWwUuTWWwU	C-213	WTWwUttUttUttU	C-263	WTTWTTuWwUWwU
C-164	WwWuTTuWwUWwU	C-214	TWTTuTTuTTuWwU	C-264	WTTWTTWTTWwU
C-165	WwWuWTTWTTWwU	C-215	TWTTuTTuWwUttU	C-265	WTTWTTWTTWwU
C-166	WwWuWTTWTTWwU	C-216	TWTTuWTTWTTTU	C-266	WTTWTTWTTWwU
C-167	WwWuTWTTWTTU	C-217	TWTTuWTTWTTTU	C-267	WTTWTTWTTWwU
C-168	WwWuTWTTWTTU	C-218	TWTTuWTTWTTTU	C-268	WTTWTTWTTWwU
C-169	WwWuWwUttUWwU	C-219	TWTTuTWTTWTTU	C-269	WTTWTTWwUttUWwU
C-170	WwWuWwUttUttU	C-220	TWTTuWwUttUttU	C-270	WTWwUttUWTTWwU

Index	Tie string	Index	Tie string
C-271	WTWwuTTuTWWWU	C-321	WWuTWWTTTuWWU
C-272	WTWwuwTTTuWWU	C-322	WWuTWWTWwUttU
C-273	WTWwuwTWwUttU	C-323	WWuTWTWTTuWWU
C-274	WTWwuwTWTTuWWU	C-324	WWuTWTWWwUttU
C-275	WTWwuwTWWWuTTU	C-325	WWuTWWWuWTTTTU
C-276	WTWwuwWwUTTTTU	C-326	WWuTWWWuTWTTU
C-277	WTWwuwWwUtwTTU	C-327	WWuWwUttUttUWWU
C-278	TWTTuTWwWuWWU	C-328	WWuWwUttUWwUttU
C-279	TWTTuTWWWuWWU	C-329	WWuWwUWTWTTTTU
C-280	TWTTuWwUWTWWU	C-330	WWuWwUWTTWTTU
C-281	TWTTuWwUWWWU	C-331	WWuWwUWWWTTTTU
C-282	TWWWTTuWwUWWU	C-332	WWuWwUWTWTTU
C-283	TWWTWTWTTWU	C-333	WWuWwUWwUttUttU
C-284	TWWTWTWWWU	C-334	WTWwUWwUWwUWWU
C-285	TWWTWWWTTWU	C-335	TWWWuWwUWwUWWU
C-286	TWWTTWTTWWWU	C-336	WWuWTWwUWwUWWU
C-287	TWWTWwUttUWWU	C-337	WWuTWWwUWwUWWU
C-288	TWWTWwUWwUttU	C-338	WWuWwUWTWwUWWU
C-289	TWTWTTuWwUWWU	C-339	WWuWwUWWWuWWU
C-290	TWTWWTWTTWU	C-340	WWuWwUWwUWTWwU
C-291	TWTWWTWWWU	C-341	WWuWwUWwUWWWU
C-292	TWTWTWTTWwU	C-342	TTTuTTuTTuTTuWWU
C-293	TWTWTWTTWWWU	C-343	TTTuTTuTTuWwUttU
C-294	TWTWWWuTTuWWU	C-344	TTTuTTuWTWTTTTU
C-295	TWTWWWuWwUttU	C-345	TTTuTTuWTTWTTU
C-296	TWWWuTTuTWTTWU	C-346	TTTuTTuTWTTTTU
C-297	TWWWuTTuTWWWU	C-347	TTTuTTuTWTTWTTU
C-298	TWWWuWTTTuWWU	C-348	TTTuTTuWwUttUttU
C-299	TWWWuTWwUttU	C-349	TTTuWTTTTuWTTTTU
C-300	TWWWuTWTTuWWU	C-350	TTTuWTTTTuTWTTU
C-301	TWWWuTWWWuTTU	C-351	TTTuWTWTTTTuTTU
C-302	TWWWuWwUTTTTU	C-352	TTTuWTTWTTTuTTU
C-303	TWWWuWwUtwTTU	C-353	TTTuTWTTuWTTTTU
C-304	WwUttUttUWwUWWU	C-354	TTTuTWTTuTWTTU
C-305	WwUttUWTWTTWU	C-355	TTTuTWTTTTuTTU
C-306	WwUttUWTTWWWU	C-356	TTTuTWTTWTTuTTU
C-307	WwUttUWWTWWU	C-357	TTTuWwUttUttUttU
C-308	WwUttUWTWWWU	C-358	TWTTTuTTuWTTTTU
C-309	WwUttUWwUttUWWU	C-359	TWTTTuTTuTWTTU
C-310	WwUttUWwUWwUttU	C-360	TWTTTuWTTTTuTTU
C-311	WwUWTTTTuTWTTWU	C-361	TWTTTuTWTTuTTU
C-312	WwUWTTTTuTWWWU	C-362	TWTWTTTuTTuTTU
C-313	WwUWTWTTTuWWU	C-363	TWTTWTTuTTuTTU
C-314	WwUWTWTTWwUttU	C-364	TTWTTuTTuWTTTTU
C-315	WwUWTTWTTuWWU	C-365	TTWTTuTTuTWTTU
C-316	WwUWTTWWWuTTU	C-366	TTWTTuWTTTTuTTU
C-317	WwUWTWwUWTTTTU	C-367	TTWTTuTWTTuTTU
C-318	WwUWTWwUtwTTU	C-368	TTWTTTTuTTuTTU
C-319	WwUWTWTTuTWTTWU	C-369	TTWTTWTTuTTuTTU
C-320	WwUWTWTTuTWWWU	C-370	TWwUttUttUttUttU

Index	Tie string	Index	Tie string
C-371	WTTuTTuTTuWTTTU	C-421	TWTTWTTuTWWWU
C-372	WTTuTTuTTuTWTTU	C-422	TWTTWWTtTuWWU
C-373	WTTuTTuWTTTtuTTU	C-423	TWTTWWTWWuTTU
C-374	WTTuTTuTWTTuTTU	C-424	TWTTWTTWTTuWWU
C-375	WTTuWTTTtuTTuTTU	C-425	TWTTWTTWWuTTU
C-376	WTTuTWTTuTTuTTU	C-426	TWTTWwwuWTTTU
C-377	WWTTTuTTuTTuTTU	C-427	TWTTWwwuTWTTU
C-378	WTWTTuTTuTTuTTU	C-428	TWTWwuTTuTTuWWU
C-379	TTTuTTuWTWWuWWU	C-429	TWTWwuTTuWWuTTU
C-380	TTTuTTuTWWWuWWU	C-430	TWTWwuWTWTTTU
C-381	TTTuTTuWWuWTWWU	C-431	TWTWwuWTTWTTU
C-382	TTTuTTuWWuTWWWU	C-432	TWTWwuTWWTTTU
C-383	TTTuWTTTtuWWuWWU	C-433	TWTWwuTWTWTTU
C-384	TTTuWTWTTWTTWU	C-434	TWTWwuWWuTTuTTU
C-385	TTTuWTWTTWWWU	C-435	TTWTTuTTuWWuWWU
C-386	TTTuWTTWTTWU	C-436	TTWTTuWTWTTWU
C-387	TTTuWTTWTTWWU	C-437	TTWTTuWTTWWU
C-388	TTTuWTWwuTTuWWU	C-438	TTWTTuTWWTTWU
C-389	TTTuWTWwuWWuTTU	C-439	TTWTTuTWTTWWU
C-390	TTTuTWTTuWWuWWU	C-440	TTWTTuWWuTTuWWU
C-391	TTTuTWTTWTTWU	C-441	TTWTTuWWuWWuTTU
C-392	TTTuTWTTWWWU	C-442	TTWTTTtuWTWU
C-393	TTTuTWTTWTTWU	C-443	TTWTTTtuTWWU
C-394	TTTuTWTTWTTWU	C-444	TTWTTWTTTtuWWU
C-395	TTTuTWWWuTTuWWU	C-445	TTWTTWTTWuTTU
C-396	TTTuTWWWuWWuTTU	C-446	TTWTTWTTuWWU
C-397	TTTuWWuTTuWTWU	C-447	TTWTTWwwuTTU
C-398	TTTuWWuTTuTWWU	C-448	TTWTTWwuWTTTU
C-399	TTTuWWuWTTTtuWWU	C-449	TTWTTWwuTWTTU
C-400	TTTuWWuWTWwuTTU	C-450	TTWTTWTTuWTWU
C-401	TTTuWWuTWTTuWWU	C-451	TTWTTWTTuTWWU
C-402	TTTuWWuTWWuTTU	C-452	TTWTTWTTTtuWWU
C-403	TTTuWWuWWuWTTTU	C-453	TTWTTWTTWuTTU
C-404	TTTuWWuWWuTWTTU	C-454	TTWTTWTTTuWWU
C-405	TWTTTuTTuWWuWWU	C-455	TTWTTWTTWWuTTU
C-406	TWTTTuWTWTTWU	C-456	TTWTTWwuWTTTU
C-407	TWTTTuWTTWWU	C-457	TTWTTWwuTWTTU
C-408	TWTTTuTWTTWU	C-458	TTWwuTTuTTuWWU
C-409	TWTTTuTWTTWU	C-459	TTWwuTTuWWuTTU
C-410	TWTTTuWWuTTuWWU	C-460	TTWwuWTWTTTU
C-411	TWTTTuWWuWWuTTU	C-461	TTWwuWTTWTTU
C-412	TWTWTTTuWTWU	C-462	TTWwuTWTTTU
C-413	TWTWTTTuTWWU	C-463	TTWwuTWTTTU
C-414	TWTWTTTtuWWU	C-464	TTWwuWWuTTuTTU
C-415	TWTWTTWwuTTU	C-465	TWwuTTuTTuWTWU
C-416	TWTWTTWTTuWWU	C-466	TWwuTTuTTuTWWU
C-417	TWTWTTWWuTTU	C-467	TWwuTTuWTTTuWWU
C-418	TWTWTTWuWTTTU	C-468	TWwuTTuWTWwuTTU
C-419	TWTWTTWuTWTTU	C-469	TWwuTTuTWTTuWWU
C-420	TWTTWTTuWTWU	C-470	TWwuTTuTWWwuTTU

Index	Tie string	Index	Tie string
C-471	TWWuTTuWWuWTTTU	C-521	WWTtTTuTTuWTWWU
C-472	TWWuTTuWWuTWTTU	C-522	WWTtTTuTTuTWWWU
C-473	TWWuWTTTtuTTuWWU	C-523	WWTtTTuWTTtuWWU
C-474	TWWuWTTTtuWWuTTU	C-524	WWTtTTuWTWWuTTU
C-475	TWWuWTWTWTTTU	C-525	WWTtTTuTWTTuWWU
C-476	TWWuWTWTTWTTU	C-526	WWTtTTuTWWWuTTU
C-477	TWWuWTTTWTTTU	C-527	WWTtTTuWWuWTTTU
C-478	TWWuWTTTWTTU	C-528	WWTtTTuWWuTWTTU
C-479	TWWuWTWWuTTuTTU	C-529	WWTWTTTtuTTuWWU
C-480	TWWuTWTTtuTTuWWU	C-530	WWTWTTTtuWWuTTU
C-481	TWWuTWTTtuWWuTTU	C-531	WWTWTWTWTTTU
C-482	TWWuTWWTWTTTU	C-532	WWTWTWTWTTU
C-483	TWWuTWWTTWTTU	C-533	WWTWTTWWTTTU
C-484	TWWuTWTWTTTU	C-534	WWTWTTWTTWTTU
C-485	TWWuTWTWTWTTU	C-535	WWTWTWWuTTuTTU
C-486	TWWuTWWWuTTuTTU	C-536	WWTTWTTtuTTuWWU
C-487	TWWuWWuTTuWTTTU	C-537	WWTTWTTtuWWuTTU
C-488	TWWuWWuTTuTWTTU	C-538	WWTTWWTWTTTU
C-489	TWWuWWuWTTTtuTTU	C-539	WWTTWWTWTTU
C-490	TWWuWWuTWTTtuTTU	C-540	WWTTWTWWTTTU
C-491	WTTuTTuTTuWWuWWU	C-541	WWTTWTWTTWTTU
C-492	WTTuTTuWTWTWWU	C-542	WWTTWWWuTTuTTU
C-493	WTTuTTuWTTWWWU	C-543	WWTWWuTTuWTTTU
C-494	WTTuTTuTWWTWWU	C-544	WWTWWuTTuTWTTU
C-495	WTTuTTuTWTTWWWU	C-545	WWTWWuWTTTtuTTU
C-496	WTTuTTuWWuTTuWWU	C-546	WWTWWuTWTTtuTTU
C-497	WTTuTTuWWuWWuTTU	C-547	WTWTTtuTTuWTWWU
C-498	WTTuWTTTtuWTWWU	C-548	WTWTTtuTTuTWWWU
C-499	WTTuWTTTtuTWWWU	C-549	WTWTTuWTTTtuWWU
C-500	WTTuWTWTTTtuWWU	C-550	WTWTTuWTWWuTTU
C-501	WTTuWTWTWWuTTU	C-551	WTWTTuTWTTuWWU
C-502	WTTuWTTWTTtuWWU	C-552	WTWTTuTWWWuTTU
C-503	WTTuWTTWWWuTTU	C-553	WTWTTuWWuWTTTU
C-504	WTTuWTWWuWTTTU	C-554	WTWTTuWWuTWTTU
C-505	WTTuWTWWuTWTTU	C-555	WTWWTTTtuTTuWWU
C-506	WTTuTWTTtuWTWWU	C-556	WTWWTTTtuWWuTTU
C-507	WTTuTWTTtuTWWWU	C-557	WTWWTWTTWTTTU
C-508	WTTuTWTTTtuWWU	C-558	WTWWTWTTWTTU
C-509	WTTuTWTTWWuTTU	C-559	WTWWTTWTTTU
C-510	WTTuTWTTWTTuWWU	C-560	WTWWTTWTTWTTU
C-511	WTTuTWTWWWuTTU	C-561	WTWWTWWuTTuTTU
C-512	WTTuTWWWuWTTTU	C-562	WTWTWTTtuTTuWWU
C-513	WTTuTWWWuTWTTU	C-563	WTWTWTTuWWuTTU
C-514	WTTuWWuTTuTTuWWU	C-564	WTWTWWWTTTU
C-515	WTTuWWuTTuWWuTTU	C-565	WTWTWTTWTTU
C-516	WTTuWWuWTWTTTU	C-566	WTWTWTTWTTTU
C-517	WTTuWWuWTTWTTU	C-567	WTWTWTWTTTU
C-518	WTTuWWuTWTTTU	C-568	WTWTWWWuTTuTTU
C-519	WTTuWWuTWTTTU	C-569	WTWWWuTTuWTTTU
C-520	WTTuWWuWWuTTuTTU	C-570	WTWWWuTTuTWTTU

Index	Tie string	Index	Tie string
C-571	WTWWWuWTTTuTTU	C-621	WTTuWWuTWWWuWWU
C-572	WTWWWuTWTTuTTU	C-622	WTTuWWuWWuWTWWU
C-573	WWWuTTuTTuTTuWWU	C-623	WTTuWWuWWuTWWWU
C-574	WWWuTTuTTuWWuTTU	C-624	WWTTTuWWuWWuWWU
C-575	WWWuTTuWTTTtu	C-625	WWTWTWTWWuWWU
C-576	WWWuTTuWTTWTTU	C-626	WWTWTWWWuWWU
C-577	WWWuTTuTWWTTTU	C-627	WWTWTWWuWTWWU
C-578	WWWuTTuTWTWTTU	C-628	WWTWTWWuTWWWU
C-579	WWWuTTuWWuTTuTTU	C-629	WWTTWWTWWuWWU
C-580	WWWuWTTTuWTTTU	C-630	WWTTWTWWWuWWU
C-581	WWWuWTTTuTWTTu	C-631	WWTTWWWuWTWWU
C-582	WWWuWTTTtuTTU	C-632	WWTTWWWuTWWWU
C-583	WWWuWTTWTTtuTTU	C-633	WWTWWuTTuWWuWWU
C-584	WWWuTWTTuWTTTU	C-634	WWTWWuWTWTWWU
C-585	WWWuTWTTuTWTTu	C-635	WWTWWuWTTWWWU
C-586	WWWuTWWTTTuTTU	C-636	WWTWWuTWWTWWU
C-587	WWWuTWTWTTtuTTU	C-637	WWTWWuTWTWWWU
C-588	WWWuWWuTTuTTuTTU	C-638	WWTWWuWWuTTuWWU
C-589	TTTuWWuWWuWWuWWU	C-639	WWTWWuWWuWWuTTU
C-590	TWTWTWWuWWuWWU	C-640	WTWTTuWWuWWuWWU
C-591	TWTWWWuWWuWWU	C-641	WTWWTWTWWuWWU
C-592	TWTWWuWTWWuWWU	C-642	WTWWTWWWuWWU
C-593	TWTWWuTWWWuWWU	C-643	WTWWTWWuWTWWU
C-594	TWTWWuWWuWTWWU	C-644	WTWWTWWuTWWWU
C-595	TWTWWuWWuTWWWU	C-645	WTWTWWTWWuWWU
C-596	TTWWTWWuWWuWWU	C-646	WTWTWTWWuWWU
C-597	TTWTWWWuWWuWWU	C-647	WTWTWWWuWTWWU
C-598	TTWWWuWTWWuWWU	C-648	WTWTWWWuTWWWU
C-599	TTWWWuTWWWuWWU	C-649	WTWWWuTTuWWuWWU
C-600	TTWWWuWWuWTWWU	C-650	WTWWWuWTWTWWU
C-601	TTWWWuWWuTWWWU	C-651	WTWWWuWTTWWWU
C-602	TWWuTTuWWuWWuWWU	C-652	WTWWWuTWWTWWU
C-603	TWWuWTTWTTWWuWWU	C-653	WTWWWuTWTWWWU
C-604	TWWuWTTWWWuWWU	C-654	WTWWWuWWuTTuWWU
C-605	TWWuWTWWuWTWWU	C-655	WTWWWuWWuWWuTTU
C-606	TWWuWTWWuTWWWU	C-656	WWWuTTuWTWWuWWU
C-607	TWWuTWWTWWuWWU	C-657	WWWuTTuTWWWuWWU
C-608	TWWuTWTWWWuWWU	C-658	WWWuTTuWWuWTWWU
C-609	TWWuTWWWuWTWWU	C-659	WWWuTTuWWuTWWWU
C-610	TWWuTWWWuTWWWU	C-660	WWWuWTTTuWWuWWU
C-611	TWWuWWuTTuWWuWWU	C-661	WWWuWTWTWTWWU
C-612	TWWuWWuWTTWTTWWU	C-662	WWWuWTWTTWWWU
C-613	TWWuWWuWTTWWWU	C-663	WWWuWTTWTTWWU
C-614	TWWuWWuTWWTWWU	C-664	WWWuWTTWTWWWU
C-615	TWWuWWuTWTWWWU	C-665	WWWuWTWWuTTuWWU
C-616	TWWuWWuWWuTTuWWU	C-666	WWWuWTWWuWWuTTU
C-617	TWWuWWuWWuWWuTTU	C-667	WWWuTWTTuWWuWWU
C-618	WTTuWTTWWuWWuWWU	C-668	WWWuTWWTWTWWU
C-619	WTTuTWWWuWWuWWU	C-669	WWWuTWWTWWWU
C-620	WTTuWWuWTWWuWWU	C-670	WWWuTWTWTTWWU



Index	Tie string
C-671	WWWuTWTWTWWWU
C-672	WWWuTWWWuTTuWWU
C-673	WWWuTWWWuWWuTTU
C-674	WWWuWWuTTuWTWWU
C-675	WWWuWWuTTuTWWWU
C-676	WWWuWWuWTTTuWWU
C-677	WWWuWWuWTWWuTTU
C-678	WWWuWWuTWTTuWWU
C-679	WWWuWWuTWWWuTTU
C-680	WWWuWWuWWuWTTTU
C-681	WWWuWWuWWuTWTU
C-682	WWWuWWuWWuWWuWWU

Next, we include the Python code listing that we used to generate these enumerations. With some utility functions, the core of this code is in the combinatorics generated from the `itertools` package and in the `mod 3` tests on final segments of strings.

```
# ties.py
# (c) 2013 Mikael Vejdemo-Johansson
# Released as CC-BY

import itertools
from scipy.special import binom

"""
Reproducing Fink & Mao is done by the following:

finkmao = classical(prototies(wtpairs(9)))
"""

def wtpairs(maxN, mod3=1):
    return [(i, n-i) for n in range(maxN)
            for i in range(n+1) if (n-2*i) % 3 == mod3]

def prototies(wtps):
    ret = []
    for (n,k) in wtps:
        if k==0:
            ret.append(['W']*(n+k))
            continue
        poss = itertools.combinations(range(n+k), k)
        for pos in poss:
            tie = ['W']*(n+k)
            for p in pos:
                tie[p] = 'T'
            ret.append(tie)
    return ret

def classical(protos):
    return [p for p in protos if len(p)>=2 and p[-2] == p[-1]]

def ktuckfinals(protos, k):
    def wort(t):
        if t == 'W':
            return 1
        elif t == 'T':
            return -1
        else:
            return 0
    def wmint(tt):
        return sum(map(wort, tt)) % 3
    return [p for p in protos
```

```

    if len(p) >= 2*k and wmint(p[-2*k:]) == ((-wort(p[-2*k])) % 3)
"""
States & cyclic transitions:
      W      T
L (left)   ^      v
C (center) ^      v
R (right)  ^      v
"""
statevect = {'T': {'L': 'C', 'C': 'R', 'R': 'L'},
             'W': {'L': 'R', 'R': 'C', 'C': 'L'}}

def WTtoCLR(tie, startstate='L'):
    state = startstate
    ret = state
    for wt in tie:
        state = statevect[wt][state]
        ret += state
    return ret

def balance(tie):
    return len([i for i in range(1, len(tie)) if tie[i] != tie[i-1]])

def symmetry(tie, startstate='L'):
    clrtie = WTtoCLR(tie, startstate=startstate)
    return len([i for i in range(len(clrtie)) if clrtie[i] == 'L']) - len([i for i in range(len(tie)) if tie[i] == 'L'])

def CLRtoWT(tie):
    ret = []
    fr = tie[0]
    for st in tie[1:]:
        to = st
        if to == 'T':
            ret.append('U')
        elif statevect['W'][fr] == to:
            ret.append('W')
        elif statevect['T'][fr] == to:
            ret.append('T')
        fr = to
    return ret

def countWT(w, t):
    if w+t < 2:
        return 0
    return binom(w+t, t) - 2*binom(w+t-2, t-1)

```

```

def tucks(t):
    return ktucks(t)

def ktuck(t,k):
    """
    Tests whether the end of the winding string t is a valid site for
    a k-fold tuck
    """
    if len(t) < 2*k:
        return False
    ws = [c for c in t[-2*k:] if c == 'W']
    ts = [c for c in t[-2*k:] if c == 'T']
    diff = (len(ws)-len(ts)) % 3
    if t[-2*k] == 'W':
        return diff == 2
    elif t[-2*k] == 'T':
        return diff == 1
    return False

def ktucks(t,k=1):
    """
    Annotates all valid k-fold tuck sites in a winding string
    """
    revt = list(t)
    ret = []
    while len(revt) > 0:
        if ktuck(revt,k):
            ret.insert(0,'u')
            revt.insert(0,revt.pop())
    return ret

def printall(maxN,k=1,p=lambda tt: True):
    """
    Prints out a TeX-able table of all tie knots with k-fold tuck
    sites marked. Defaults to k=1. Separated by final tuck direction.
    """
    for j,c in enumerate(['L','C','R']):
        ties = [ktucks(tt,k) for tt in prototies(wtpairs(maxN,mod3=j))
        if len(tt) > 1 and p(tt)]
        for i,t in enumerate(ties):
            print ('%s-%d' % (c,i+1)), '&', ''.join(t), '\\\\'

```

finkmao = ['LRCT',