A Computational Approach to String Figures

Yulong Liu

2023-11-29

A Computational Approach to String Figures

2023-11-29

A Computational Approach to String Figures

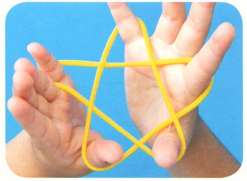
Yulong Liu

2023-11-29

String Figures

1. what are strings figures, i want to give some exmaples

String Figures



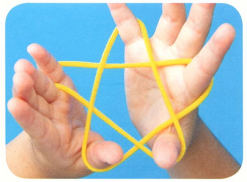
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

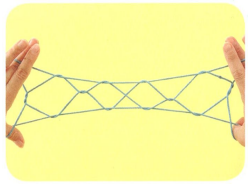
String Figures

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

1. these photos comes from this book, translated as "String Figure Encyclopedia" by Noguchi

String Figures



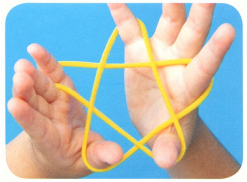
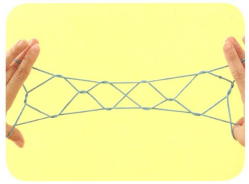
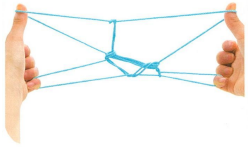


Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

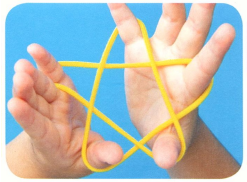
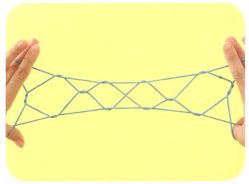
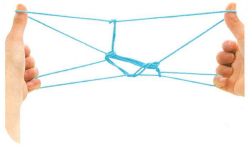
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

1. i can play the first two, the last one takes like 40 steps to make (CAMERA)

String Figures

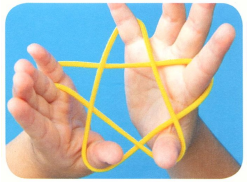
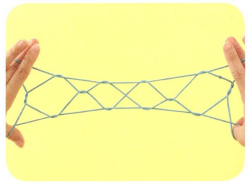
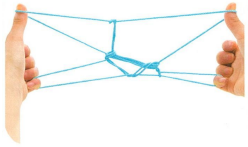




- Designs formed from a loop of string

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures
- Designs formed from a loop of string
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

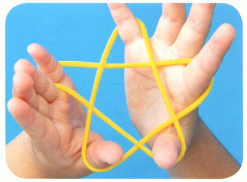
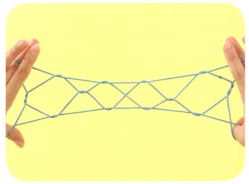
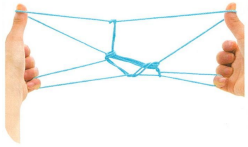
- Designs formed from a loop of string
- Commonly known as a children's game

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

1. i mean, look at how young those hands are

String Figures
- Designs formed from a loop of string
- Commonly known as a children's game
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

- Designs formed from a loop of string
- Commonly known as a children's game

People have also been playing with the string throughout history.

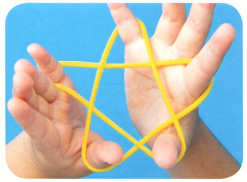
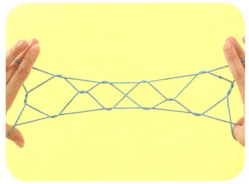
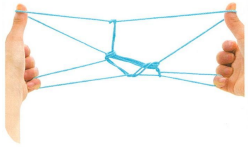
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures
- Designs formed from a loop of string
- Commonly known as a children's game

People have also been playing with the string throughout history.

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures

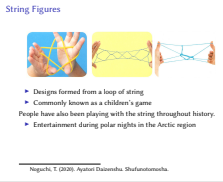




- Designs formed from a loop of string
- Commonly known as a children's game

People have also been playing with the string throughout history.

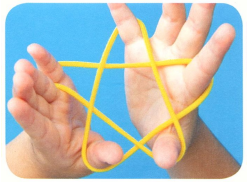
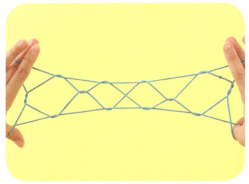
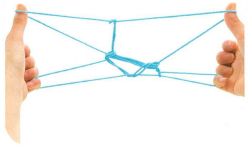
- Entertainment during polar nights in the Arctic region

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.



1. the native inhabitants in the arctic region play string figures for entertainment

String Figures

- Designs formed from a loop of string
- Commonly known as a children's game

People have also been playing with the string throughout history.

- Entertainment during polar nights in the Arctic region
- Storytelling and illustrating scenes from myths and legends

Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

String Figures
Designs formed from a loop of string
Commonly known as a children's game
People have also been playing with the string throughout history.
Entertainment during polar nights in the Arctic region
Storytelling and illustrating scenes from myths and legends
Noguchi, T. (2020). Ayatori Daizenshu. Shufunotomosha.

1. the indigenous people in New Zealand play string figures for storytelling and illustrating scenes from myths and legends
2. and i play string figures when overleaf takes forever to compile my slides

A Computational Approach

1. we want to answer the question: how to make string figures

A Computational Approach

To make a string figure:

A Computational Approach

To make a string figure:

A Computational Approach

To make a string figure:

- Start with an initial position (opening)

1. show opening for star

A Computational Approach

To make a string figure:
- Start with an initial position (opening)

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves

1. make star

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

1. show SF after each move
2. ...but it has always been a challenge to describe the SFs and these movements to someone else
3. and maybe do some calculations to predict what's the result of applying certain movements

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

- Represent string figures: simple, precise

A Computational Approach
To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another
String figures computations
- Represent string figures: simple, precise

1. it'd be good to have a way that computers understand, so we can store them easily in computers

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

- Represent string figures: simple, precise
- Apply moves directly to the representations

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations
- Represent string figures: simple, precise
- Apply moves directly to the representations

1. instead of doing the moves physically on a string, then writing the repr down, it'd be good to have an alg to calculate it directly
2. this is essentially teaching computers how to make string figures

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

- Represent string figures: simple, precise
- Apply moves directly to the representations

Motivation

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations
- Represent string figures: simple, precise
- Apply moves directly to the representations

Motivation

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

- Represent string figures: simple, precise
- Apply moves directly to the representations

Motivation

- Precise language of describing string figures

A Computational Approach
To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another
String figures computations
- Represent string figures: simple, precise
- Apply moves directly to the representations
Motivation
- Precise language of describing string figures

1. think of it like music scores
2. if i want to show you how to play this new piece of music
3. i don't have to play it physically with an instrument
4. i can just give you the music sheet
5. which is much more convenient than, say, a recording of me playing the music
6. ...
7. music sheet is very visual and hard for computers to read, that's why we have MIDI, and now computers can play MIDI files to simulate what they would sound like on an instrument

A Computational Approach

To make a string figure:

- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations

- Represent string figures: simple, precise
- Apply moves directly to the representations

Motivation

- Precise language of describing string figures
- Computer simulations & animations

A Computational Approach

To make a string figure:
- Start with an initial position (opening)
- Apply a sequence of moves
- Each move transforms a string figure to another

String figures computations
- Represent string figures: simple, precise
- Apply moves directly to the representations

Motivation
- Precise language of describing string figures
- Computer simulations & animations

1. same idea applies with strings figures
2. with a carefully designed language to describe string figures, we can let computers do simulations as well
3. and there are currently implementations of this, made by Alfredo, a physics professor in italy.
4. i can show some features that he implemented at the end of the talk
5. discussing about his code and meeting with parker and another math professor in Université Paris Cité, eric, was part of my reading course for parker

Representation: Diagrams

Representation: Diagrams

1. let's look at some ways of representing string figures
2. we will start with a visual one, diagrams

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

Representation: Diagrams
Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

1. L for left hand, R for right hand

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

- Ordered from nearest to furthest

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky
- Ordered from nearest to furthest

1. i.e. order by the number

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

- Ordered from nearest to furthest

$$L1 < L2 < L3 < L4 < L5$$
$$R1 < R2 < R3 < R4 < R5$$

Representation: Diagrams
Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky
Ordered from nearest to furthest
$L1 < L2 < L3 < L4 < L5$
$R1 < R2 < R3 < R4 < R5$

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

- Ordered from nearest to furthest

$$L1 < L2 < L3 < L4 < L5$$
$$R1 < R2 < R3 < R4 < R5$$

String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$

Representation: Diagrams
Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky
Ordered from nearest to furthest
$L1 < L2 < L3 < L4 < L5$
$R1 < R2 < R3 < R4 < R5$
String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

- Ordered from nearest to furthest

$$L1 < L2 < L3 < L4 < L5$$
$$R1 < R2 < R3 < R4 < R5$$

String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$

- Fn is the near string, Ff is the far string

Representation: Diagrams
Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky
Ordered from nearest to furthest
$L1 < L2 < L3 < L4 < L5$
$R1 < R2 < R3 < R4 < R5$
String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$
Fn is the near string, Ff is the far string

Representation: Diagrams

Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky

- Ordered from nearest to furthest

$$L1 < L2 < L3 < L4 < L5$$
$$R1 < R2 < R3 < R4 < R5$$

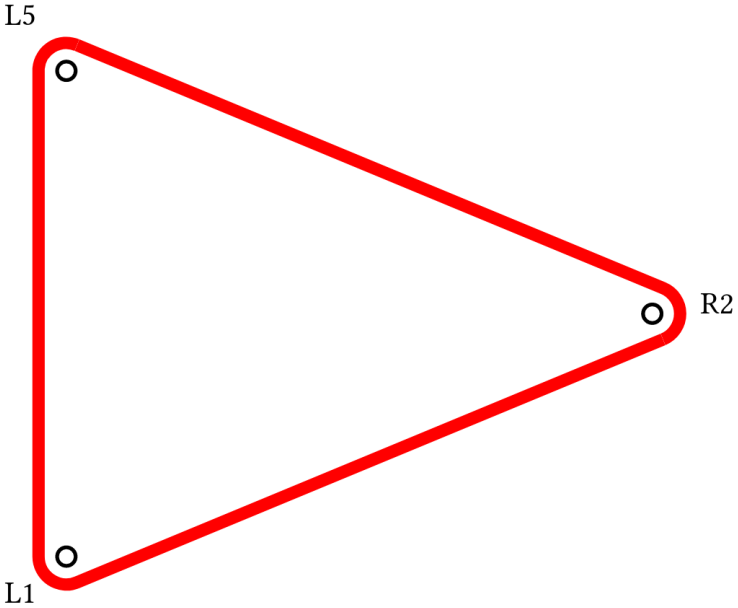
String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$

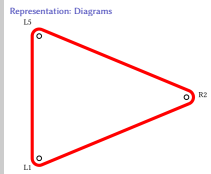
- Fn is the near string, Ff is the far string
- Lp and Rp are palmar strings

Representation: Diagrams
Fingers are named $L1, \cdots, L5$ and $R1, \cdots, R5$ from thumb to pinky
Ordered from nearest to furthest
$L1 < L2 < L3 < L4 < L5$
$R1 < R2 < R3 < R4 < R5$
String segments are named by finger $F \in \{L1, \cdots, L5, R1, \cdots, R5\}$
Fn is the near string, Ff is the far string
Lp and Rp are palmar strings

1. when we have a string between $L1$ and $L5$, we give a special name, called palmar string

Representation: Diagrams





- an actual diagram would look like this
- top down view of the fingers
- write segments, write *Lp*
- ARE WE OK WITH THIS???
- ...
- the diagrams are intuitive and very visual, but they are less computer friendly
- what do computers like? they like array of symbols

Representation: Linear Sequences

Two components

Representation: Linear Sequences
Two components

Representation: Linear Sequences

Two components

- Fingers that hold the string

Representation: Linear Sequences

Two components
- Fingers that hold the string

Representation: Linear Sequences

Two components

- Fingers that hold the string
- Crossings between two segments

Representation: Linear Sequences

Two components
- Fingers that hold the string
- Crossings between two segments

Representation: Linear Sequences

Two components

- Fingers that hold the string
- Crossings between two segments

Diagram → linear sequence

Representation: Linear Sequences

Two components
- Fingers that hold the string
- Crossings between two segments

Diagram → linear sequence

Representation: Linear Sequences

Two components

- Fingers that hold the string
- Crossings between two segments

Diagram → linear sequence

- Start with left nearest finger and travel clockwise

Representation: Linear Sequences

Two components
- Fingers that hold the string
- Crossings between two segments

Diagram → linear sequence
- Start with left nearest finger and travel clockwise

1. travel along the string clockwise

Representation: Linear Sequences

Two components

- Fingers that hold the string
- Crossings between two segments

Diagram → linear sequence

- Start with left nearest finger and travel clockwise
- Visit fingers and crossings

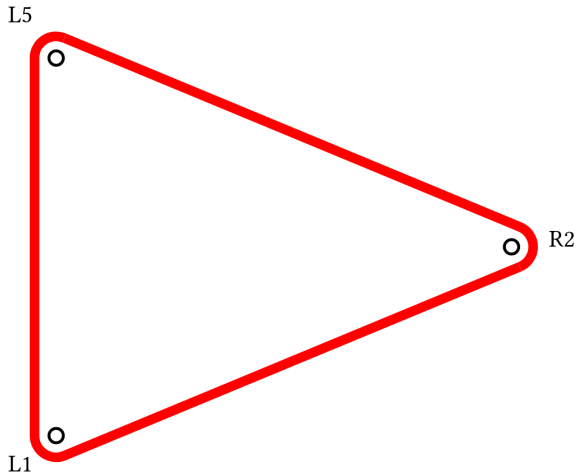
Representation: Linear Sequences

Two components
- Fingers that hold the string
- Crossings between two segments

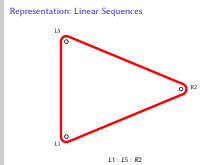
Diagram → linear sequence
- Start with left nearest finger and travel clockwise
- Visit fingers and crossings

1. we would write down the fingers and crossings to make a linear sequence

Representation: Linear Sequences



$L1 : L5 : R2$



- an algorithm that converts diagrams to linear sequences would map this to $L1 : R5 : R2$
- ARE WE OK WITH THIS

Linear Sequences with Crossings

Diagram → linear sequence

Linear Sequences with Crossings
Diagram → linear sequence

1. what do we do if we see crossings

Linear Sequences with Crossings

Diagram → linear sequence

- Name each crossing as x_i for some i

Linear Sequences with Crossings
Diagram → linear sequence
Name each crossing as x_i for some i

Linear Sequences with Crossings

Diagram $\longrightarrow$ linear sequence

- Name each crossing as x_i for some i
- Visit overcrossing $\implies$ write $x_i(o)$

Linear Sequences with Crossings
Diagram → linear sequence
- Name each crossing as x_i for some i
- Visit overcrossing $\implies$ write $x_i(o)$

Linear Sequences with Crossings

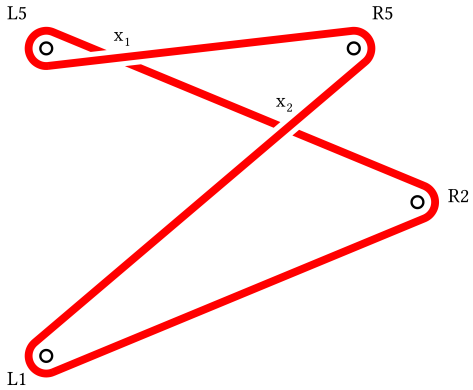
Diagram $\longrightarrow$ linear sequence

- Name each crossing as x_i for some i
- Visit overcrossing $\implies$ write $x_i(o)$
- Visit undercrossing $\implies$ write $x_i(u)$

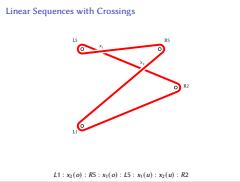
- draw overcrossing and undercrossing

Linear Sequences with Crossings
Diagram → linear sequence
Name each crossing as x_i for some i
Visit overcrossing $\implies$ write $x_i(o)$
Visit undercrossing $\implies$ write $x_i(u)$

Linear Sequences with Crossings



$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$



- if we have n fingers holding the string figure that has m crossings, the sequence will have $n + 2m$ symbols
- in this case we have 4 fingers and 2 crossings, so the sequence is $4 + 2 * 2 = 8$ symbols long
- in most cases, we are making more and more crossings in each step, so it would be tedious to write down the entire sequence by hand
- which gives us more motivation to develop a systematic way of computing the movements so we can let computer do the calculation
- ...
- the most common movements involve a finger and a near/far string segment
- eg this string figure is made from the opening with pick on Lp
- so if we want to pick Lp on the linear sequence, we need to first identify the segment from the sequence alone

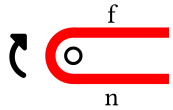
Identifying String Segments from Linear Sequences

Consider a left-hand finger L_i in the sequence

Identifying String Segments from Linear Sequences
Consider a left-hand finger L_i in the sequence

Identifying String Segments from Linear Sequences

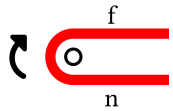
Consider a left-hand finger L_i in the sequence

- Traverse clockwise 

Identifying String Segments from Linear Sequences
Consider a left-hand finger L_i in the sequence
Traverse clockwise

Identifying String Segments from Linear Sequences

Consider a left-hand finger L_i in the sequence

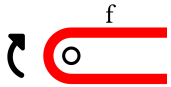
- Traverse clockwise 

$\Longrightarrow \ldots : [n]L_i[f] : \ldots$

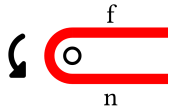
Identifying String Segments from Linear Sequences
Consider a left-hand finger L_i in the sequence
Traverse clockwise $\Longrightarrow \ldots : [n]L_i[f] : \ldots$

Identifying String Segments from Linear Sequences

Consider a left-hand finger L_i in the sequence

- Traverse clockwise 

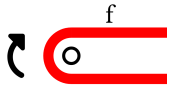
$\Longrightarrow \ldots : [n]L_i[f] : \ldots$

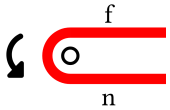
- Traverse counterclockwise 

Identifying String Segments from Linear Sequences
Consider a left-hand finger L_i in the sequence
- Traverse clockwise $\Longrightarrow \ldots : [n]L_i[f] : \ldots$
- Traverse counterclockwise

Identifying String Segments from Linear Sequences

Consider a left-hand finger L_i in the sequence

- Traverse clockwise 

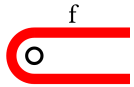
$\Longrightarrow \ldots : [n]L_i[f] : \ldots$
- Traverse counterclockwise 

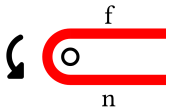
$\Longrightarrow \ldots : [f]L_i[n] : \ldots$

Identifying String Segments from Linear Sequences
Consider a left-hand finger L_i in the sequence
- Traverse clockwise $\Longrightarrow \ldots : [n]L_i[f] : \ldots$
- Traverse counterclockwise $\Longrightarrow \ldots : [f]L_i[n] : \ldots$

Identifying String Segments from Linear Sequences

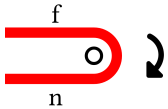
Consider a left-hand finger L_i in the sequence

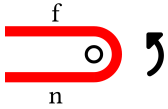
- Traverse clockwise 

$\Longrightarrow \ldots : [n]L_i[f] : \ldots$
- Traverse counterclockwise 

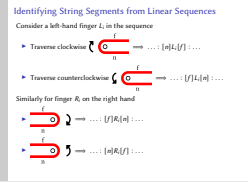
$\Longrightarrow \ldots : [f]L_i[n] : \ldots$

Similarly for finger R_i on the right hand

- 

$\Longrightarrow \ldots : [f]R_i[n] : \ldots$
- 

$\Longrightarrow \ldots : [n]R_i[f] : \ldots$



- ...
- since we always start a linear sequence by going clockwise, we can figure the segments on the first finger
- how about other fingers

Identifying String Segments : Opposite Hand

Consider $\ldots : L_i : \ldots : R_j : \ldots$

Identifying String Segments : Opposite Hand
Consider $\ldots : L_i : \ldots : R_j : \ldots$

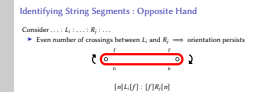
1. suppose the next finger is on the opposite hand

Identifying String Segments : Opposite Hand

Consider $\ldots : L_i : \ldots : R_j : \ldots$

- Even number of crossings between L_i and $R_j \implies$ orientation persists

$$[n]L_i[f] : [f]R_j[n]$$



Identifying String Segments : Opposite Hand

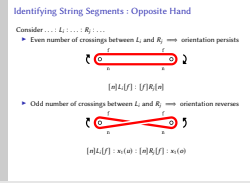
Consider . . . : L_i : . . . : R_j : . . .

- Even number of crossings between L_i and $R_j \implies$ orientation persists

$$[n]L_i[f] : [f]R_j[n]$$

- Odd number of crossings between L_i and $R_j \implies$ orientation reverses

$$[n]L_i[f] : x_1(u) : [n]R_j[f] : x_1(o)$$



- write x_1
- show 2 crossings: twist twice

Identifying String Segments : Same Hand

Consider $\ldots : L_i : \ldots : L_j : \ldots$

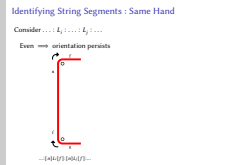
Identifying String Segments : Same Hand
Consider $\ldots : L_i : \ldots : L_j : \ldots$

Identifying String Segments : Same Hand

Consider $\ldots : L_i : \ldots : L_j : \ldots$

Even $\implies$ orientation persists

$\ldots : [n] L_i [f] : [n] L_j [f] : \ldots$



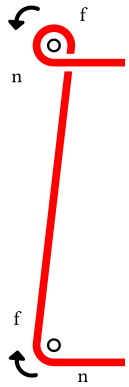
Identifying String Segments : Same Hand

Consider $\ldots : L_i : \ldots : L_j : \ldots$

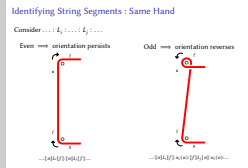
Even $\Longrightarrow$ orientation persists

$\ldots : [n] L_i [f] : [n] L_j [f] : \ldots$

Odd $\Longrightarrow$ orientation reverses



$\ldots : [n] L_i [f] : x_1(u) : [f] L_j [n] : x_1(o) : \ldots$



Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$L1\text{: } x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

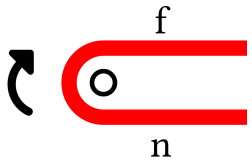
Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$L1\text{: } x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$\widetilde{L1} : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$



Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$\widetilde{L1} : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
f
n

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n] \overset{\frown}{L1} [f] : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

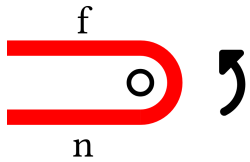
Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n] \overset{\frown}{L1} [f] : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n]\ \widetilde{L1}\ [f] : x_2(o) : \widehat{R5} : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$



Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]\ \widetilde{L1}\ [f] : x_2(o) : \widehat{R5} : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
f
n

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n]\ \widehat{L1}\ [f] : x_2(o) : [n]\ \widehat{R5}\ [f] : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

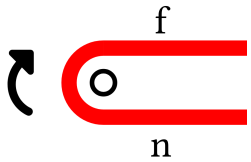
Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]\ \widehat{L1}\ [f] : x_2(o) : [n]\ \widehat{R5}\ [f] : x_1(o) : L5 : x_1(u) : x_2(u) : R2$

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n]L1[f] : x_2(o) : [n]\ \overset{\curvearrowleft}{R5}\ [f] : x_1(o) : \overset{\curvearrowright}{L5} : x_1(u) : x_2(u) : R2$$



Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]L1[f] : x_2(o) : [n]\ \overset{\curvearrowleft}{R5}\ [f] : x_1(o) : \overset{\curvearrowright}{L5} : x_1(u) : x_2(u) : R2$
f
n

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n]L1[f] : x_2(o) : [n]\ \overset{\curvearrowleft}{R5}\ [f] : x_1(o) : [n]\ \overset{\curvearrowright}{L5}\ [f] : x_1(u) : x_2(u) : R2$$

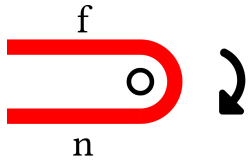
Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]L1[f] : x_2(o) : [n]\ \overset{\curvearrowleft}{R5}\ [f] : x_1(o) : [n]\ \overset{\curvearrowright}{L5}\ [f] : x_1(u) : x_2(u) : R2$

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

By convention, the first finger in the linear sequence is clockwise

$$[n]L1[f] : x_2(o) : [n]R5[f] : x_1(o) : [n]\ \widetilde{L5}\ [f] : x_1(u) : x_2(u) : \widetilde{R2}$$



Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]L1[f] : x_2(o) : [n]R5[f] : x_1(o) : [n]\ \widetilde{L5}\ [f] : x_1(u) : x_2(u) : \widetilde{R2}$
f
n

Identifying String Segments : Example

$$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$$

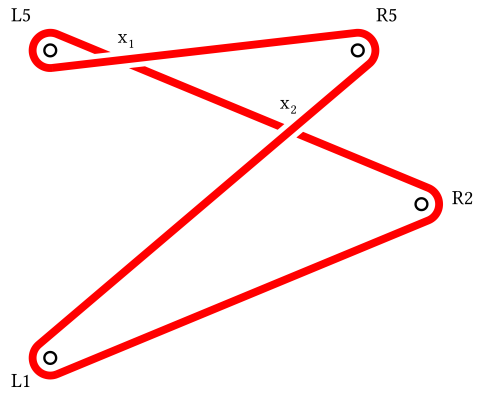
By convention, the first finger in the linear sequence is clockwise

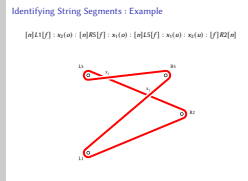
$$[n]L1[f] : x_2(o) : [n]R5[f] : x_1(o) : [n] \widetilde{L5} [f] : x_1(u) : x_2(u) : [f] \widetilde{R2} [n]$$

Identifying String Segments : Example
$L1 : x_2(o) : R5 : x_1(o) : L5 : x_1(u) : x_2(u) : R2$
By convention, the first finger in the linear sequence is clockwise
$[n]L1[f] : x_2(o) : [n]R5[f] : x_1(o) : [n] \widetilde{L5} [f] : x_1(u) : x_2(u) : [f] \widetilde{R2} [n]$

Identifying String Segments : Example

$[n]L1[f] : x_2(o) : [n]R5[f] : x_1(o) : [n]L5[f] : x_1(u) : x_2(u) : [f]R2[n]$





- trace sequence
- ARE WE OK WITH THIS
- the the point is that once we get a linear sequence, we can identify the segments without diagrams
- so when want to pick a specific string segment, we know where it is in the linear sequence
- (draw $L1 : L5 : R2$)
- so to do "pick Lp with $R5$", we need to first identify where Lp is in the sequence, and insert $R5$ there with some crossings
- ...
- to describe picking, we need to define another simpler movement first, which is twisting