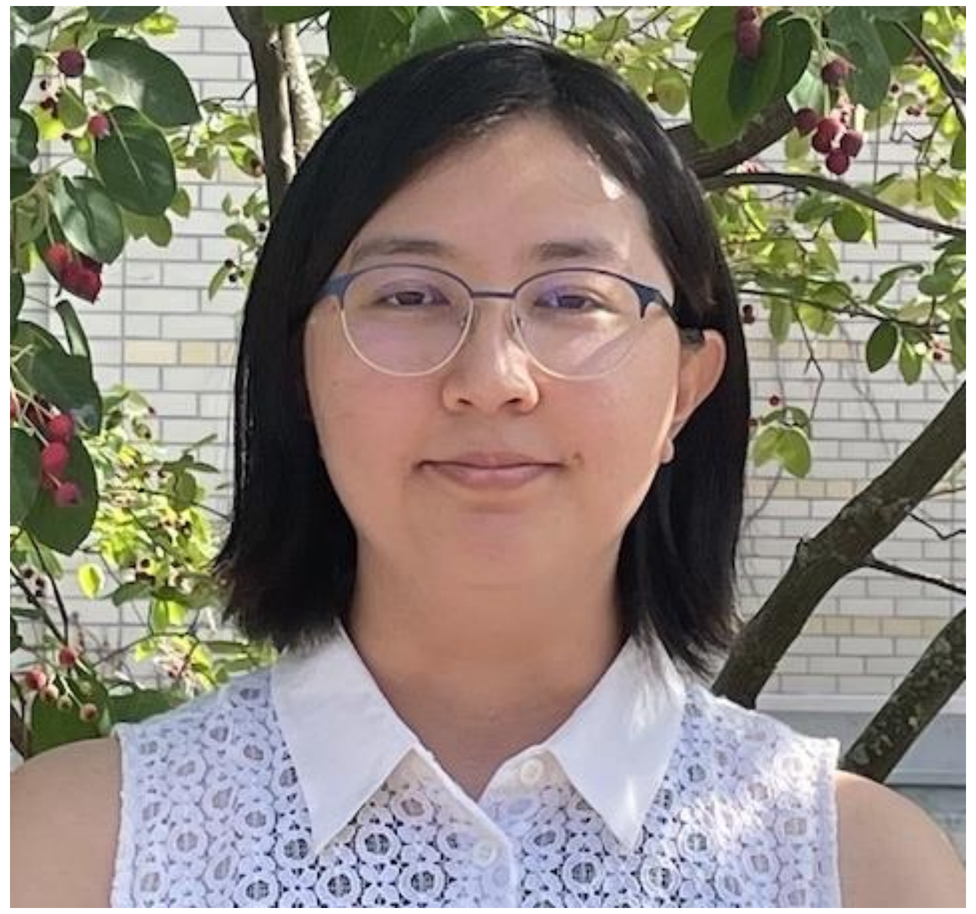
Computational Knitting



Edward Chien
Assistant Professor

Ben Jones
Postdoctoral Associate

Jenny Lin
Assistant Professor

Organization

- Knitting Background
  - What is Knitting and Why is it Important?
  - Why is Knitting Geometry?
- Computational Knitting Representations and Algorithms
  - Fabric Level
  - Stitch Level
  - Yarn Level
- Open Problems in Computational Knitting

Aardman Animations

Sources: Nike, Gymshark, Peregrine Clothing, Museum Outlets, Alexandre Kaspar

Sources: Nike, Gymshark, Peregrine Clothing, Museum Outlets, Alexandre Kaspar

Technical Knitting Applications

Saint-Gobain Aerospace

Luo et al. 2022

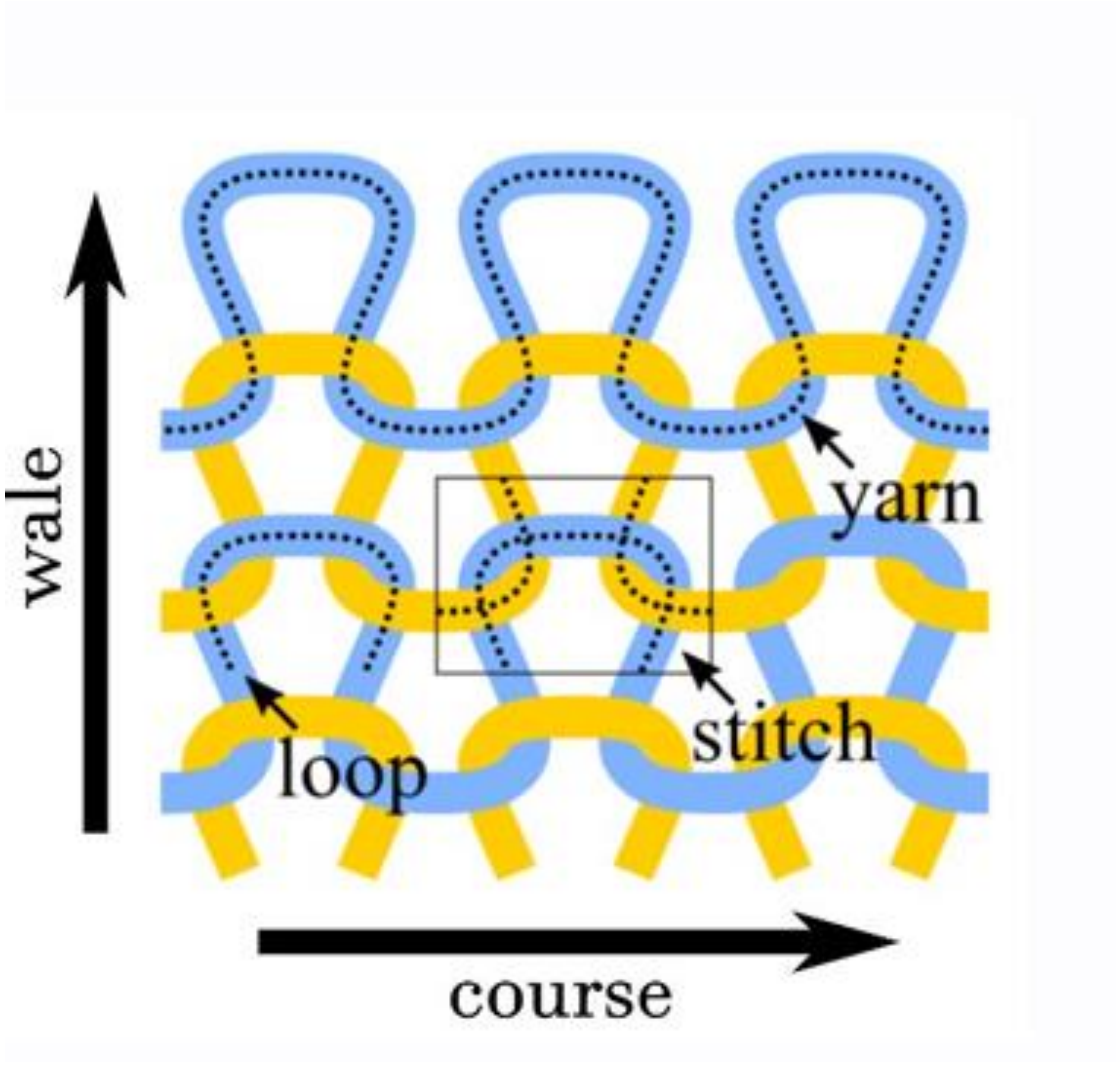
Kim et al. 2022

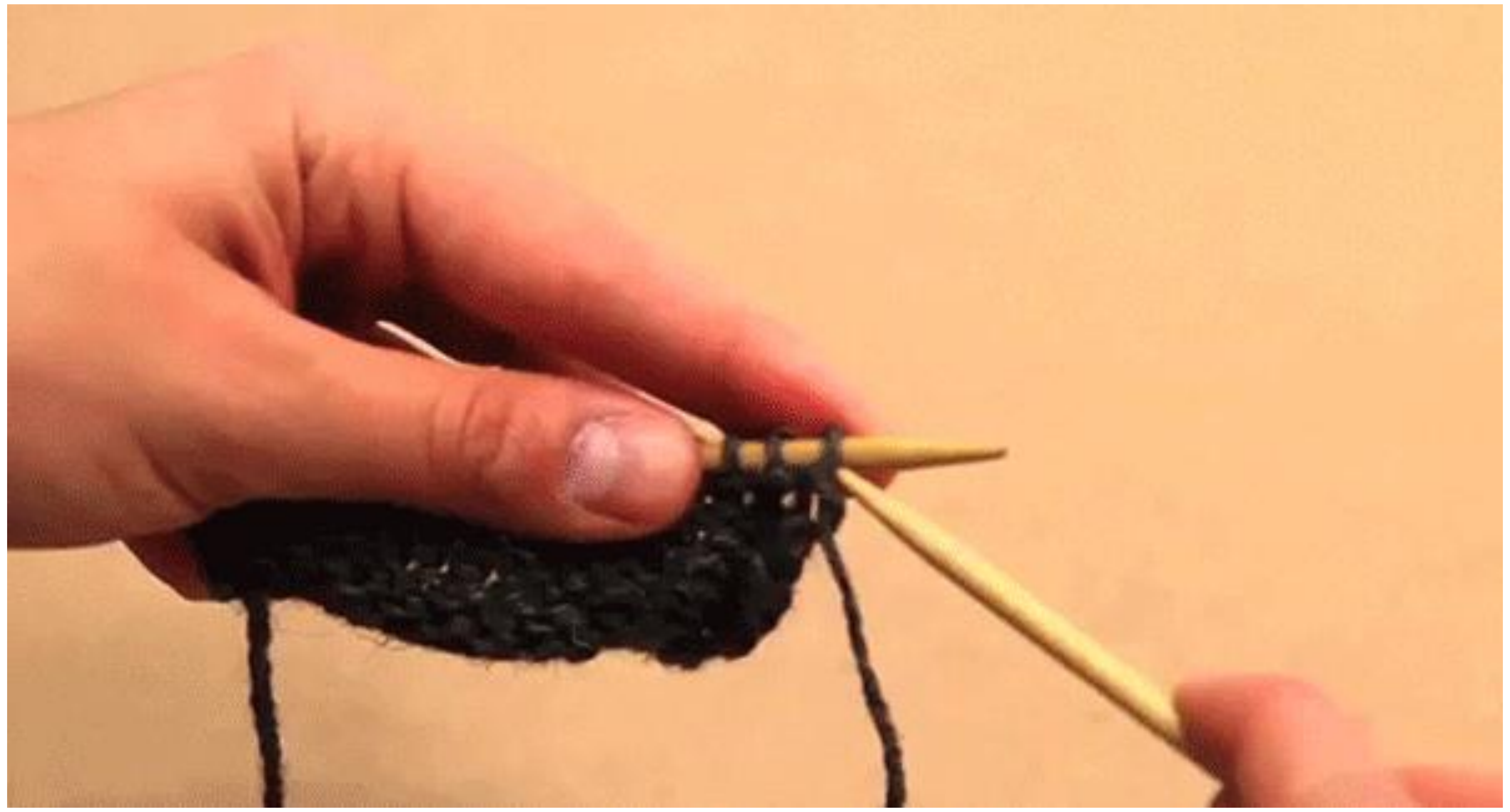
Liu et al. 2022

Zaha Hadid

Wicaksono et al. 2023

Hand Knitting

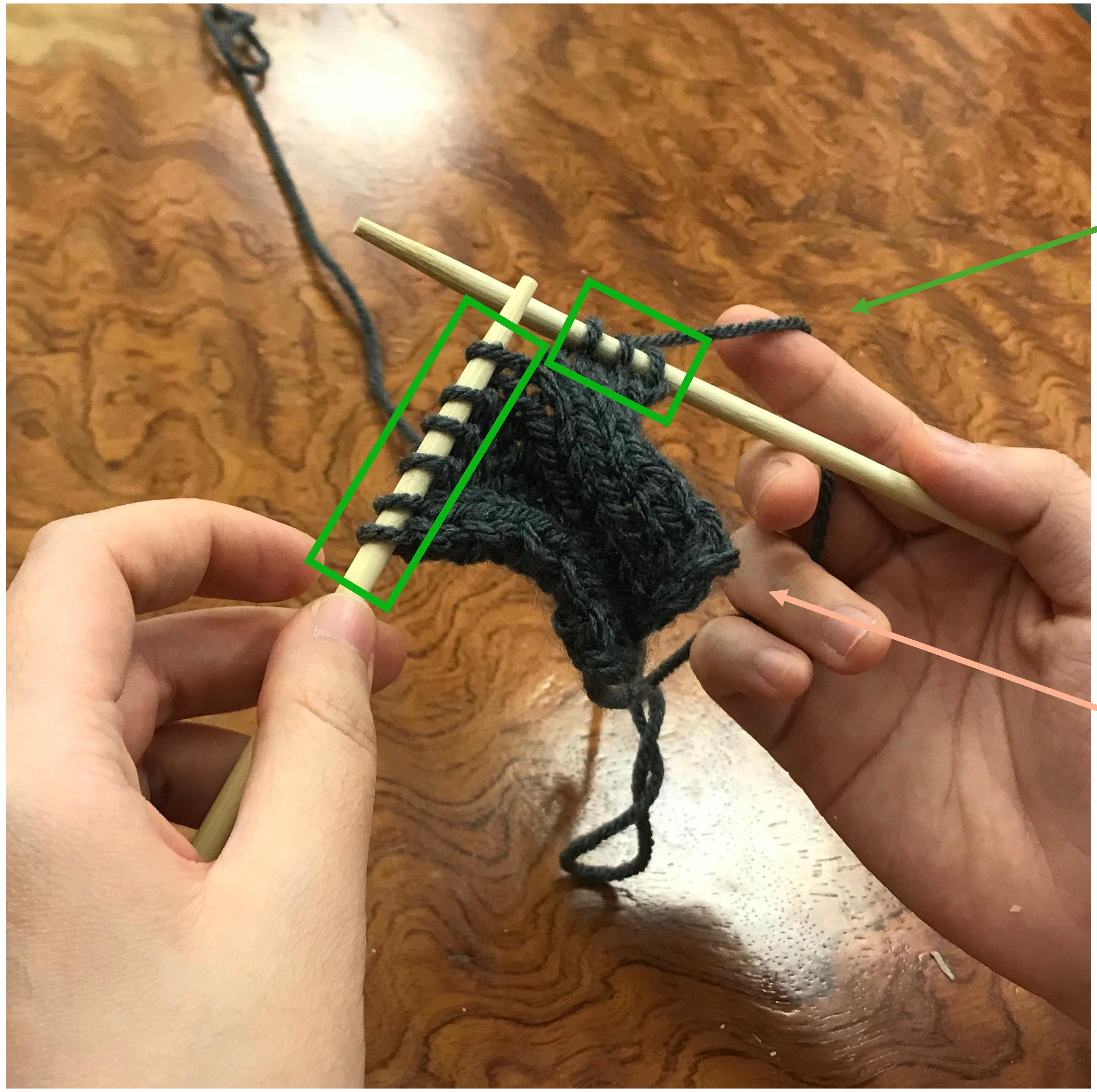




Images: Buzzfeed

Knitting (The Process)

Active Loops: would unravel without needles holding them



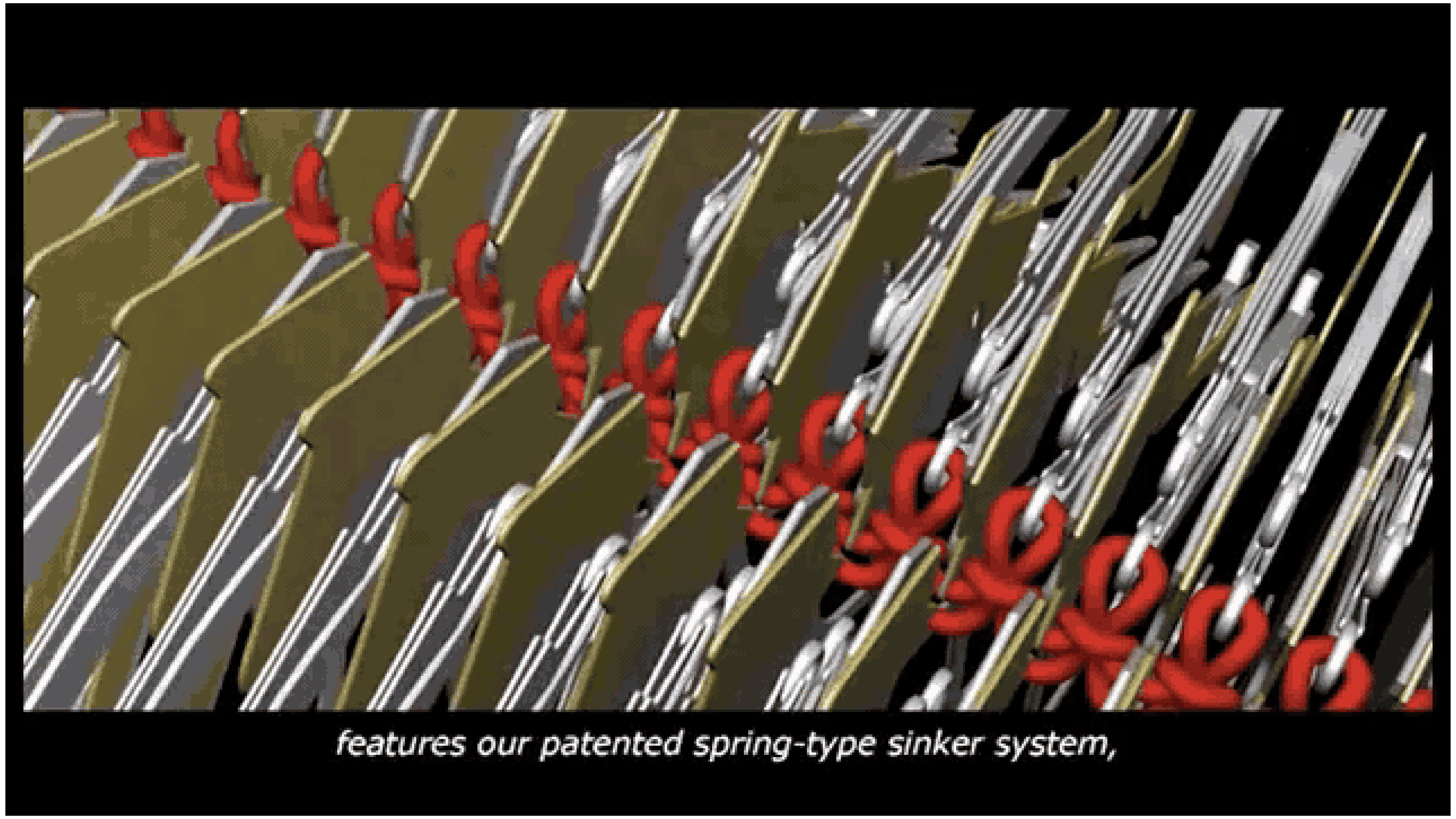
Incoming yarn: pulled through active loops to make new loops

In progress fabric: stable loops that won't unravel


Incoming yarn
Active Loops
In progress fabric
Tekstil Sayfası


KNITTING
Tekstil Sayfasi

Whole garment Knitting

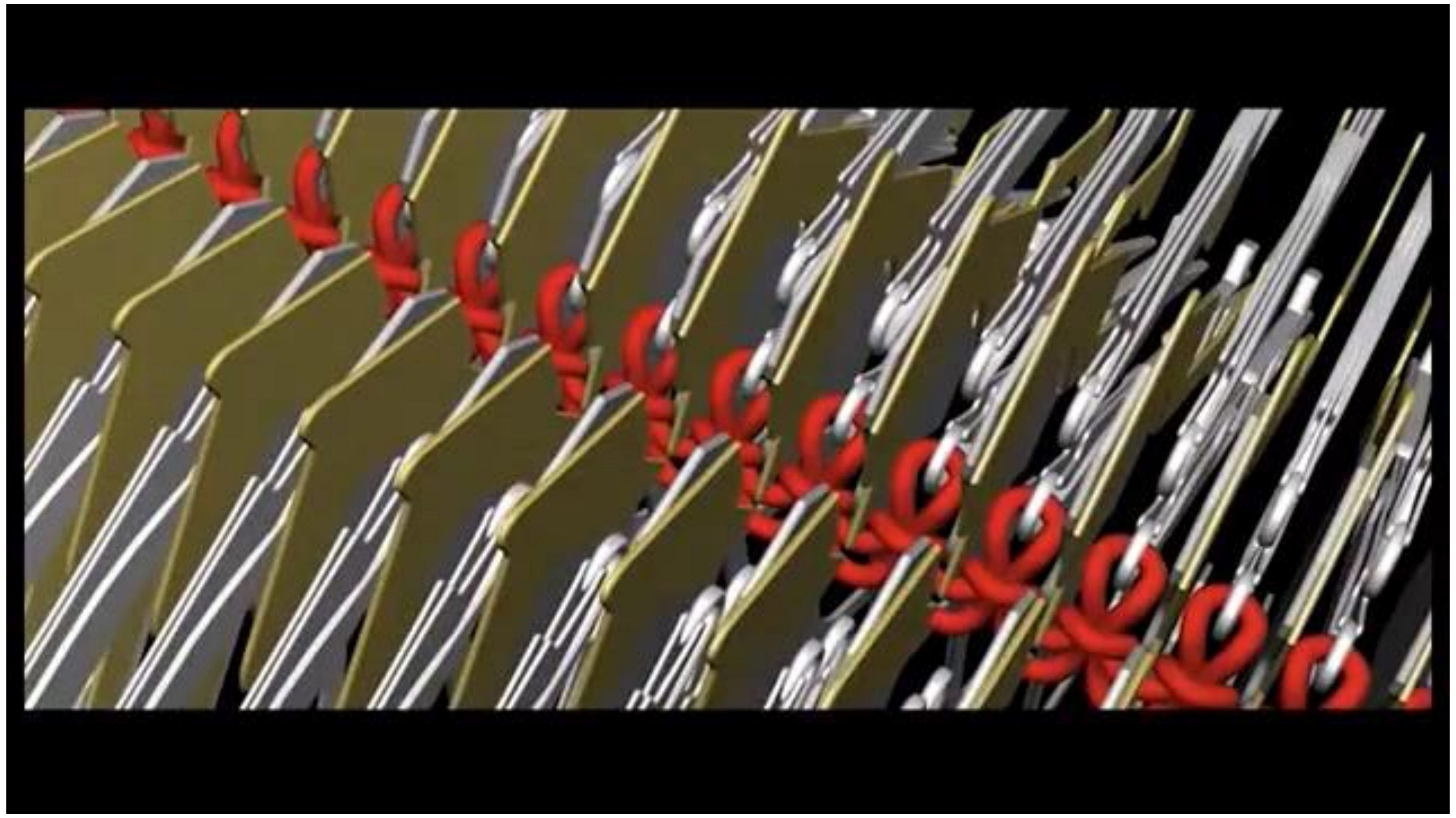


- "Whole garment" machines are akin to 3D printing a garment
- Machine knitting is much faster and more precise
  - (though less versatile than hand knitting)

Video: [Shima Seiki]

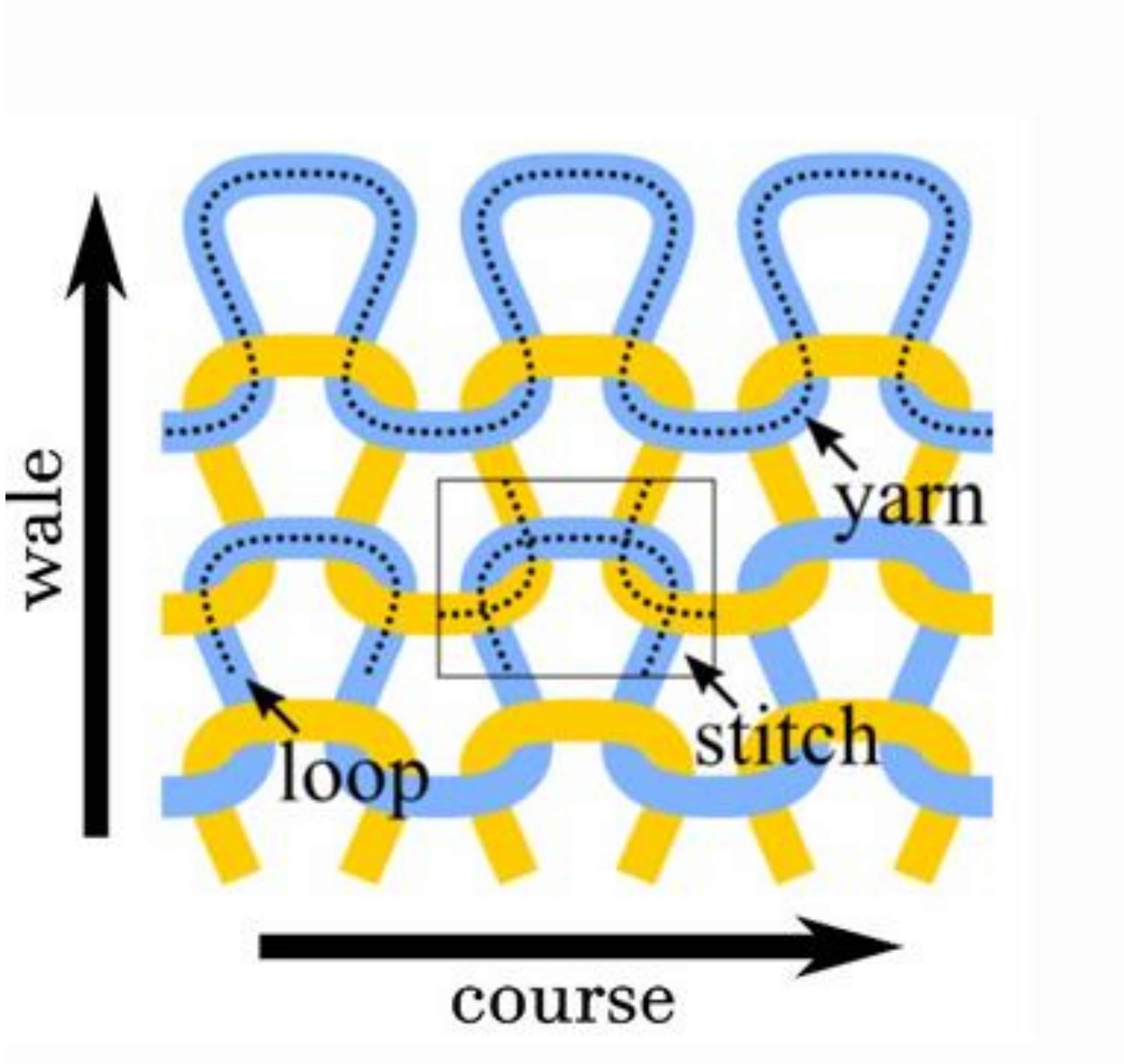
Sheets & Cylinders

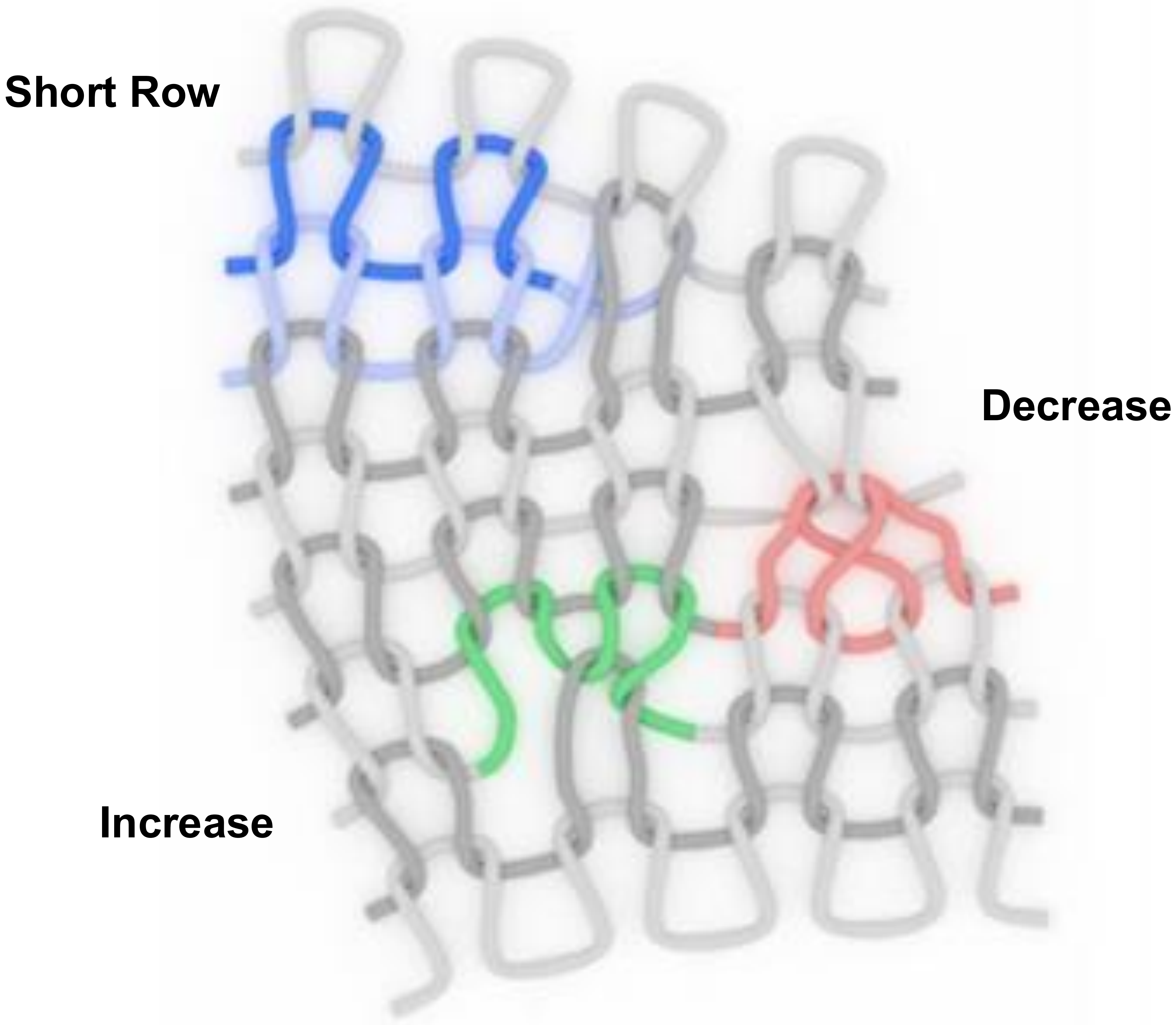
- Two basic topological structures that can be joined to make almost any surface: **sheets** and cylinders



Video: Shima Seiki, NHK

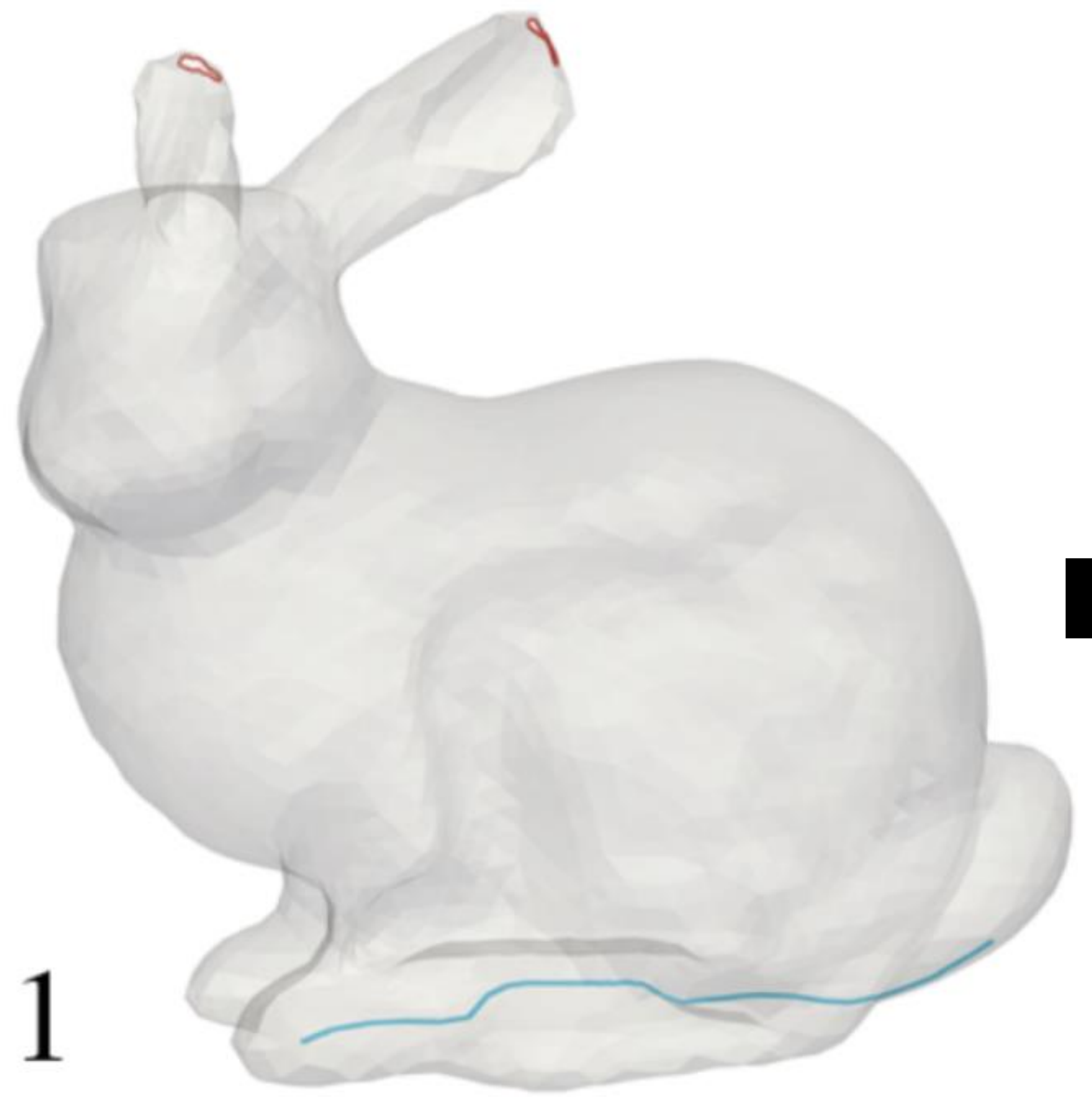
Curvature in Knitting

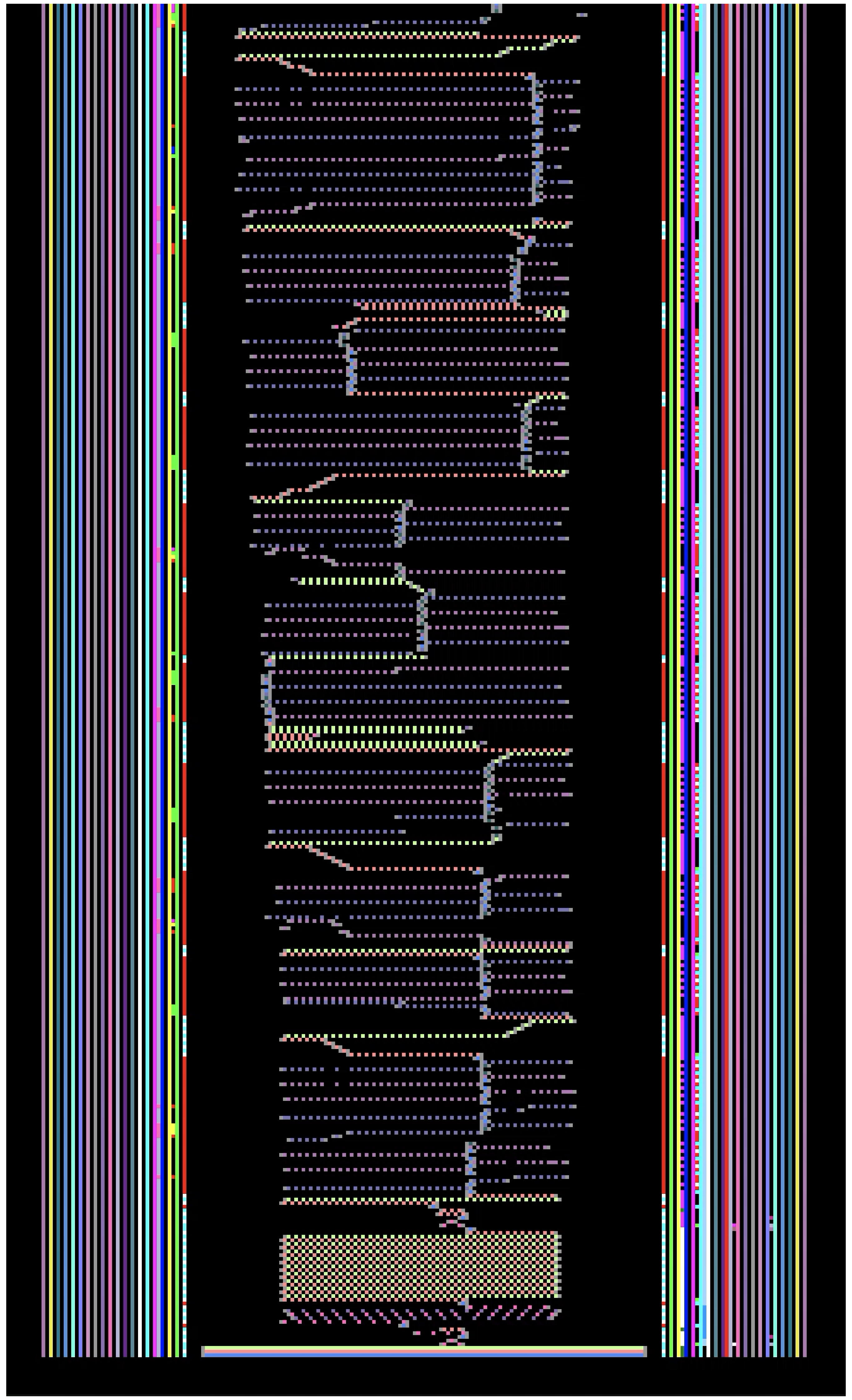




Shaping

Why is Knitting Computer Graphics?

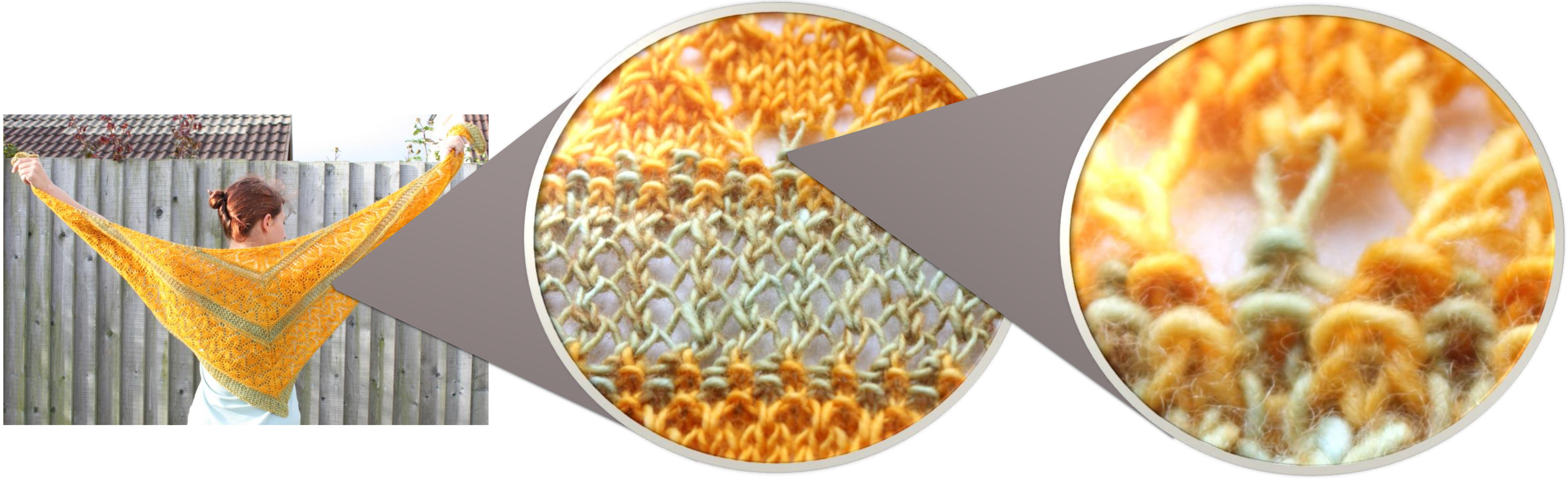




Organization

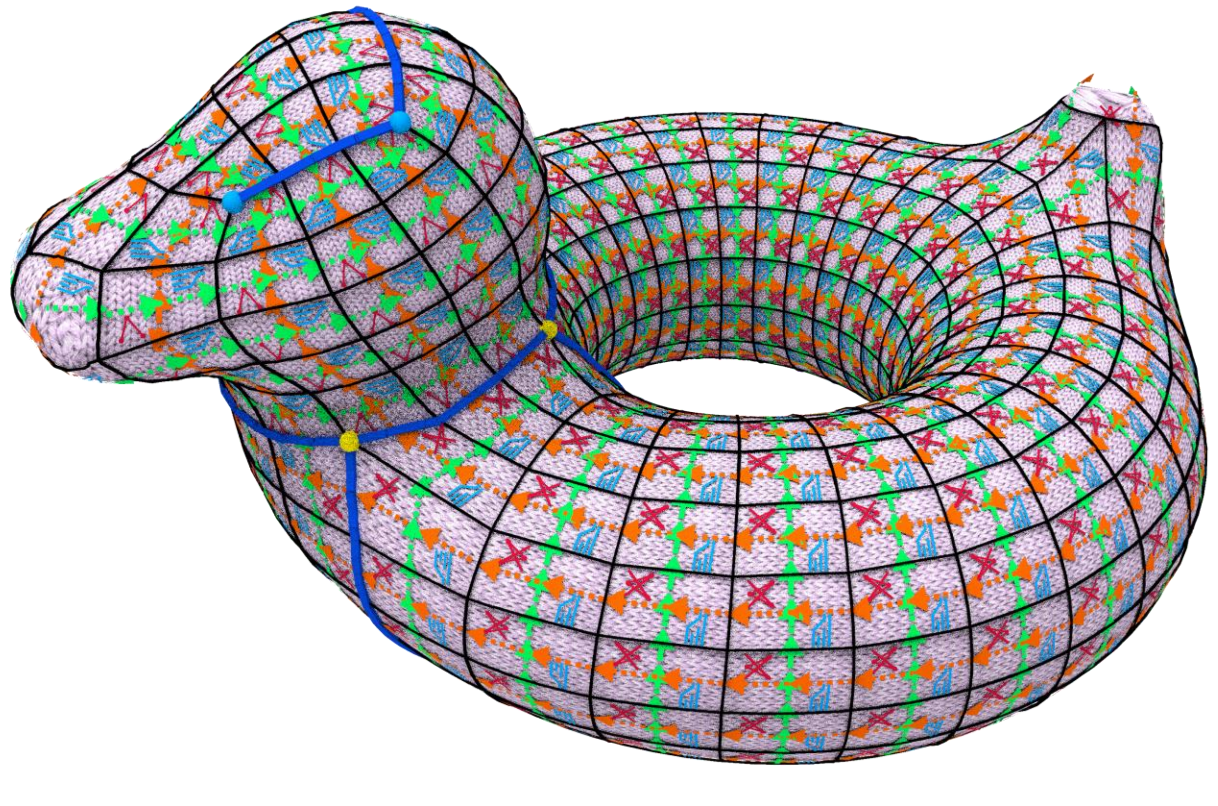
- Knitting Background
  - What is Knitting and Why is it Important?
  - Why is Knitting Geometry?
- Computational Knitting Representations and Algorithms
  - Fabric Level
  - Stitch Level
  - Yarn Level
- Open Problems in Computational Knitting

Knitting Levels of Detail

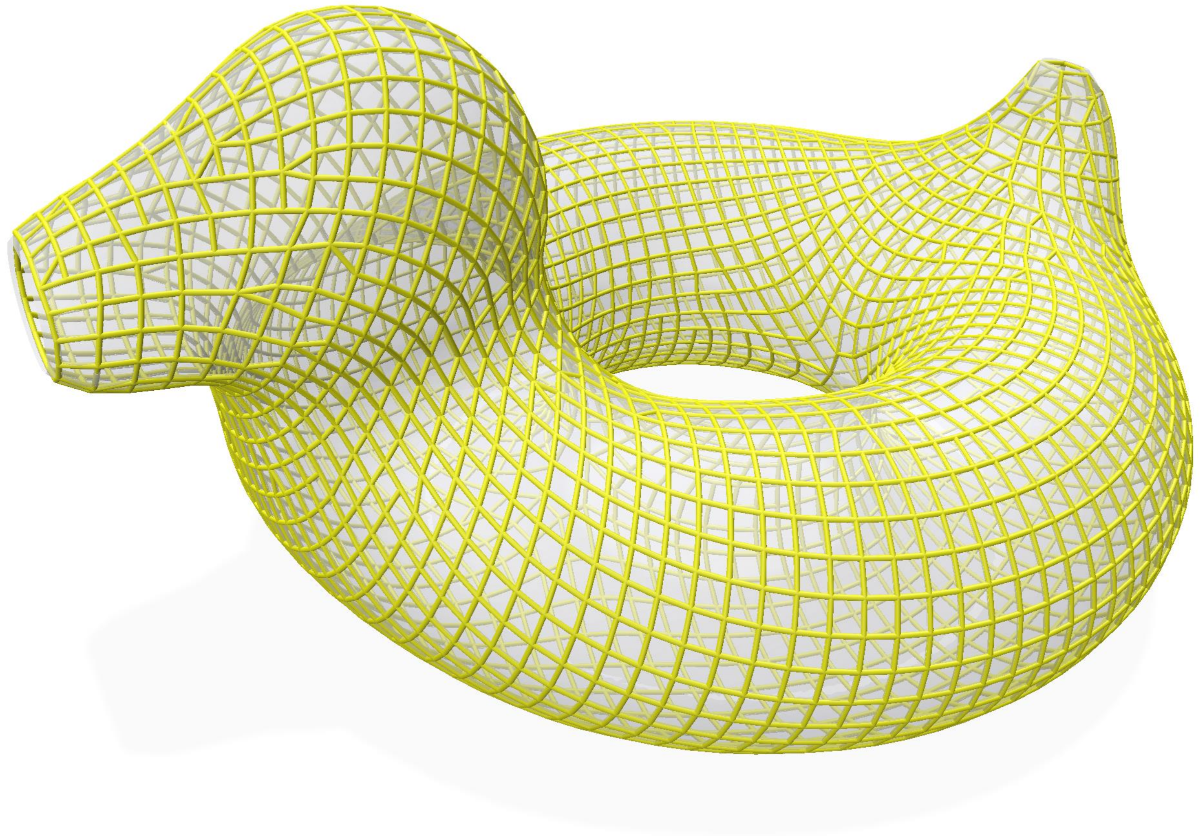


Fabric **Stitch** **Yarn**

Knitting Levels of Abstraction



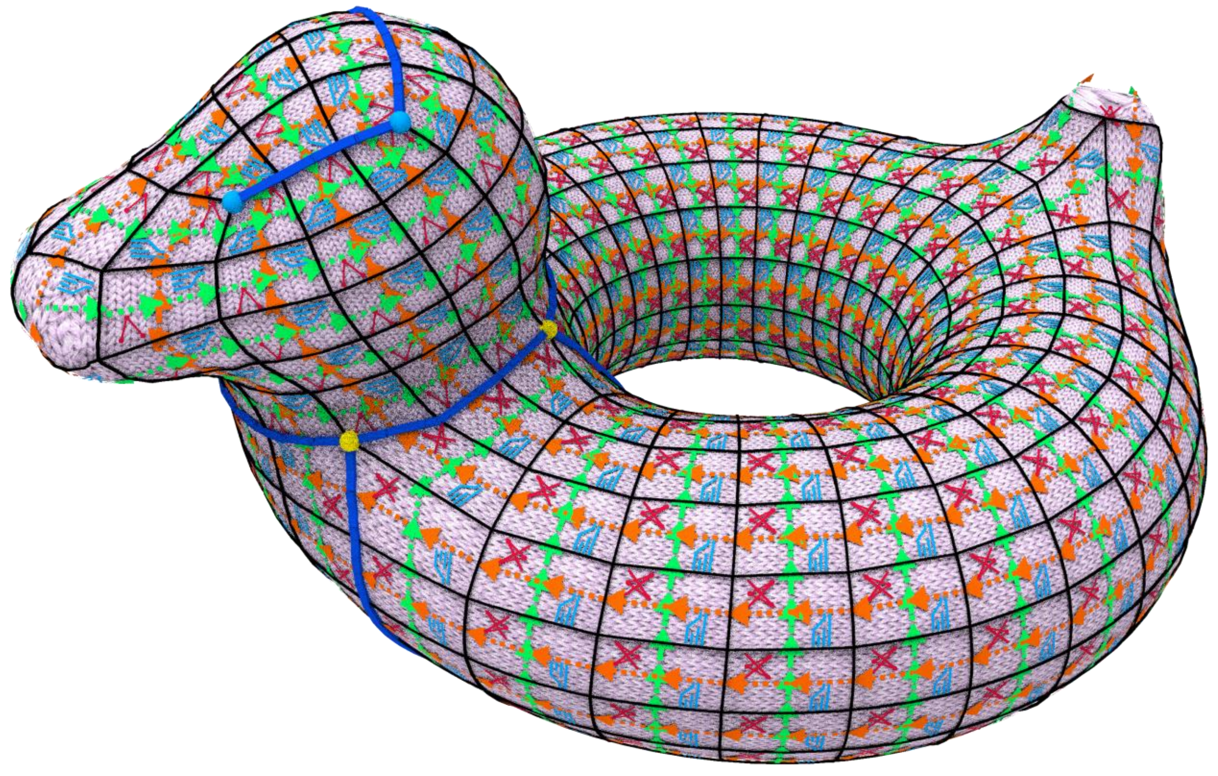
Fabric

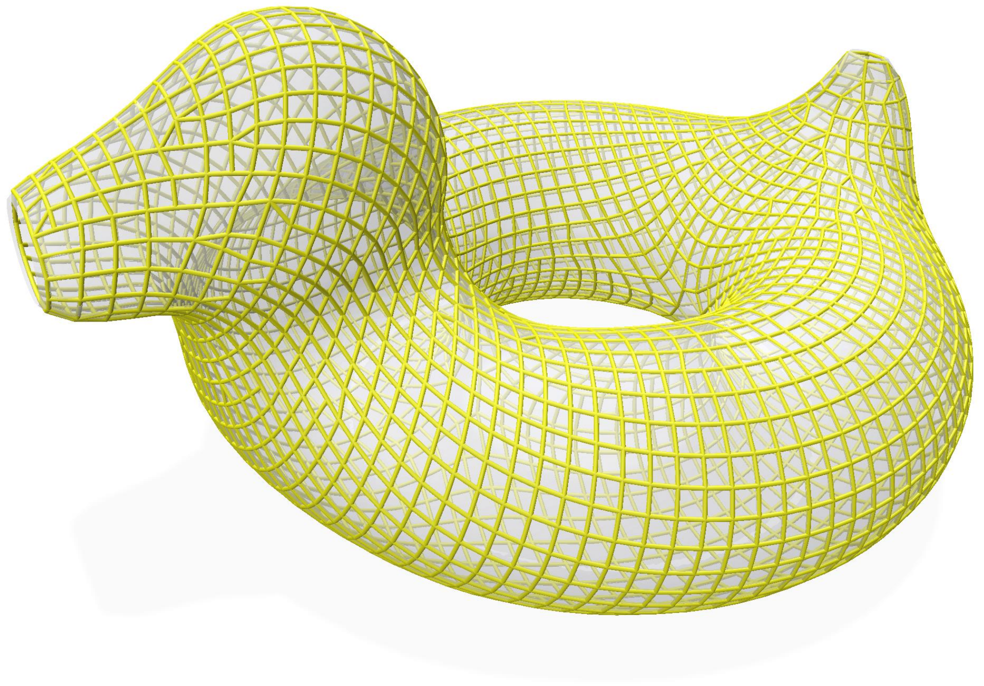


Stitch

Yarn

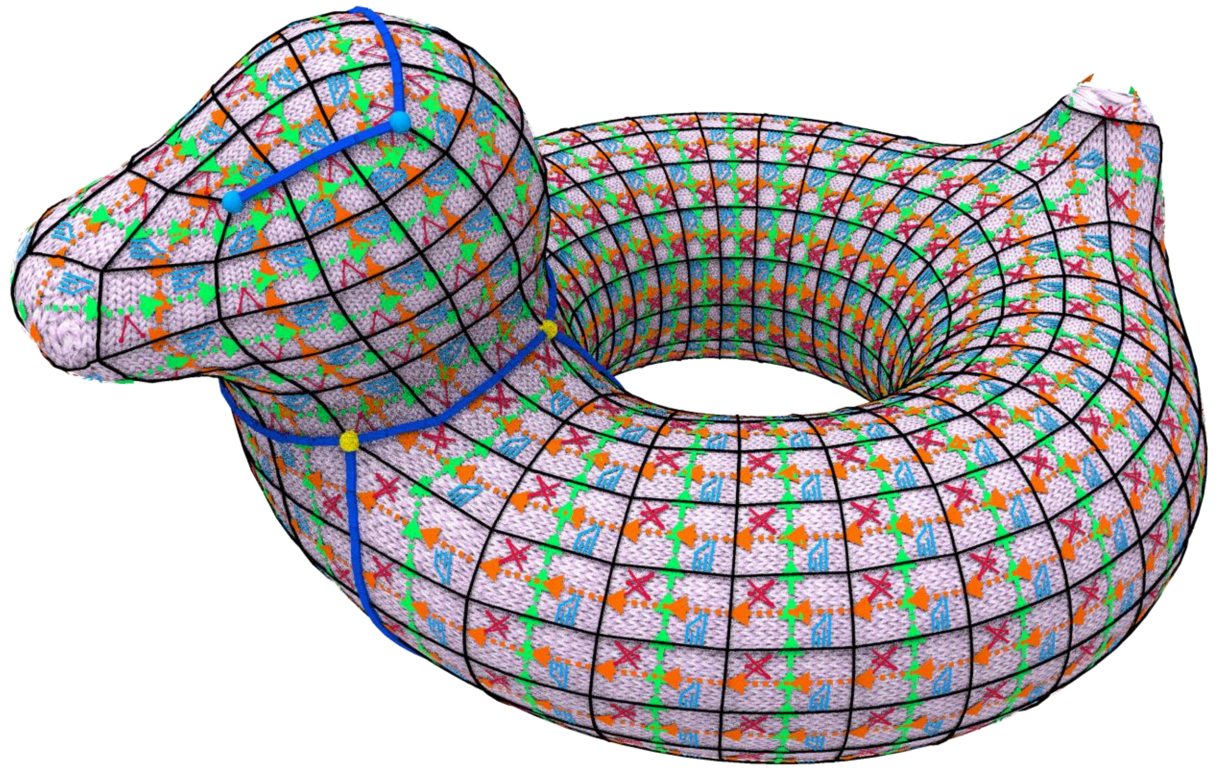
Knitting Levels of Abstraction



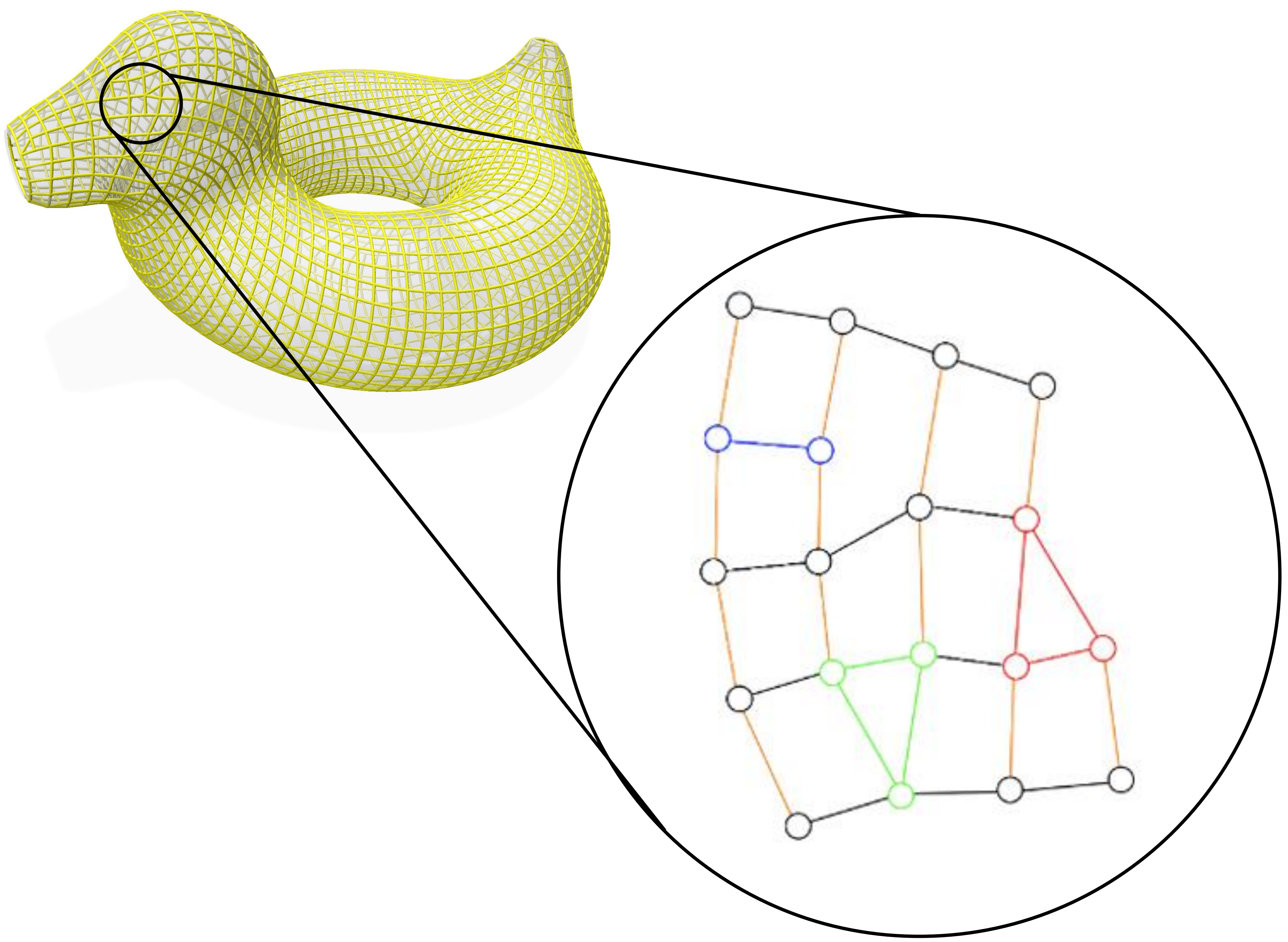


Fabric **Stitch** **Yarn**

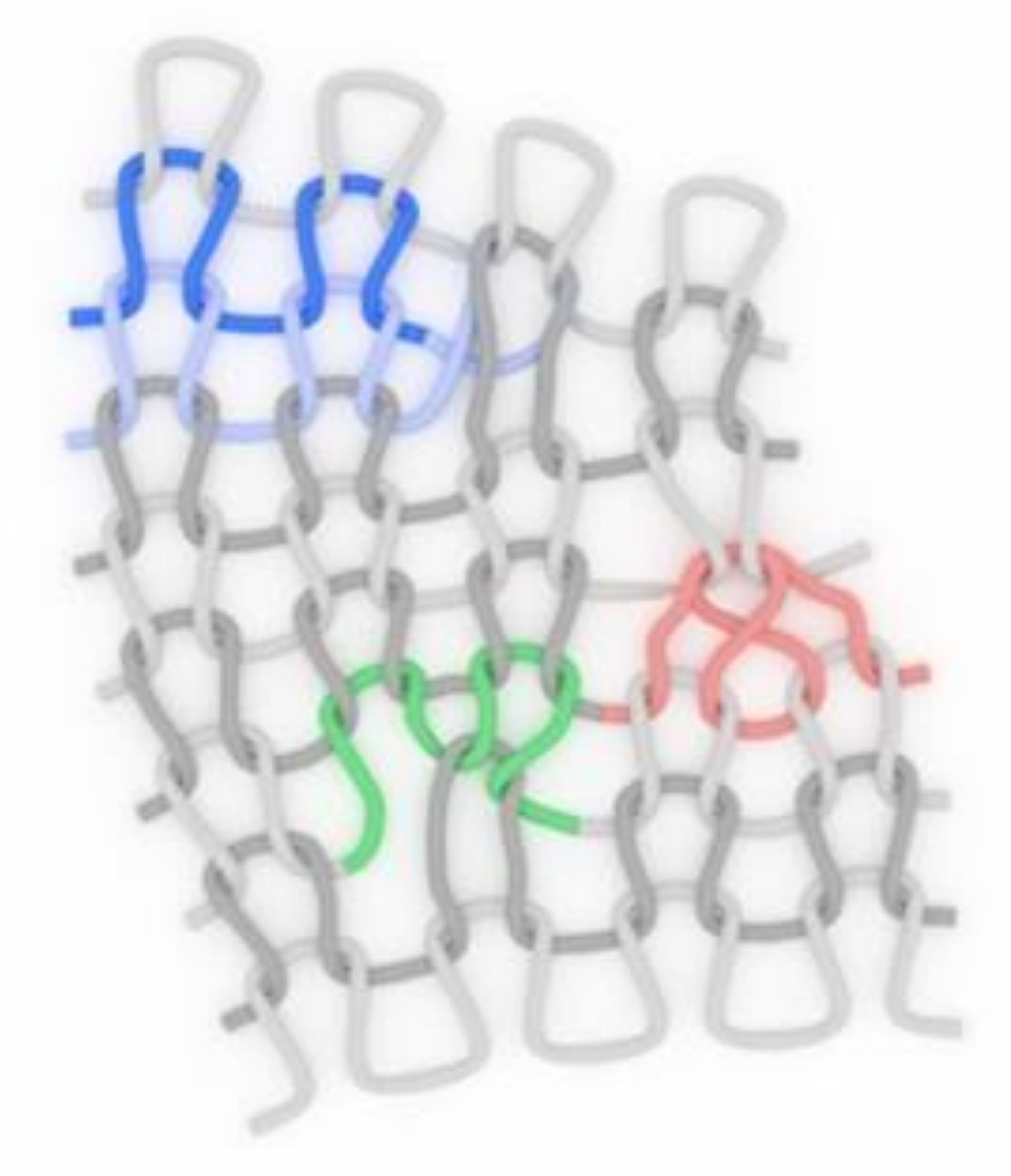
Knitting Levels of Abstraction



Fabric

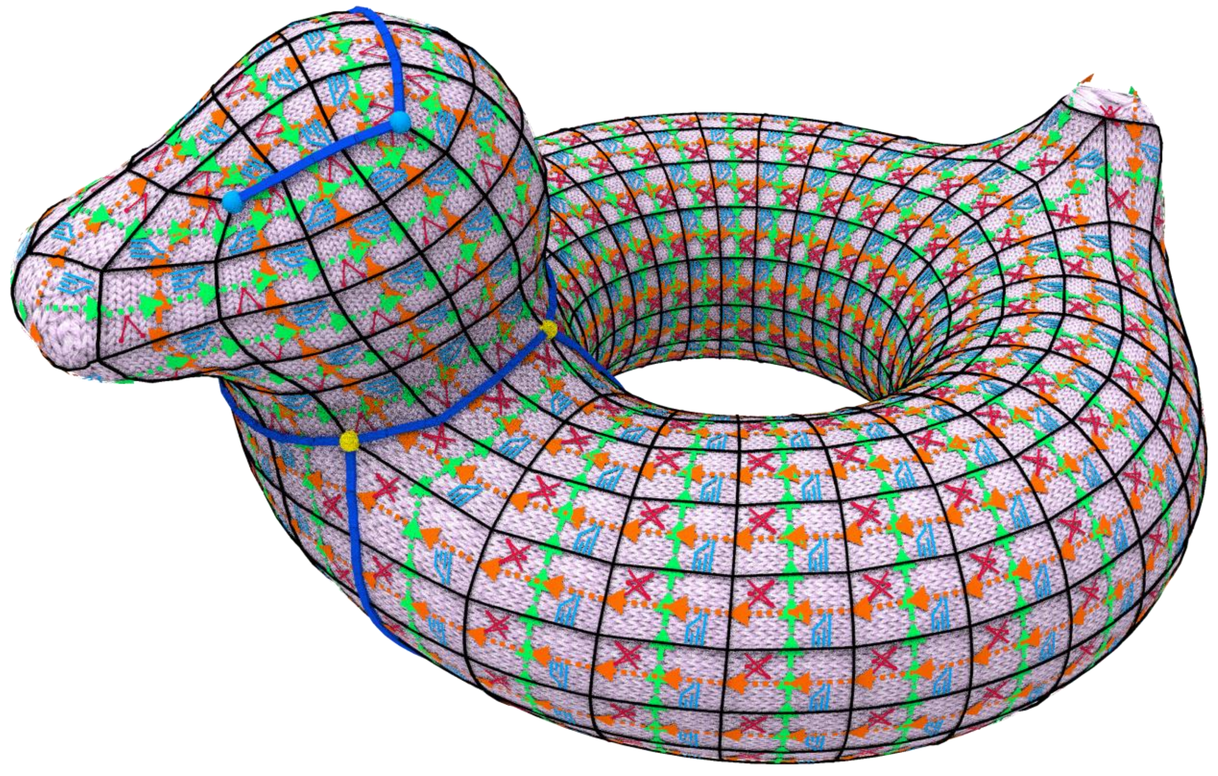


Stitch

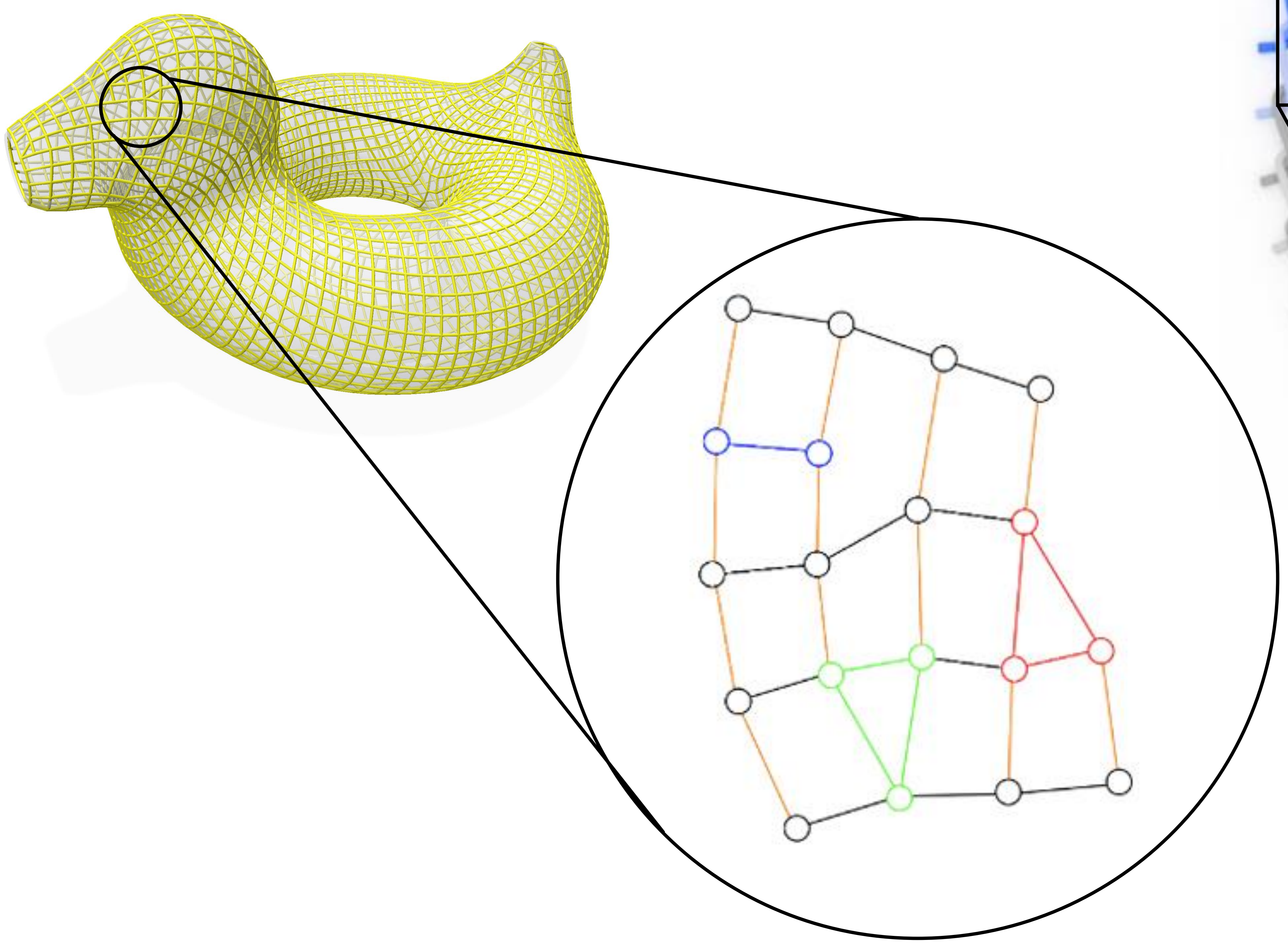


Yarn

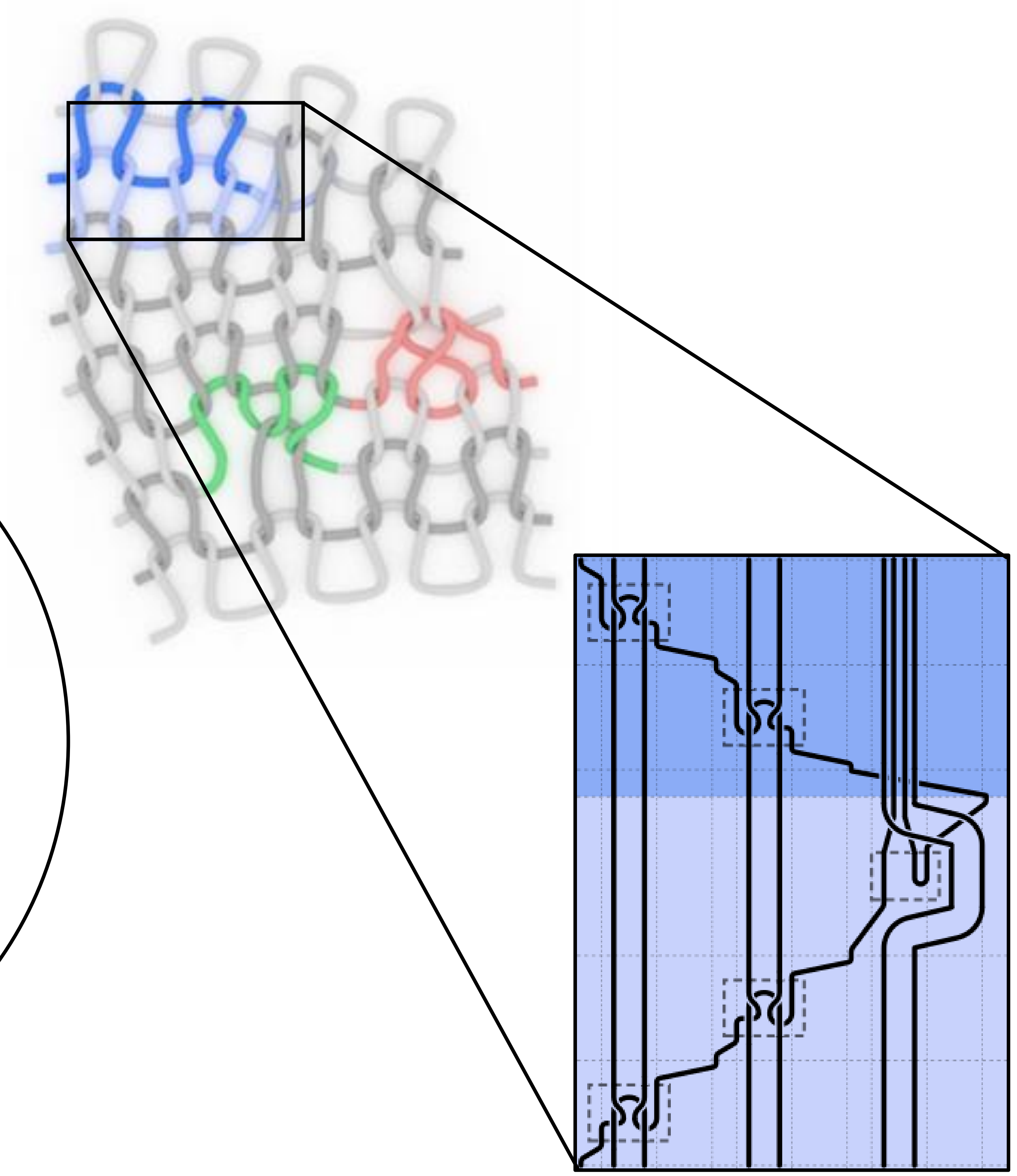
Knitting Levels of Abstraction



Fabric



Stitch

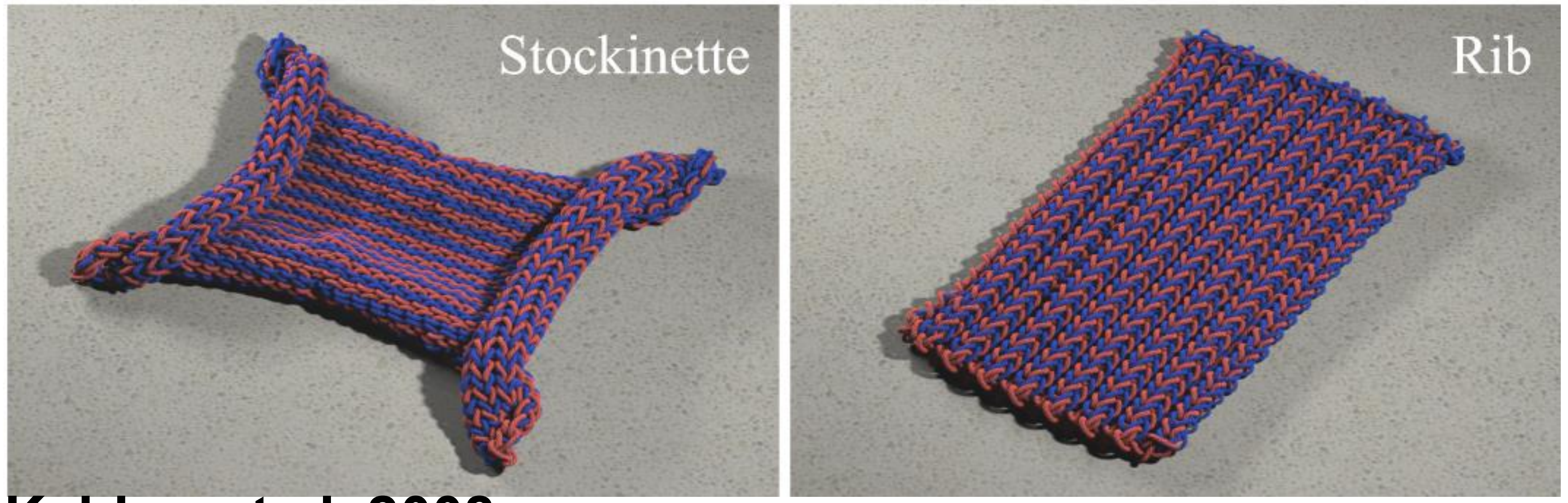


Yarn

A Brief History of Knit Representations

Knitting Levels of Abstraction

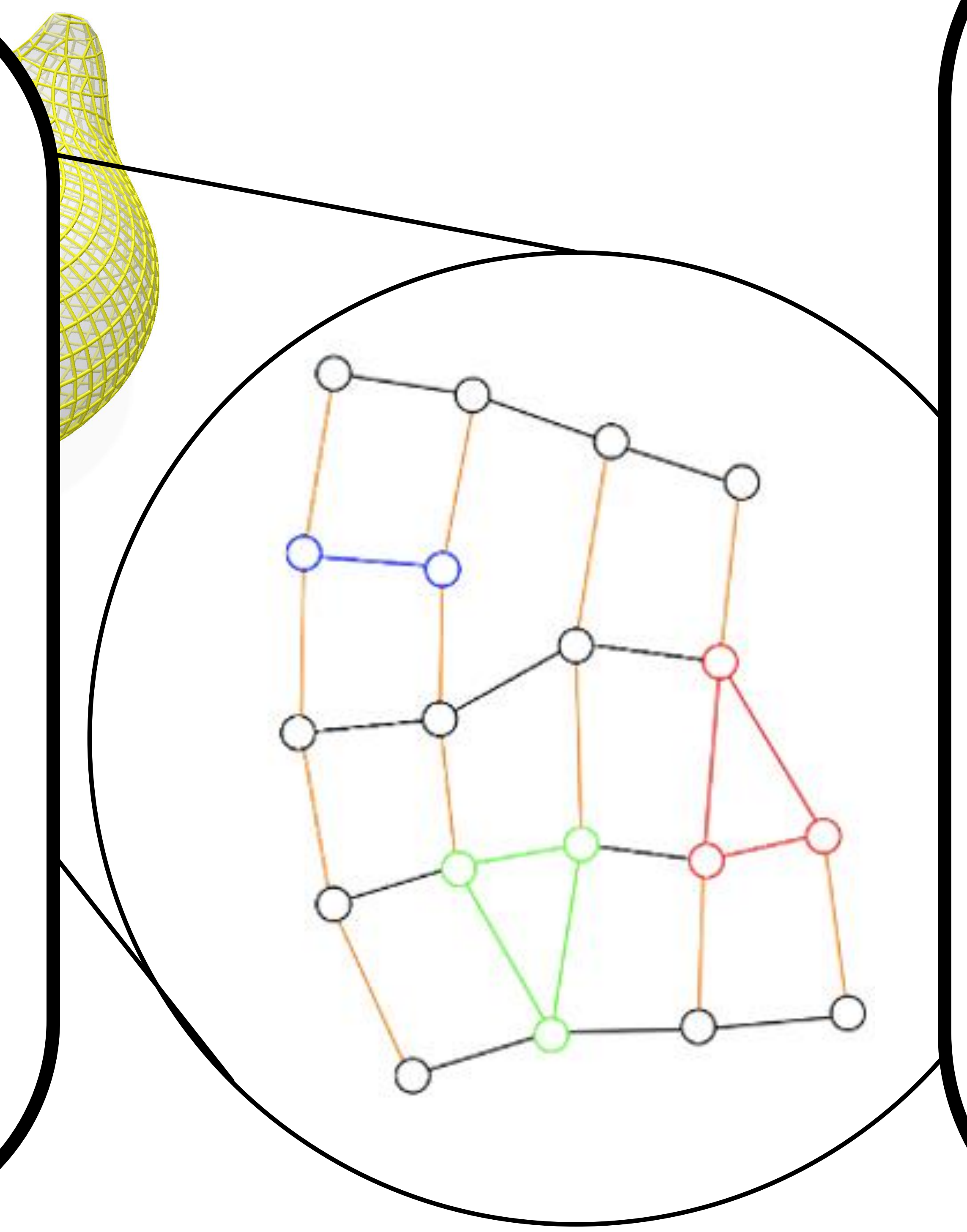
Yarn-Level Simulation



Kaldor, et al. 2008

Detailed but slow

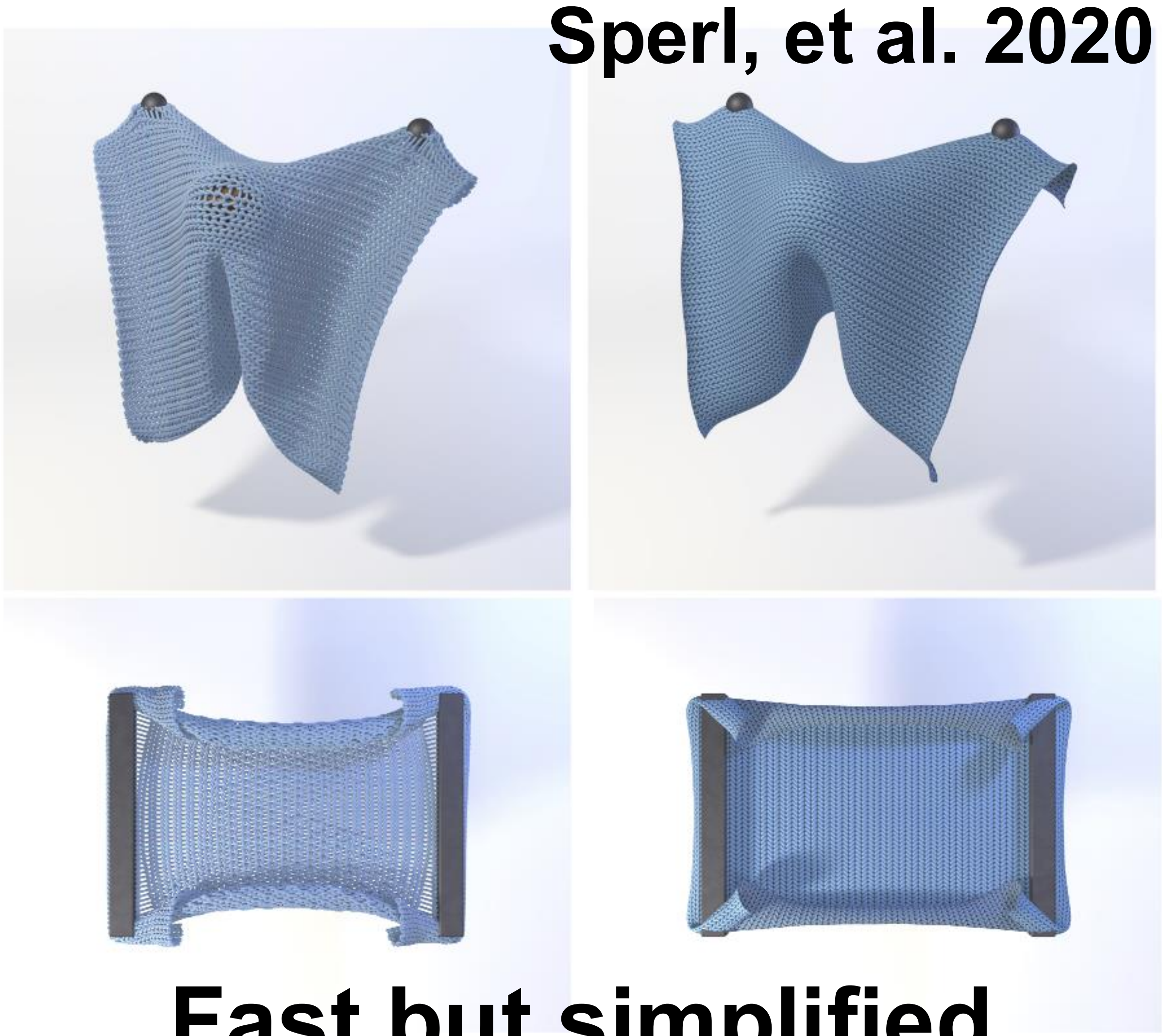
Fabric



Stitch

Thin-Sheet Simulation

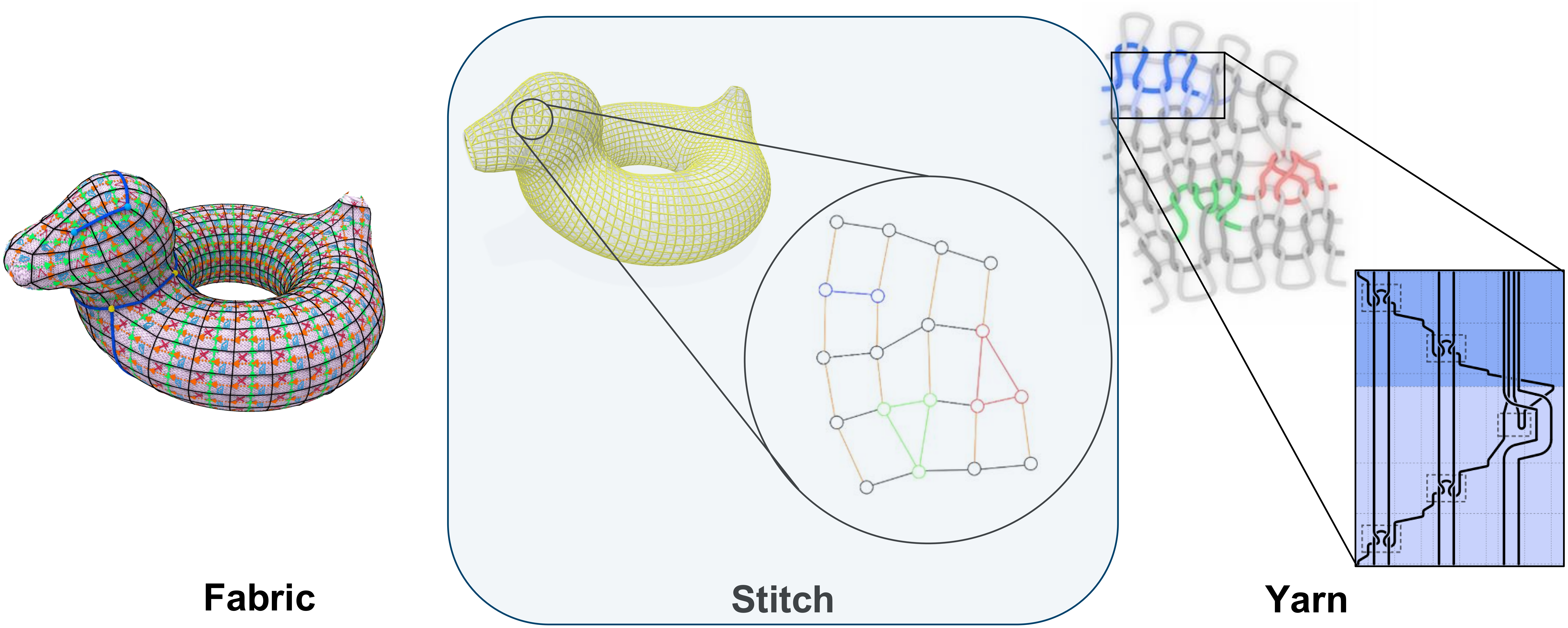
Sperl, et al. 2020



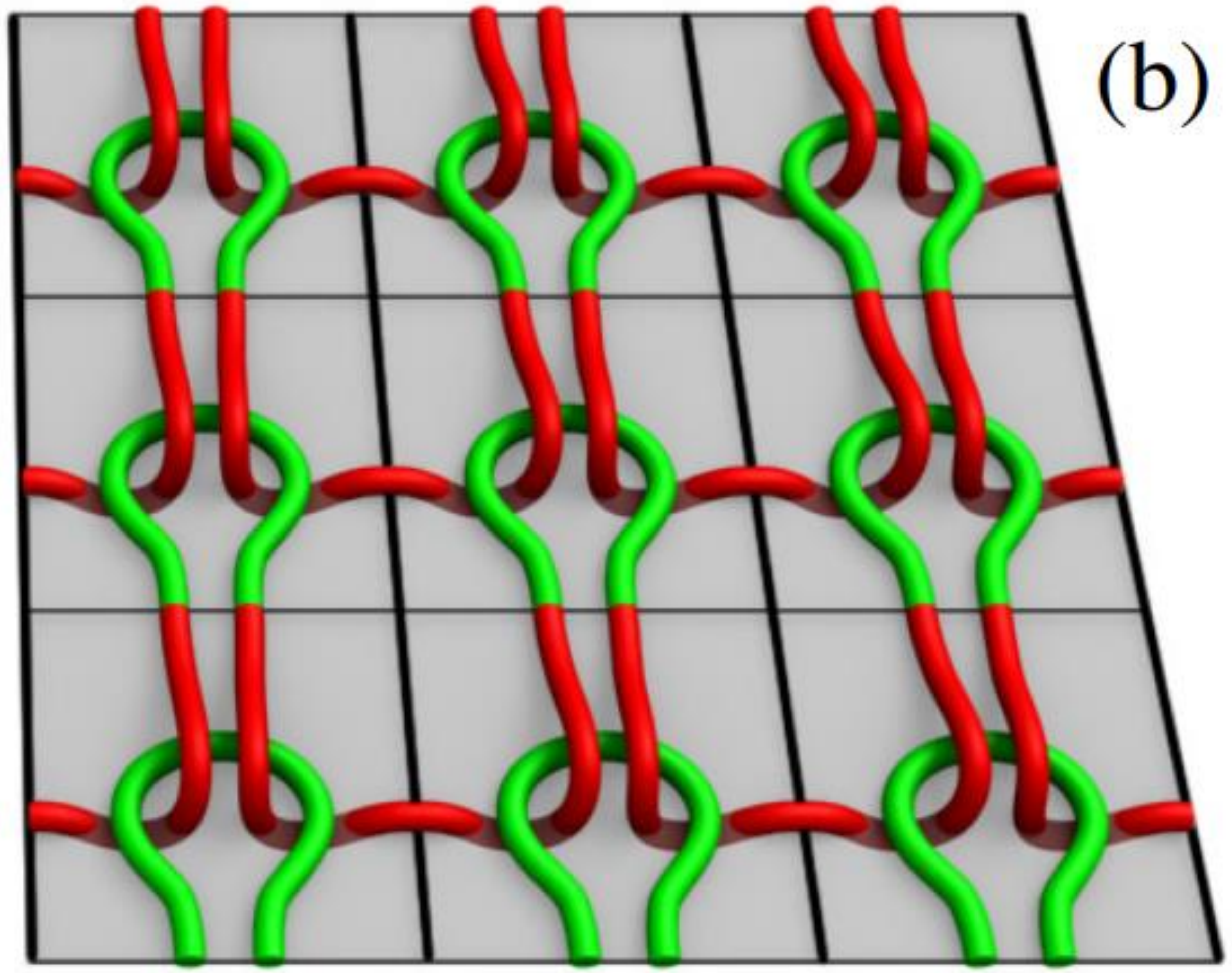
Fast but simplified

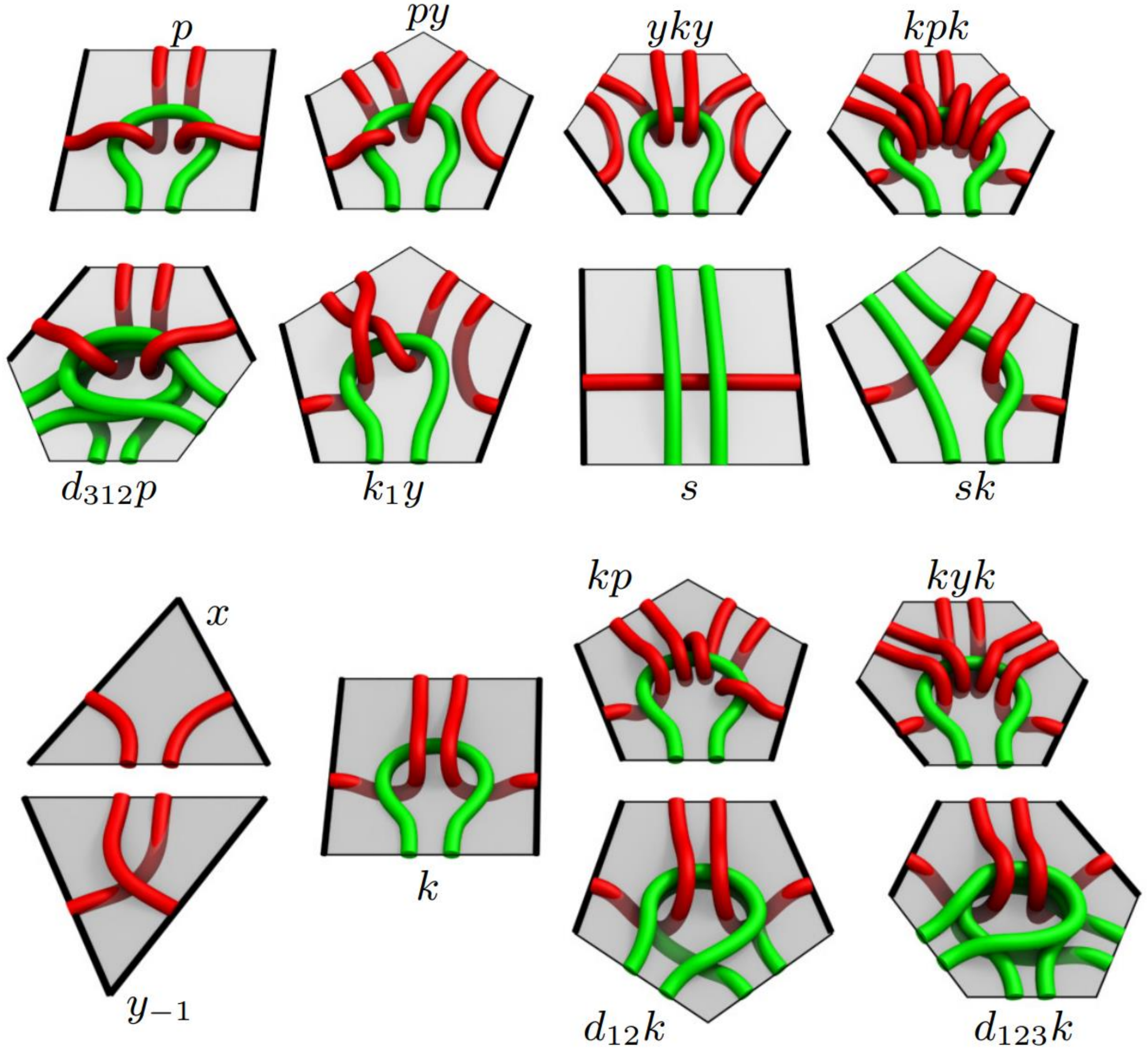
Yarn

Knitting Levels of Abstraction



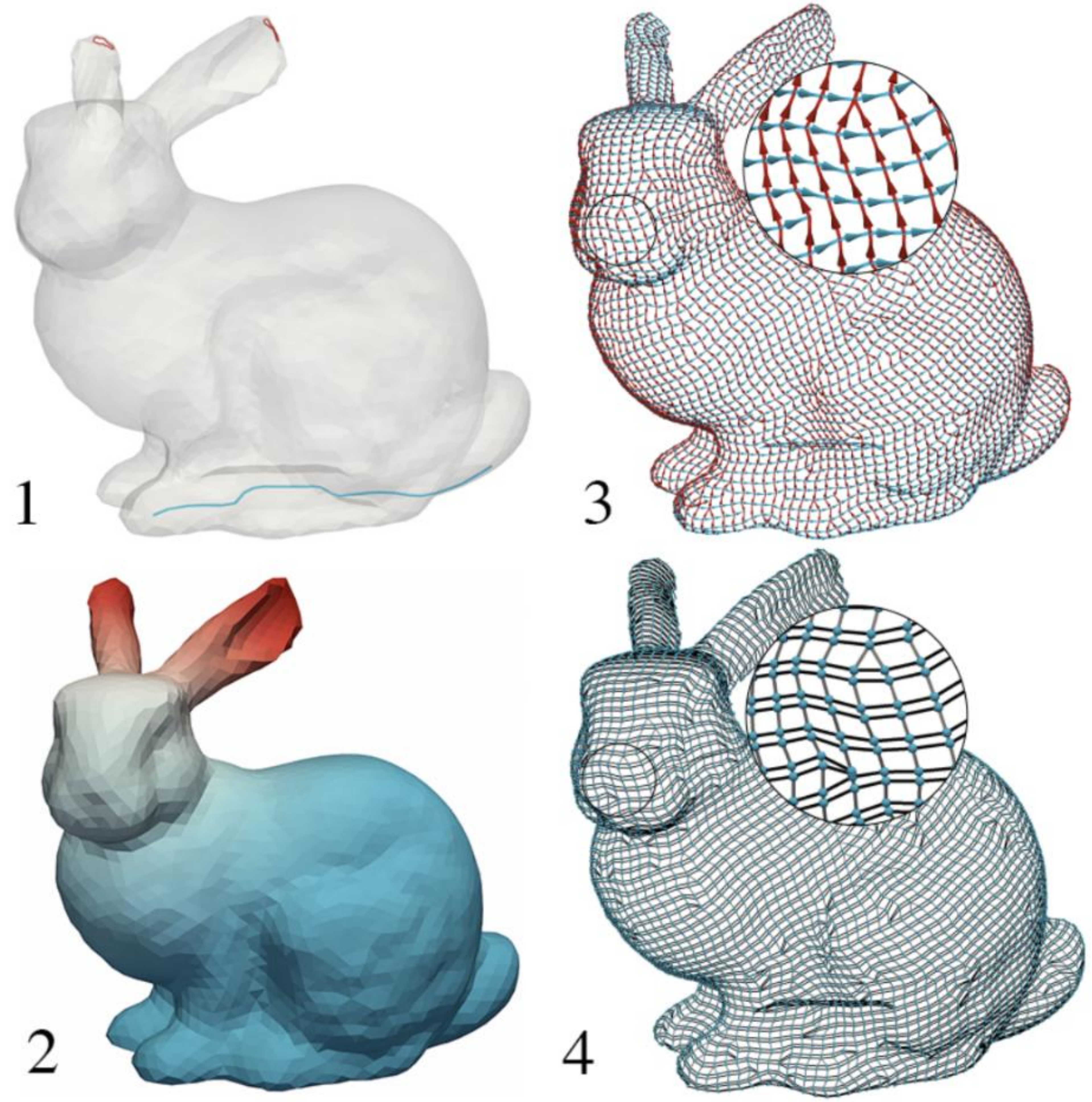
Stitch-Level Abstractions: Meshes

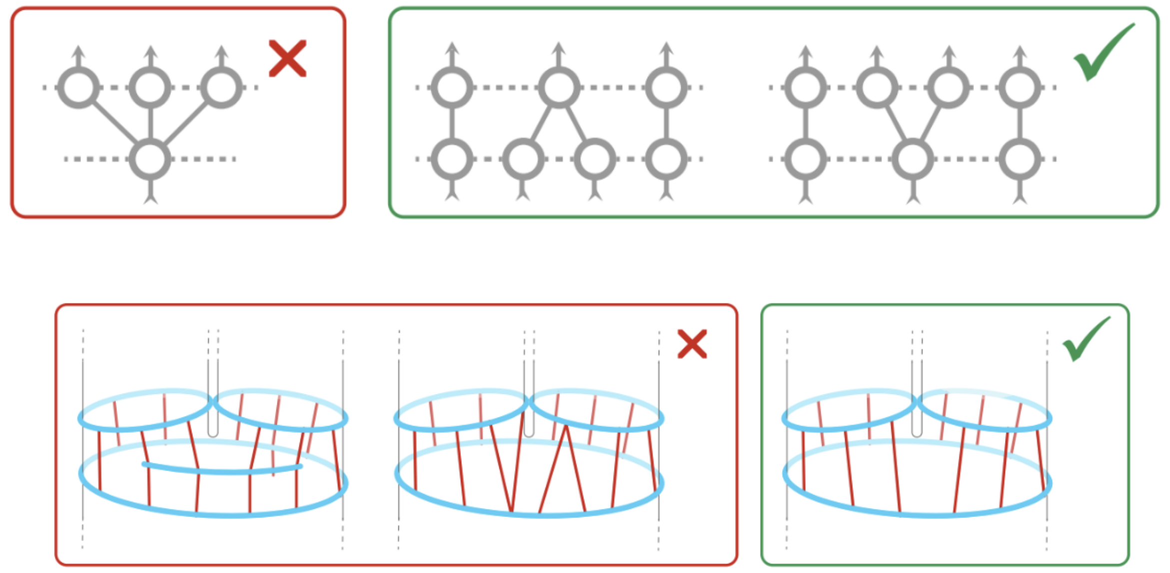


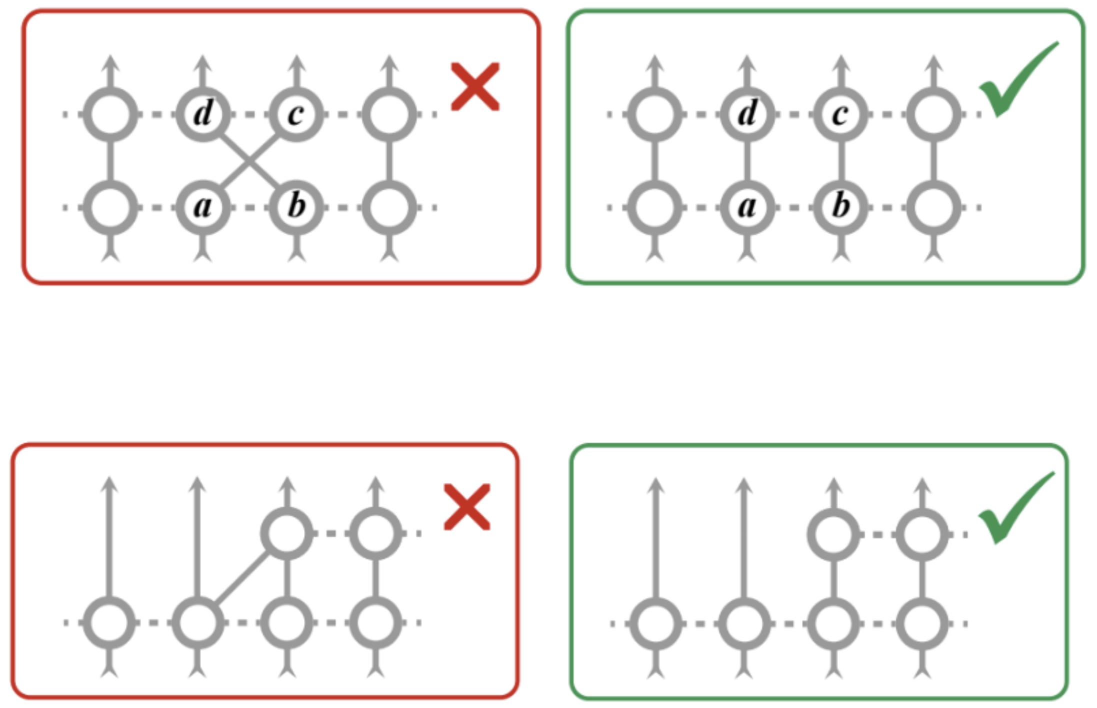


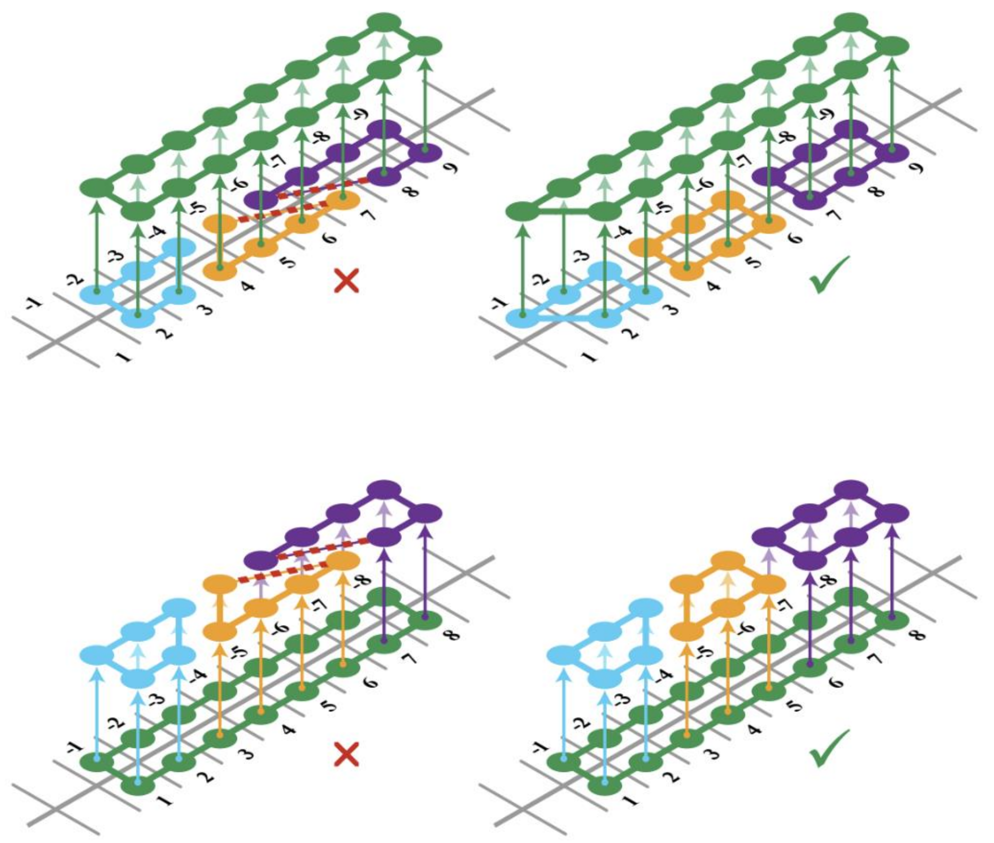
Yuksel et al 2012

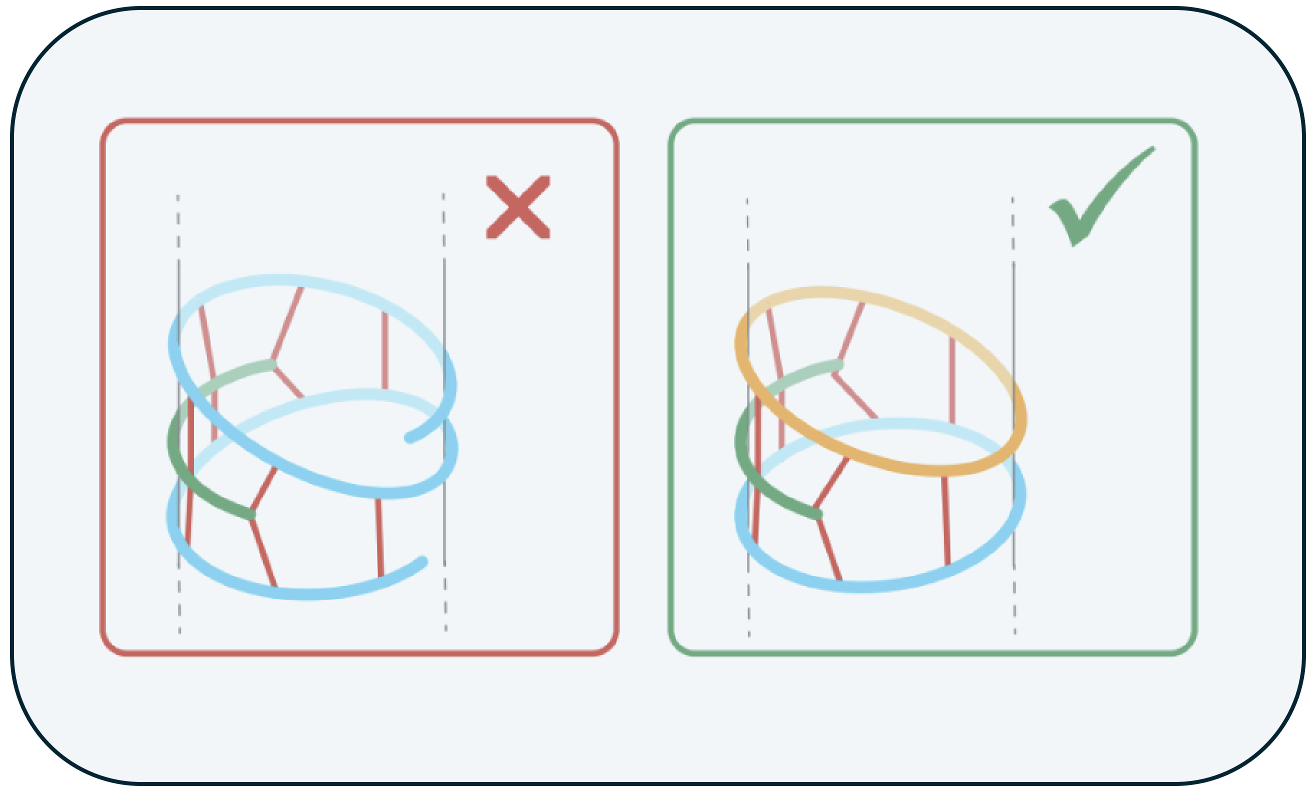
Stitch-Level Abstraction: Graphs





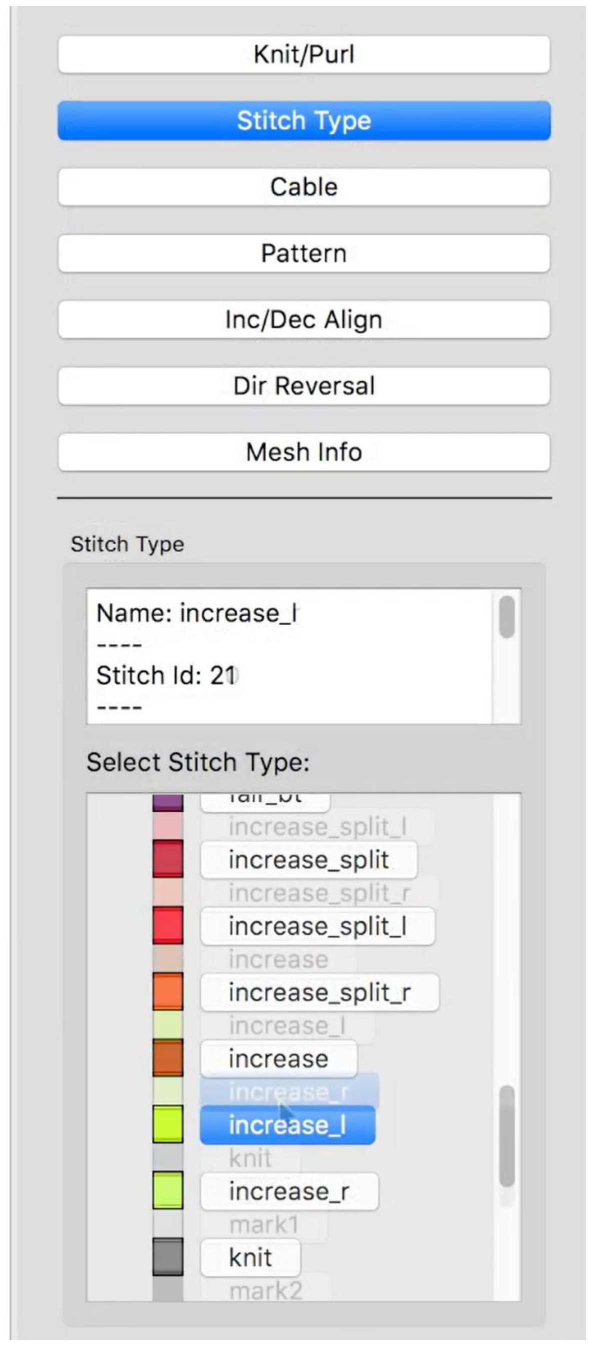


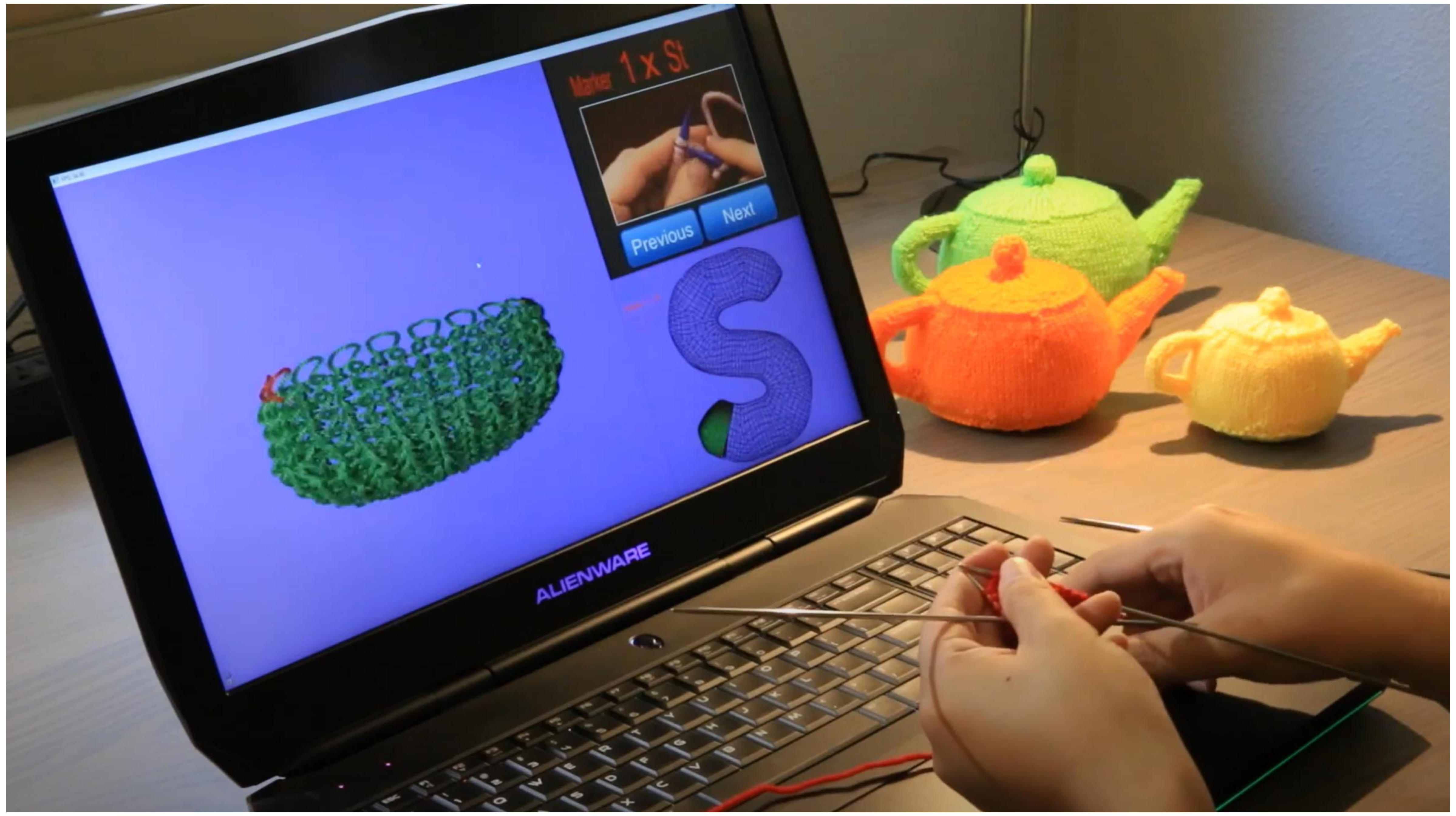




Nayaranan et al 2018

Visual Stitch-Level Programming

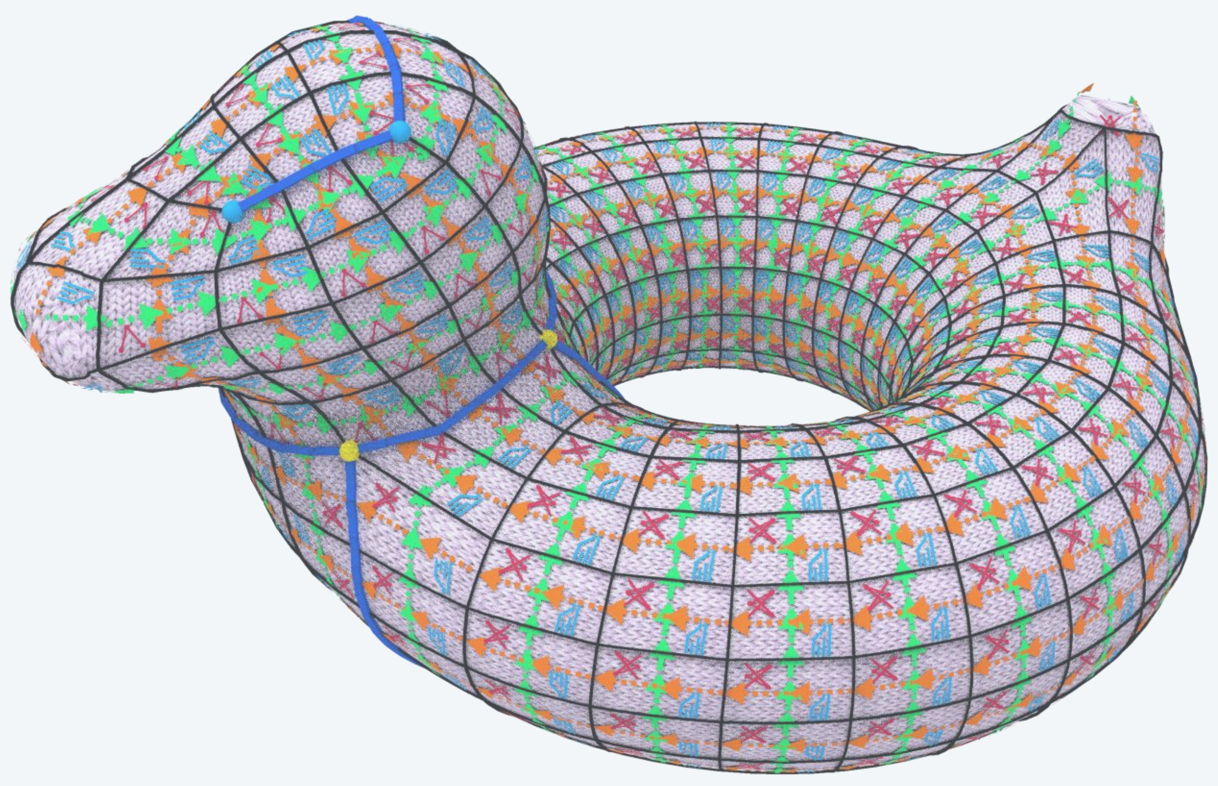




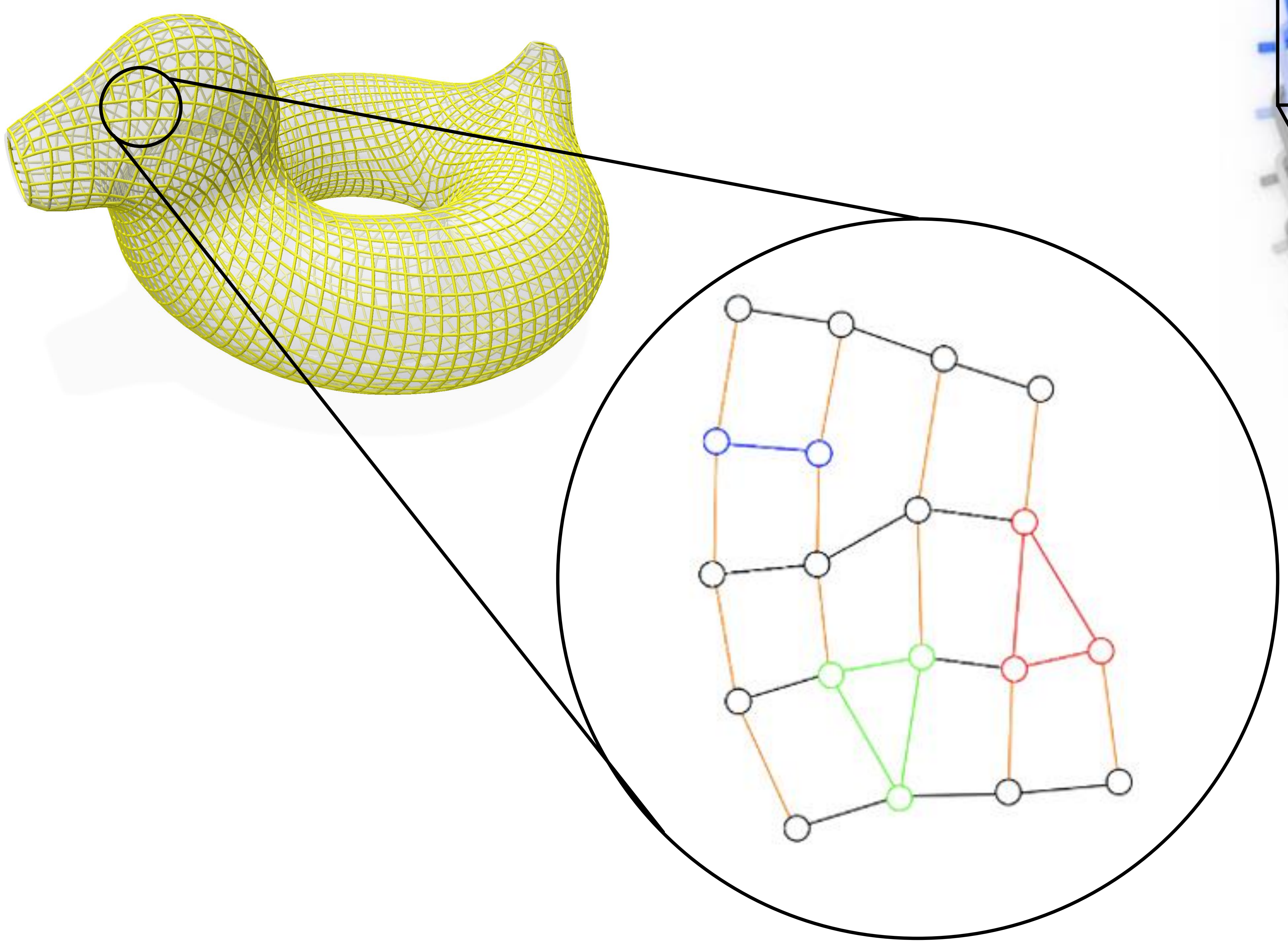
Narayanan et al 2019 (SIGGRAPH)

Wu et al 2019 (SIGGRAPH)

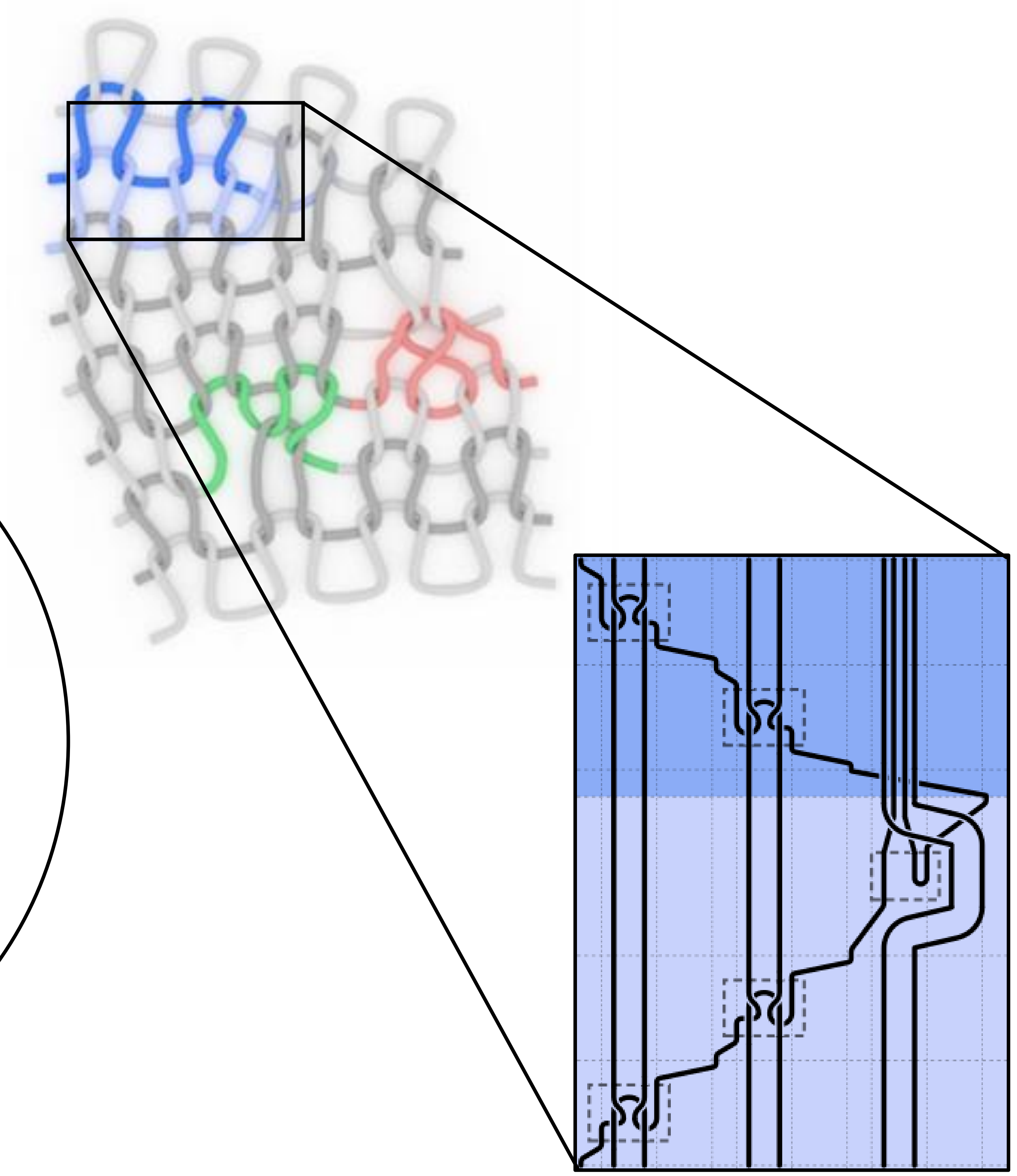
Knitting Levels of Abstraction



Fabric



Stitch



Yarn

Stitches Aren’t the Whole Story

Image: HAHA MAMA Clothing

Stitches Aren’t the Whole Story

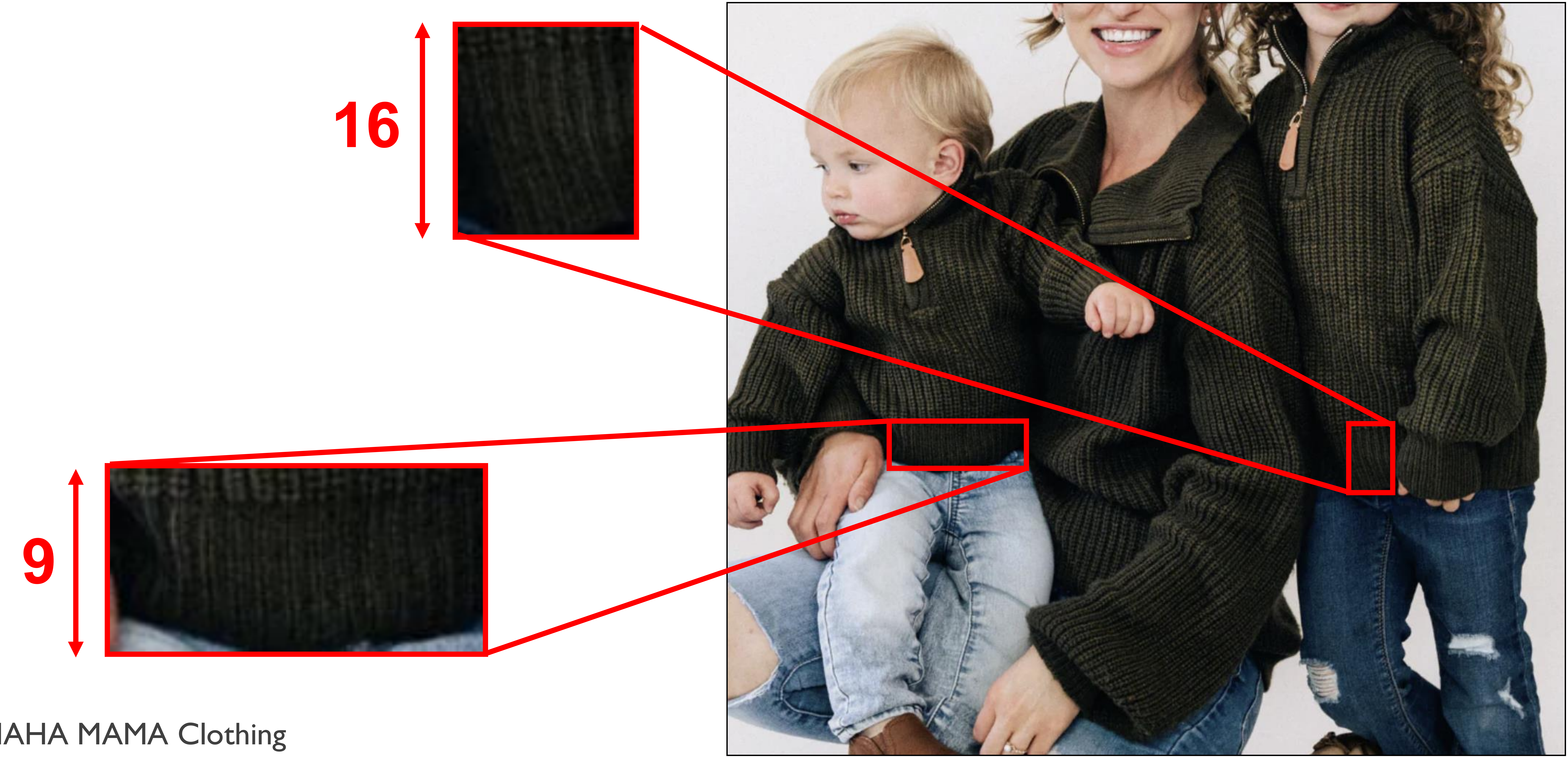


Image: HAHA MAMA Clothing

Computation Design of Knit Templates

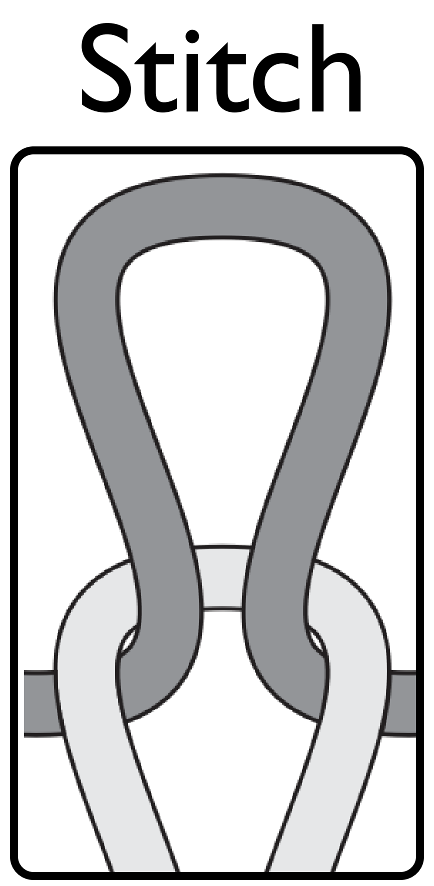
(Siggraph 2022)

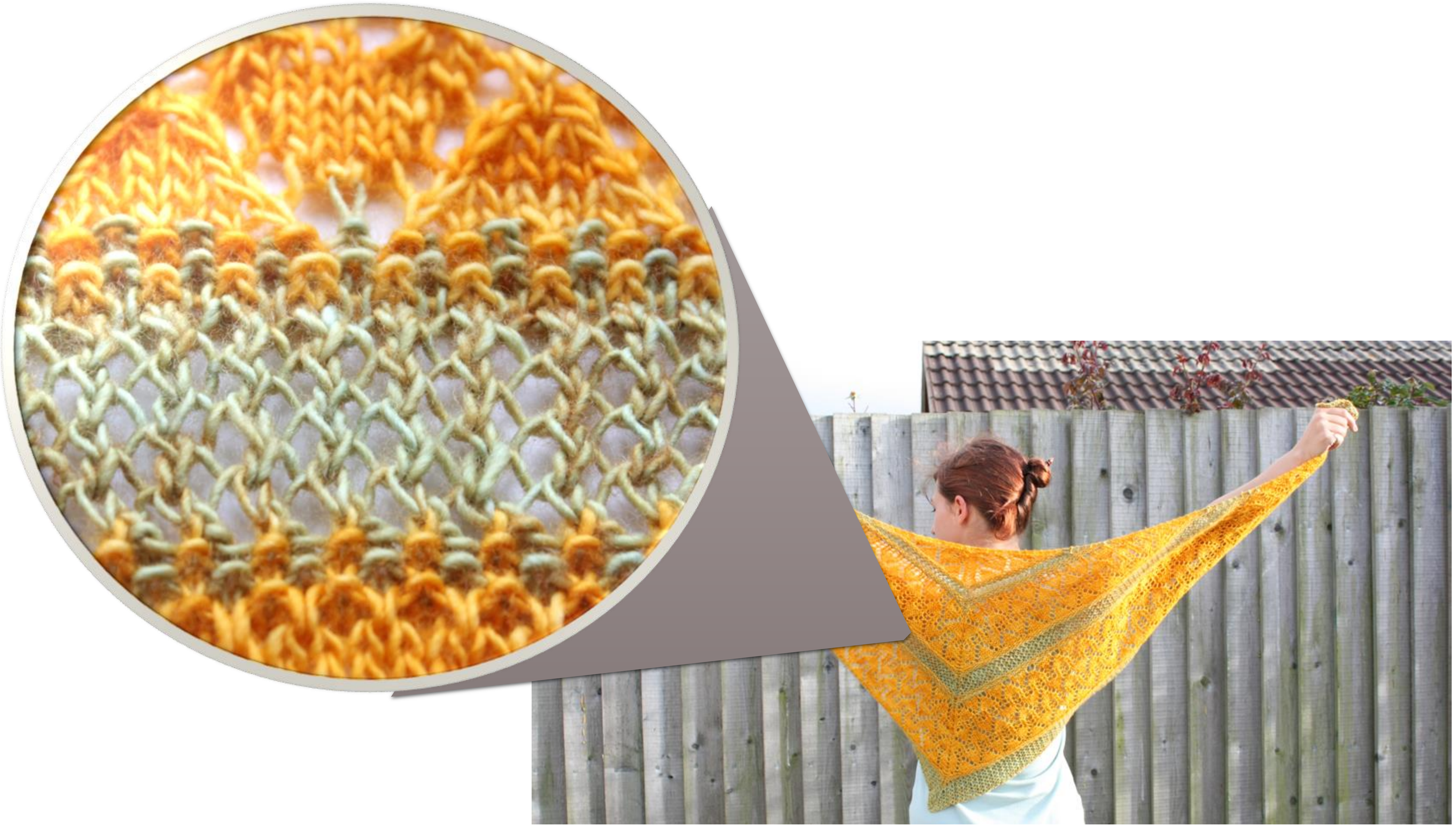
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Knitting is Customizable

Image: Kim Scarborough

Knitting is Discrete





Knit Design Axes

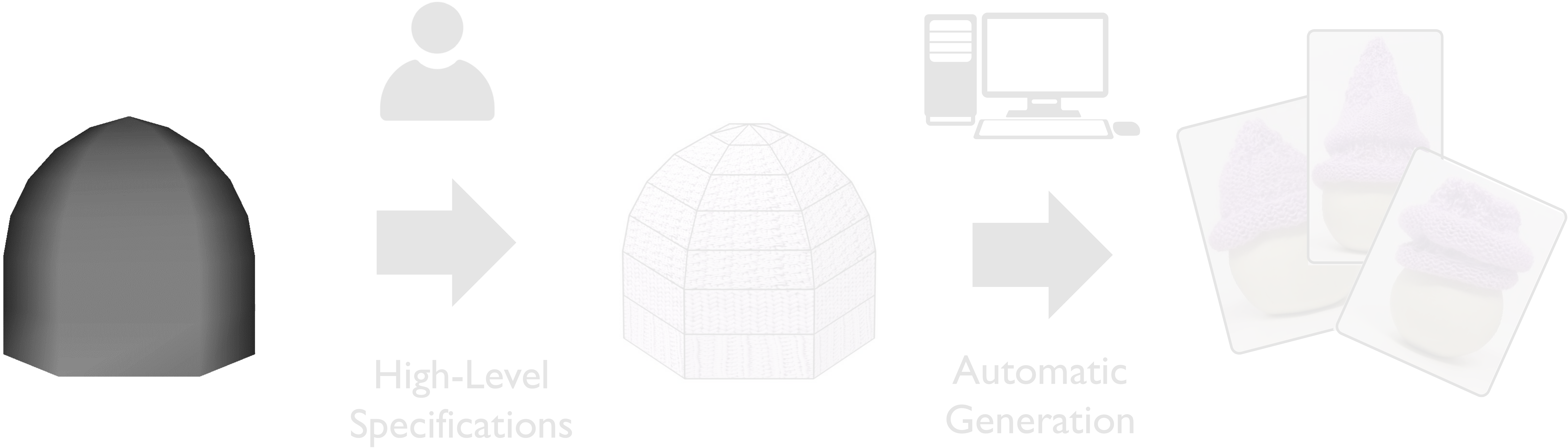
Texture

Shaping

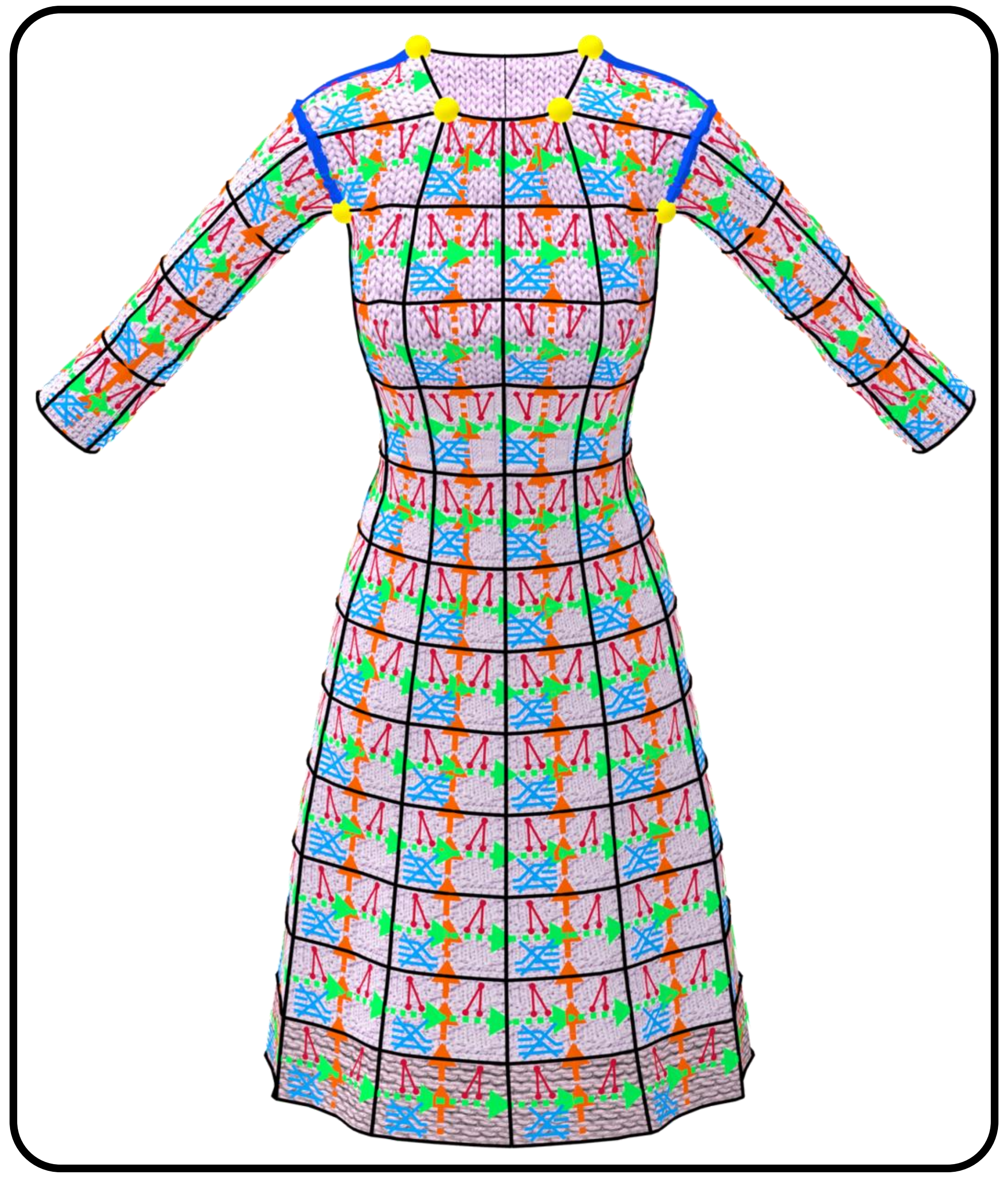
Composition

Shape Variation

Knit Templates



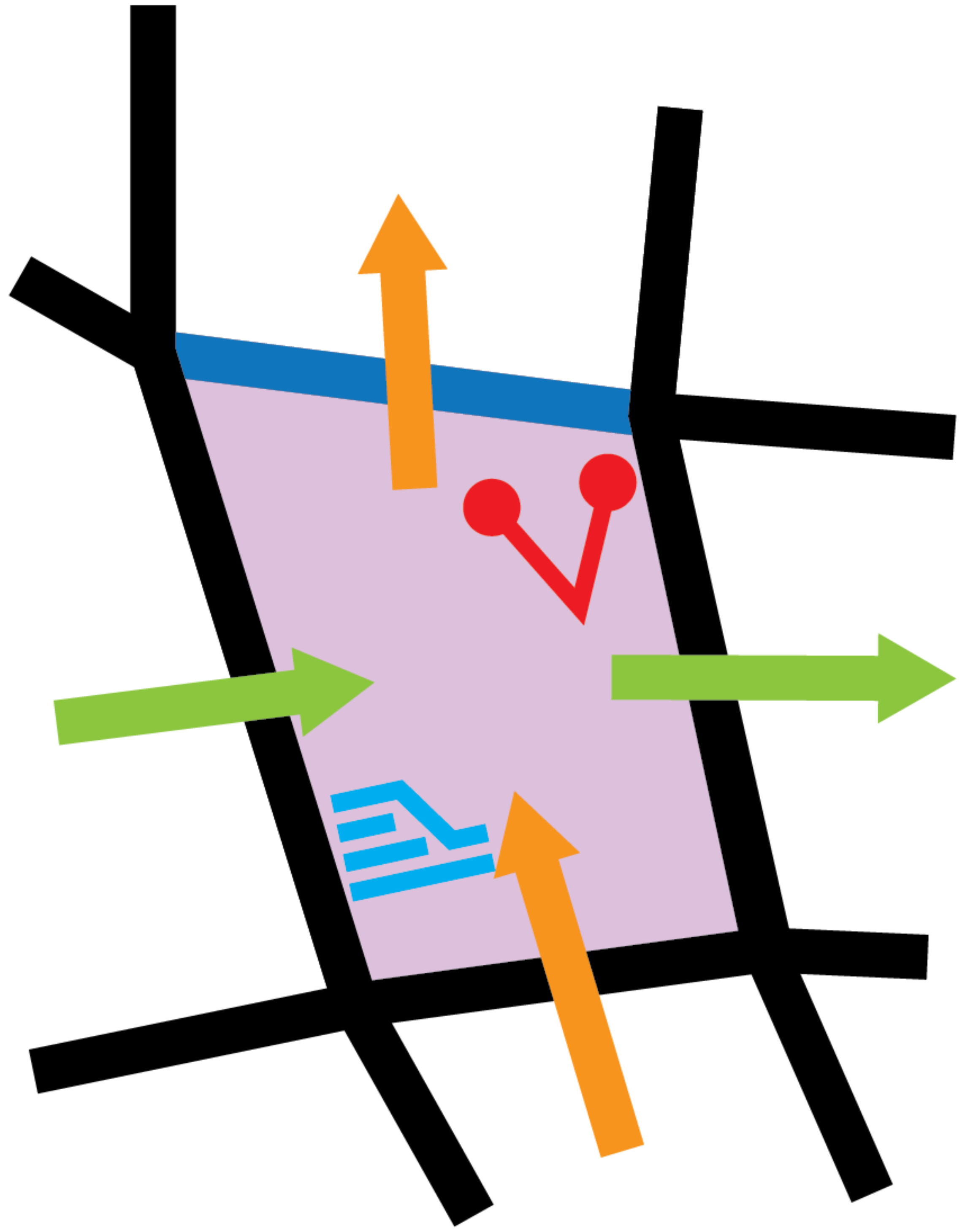
Overview

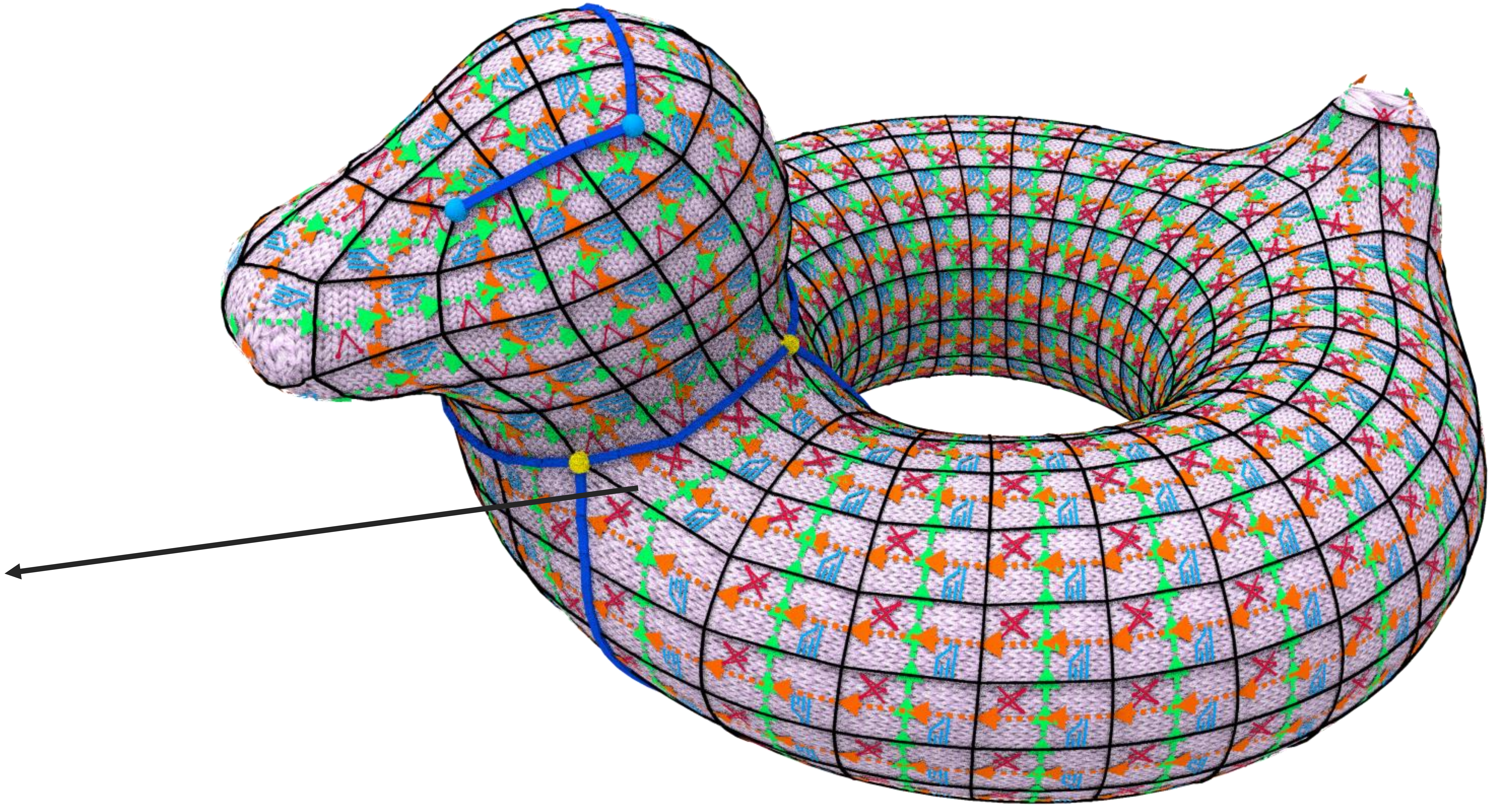


Data Structure

Design Tool

Coarse Knit Meshes

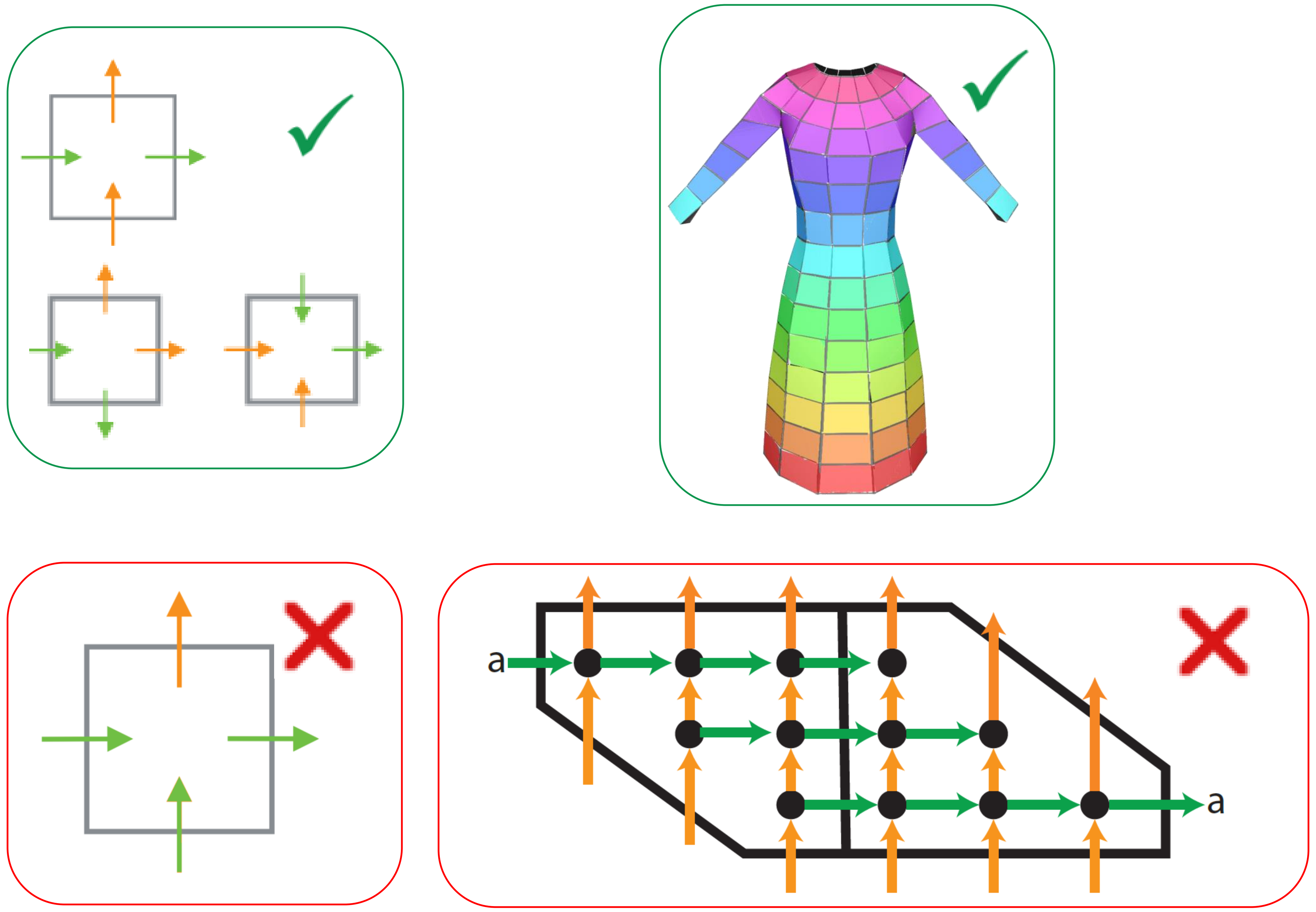




Why Quads?

Why Patches?

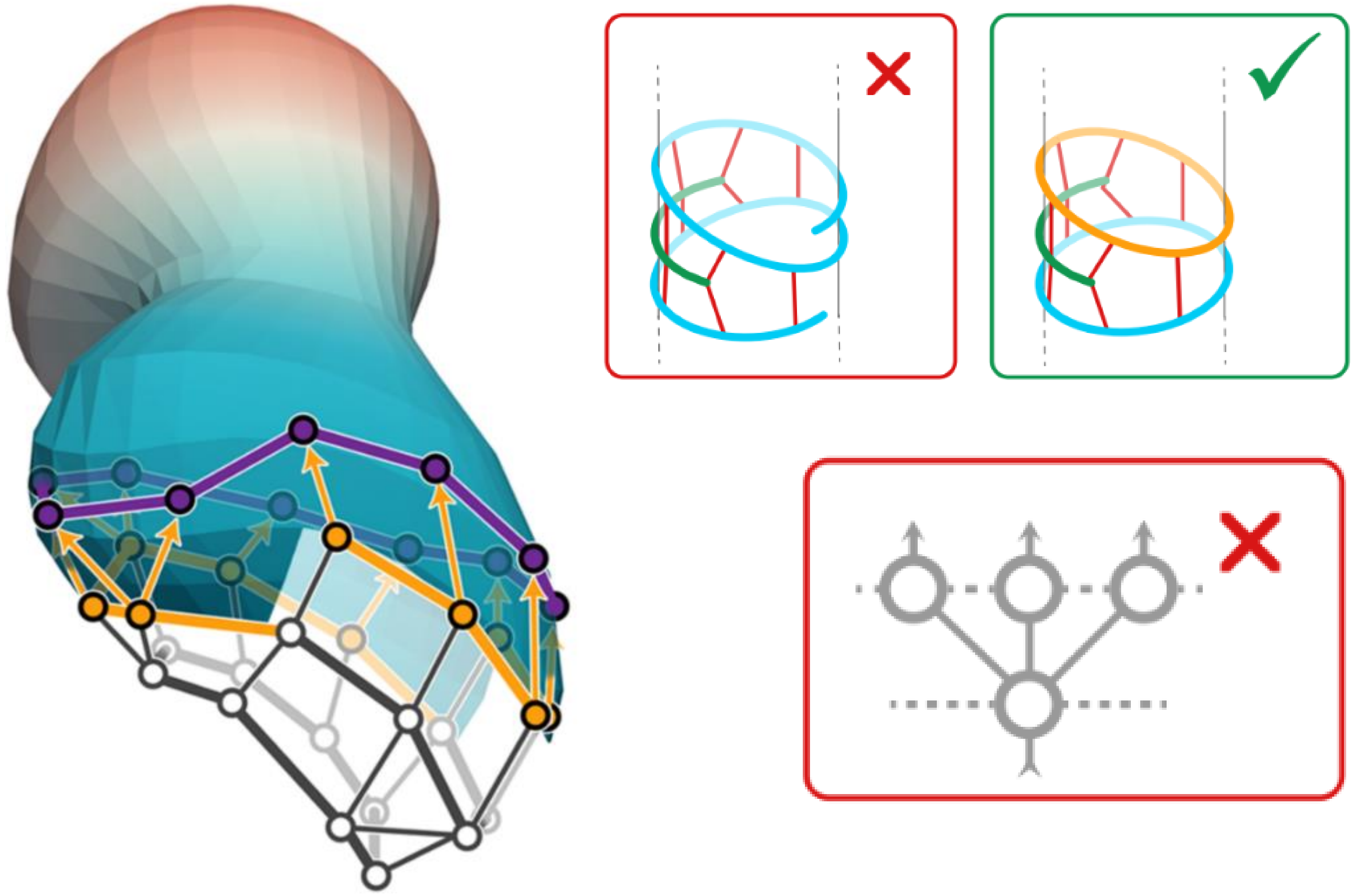
Insight 1: Why Patches?



[Our Work]

High-Level Constraints

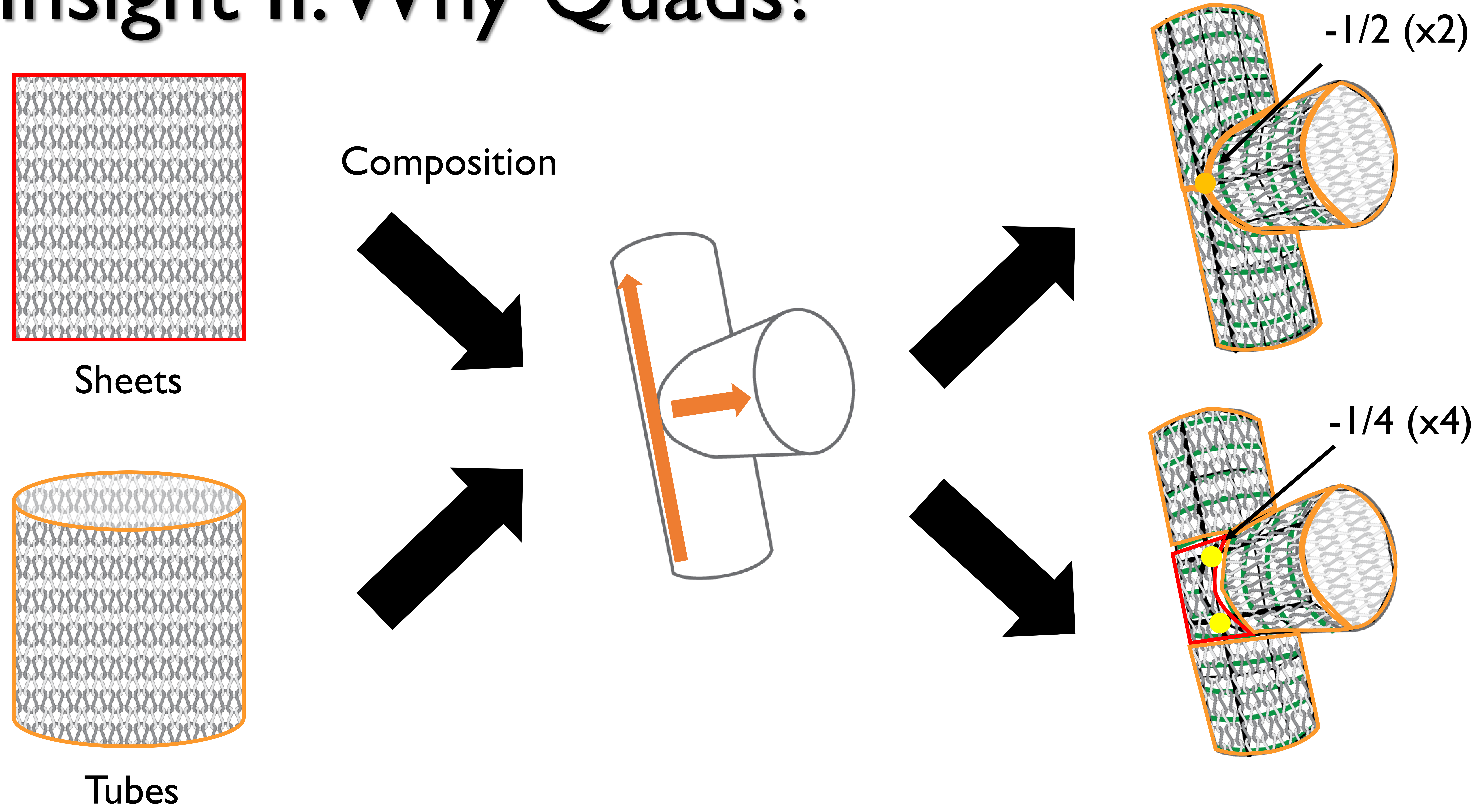
Proof



[Narayanan et al. 2018, 2019]

Low-Level Constraints

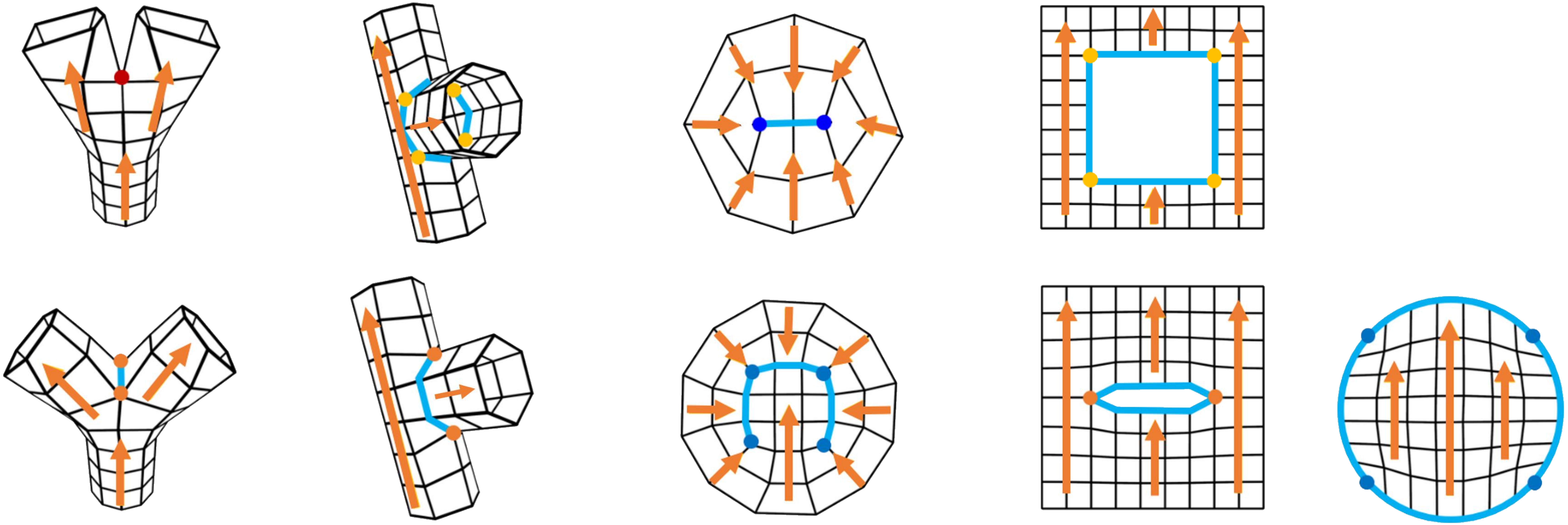
Insight II: Why Quads?



Composition Rules

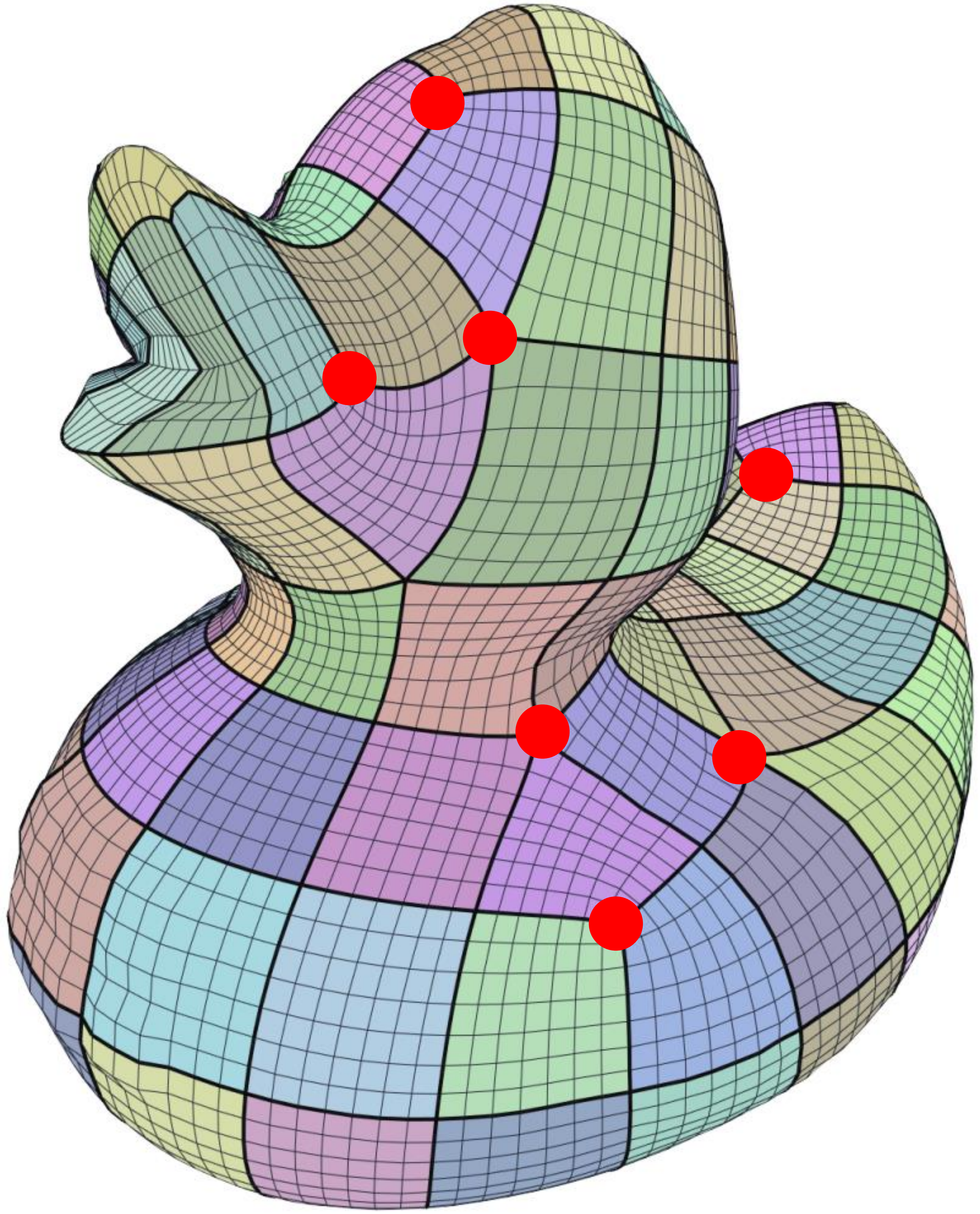
Images: emmajane

Composition Rules

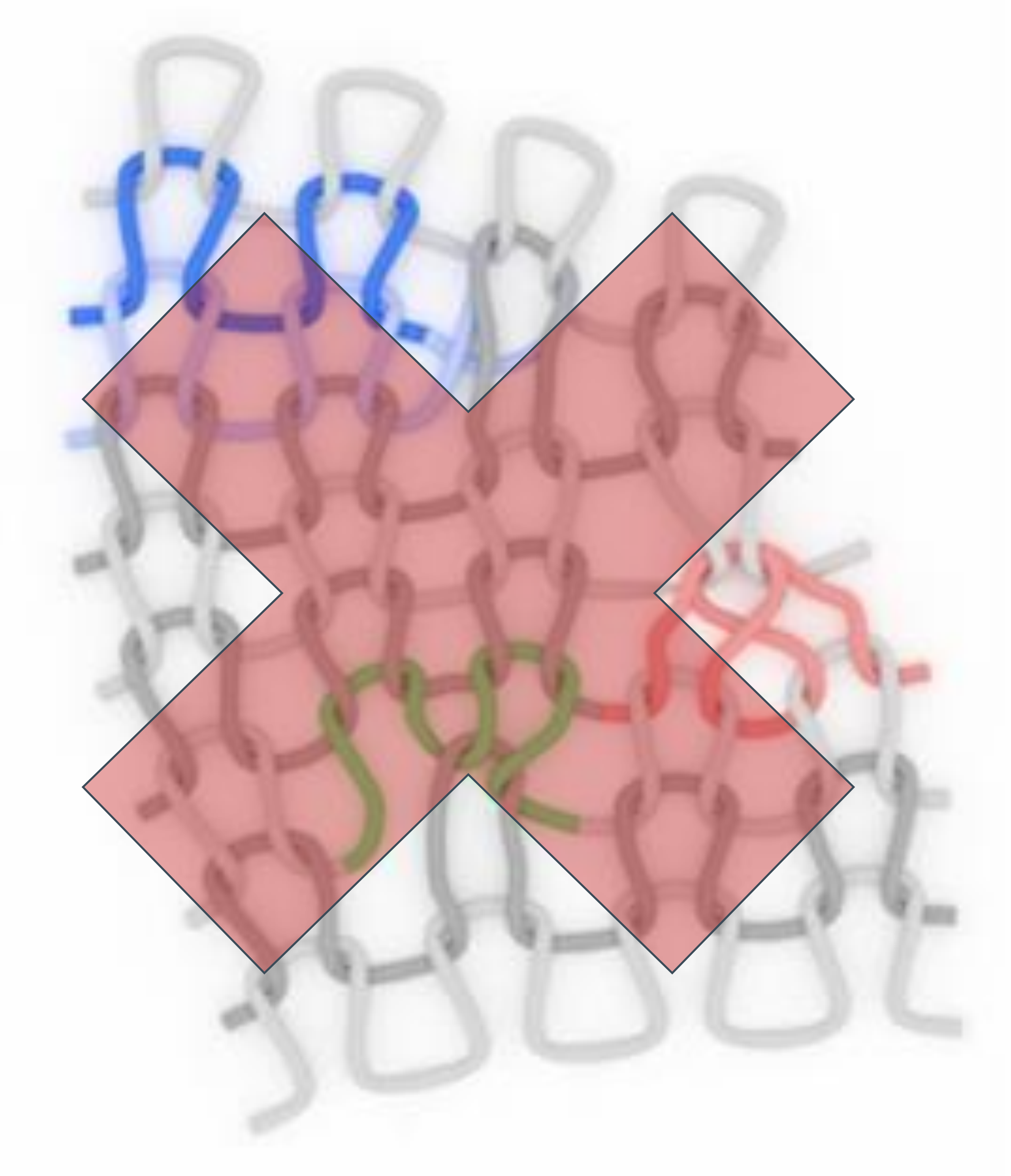


Splitting / Merging   Closing   Opening   Patch

A Note About Singularities



Fabric Level

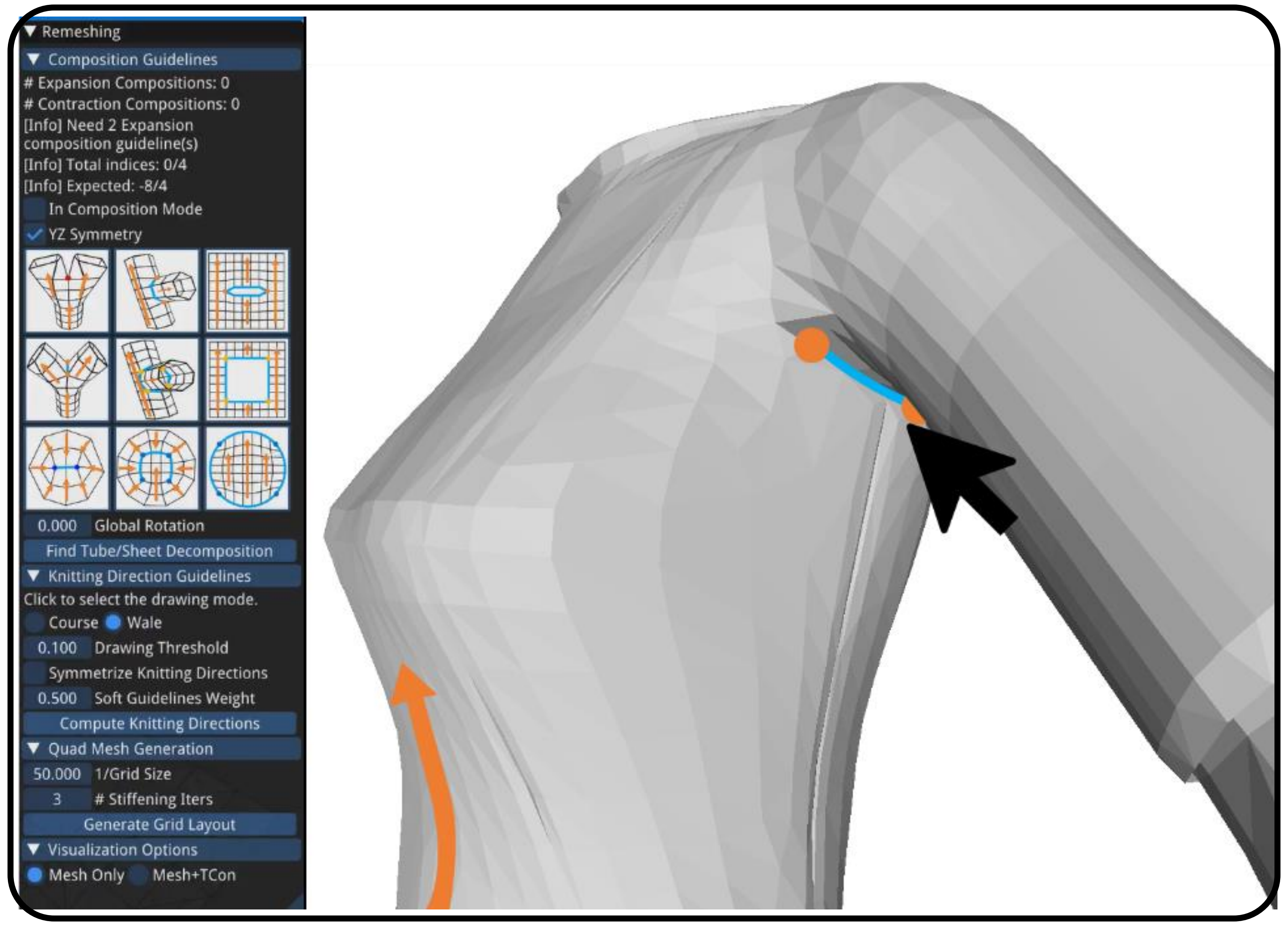


Stitch Level

Lyon et al 2021

Overview

Data Structure



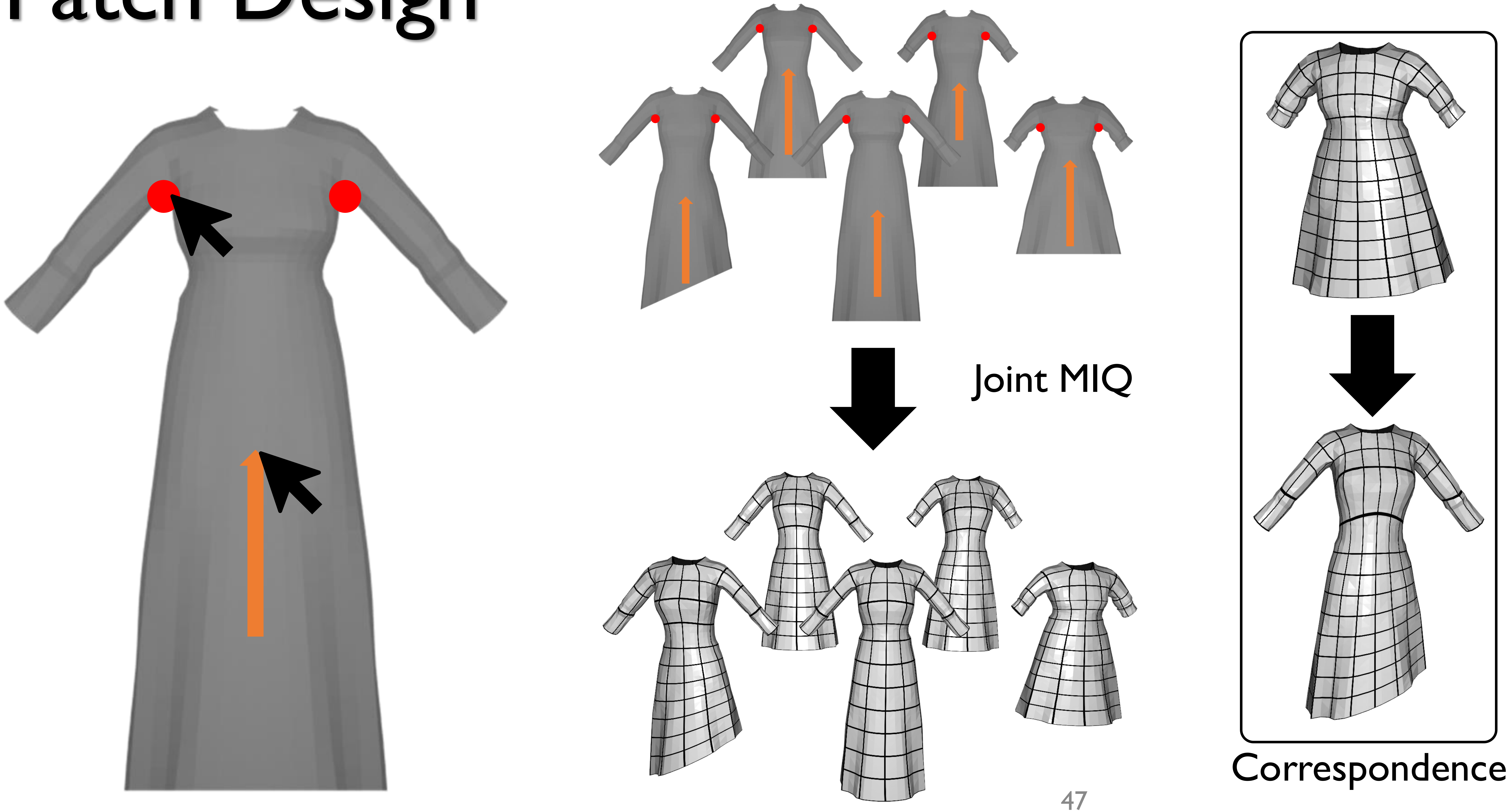
Design Tool

Design Tool Overview

Consistency

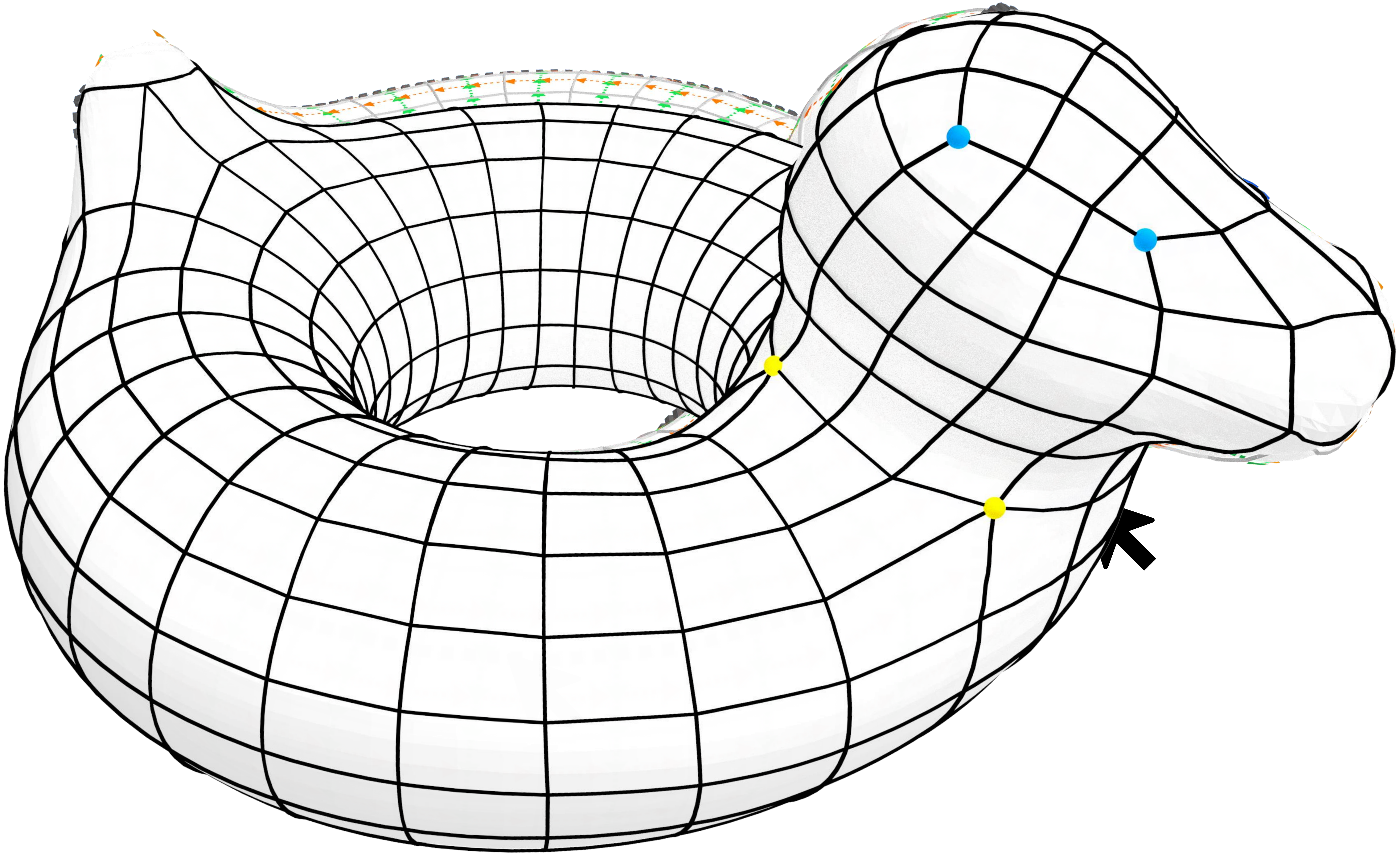
Correctness

Patch Design

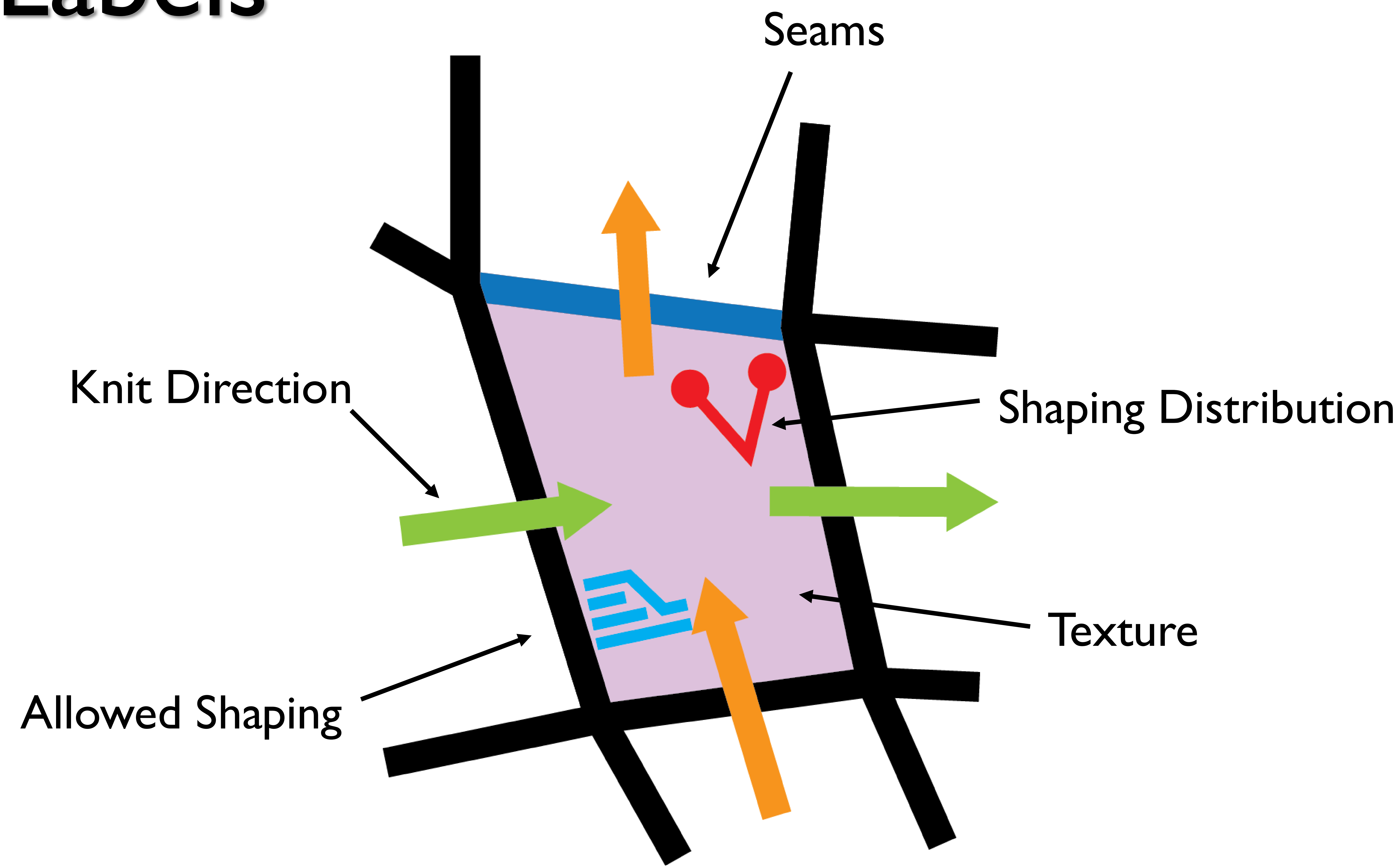


Design Tool Overview

Patch Labeling



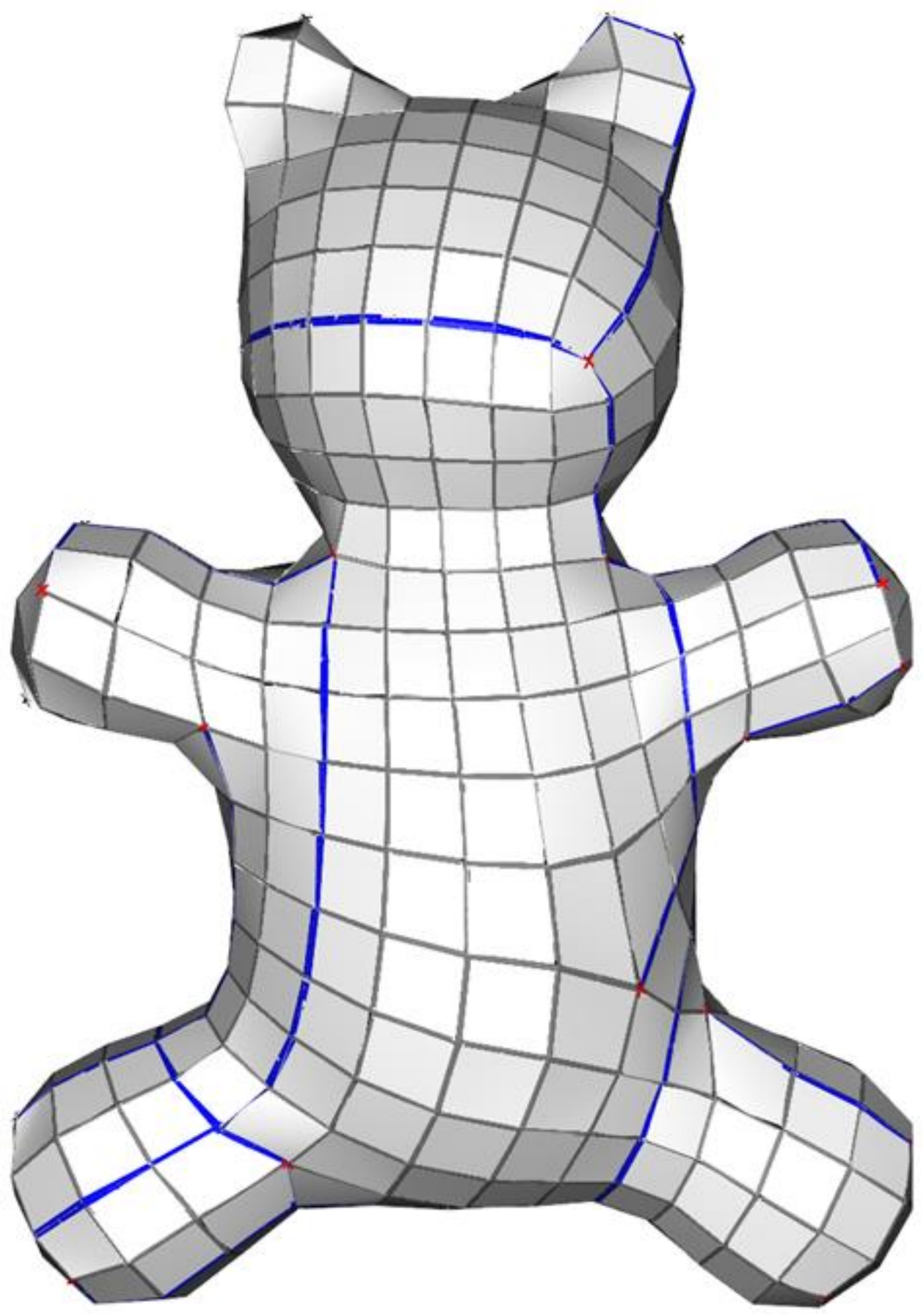
Patch Labels



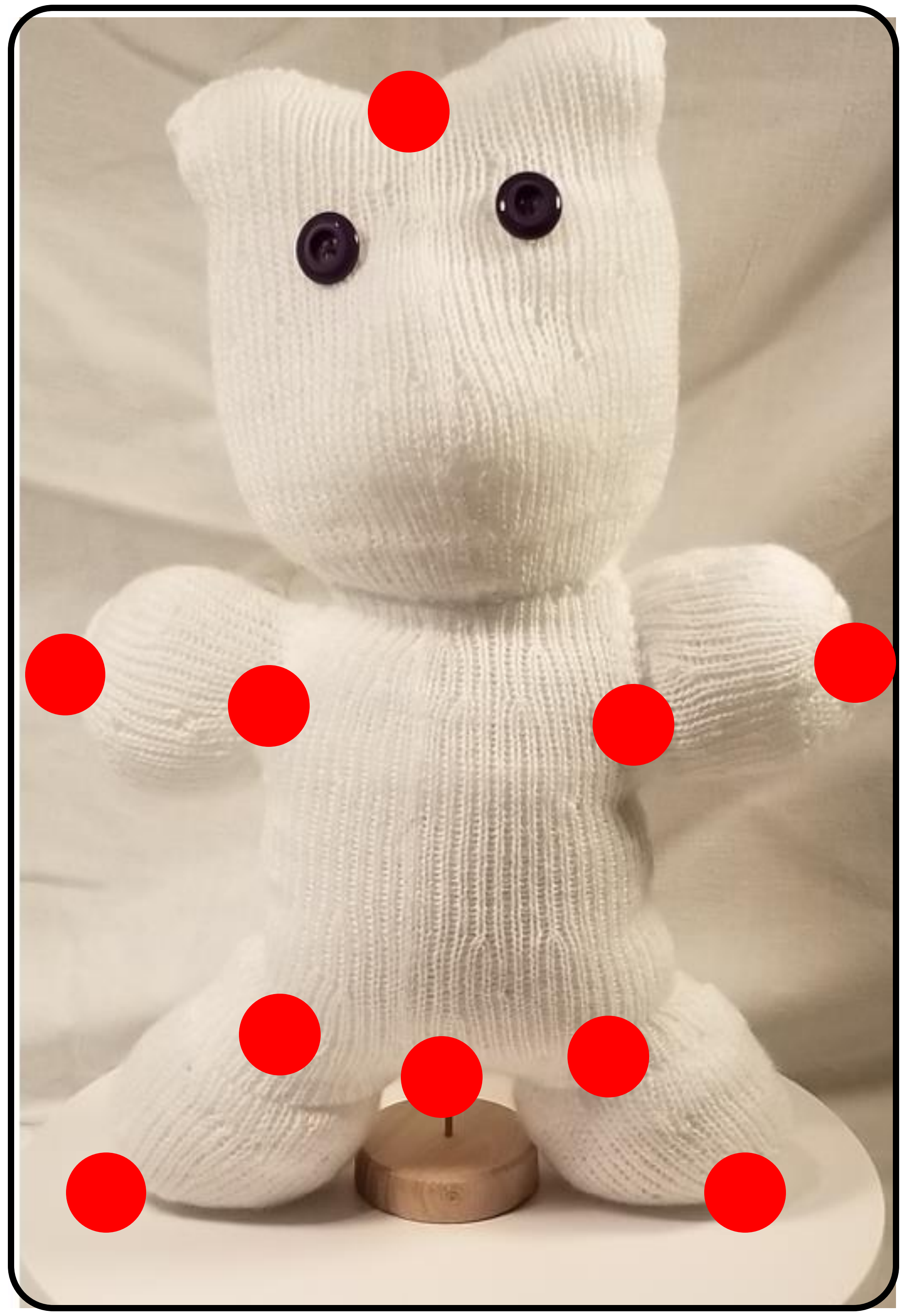
Results

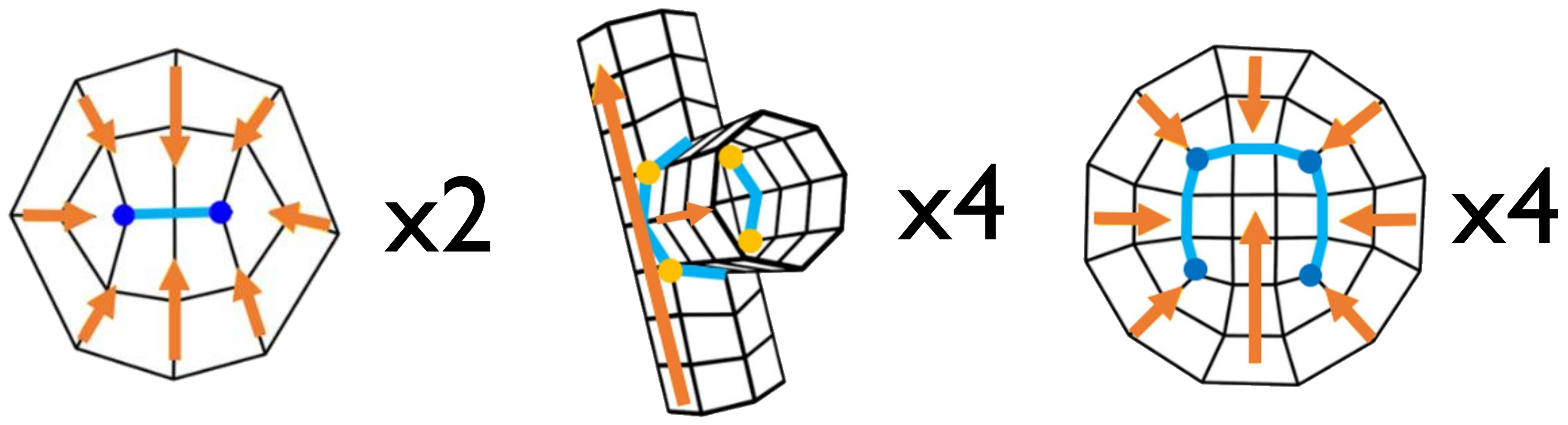
Results

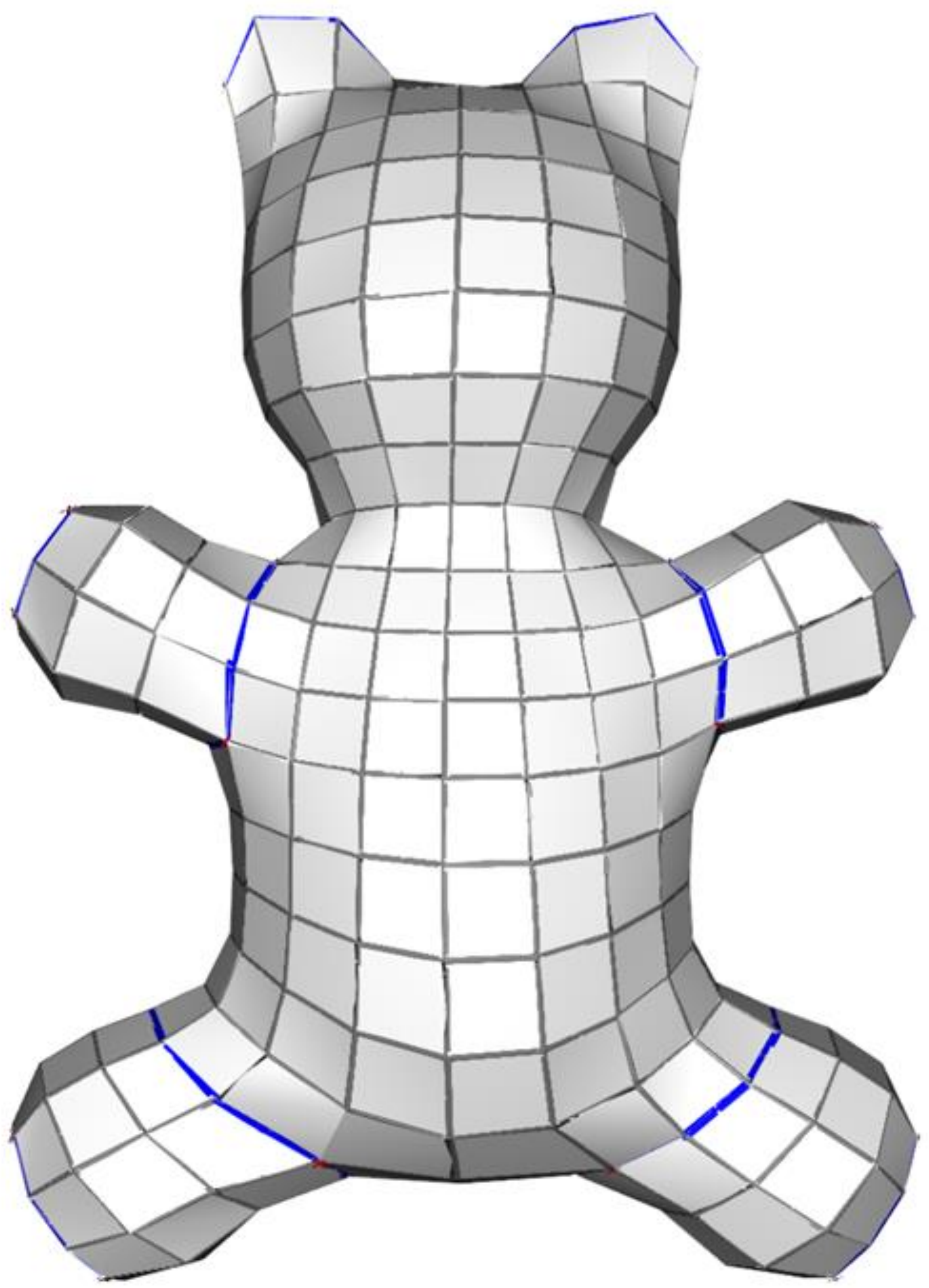
Composition Rules



No Rules



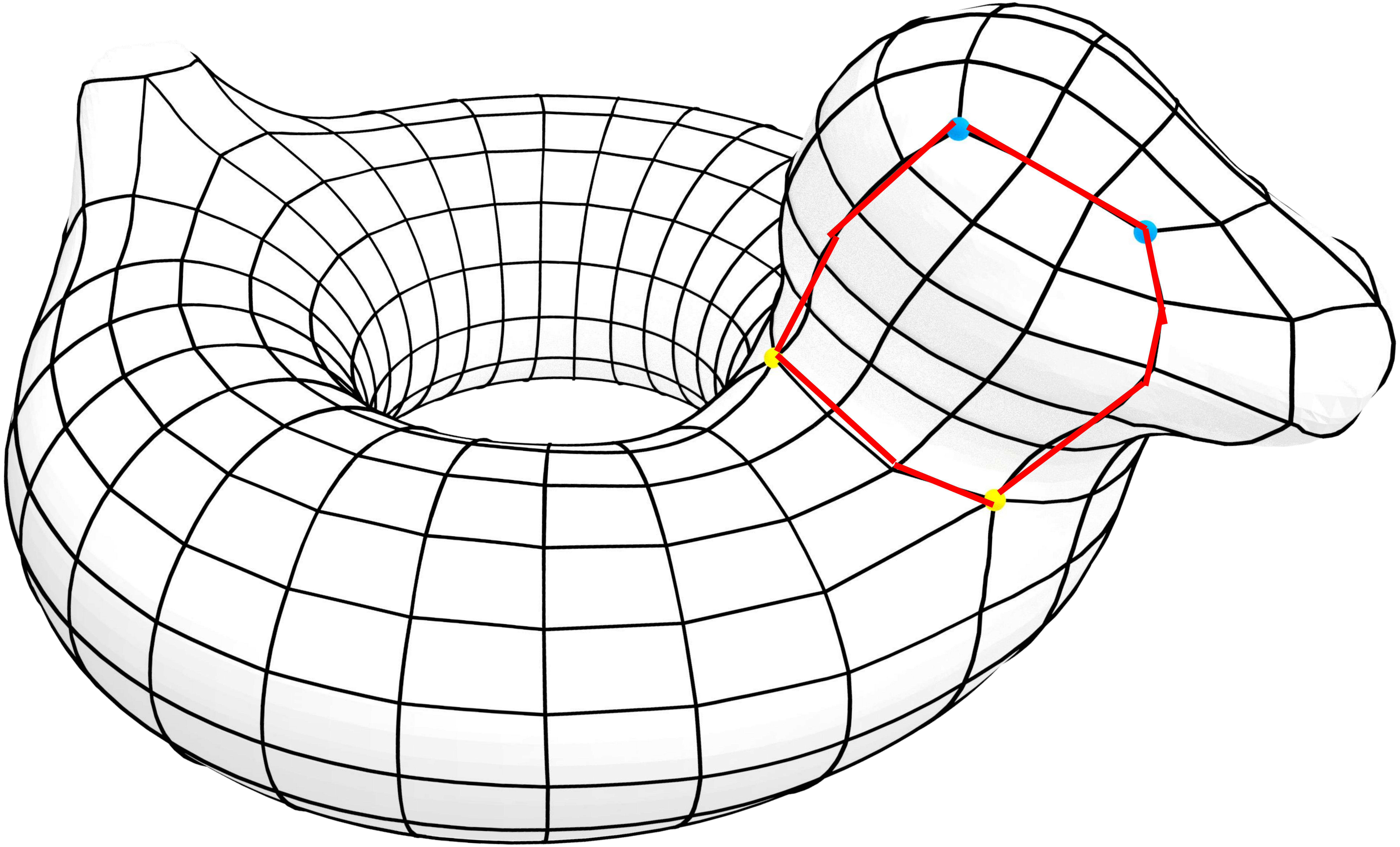




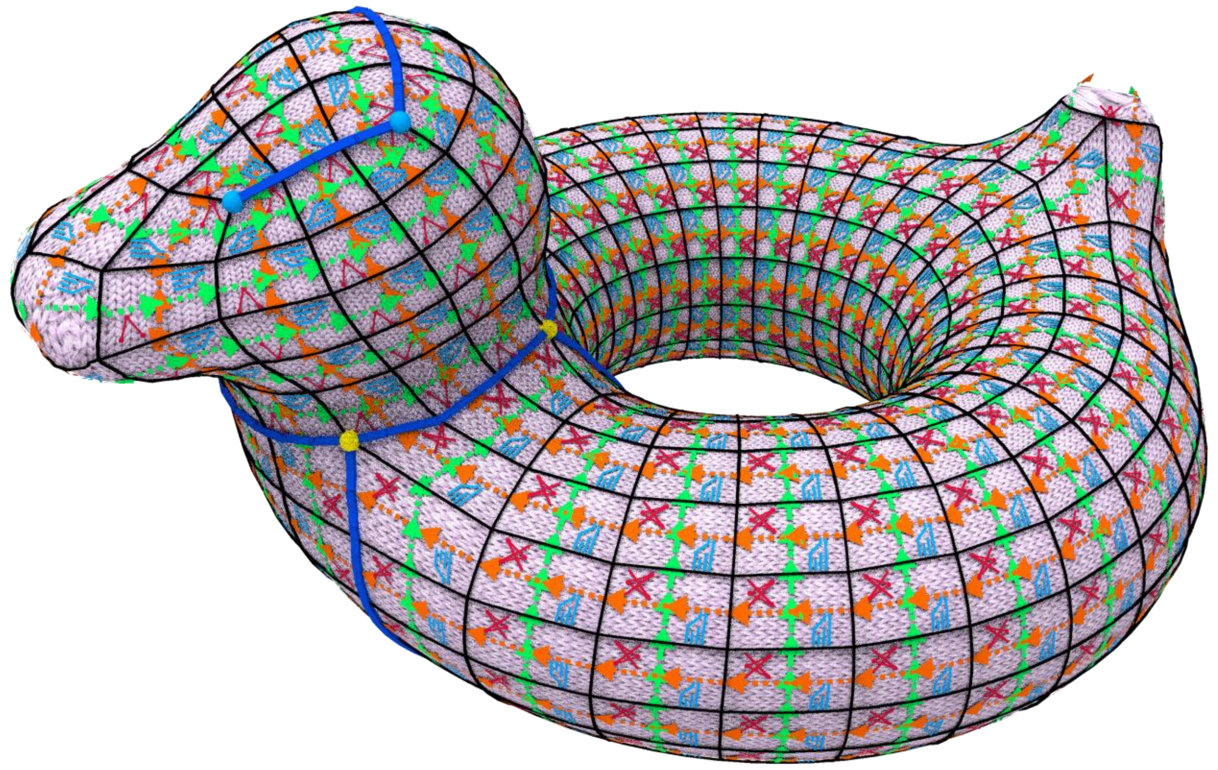
Composition Rules

Shape Variation

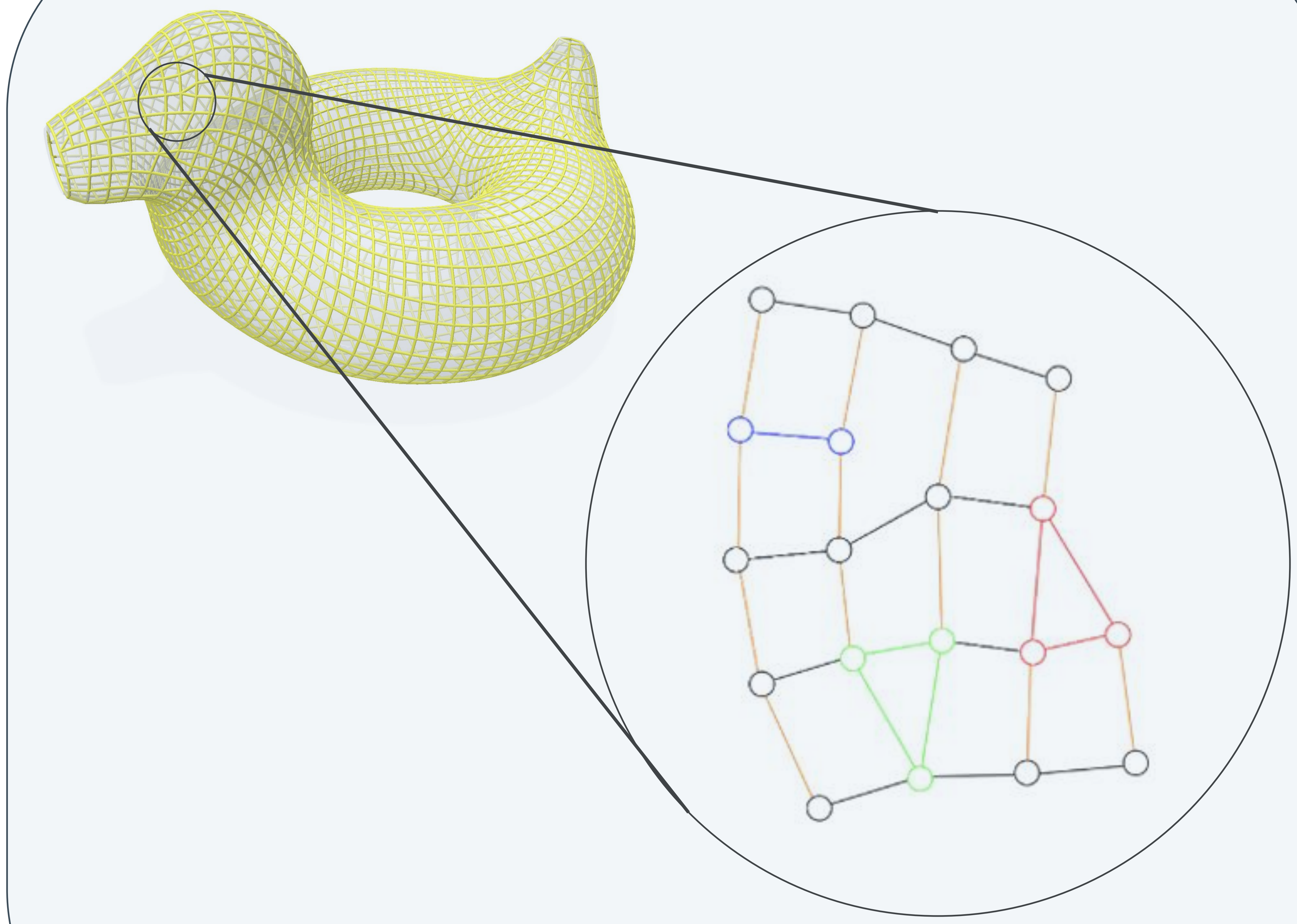
Future: Better Tesselations, Bigger Patches



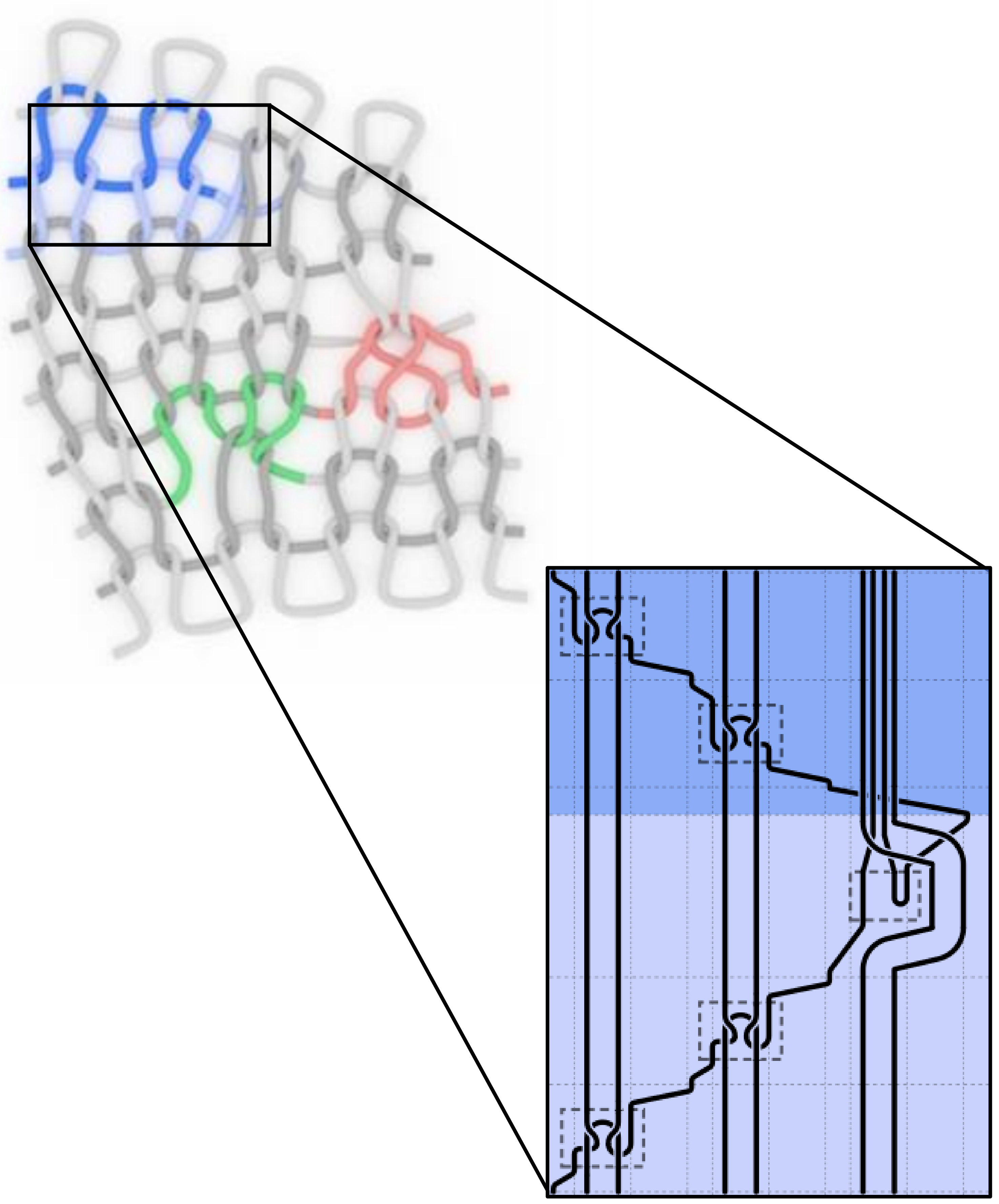
Knitting Levels of Abstraction



Fabric



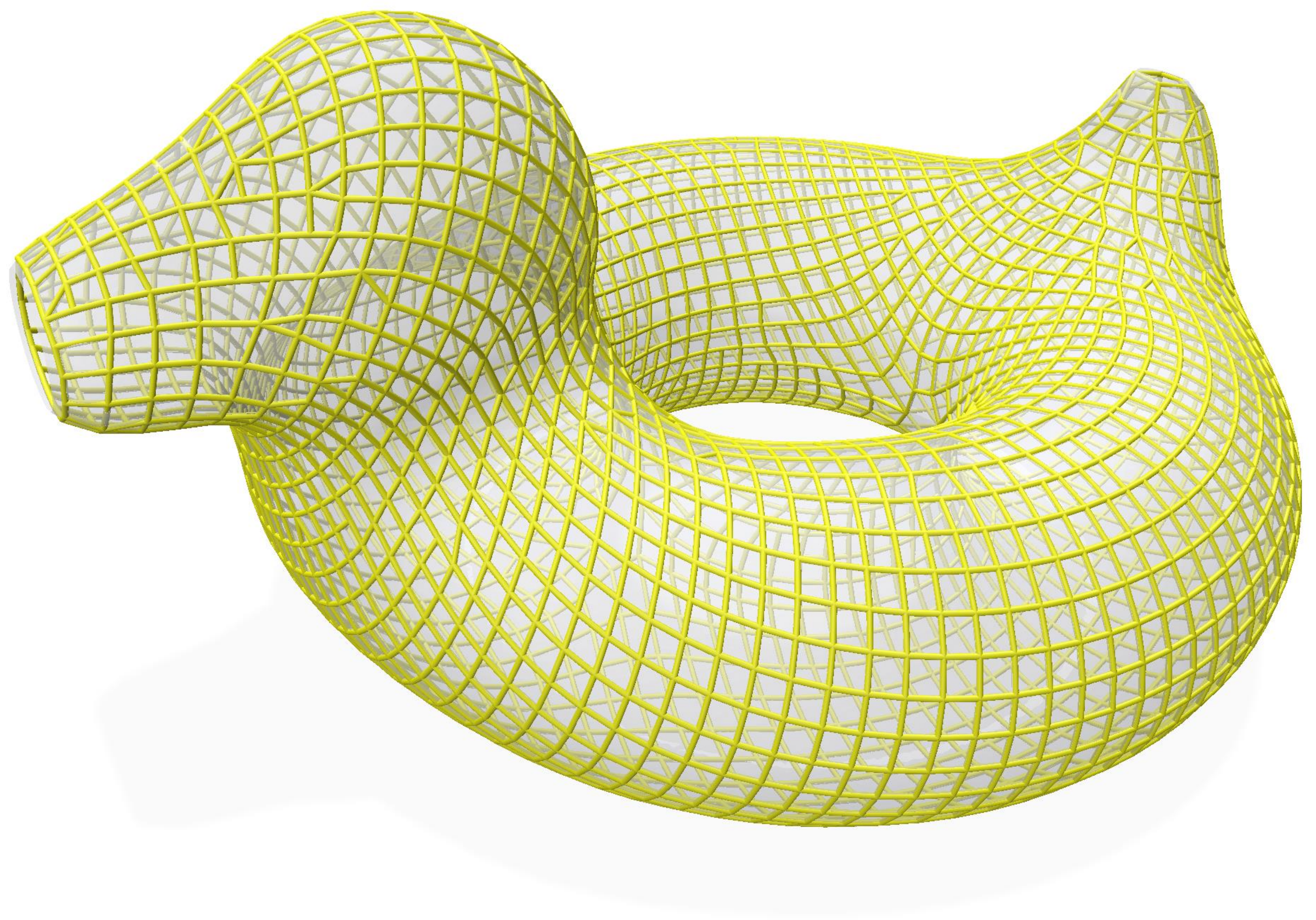
Stitch

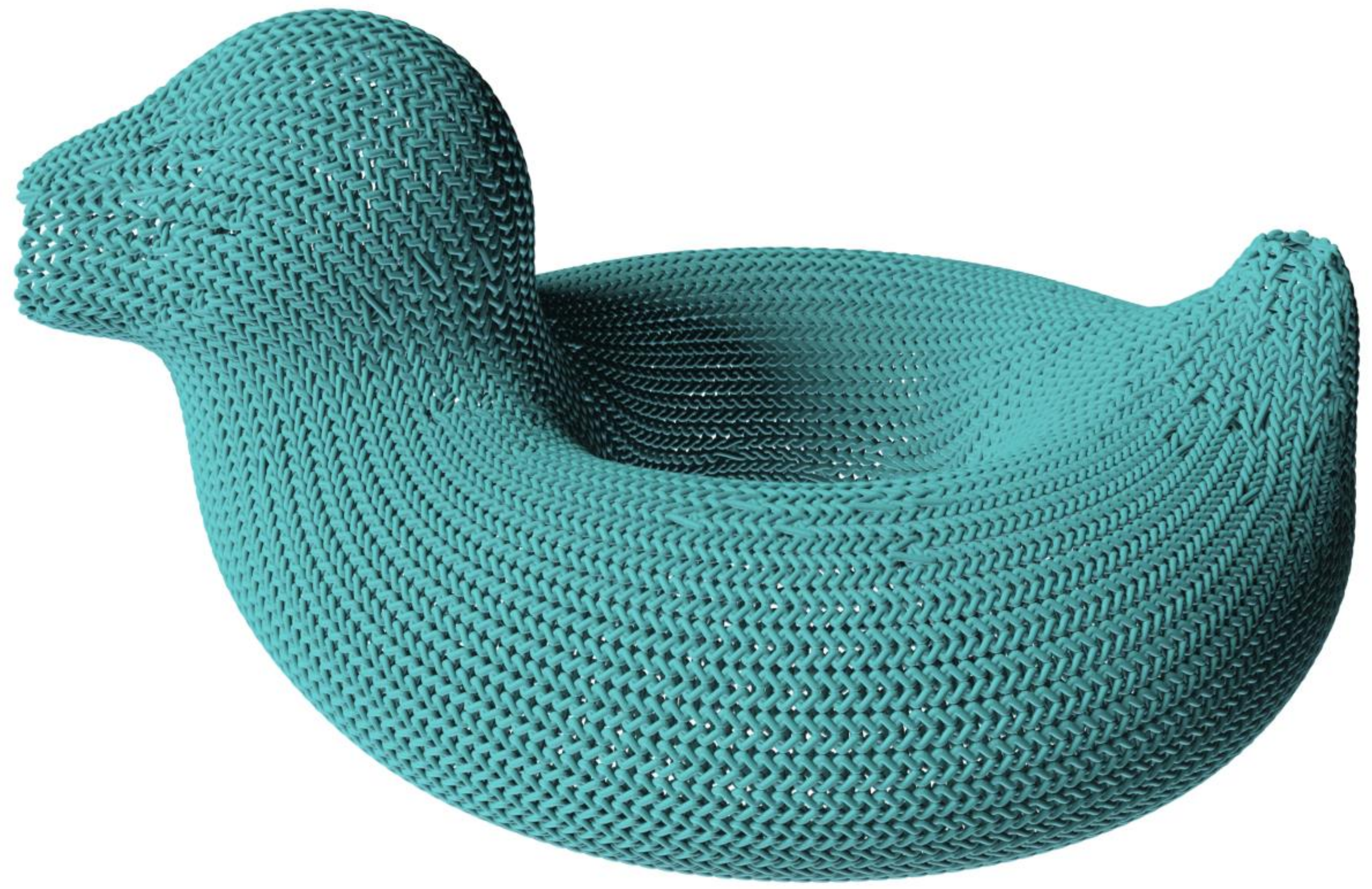


Yarn

Stitch-level abstraction

- Characterized by knit graphs or quad-dominant stitch meshes
- Nodes correspond to individual stitches (roughly)



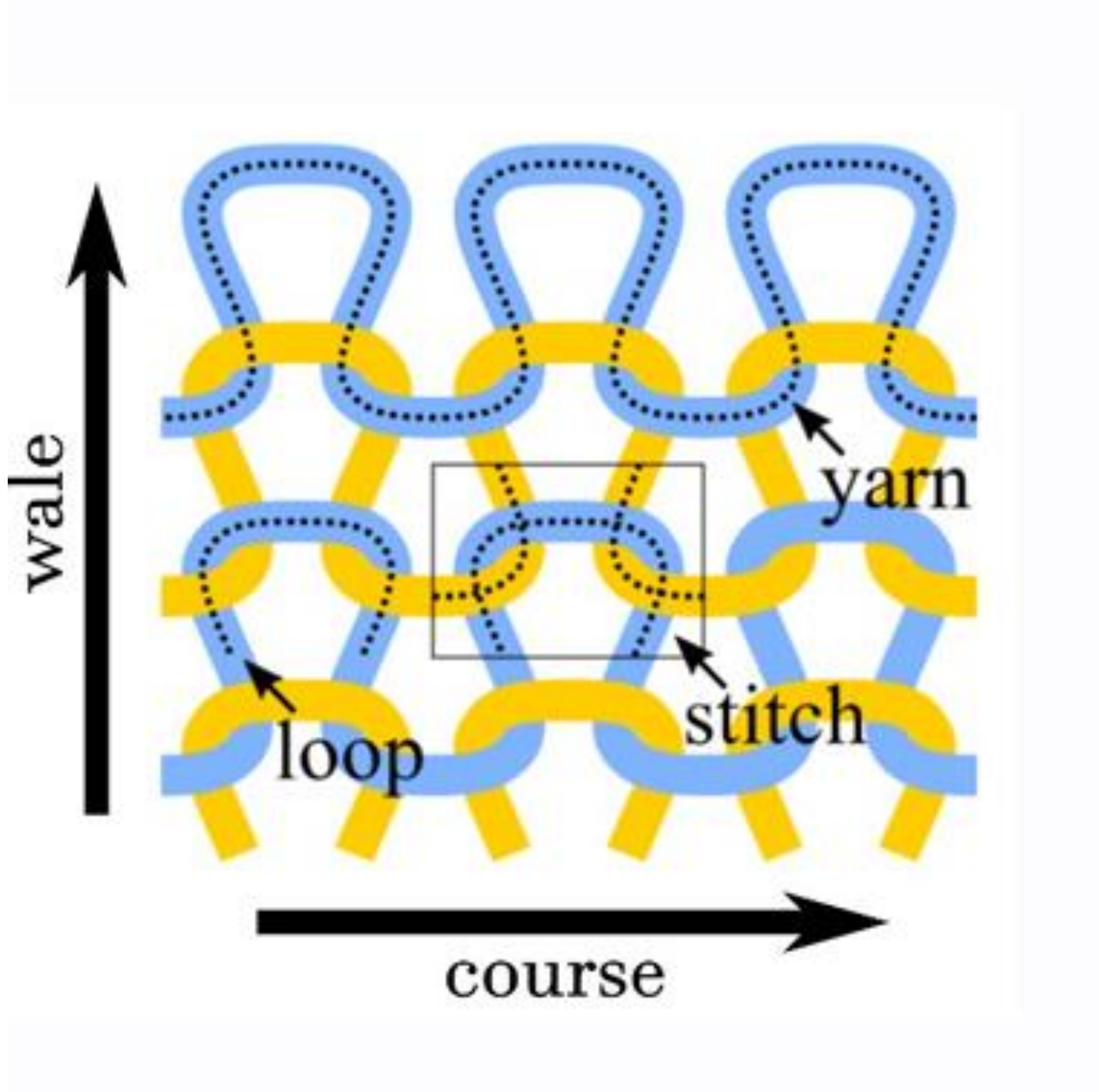


- Aim is to achieve **consistent element size** and **geometric fidelity**

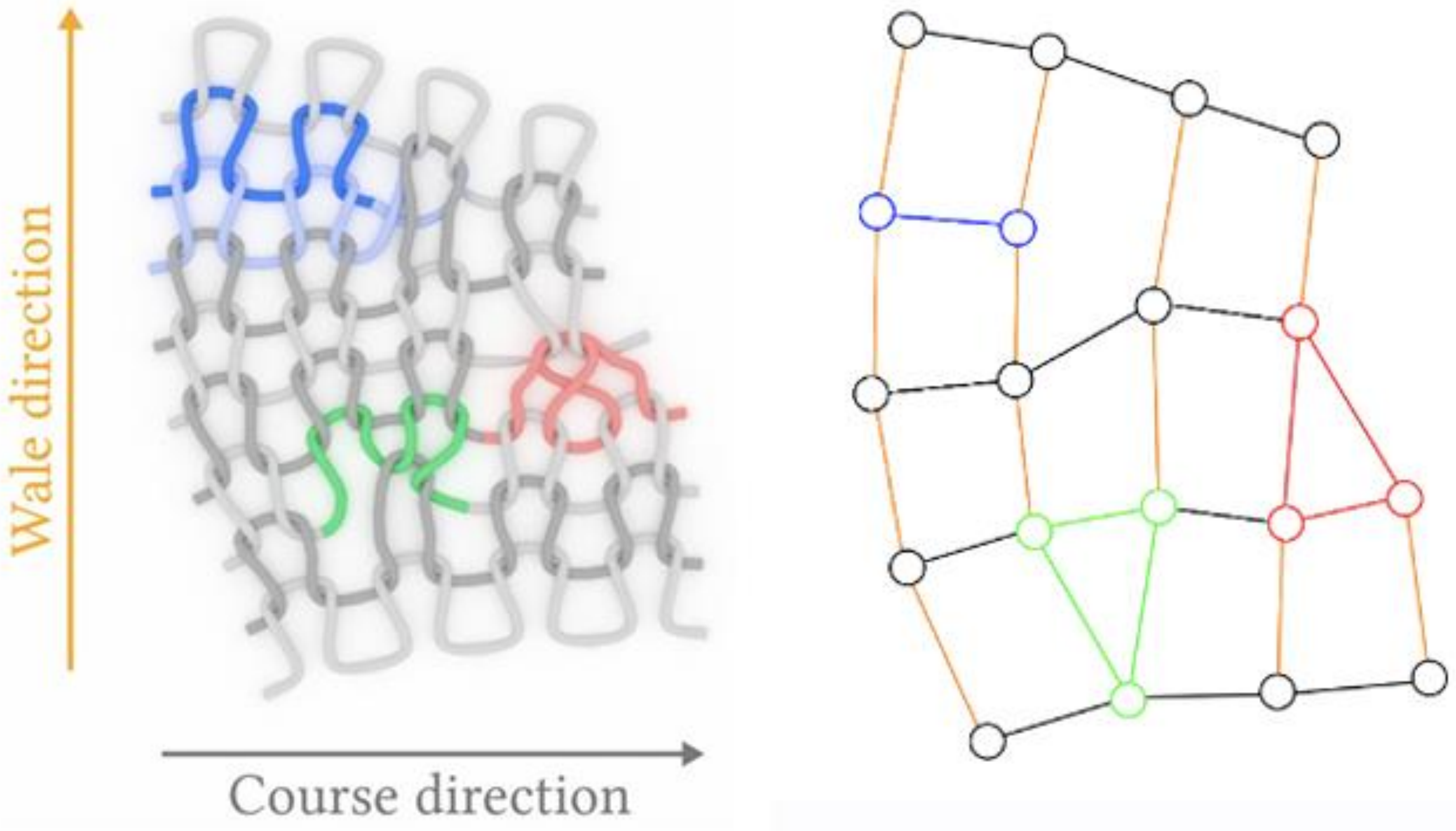
Images: Curl Quantization for Automatic Placement of Knit Singularities, Mitra et al.

Stitch-level abstraction

- Non-quad elements arise to denote shaping operations



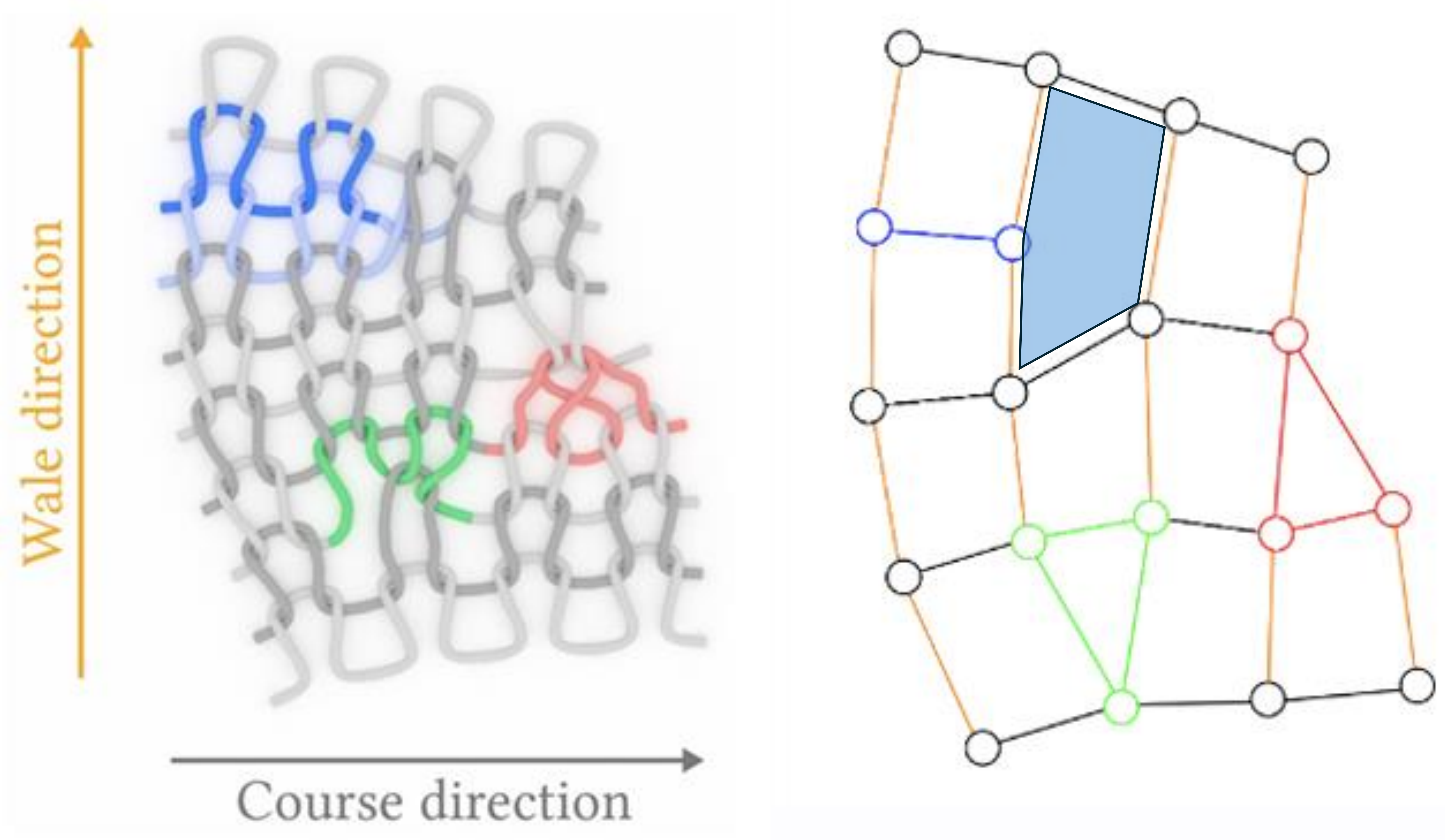
Flat geometry[1]



Stitch irregularities induce curvature[2]

[1] *from Visual Knitting Machine Programming (2019)*
[2] *from Knit Sketching: from Cut and Sew Patterns to Machine-Knit Garments (2021)*

Shaping via stitch irregularities/singularities

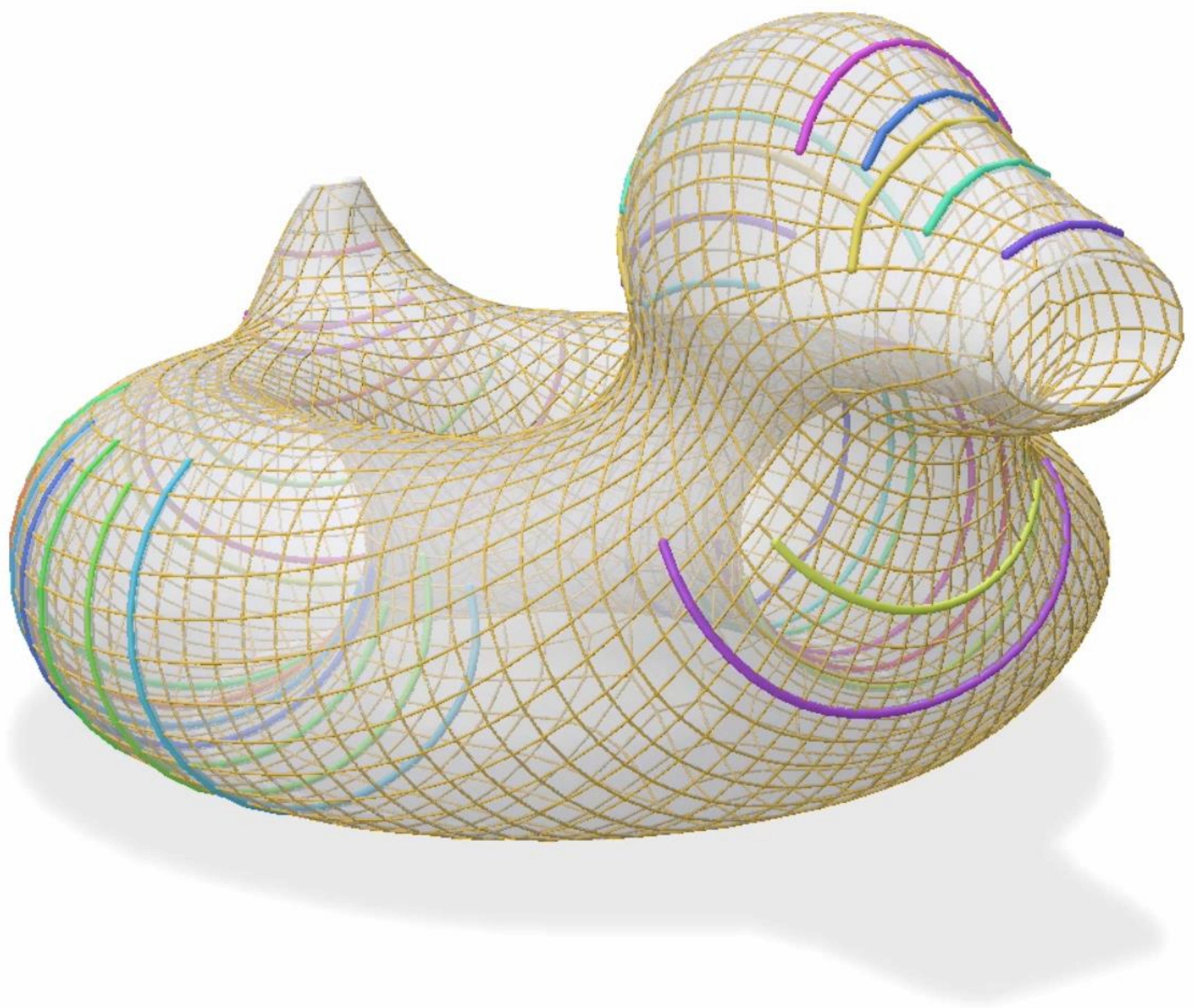


Stitch irregularities induce curvature[2]

- *Knit graph* of Autoknit has a node for every pair of stacked stitches, and edges of course and wale type
- *Singularities* correspond to non-quad faces in the mesh
  - Increase
  - Decrease
  - Short Row Ends

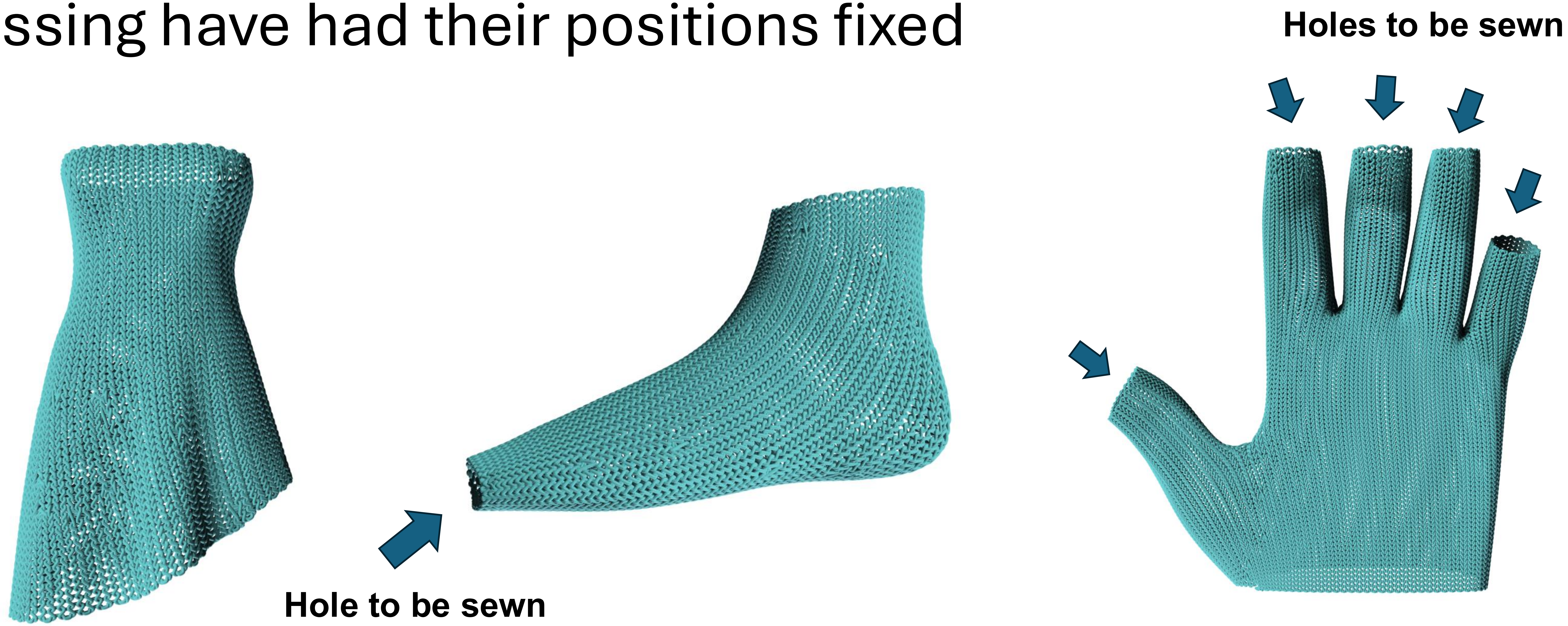
[2] *from Knit Sketching: from Cut and Sew Patterns to Machine-Knit Garments (2021)*

Short row shaping example



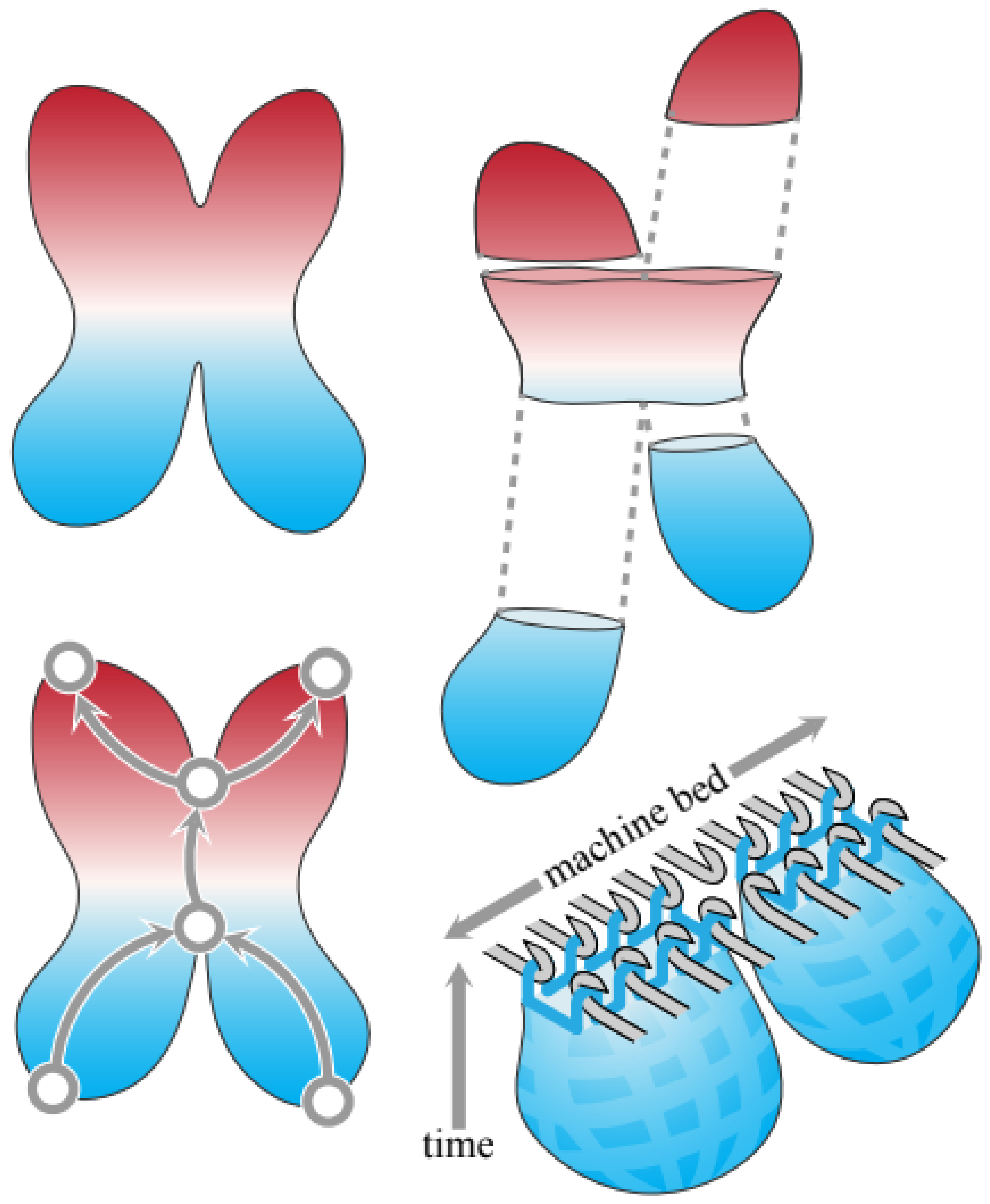
“Whole-garment” setting

- Aim is a knit structure that is fully machine-knittable
- Any seams or holes that need to be sewn together in post-processing have had their positions fixed



Images: [Curl Quantization for Automatic Placement of Knit Singularities](), Mitra et al.

Knitting time function



- The *time function* $h : S \to [0,1]$ encodes general knitting direction and dependency between course rows
- *Level sets* correspond roughly to desired course rows

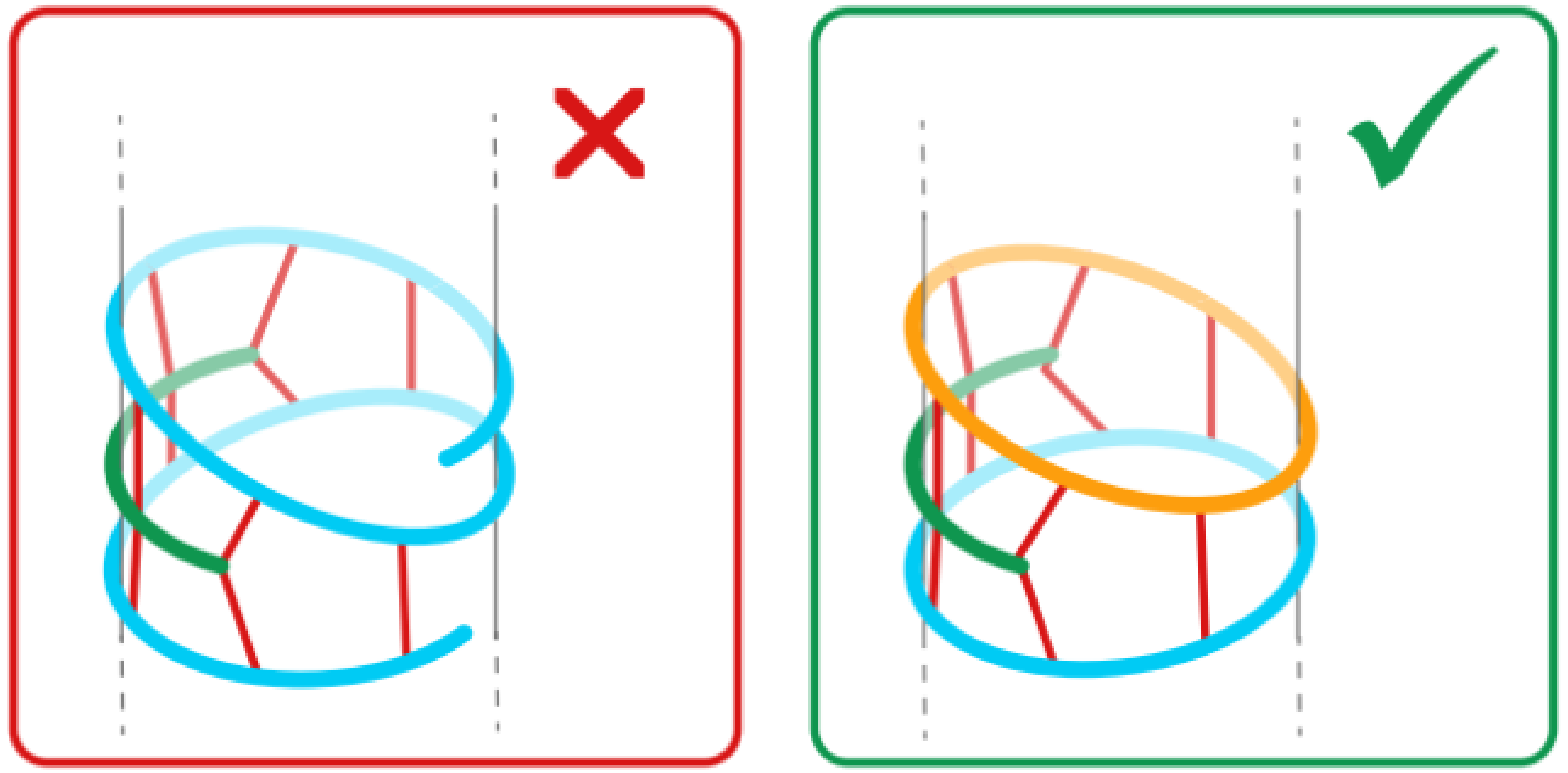
$$h^{-1}(c) = \{p \in S \mid h(p) = c\}, \quad c \in [0,1]$$

- Approximate wale direction given by the gradient ∇h
- User input or harmonic interpolation

Image: Autoknit, Narayanan et al.

Helix-free constraint (Autoknit Property 2)

- **Hard** manufacturing constraint: extracted course rows must “join up” as they come around
- Allows one to *trace* the knit graph and come up with a “self-supporting” yarn path
  - Loops need to be supported by existing loops



- Note: the actual yarn path *does helix*, but it is hard to achieve “right” amount of helicing in the knit graph generation
- Such constraints are difficult to handle in existing pipelines for quad-dominant meshing and/or stripe texturing

Image: Autoknit, Narayanan et al.

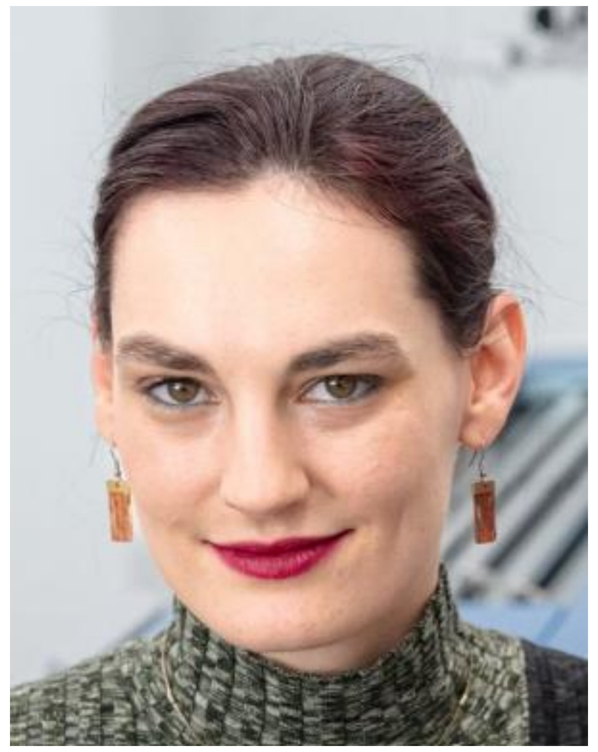
Foliations & Stripes-Based Approaches

- A line of works first-authored by my PhD student Rahul Mitra, and done in collaboration with many others


Matteo Couplet


Liane Makatura


Erick Jimenez Berumen


Megan Hofmann


Kui Wu


Emily Whiting


Rahul Mitra

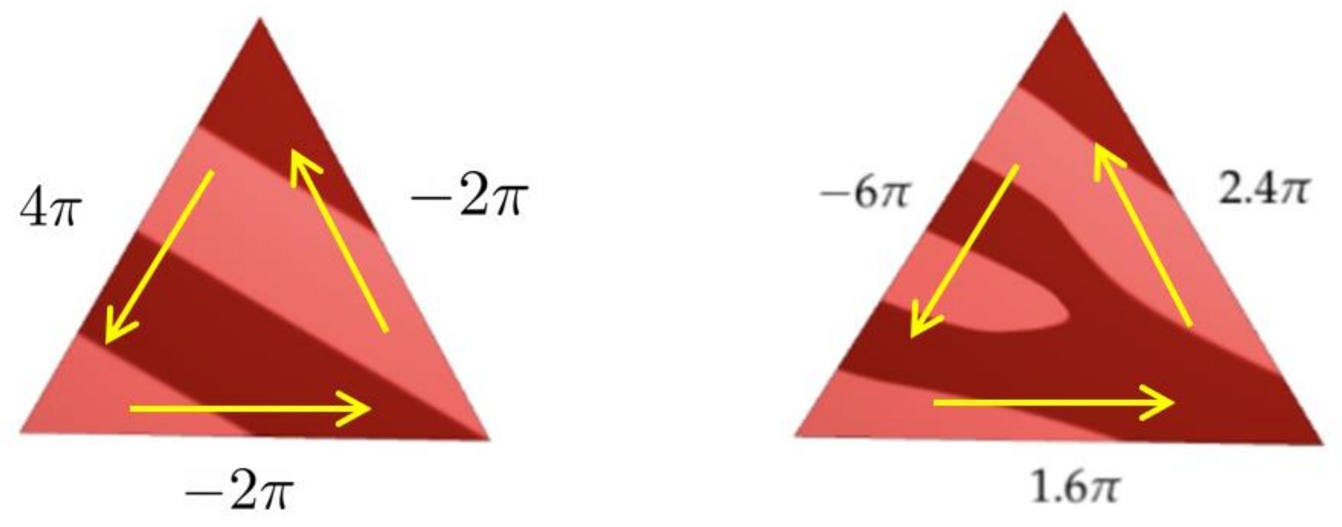
BOSTON UNIVERSITY

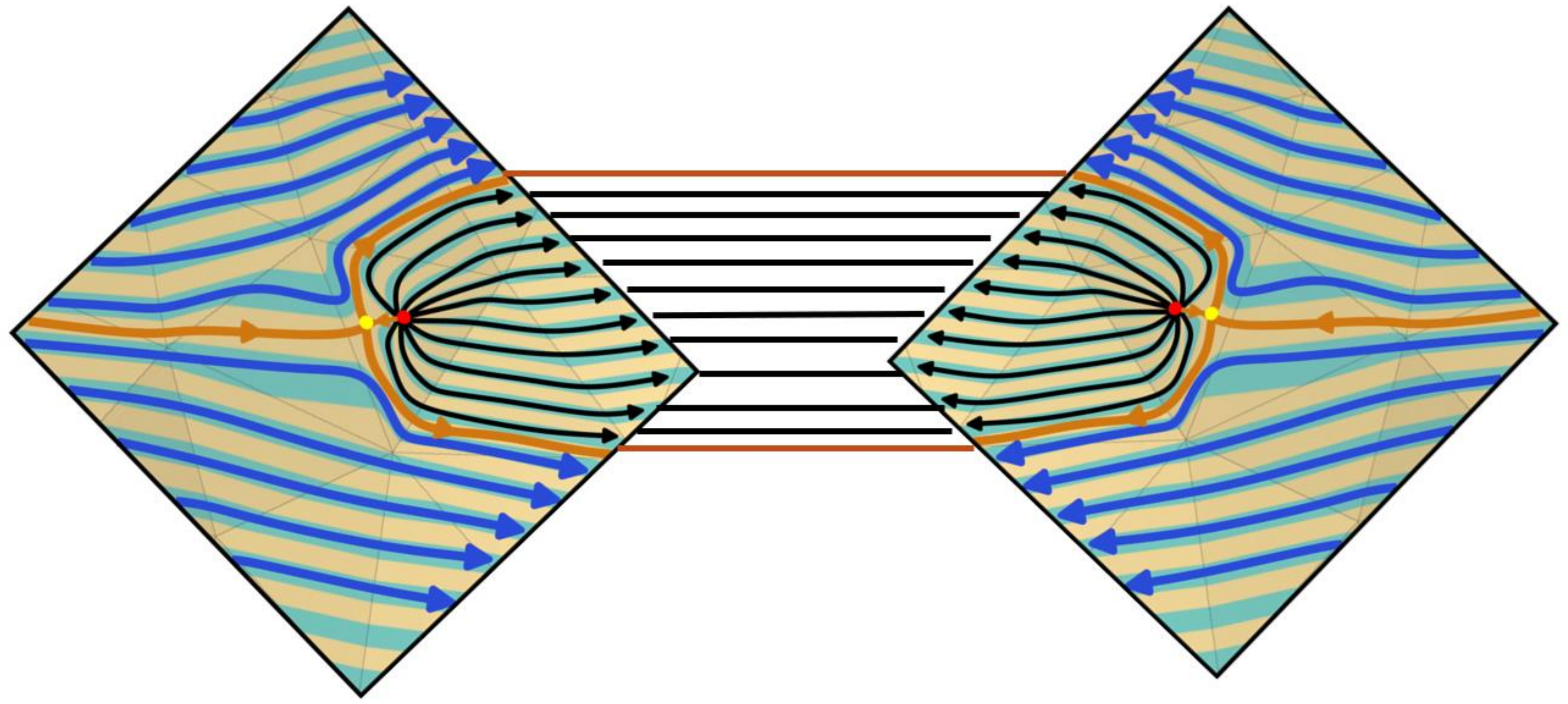
LVX VERITAS VIRTVS Northeastern University

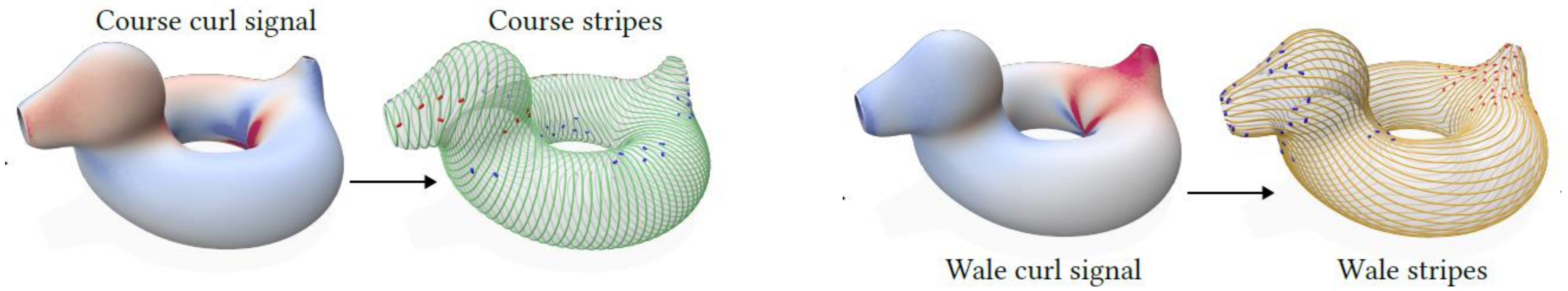
LIGHTSPEED STUDIOS

Three Main Concepts

1. Spinning Forms for Stripes
2. Global Foliation Structure: Orbit Complex
3. Stripe Singularity Placement via Curl Quantization

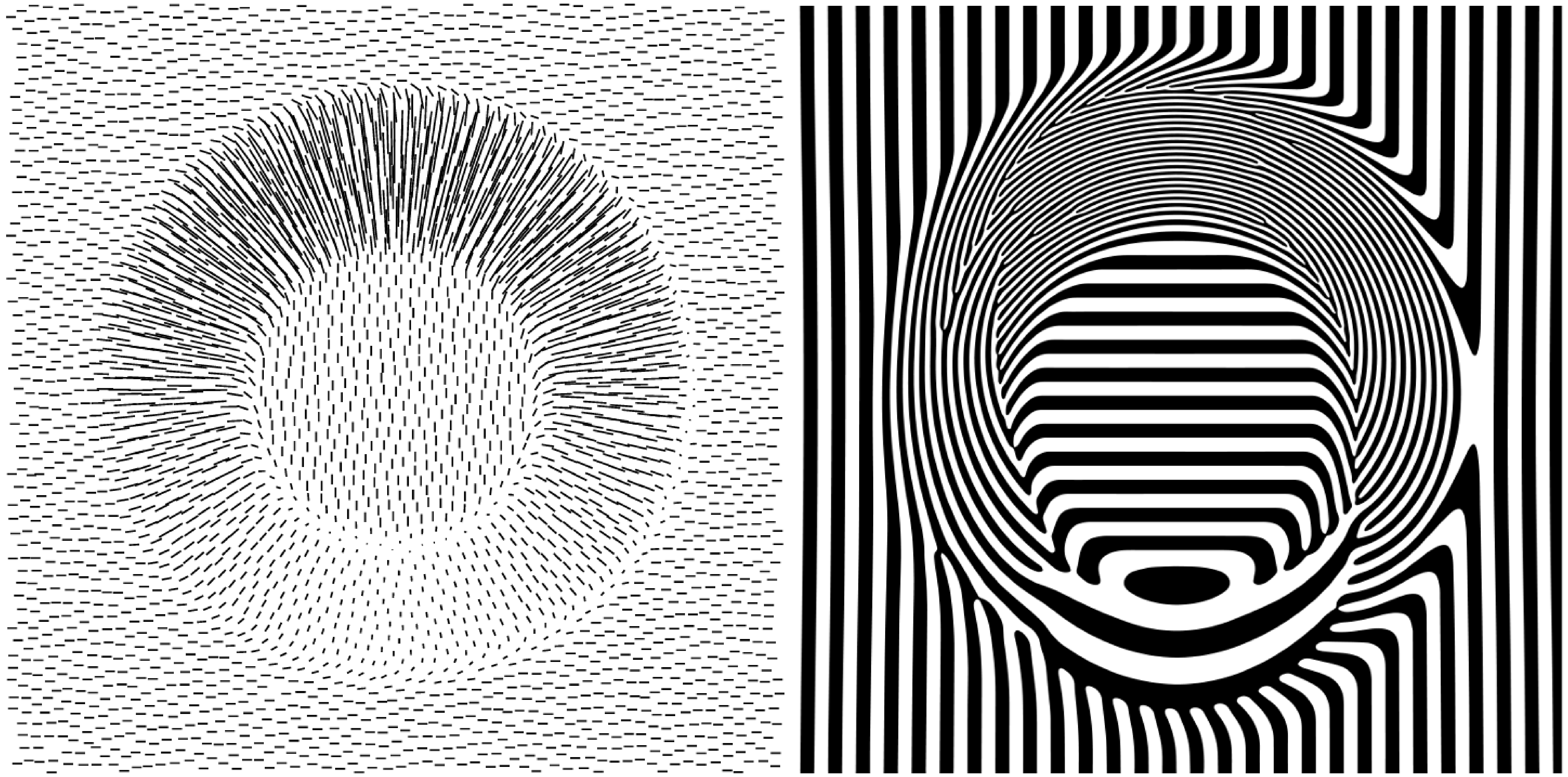






Stripe Textures on Surfaces

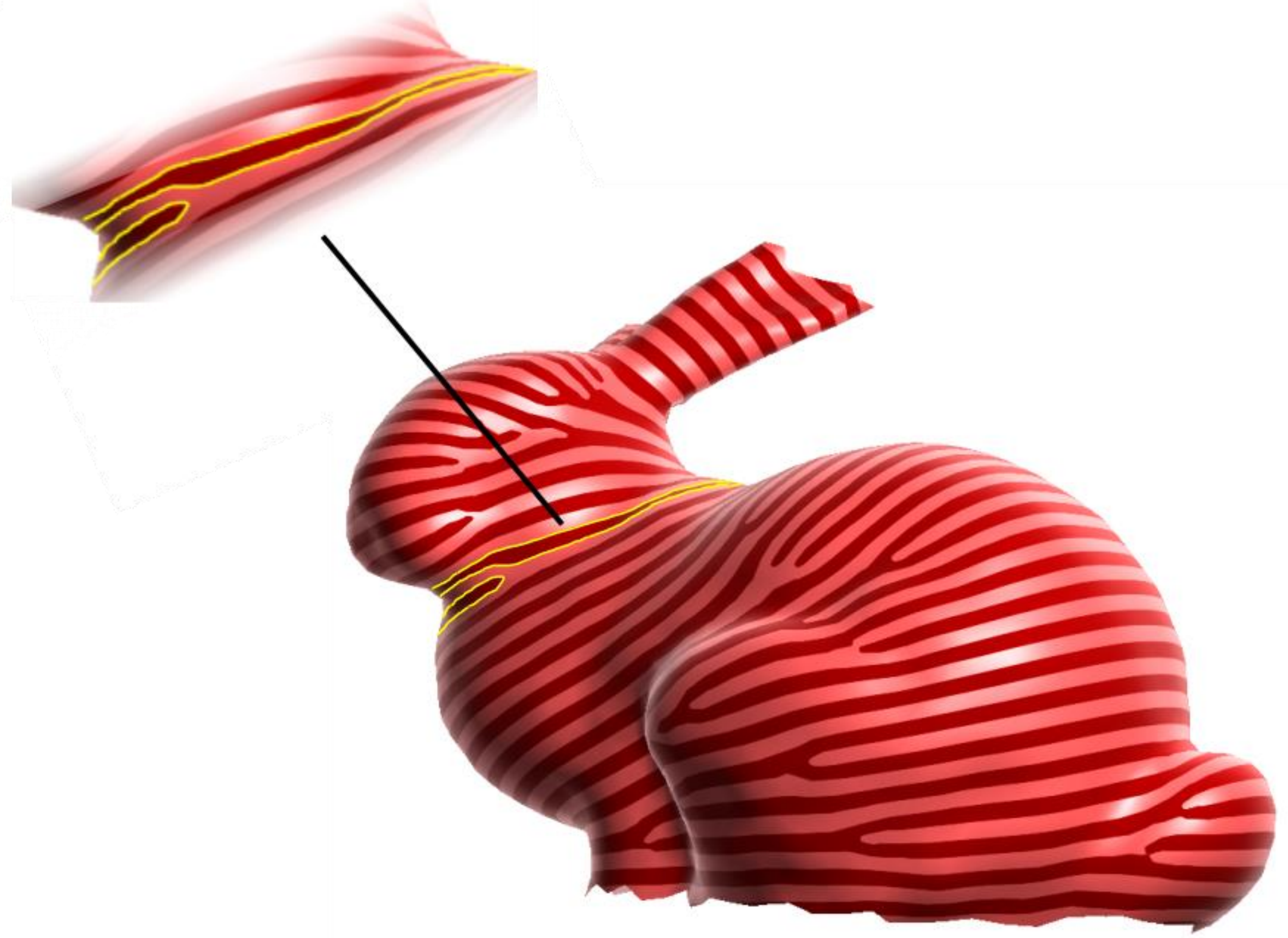
- Methods for generating stripe textures use an input vector field to guide direction and frequency of stripes, e.g., Stripe Patterns on Surfaces by Knöppel et al. 15



A plane example

Stripe Texturing

- Using a unit norm vector field achieves evenly-spaced stripes, analogous to the goal of evenly-spaced courses/wales
- **Problem:** direct application gives no simple way to get helix-free stripes
- Considered initially by KnitKit [Nader et al. 2021]; attempted remeshing operations to fix helices (sans guarantees)



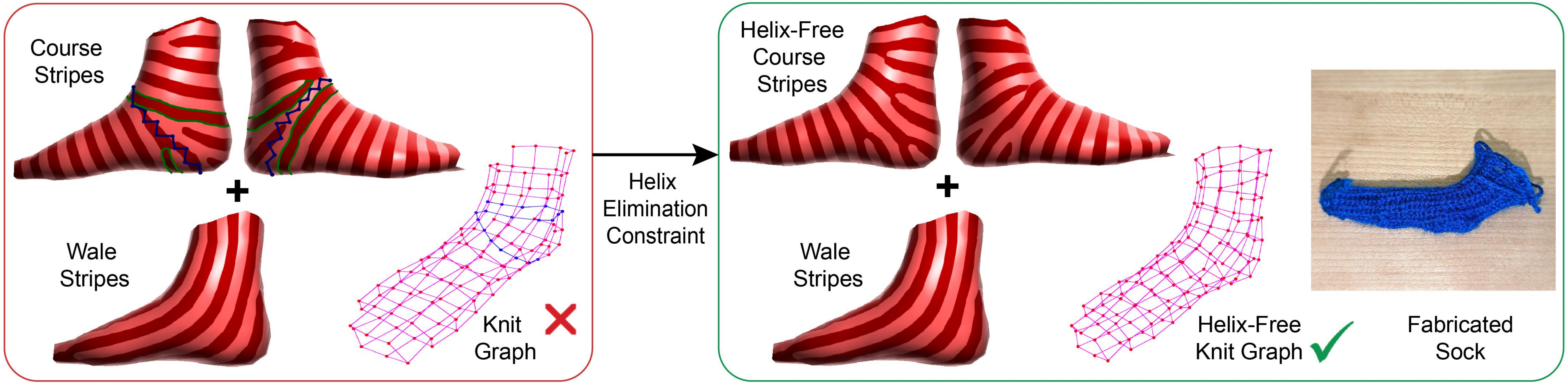
$$\overline{\nabla h} := \frac{\nabla h}{||\nabla h||}$$ for course stripes

$$\overline{\nabla h}^{\perp}$$ for wale stripes

Images: Helix-Free Stripes for Knit Graph Design, Mitra et al.

Helix-Free Stripes for Knit Graph Design

- Published at SIGGRAPH23: used simple linear constraints to achieved helix-free stripe patterns



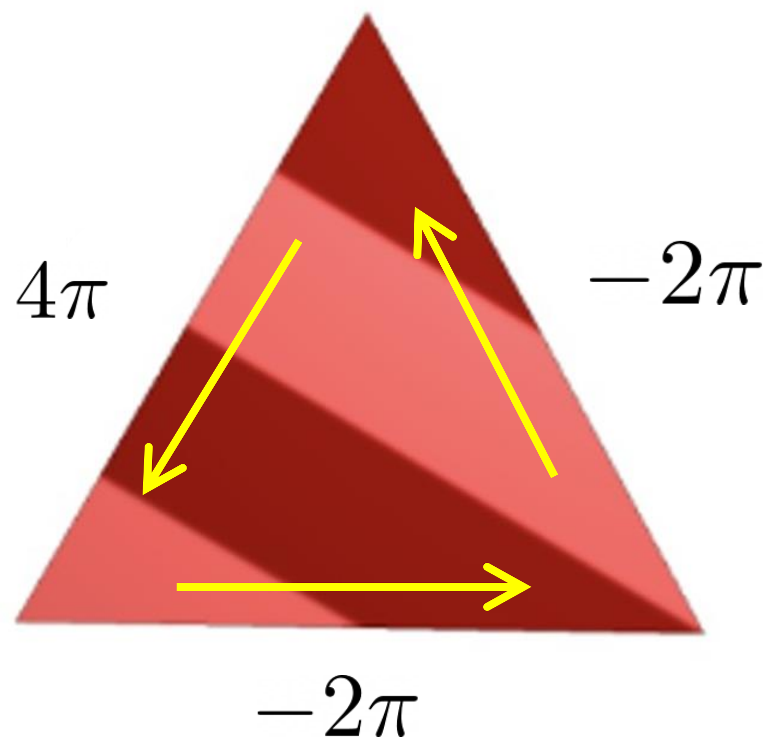
- Global optimization in the space of *spinning forms* contrasts with the iterative front-marching approach of Autoknit
  - Allows for simpler user input with linear constraints

Spinning forms

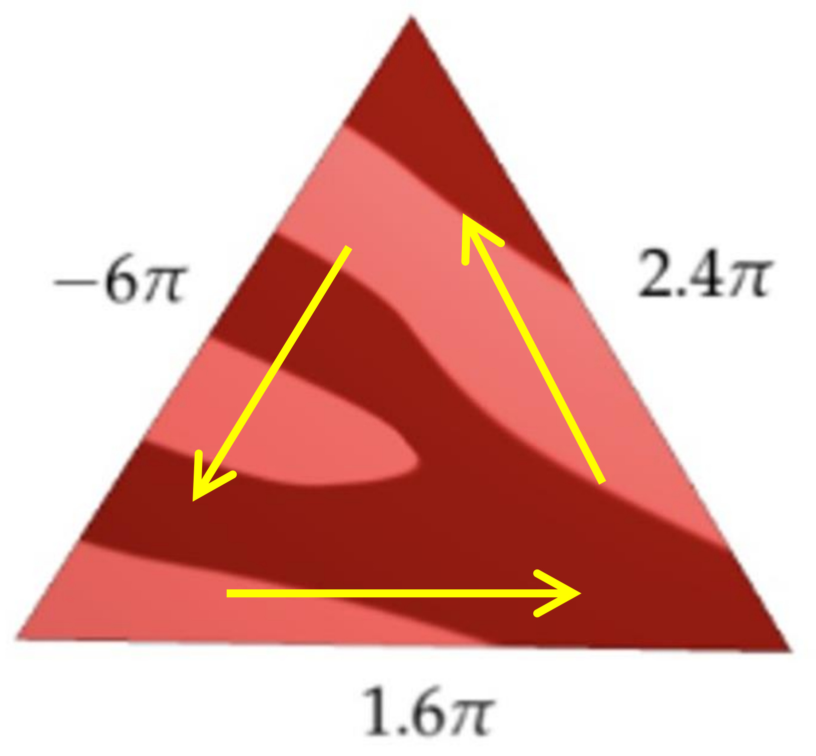
- Stripe patterns are specified via texture maps $\phi : S \to \mathbb{S}^1$

  Red when $\phi \in (0, \pi)$ Pink when $\phi \in (\pi, 2\pi)$

- Discretized in [Knöppel et al.] as *spinning forms*, discrete 1-forms $\sigma : E \to \mathbb{R}$
  - σ_e denotes the change in ϕ over edge e
- (Discrete Differential Geometry notes, Keenan Crane)



Use piecewise linear interpolant over most triangles



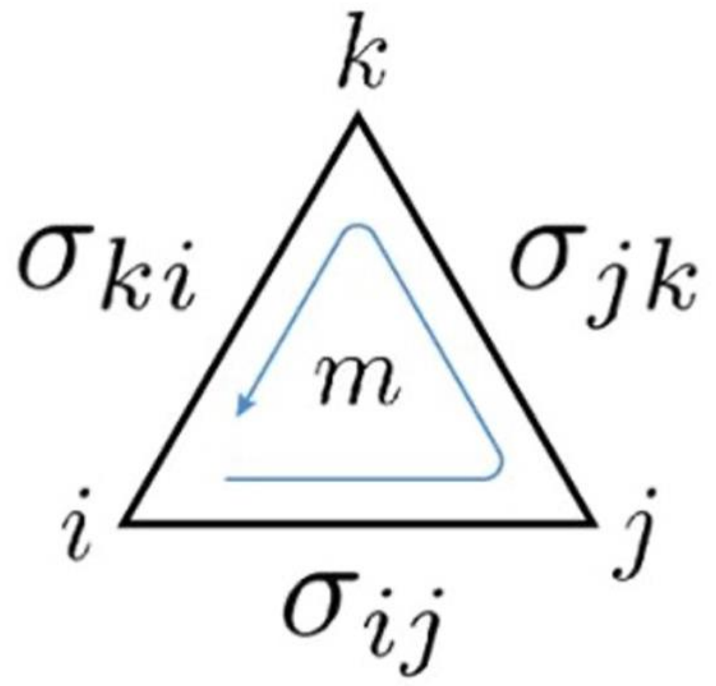
Over singular triangles use novel interpolant from [Knöppel et al!]

Spinning forms

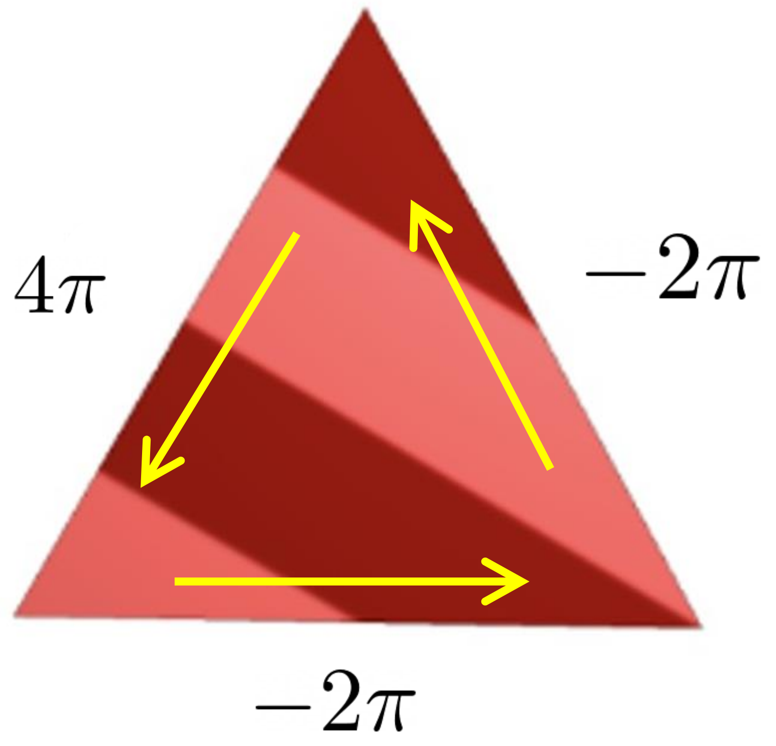
- In particular, $\sigma : E \to \mathbb{R}$ has integer curl on all faces

$$(d_1\sigma)_m \in 2\pi\mathbb{Z}$$

- If $(d_1\sigma)_m \neq 0$, we call it a *stripe singularity*
- On such faces, there is a birth or death of 2π level sets of ϕ

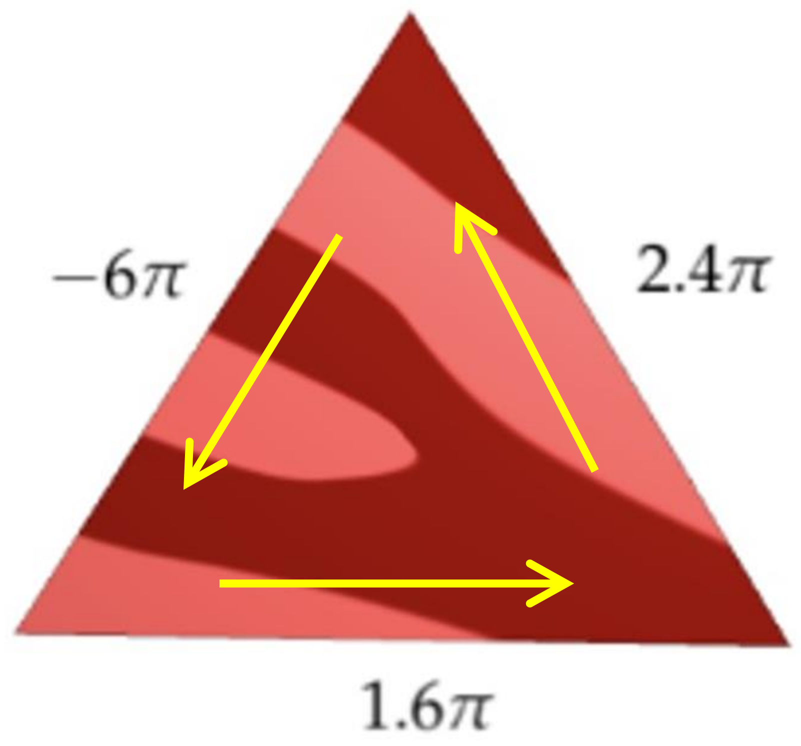


$$(d_1\sigma)_m = \sigma_{ij} + \sigma_{jk} + \sigma_{ki}$$



$$(d_1\sigma)_m = 0$$

Piecewise linear over most triangles

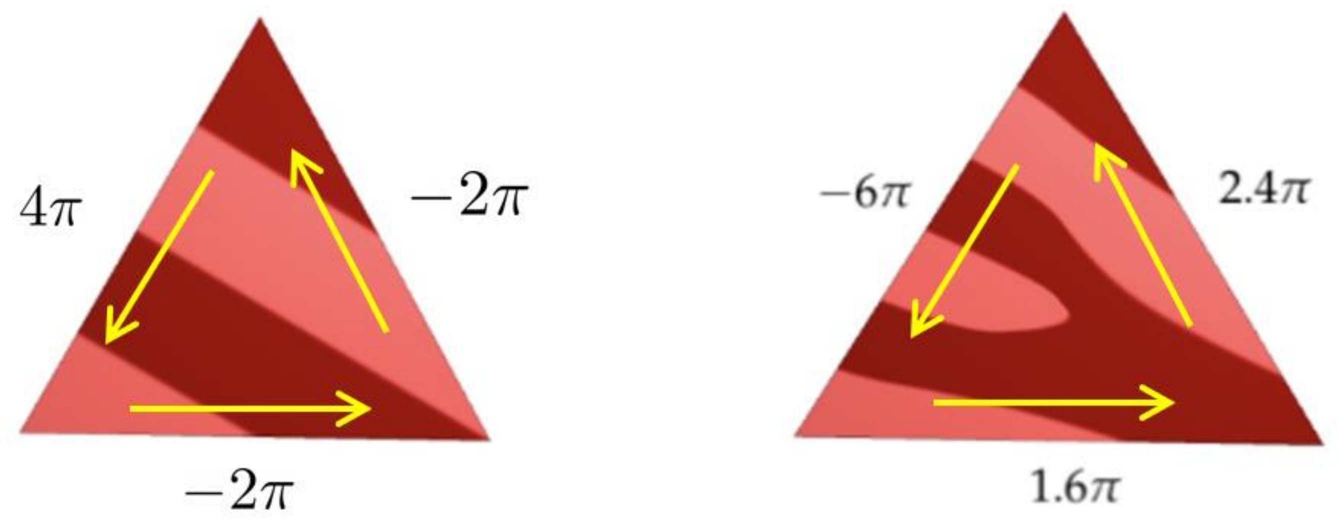


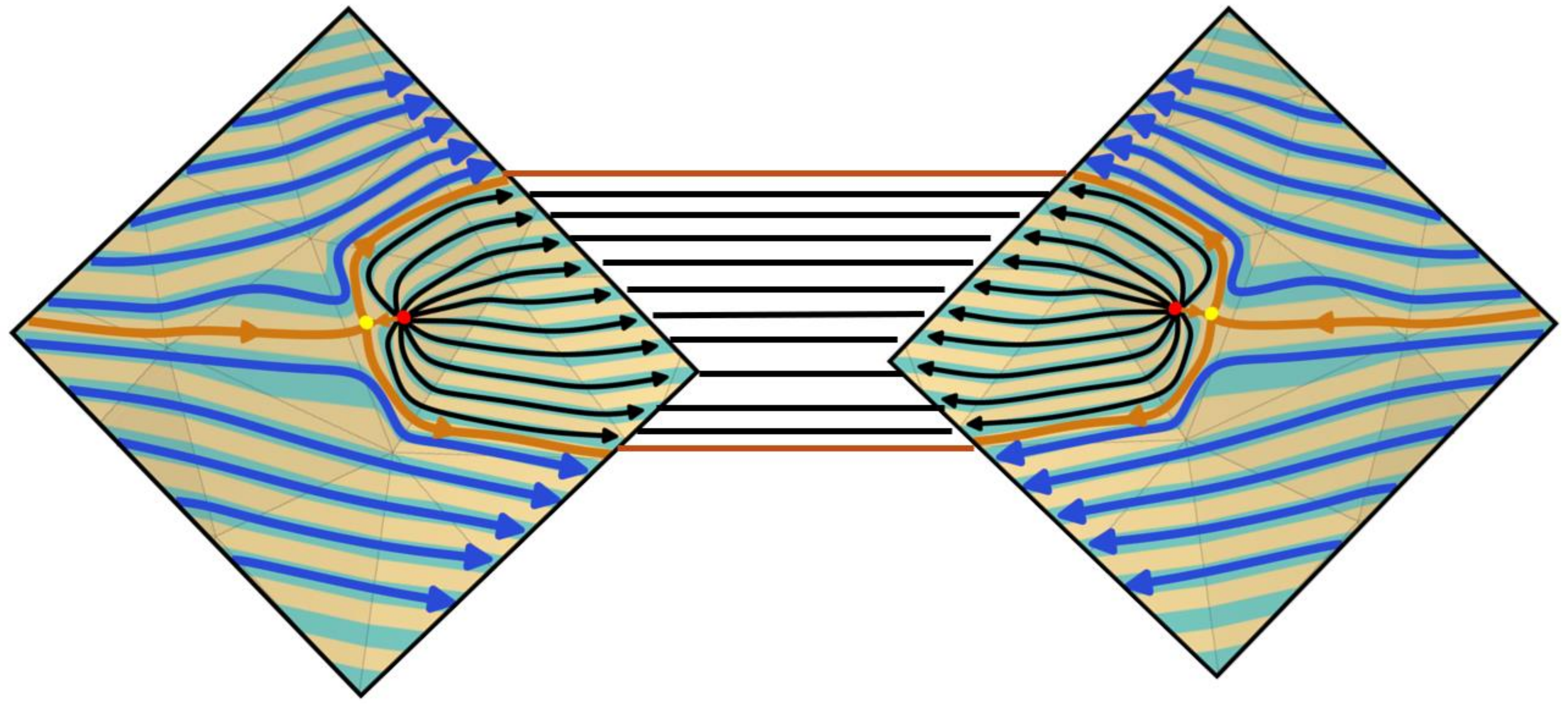
$$(d_1\sigma)_m = -2\pi$$

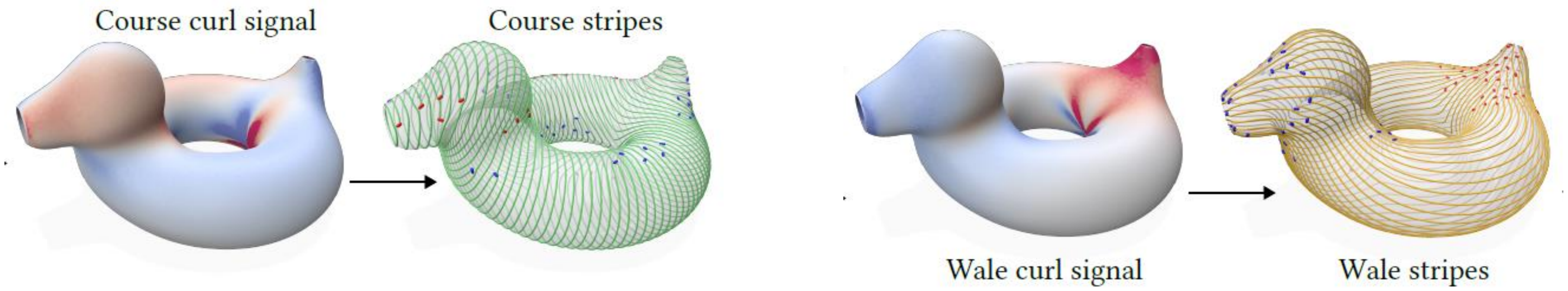
Novel interpolant from [Knöppel et al!]

Three Main Concepts

1. Spinning Forms for Stripes
2. Global Foliation Structure: Orbit Complex
3. Stripe Singularity Placement via Curl Quantization





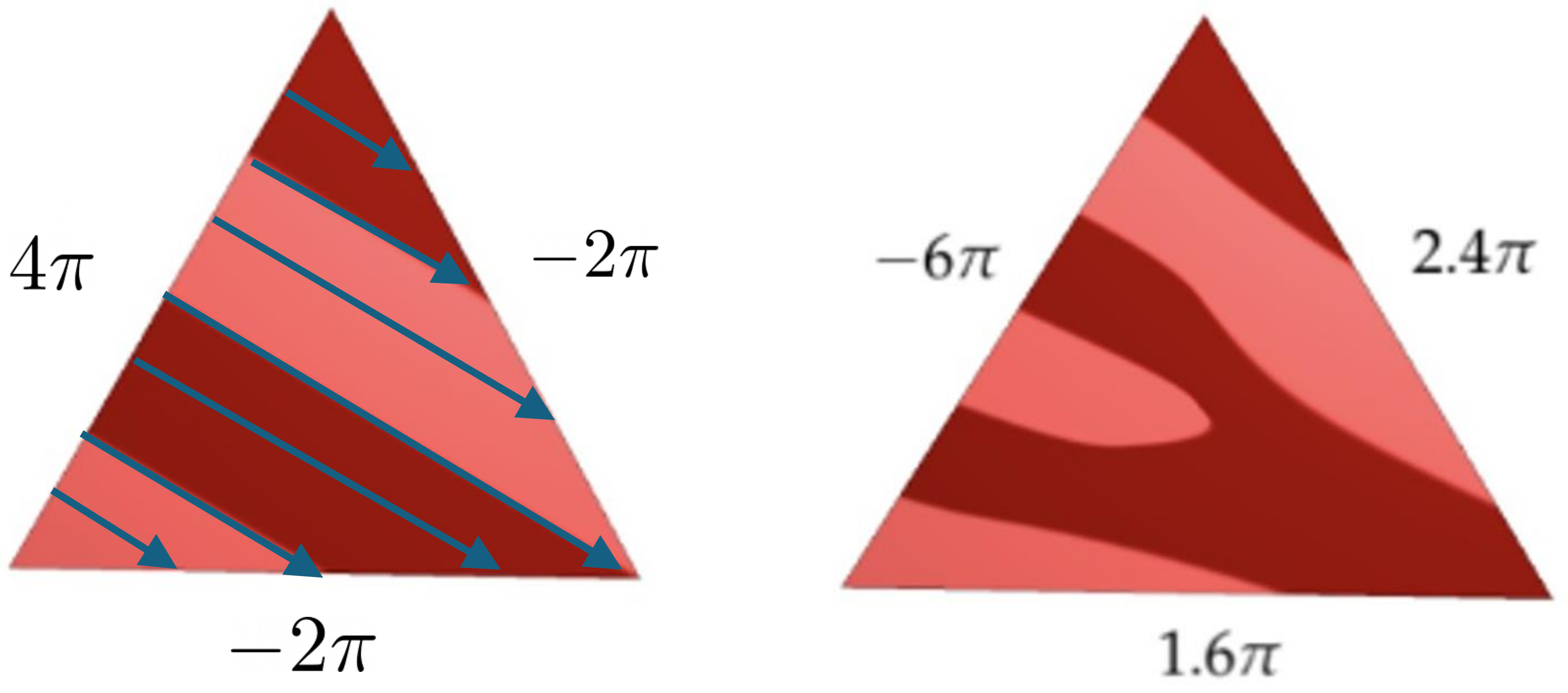


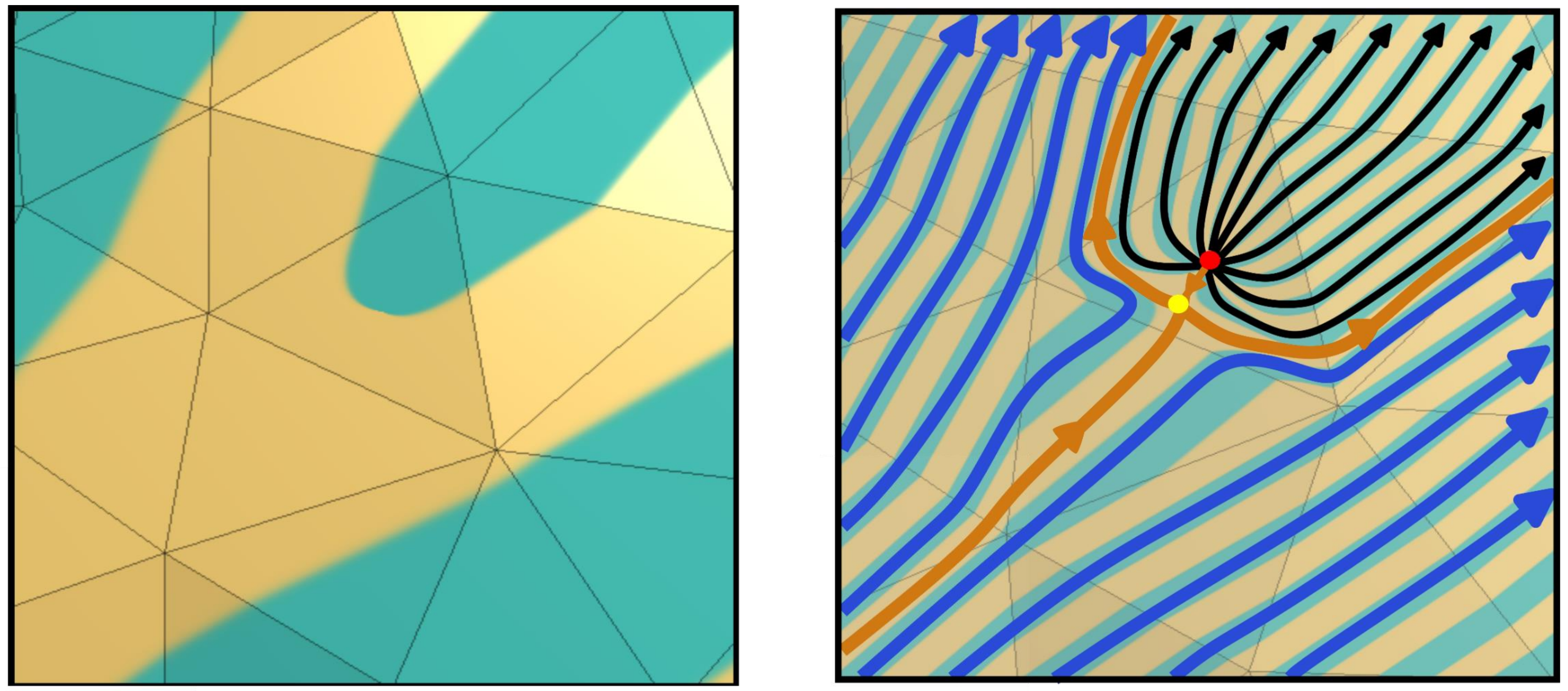
Foliations as vector field flows

- A spinning form σ can be interpreted as a discretized vector field V_σ
- Its *integral curves* form the leaves of a *foliation*

$$\gamma'(t) = V_\sigma(\gamma(t))$$

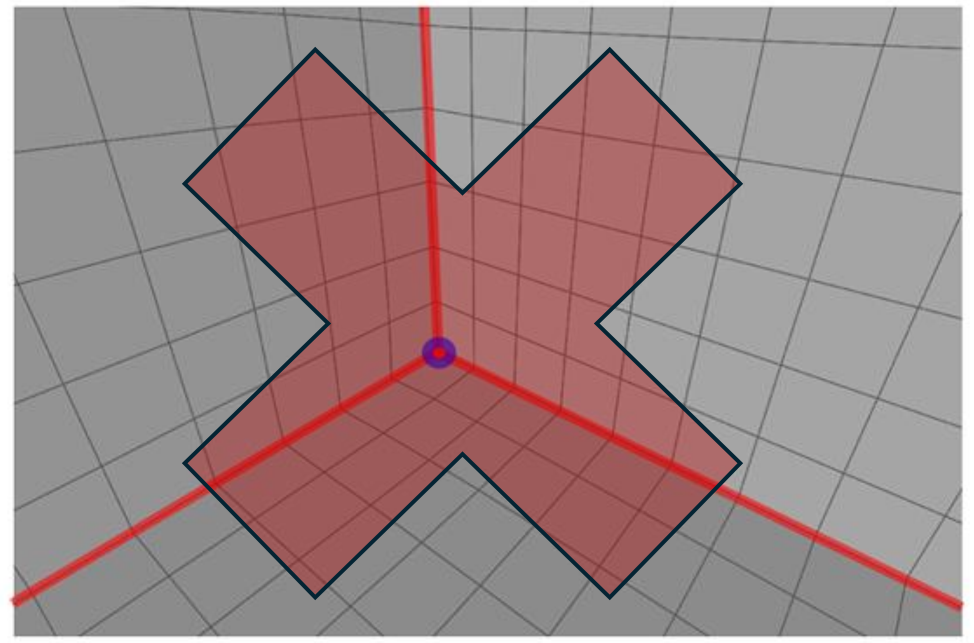
  - Stripes are collections of integral curves
  - Courses and wales in eventual knit graph are particular integral curves
- Nontrivial topological behavior only at faces with stripe singularities



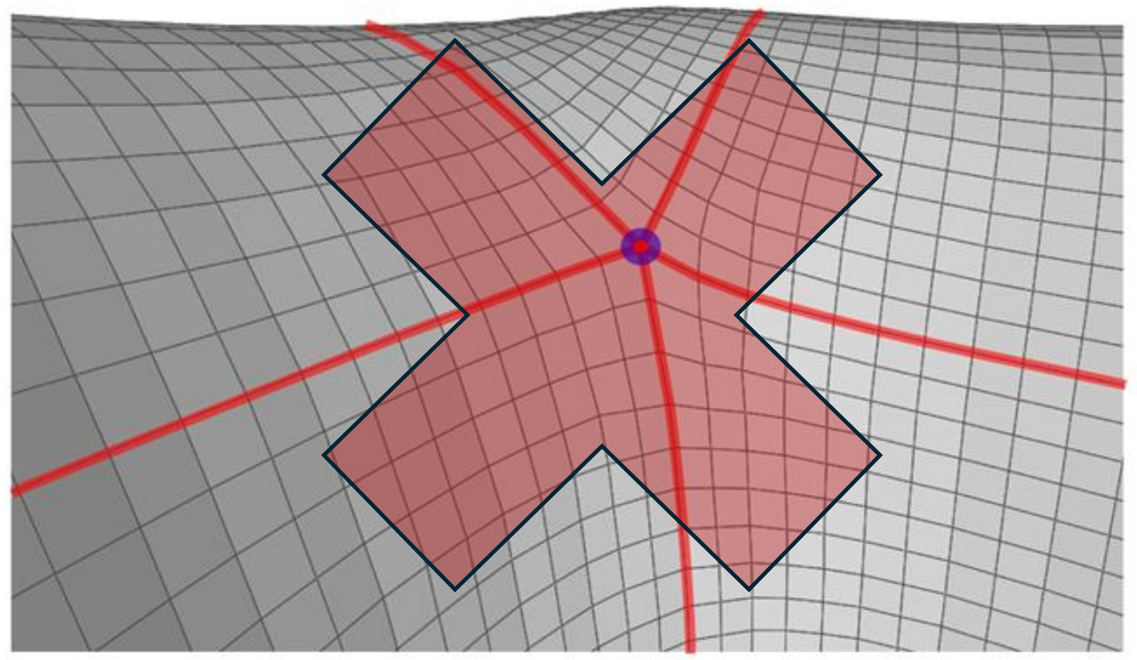


An aside

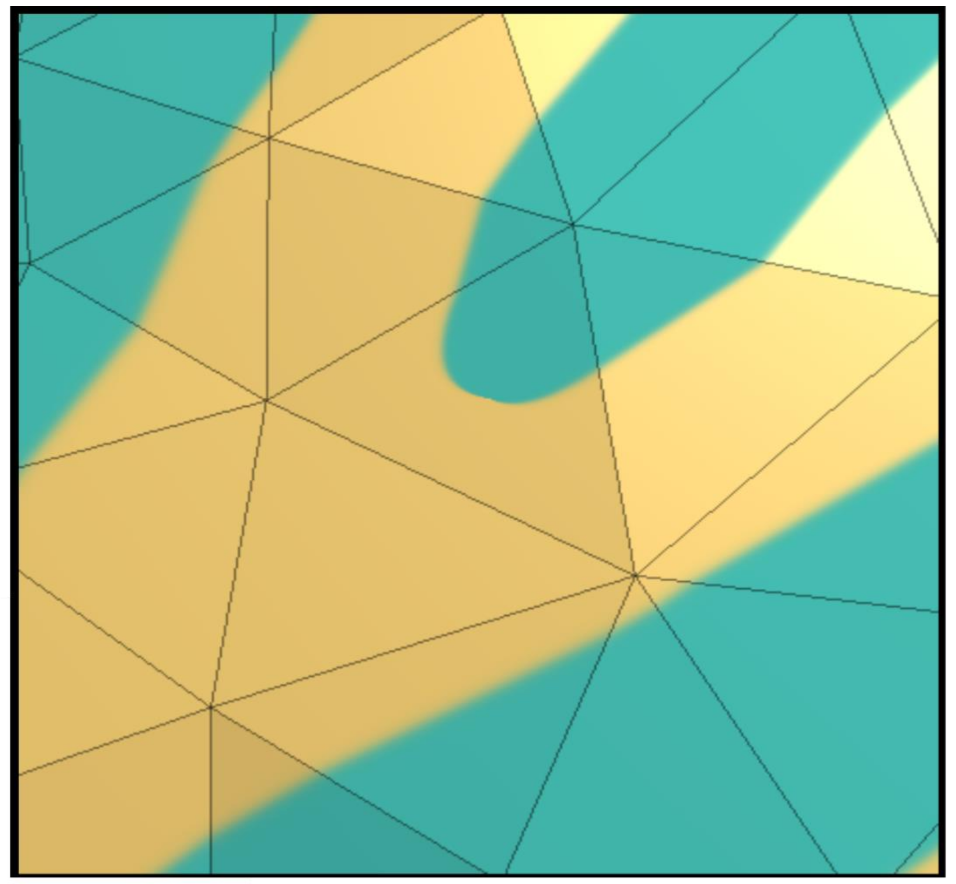
- Stripe singularities are NOT your typical quad mesh singularities
- Not machine-knittable, and lead to disagreement of course/wales, or mismatched direction
  - Present in the composition rules of Ben's work
- Akin to the "position" singularities of [Instant Meshes]()

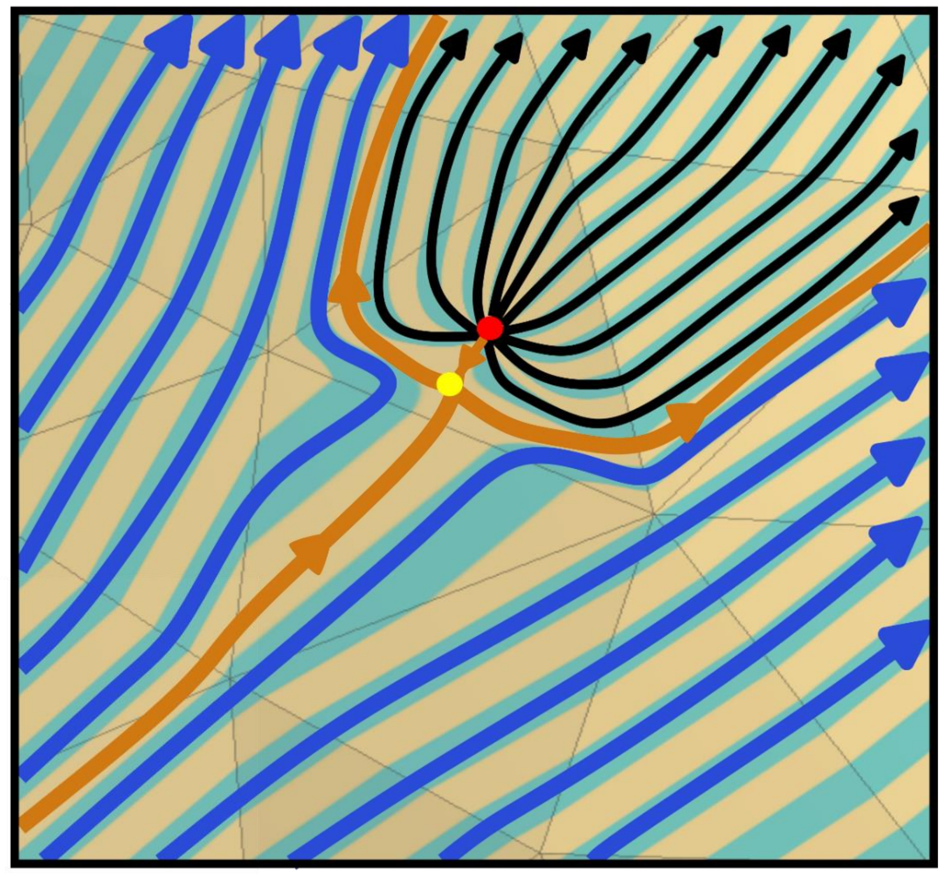


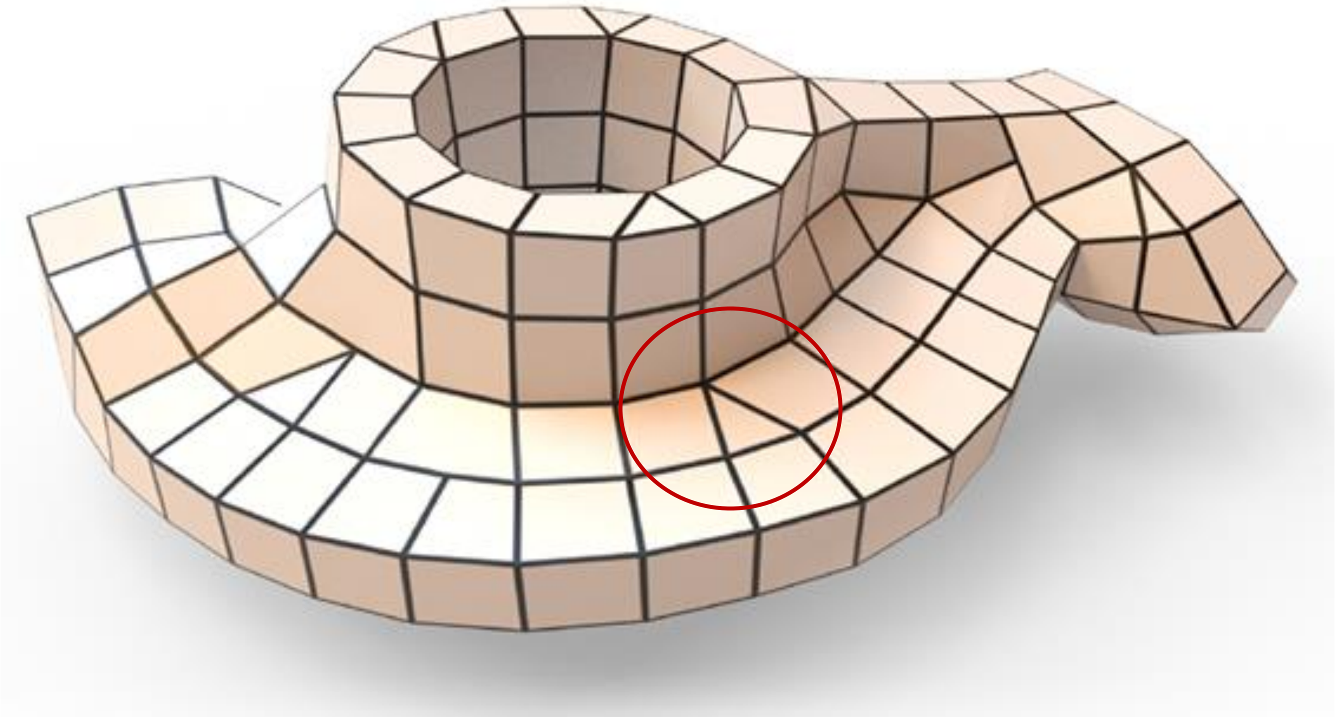
Index +1/4, valence 3



Index -1/4, valence 5



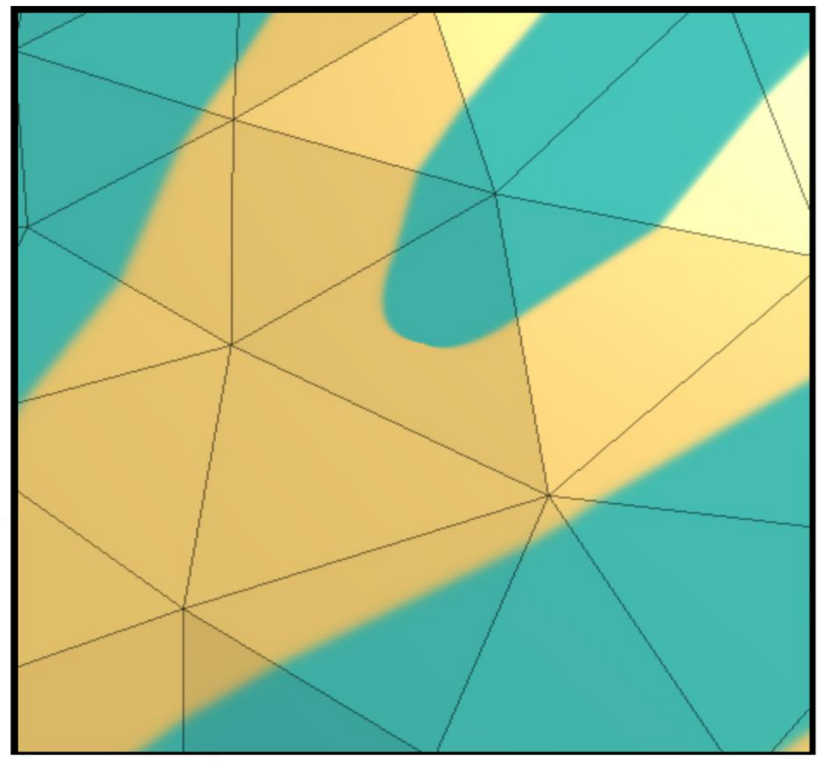




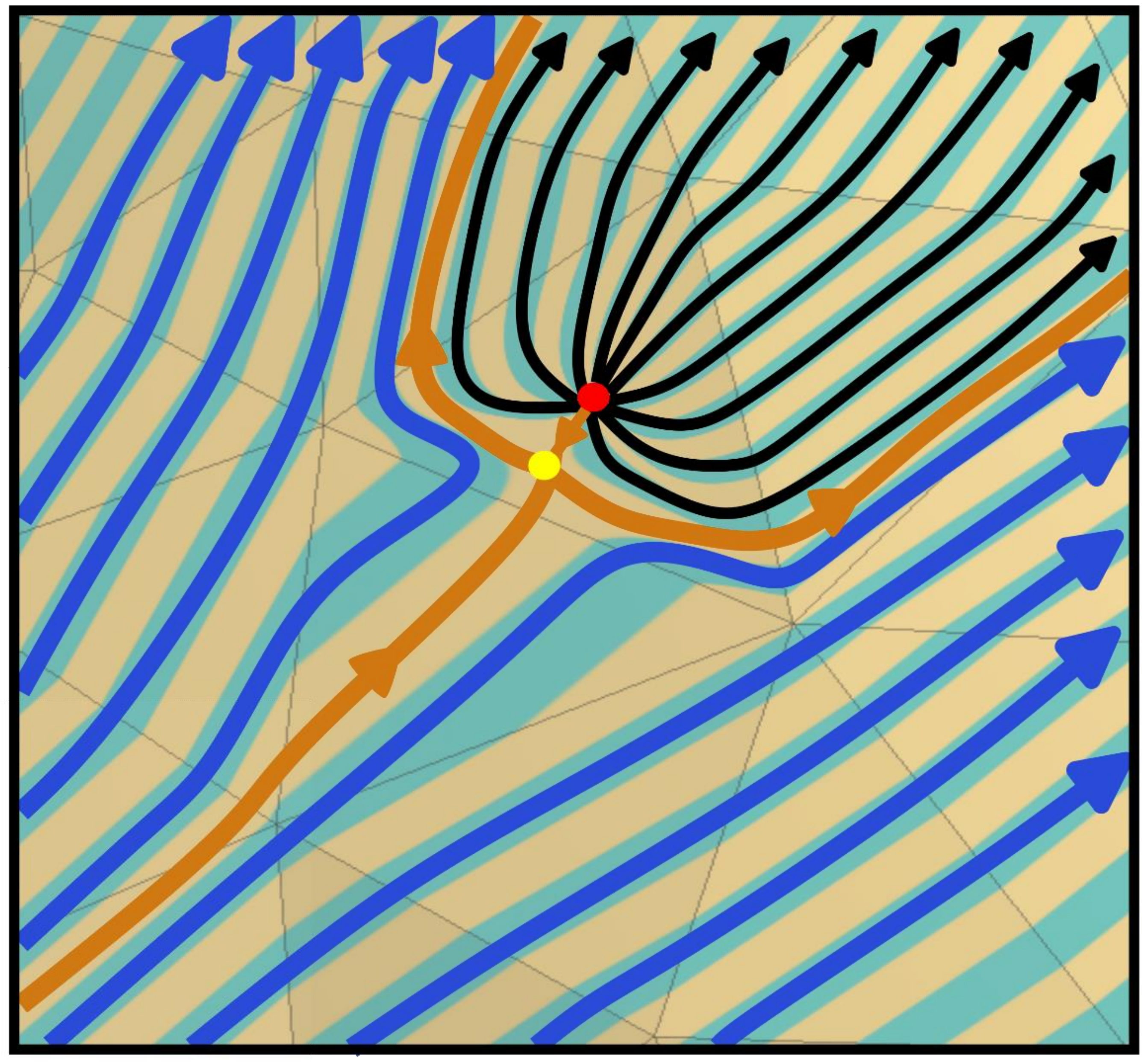
Bottom image credit: *Instant Meshes*, Jakob et al. 2015

Top right image credit: *Quad Mesh Generation and Processing*, Bommes et al. 2017

Local stripe singularity structure

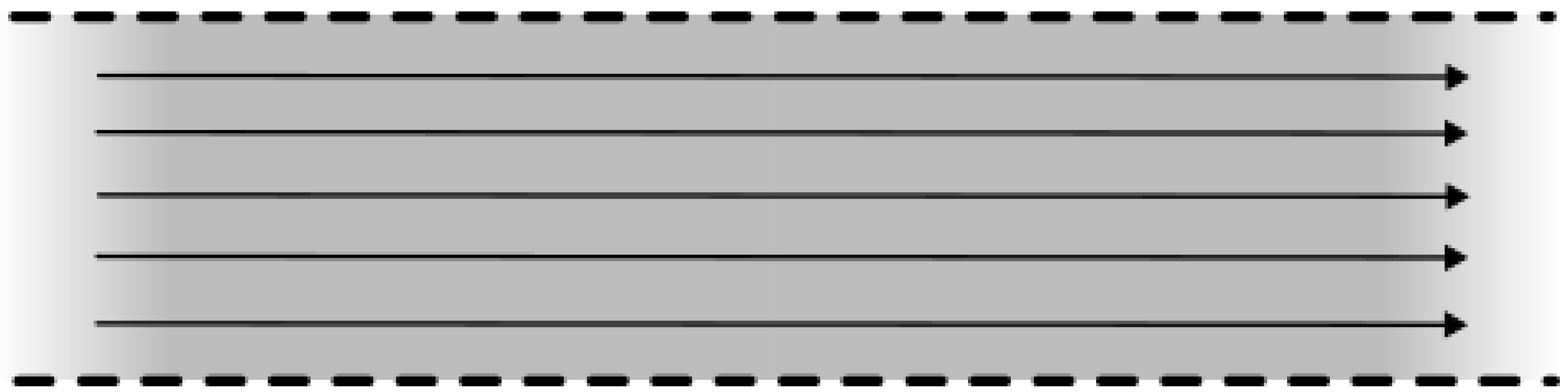


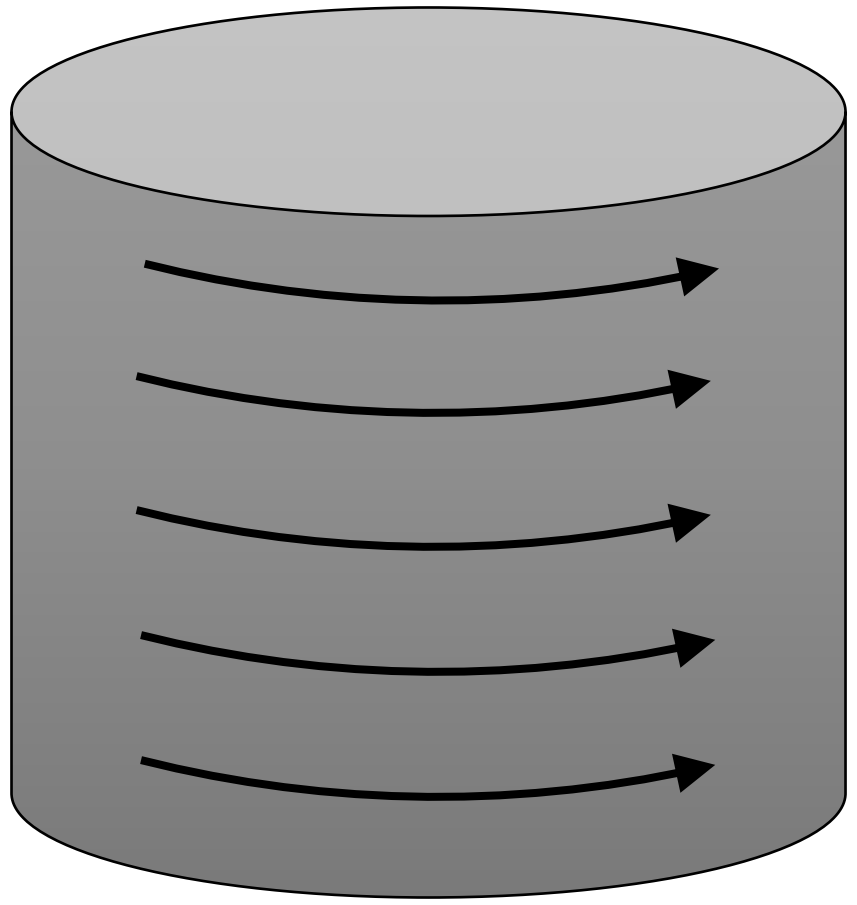
- Stripe singularities are pairs of index +/-1 (red/yellow) singularities of a vector field
  - (Index refers to the number of times the vector field rotates about a singularity)
- Separatrices are integral curves that start or end at saddle points (yellow singularities)
- They partition S into *cells* of the *orbit complex*, a global descriptor of the foliation topology

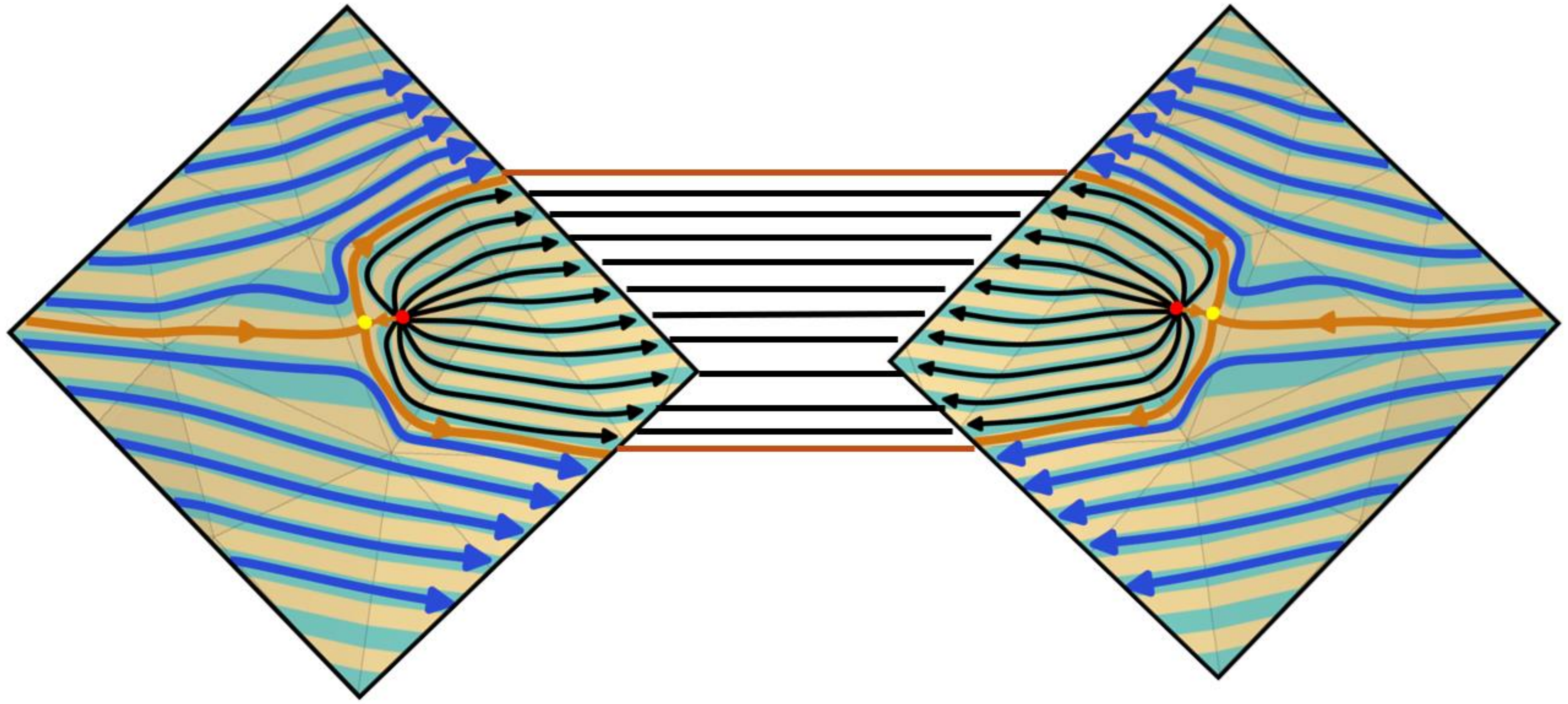


Orbit complex cell types

- Flow on all cells is topologically equivalent to:
  - A periodic flow on a cylinder
  - Or a horizontal flow on an infinite strip $\mathbb{R} \times (0,1)$



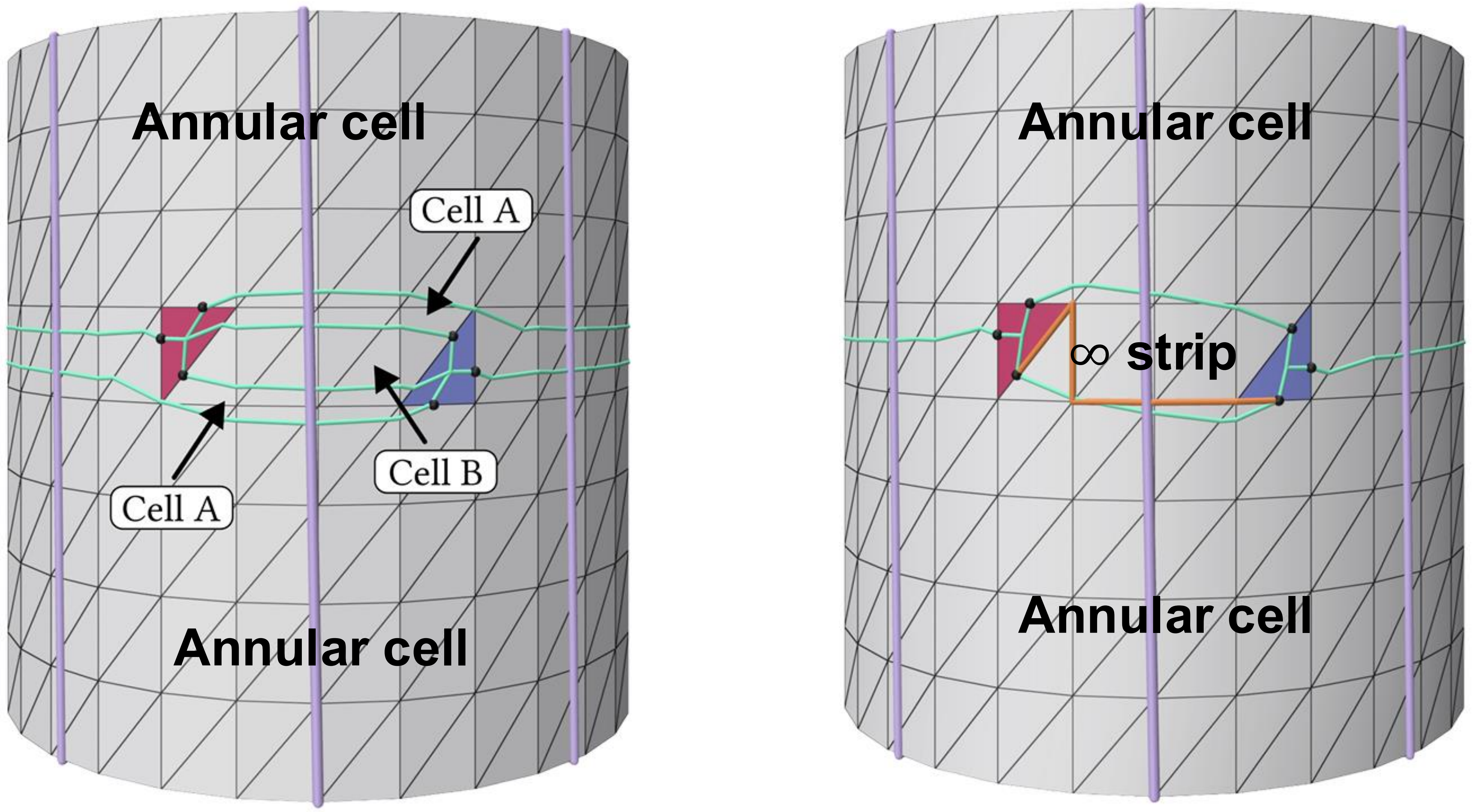




(flow slows down infinitely at singularities)

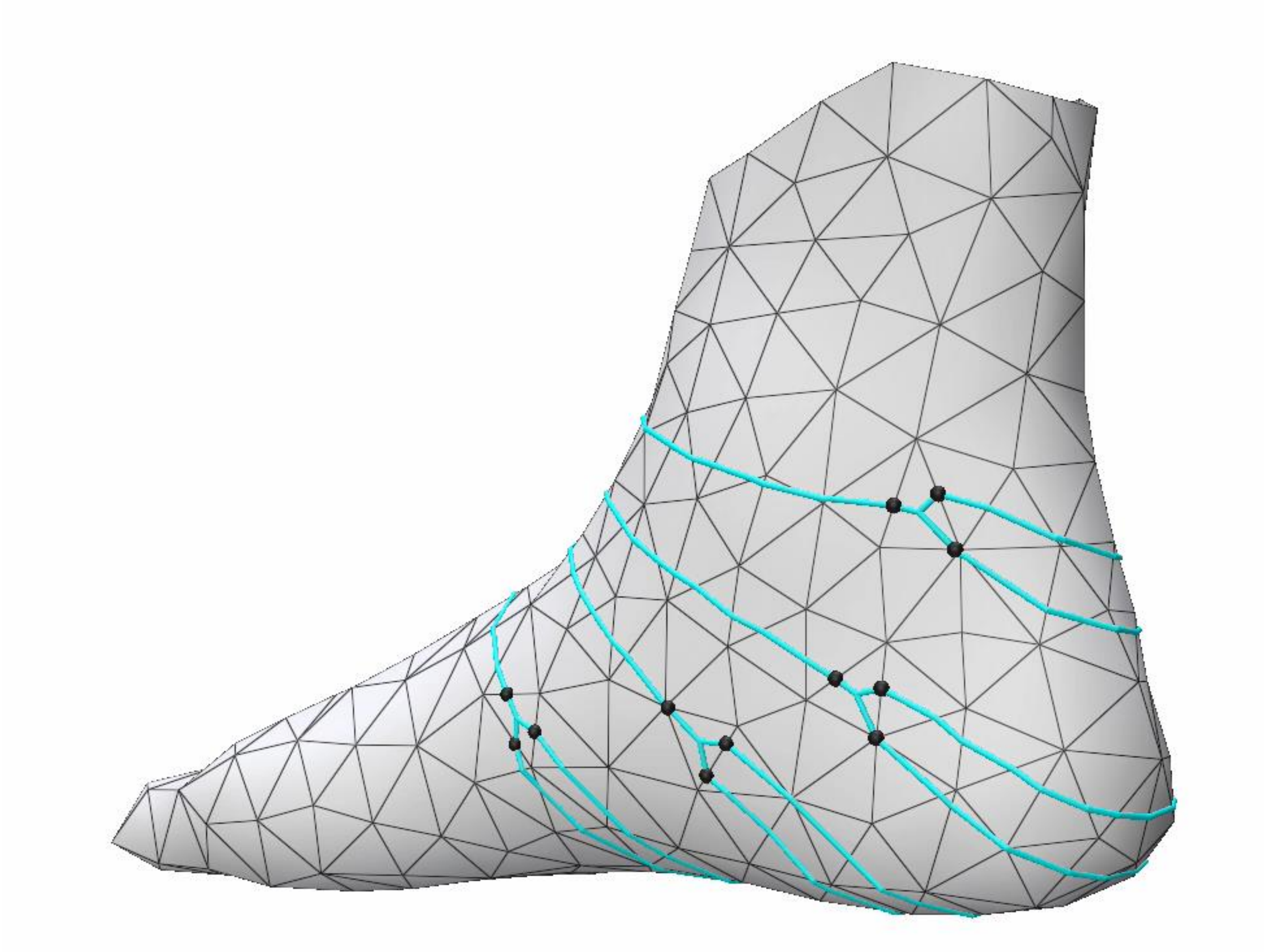
Simple orbit complex example

Cells A & B are both
∞ strip cells



- <u>Note</u>: the integral curves in Cell A helix, while those in Cell B do not
- Simple linear path integral constraints (orange above) align separatrices and prevent helicing of *any level set*

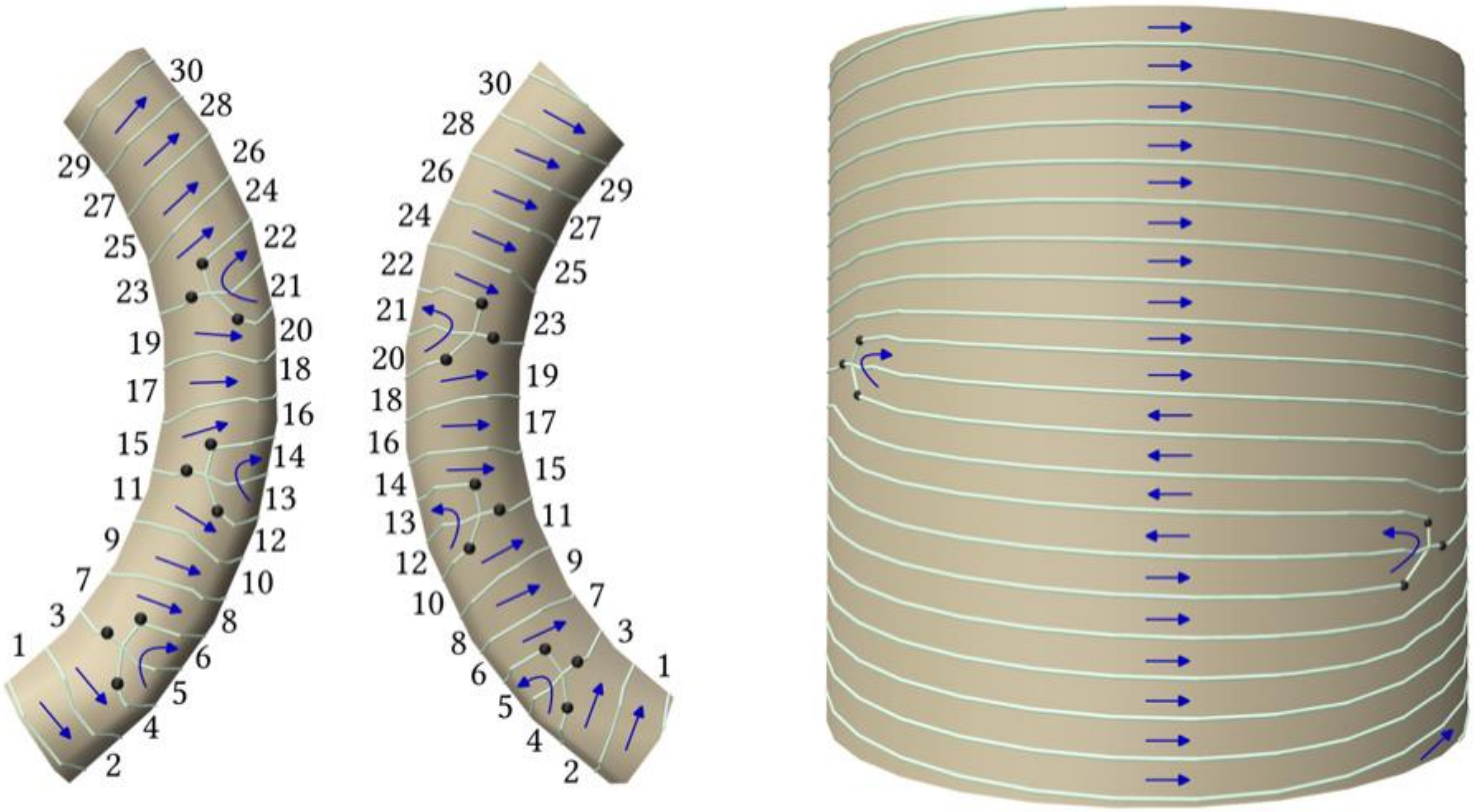
A slightly more complex example



- Further details can be found in our SIGGRAPH24 work: [Singular Foliations for Knit Graph Design]
- Mathematical concept of *orbit complex* in reference: “Introduction to the Qualitative Theory of Dynamical Systems on Surfaces” [Aranson et al. 96]

Exact helicing => Tracing-free pipeline?

- Closer control of foliation structure opens door to a *tracing-free* pipeline

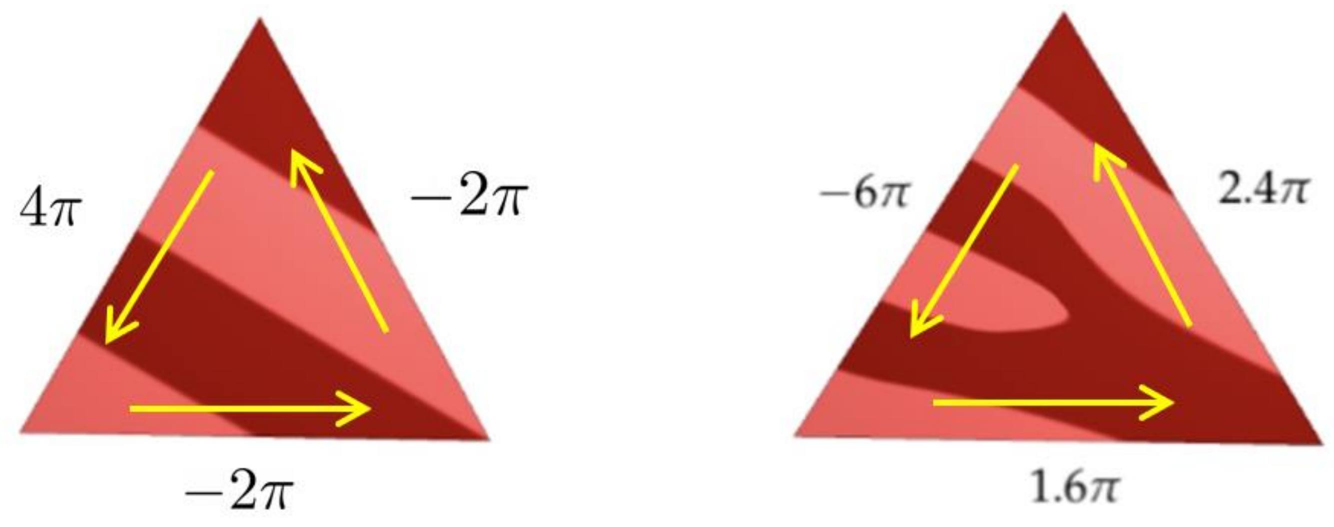


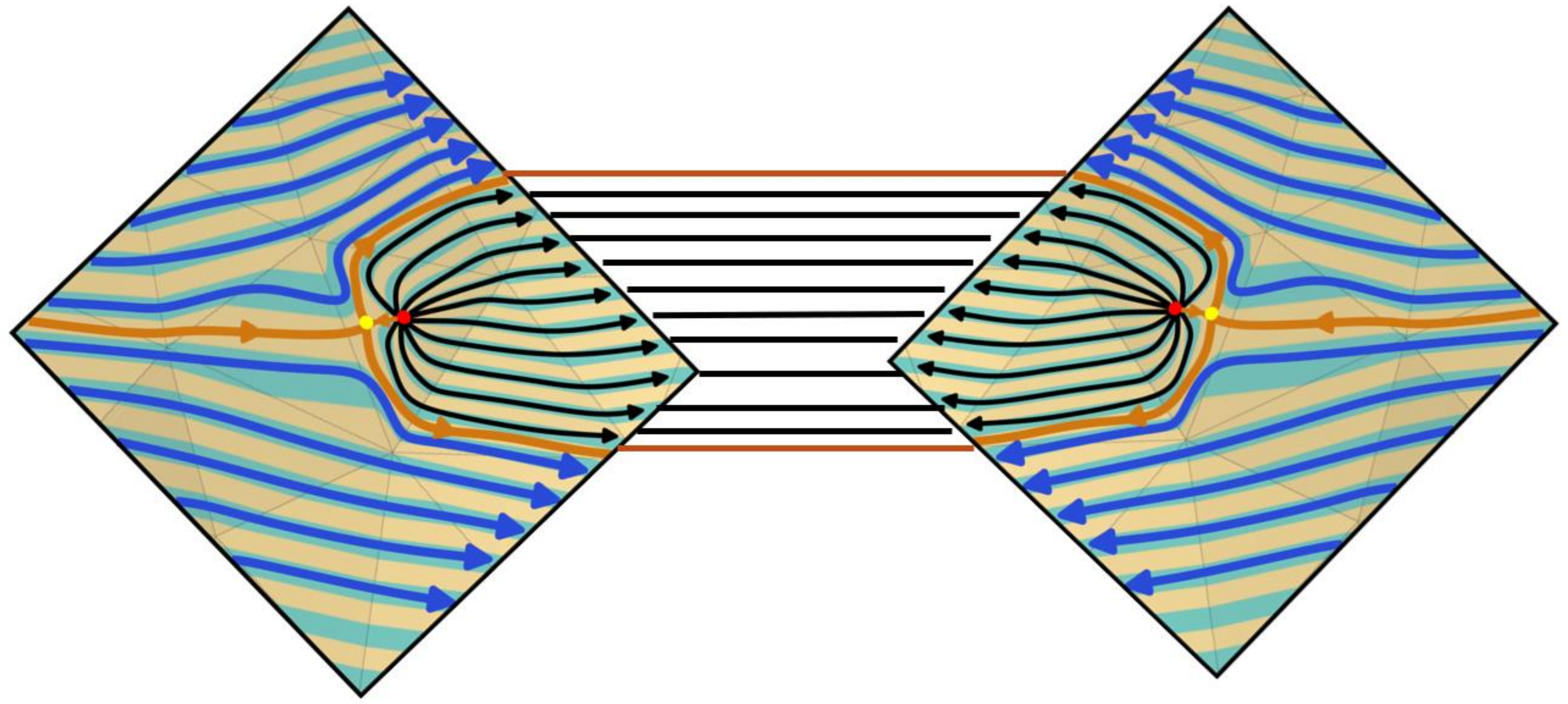
Tracing (yarn path) implied directly by foliation structure

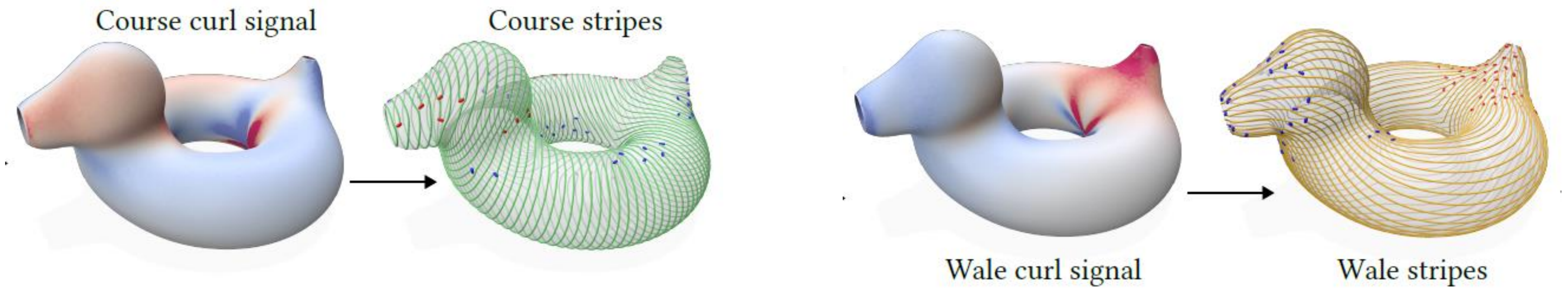
- Capable of representing a broader range of yarn paths than the knit graphs of Autoknit

Three Main Concepts

1. Spinning Forms for Stripes
2. Global Foliation Structure: Orbit Complex
3. Stripe Singularity Placement via Curl Quantization

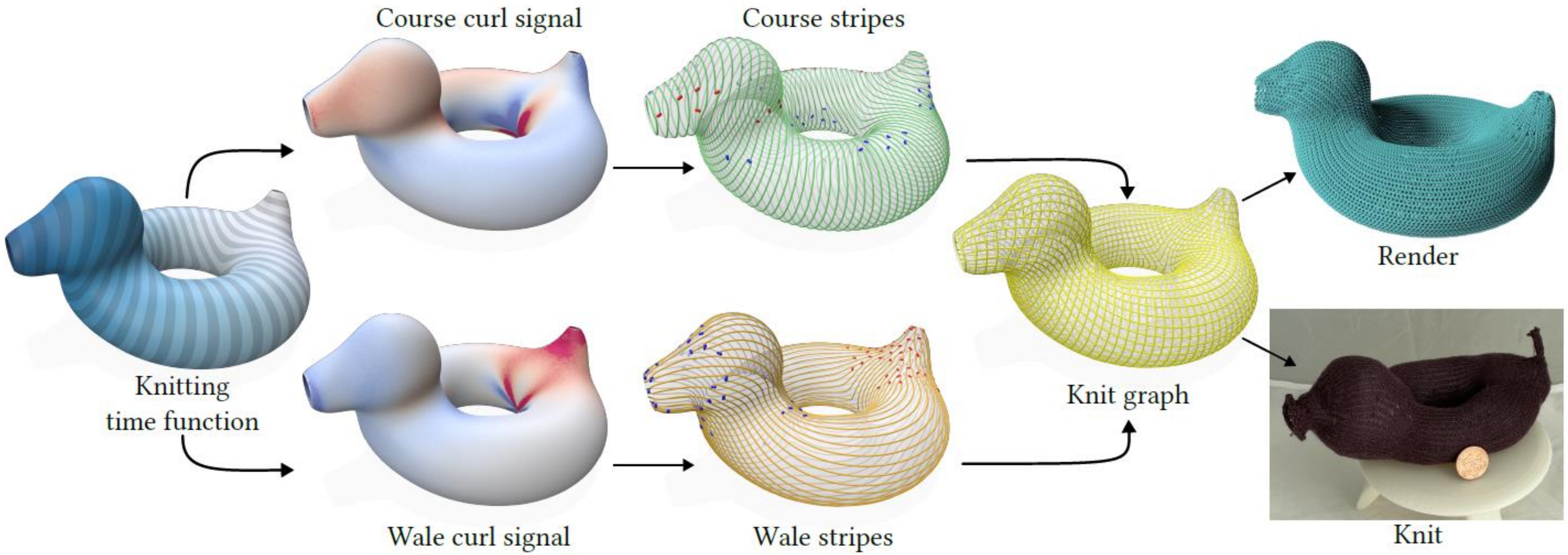






But where do we place singularities?

- We look at the problem as one of *curl quantization*



- To be presented at SIGGRAPH25: [Curl Quantization for Automatic Placement of Knit Singularities]

Curl quantization problem

- We discretize the following continuous optimization problem:

normalized time function gradient

vector field discretized by σ

$$\min_{V_\sigma, p_1, \ldots, p_n} \int_M \|V_\sigma - \overline{\nabla h}\|^2$$

curl quantization sites (stripe singularities)

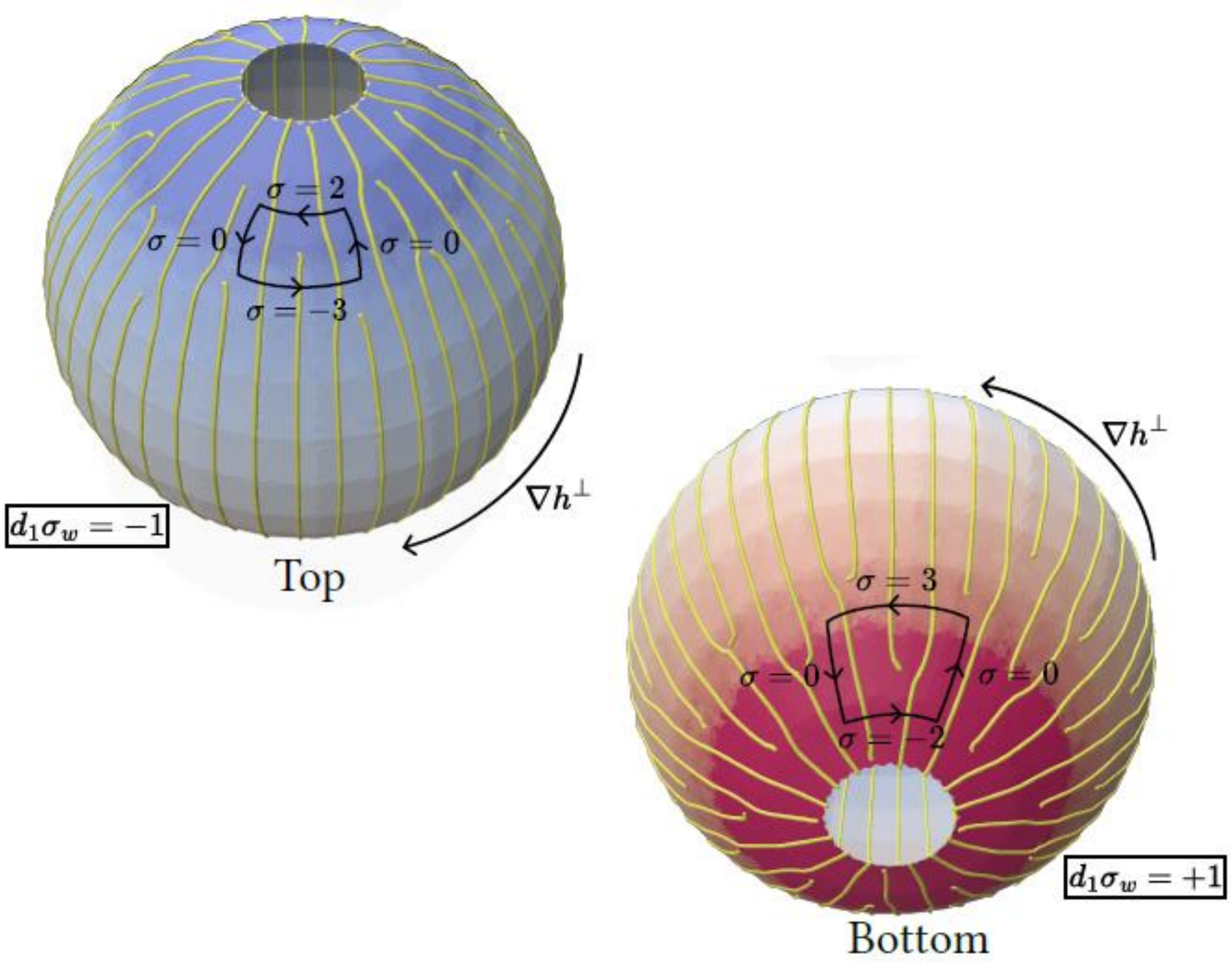
$$\text{such that} \quad \nabla \times V_\sigma = 2\pi \sum_{i=1}^{n} k_i \delta(p_i)$$

± 1

- (technically ill-posed, in the same way that Trivial Connections [Crane et al. 2010] is)

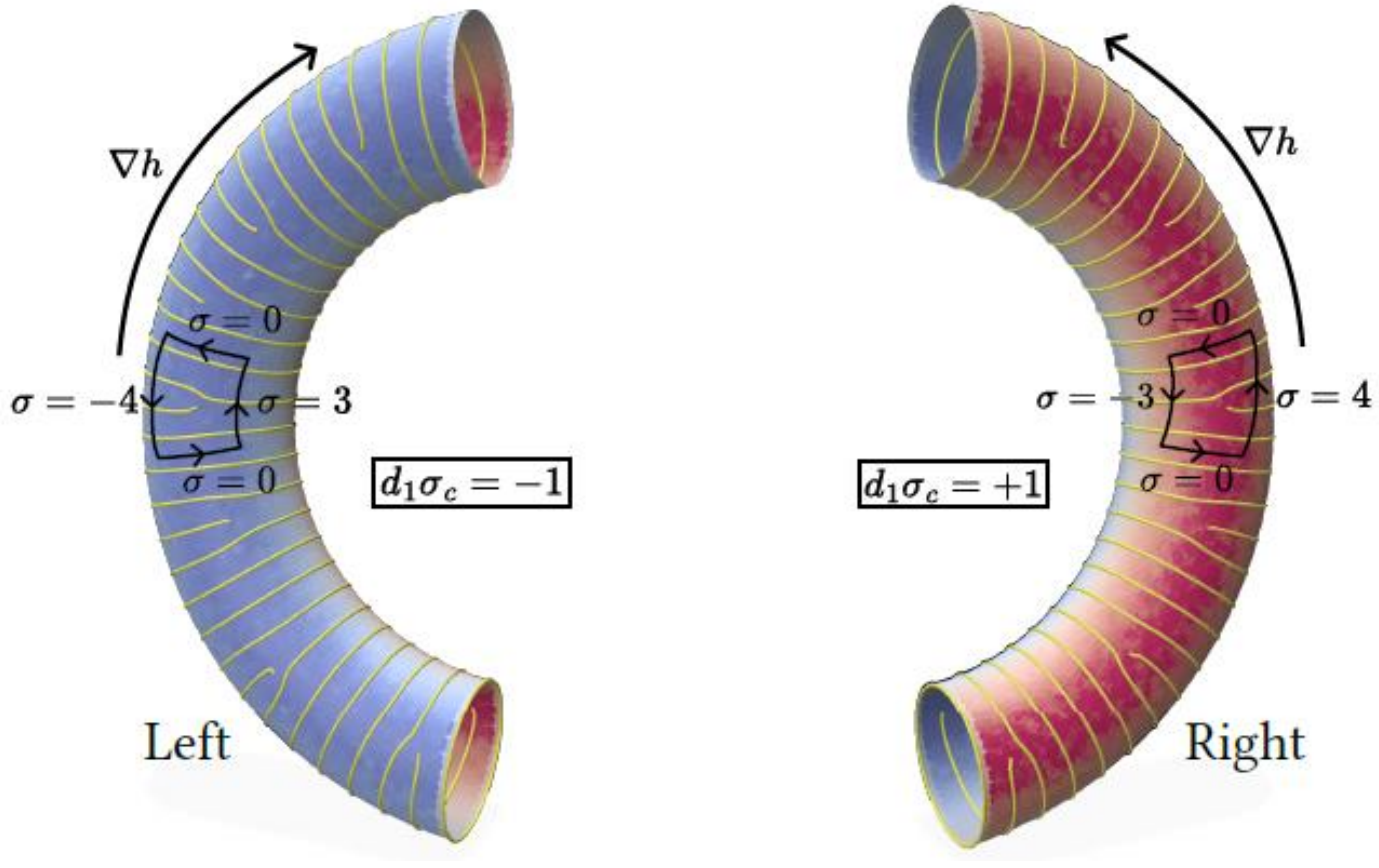
Intuitive sphere example

- Curl analogous to use of Gaussian curvature K for quad mesh singularities
- Gives a heuristic to place stripe singularities
- Note curvature same at increase and decrease placements here



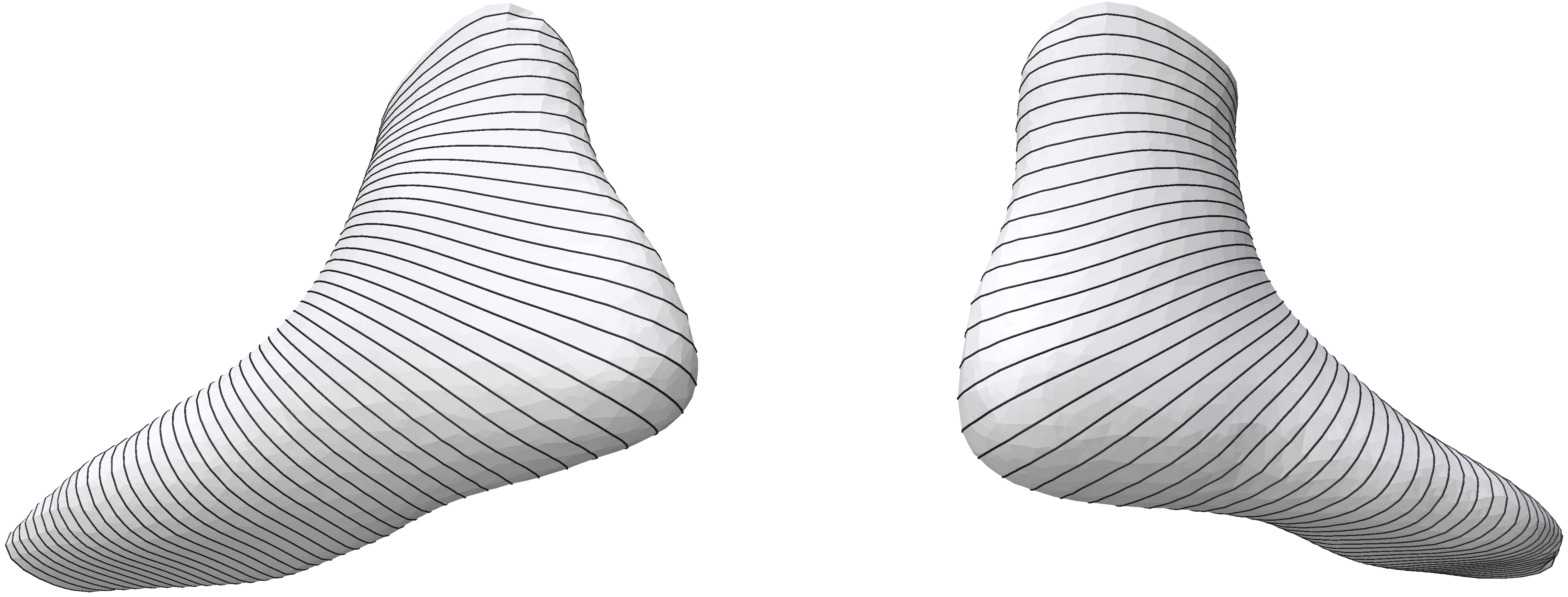
Bent cylinder example

- The curl of a vector field measures how locally *non-integrable* it is.
- When $|\nabla \times \overline{\nabla h}|$ is particularly high, it indicates a large variation in spacing of time function isolines.
- Insertion of a singularity allows for evenly-spaced stripes on either side.



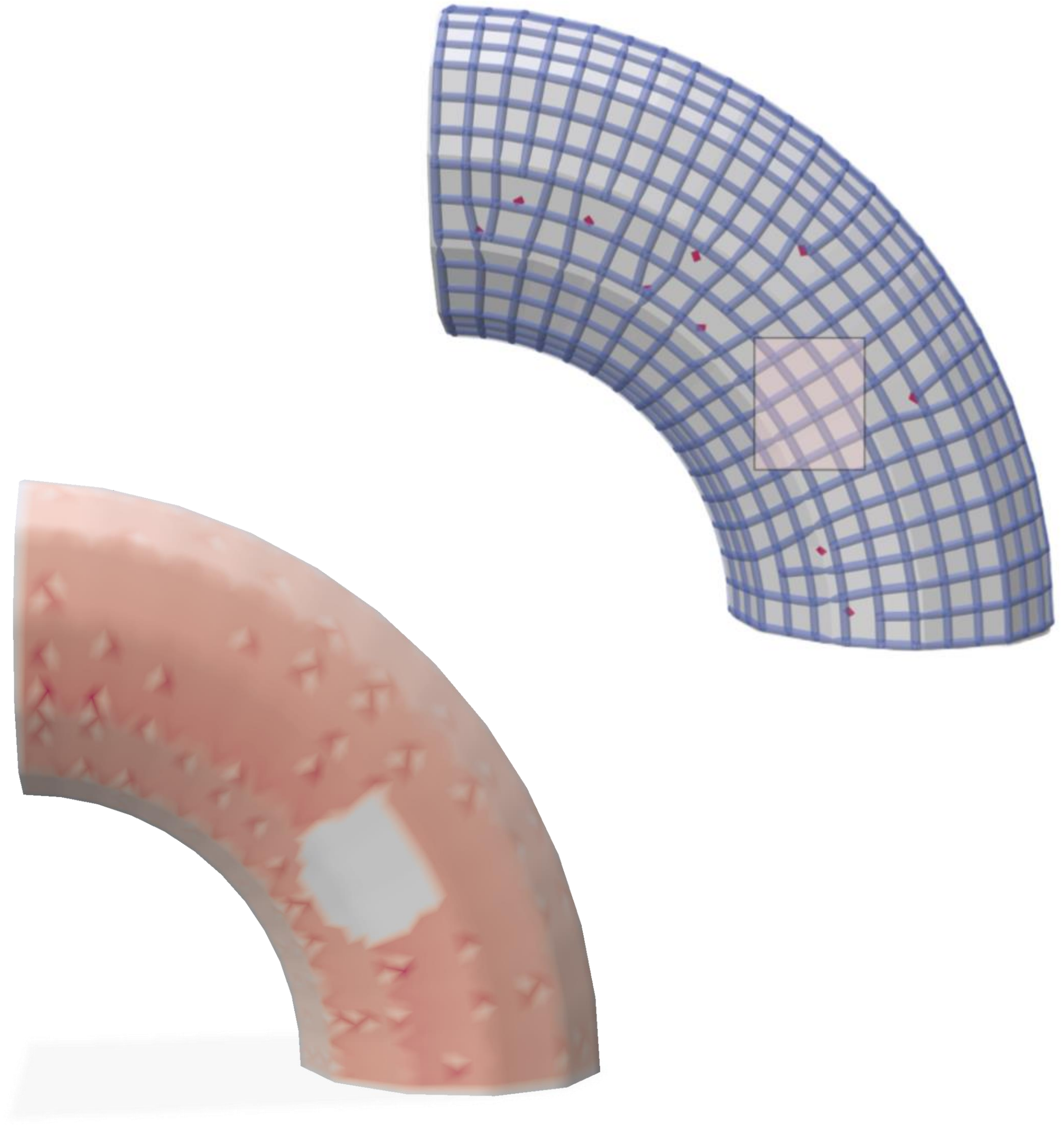
An iterative strategy

- Our solve strategy greedily places singularities at locations of high curl, accounting for prior placements in each step



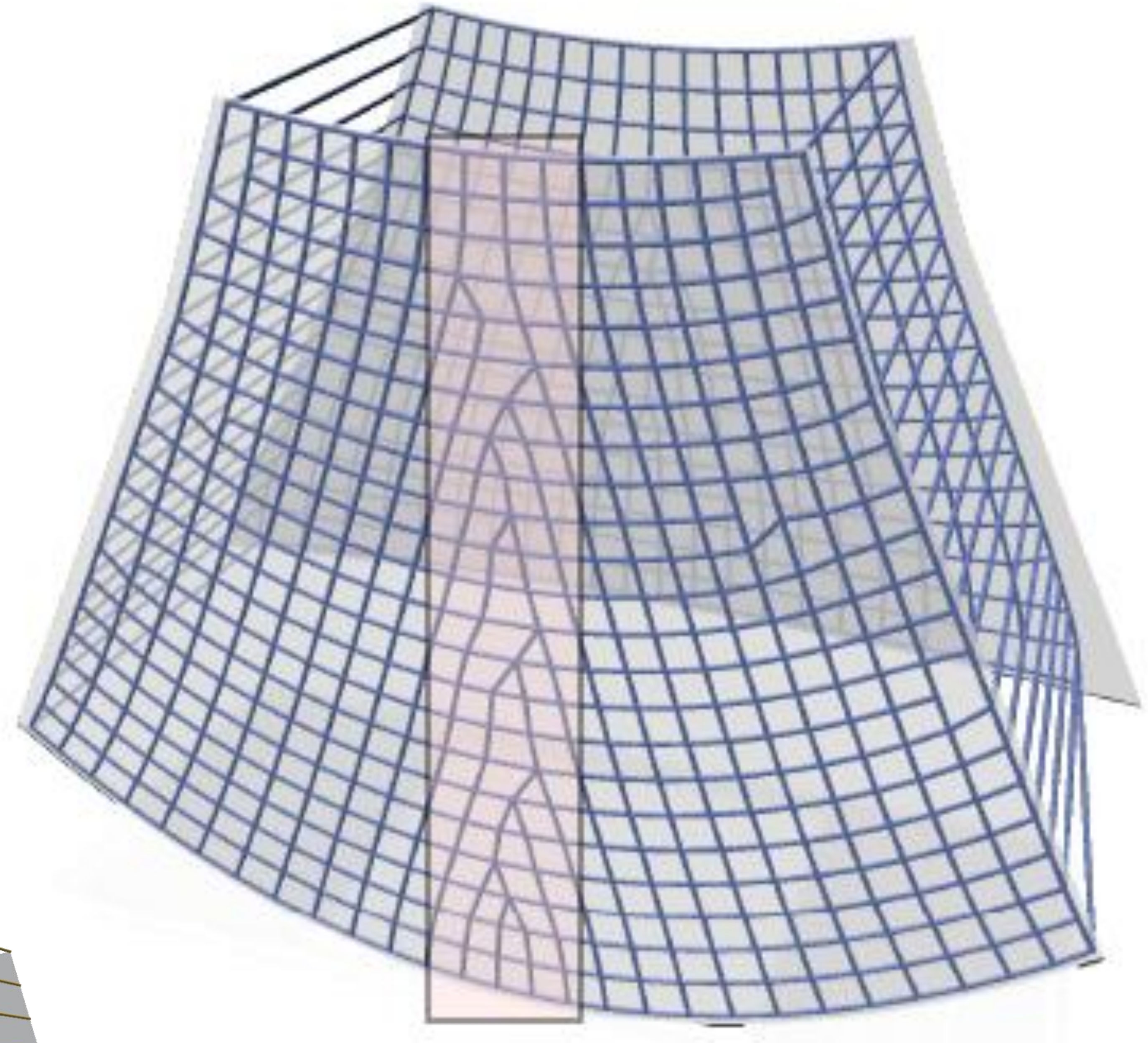
More intuitive user controls

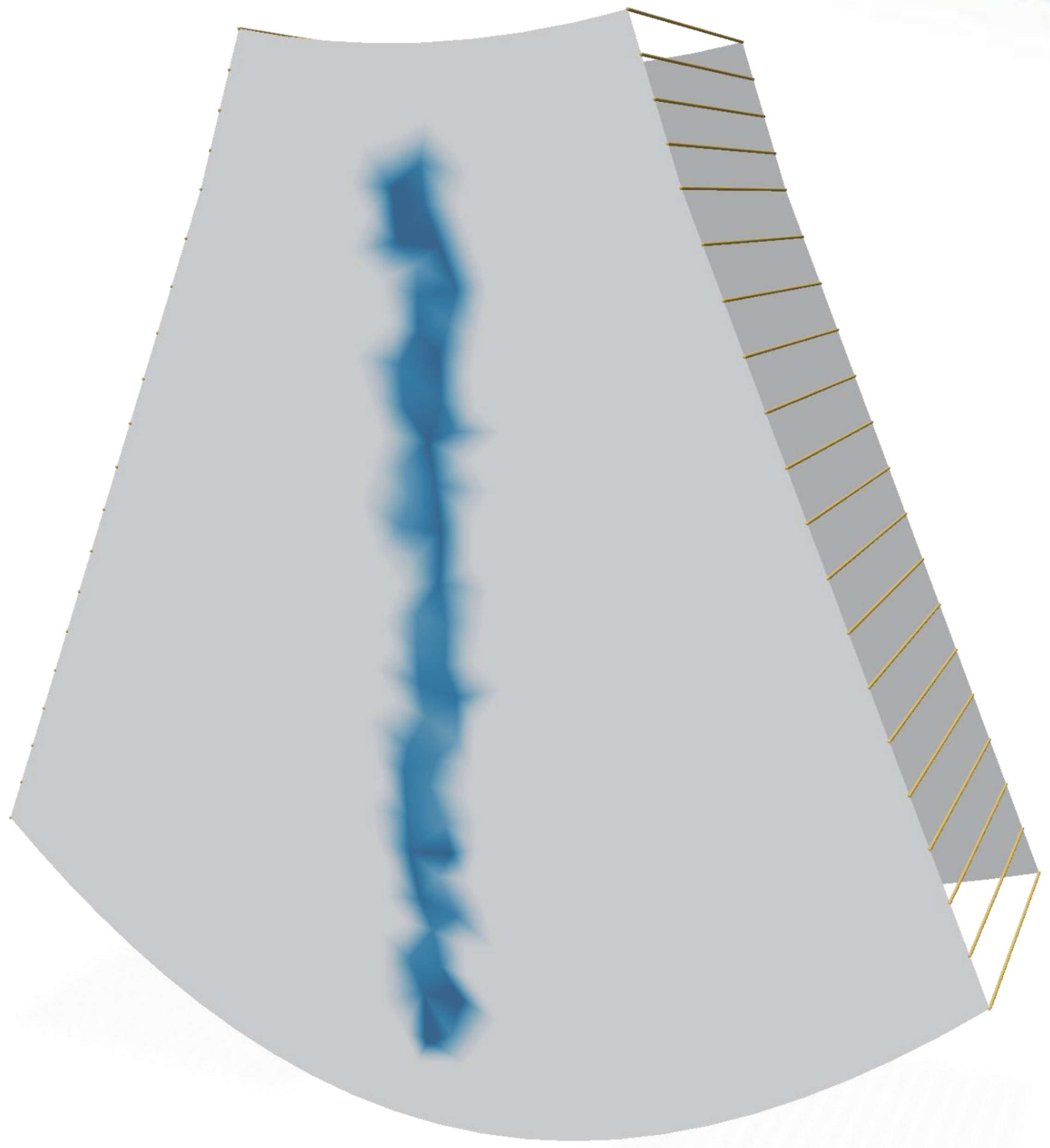
- Modification of the curl signal allows for more natural “apparent seam” placement, region masking, and other user constraints.



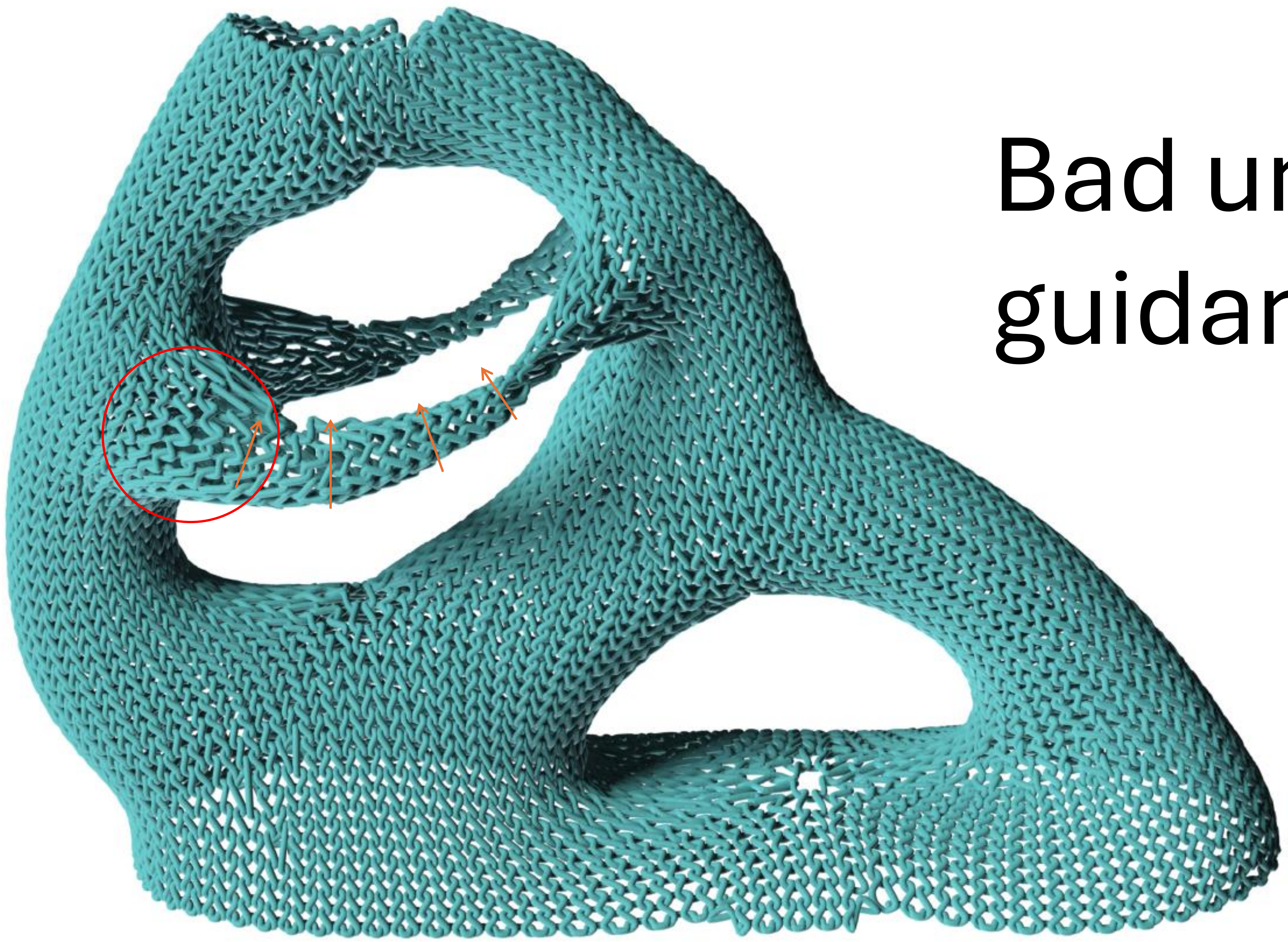
More intuitive user controls

- Modification of the curl signal allows for more natural "apparent seam" placement, region masking, and other user constraints.
- Prior work required specific face-by-face specification of singularity locations.





An even greater generalization?

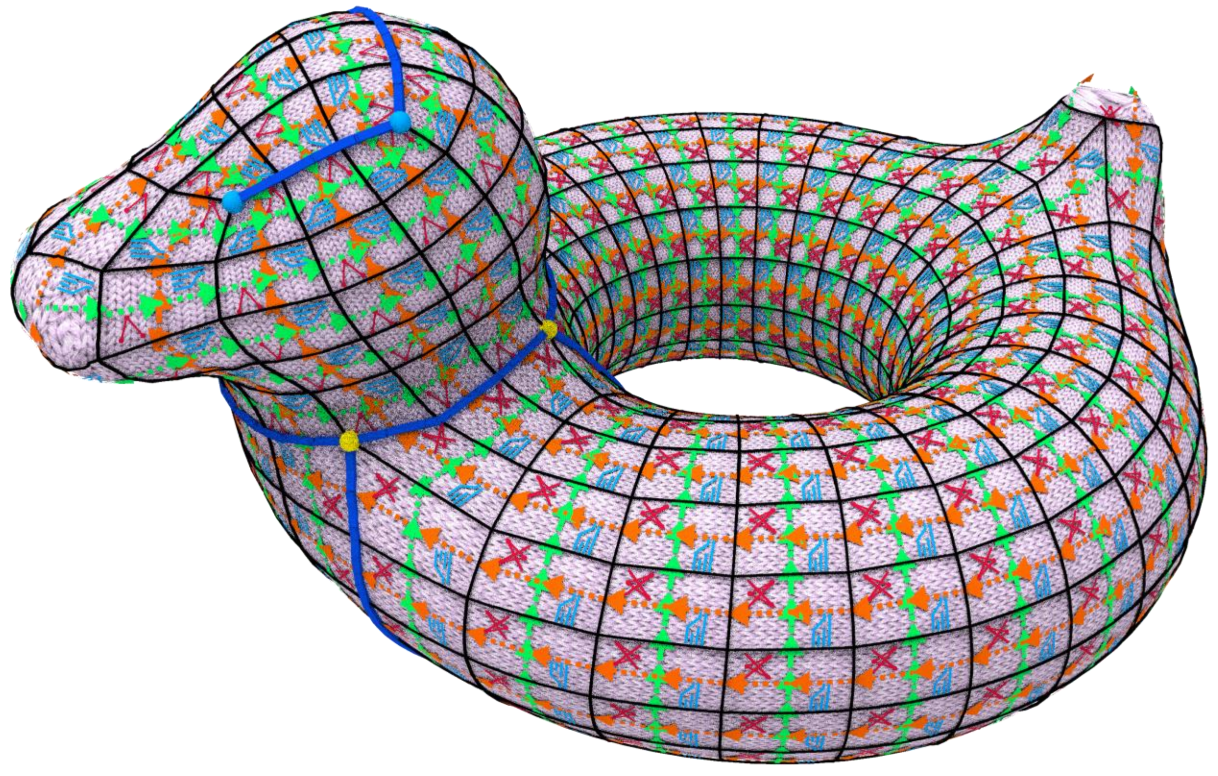


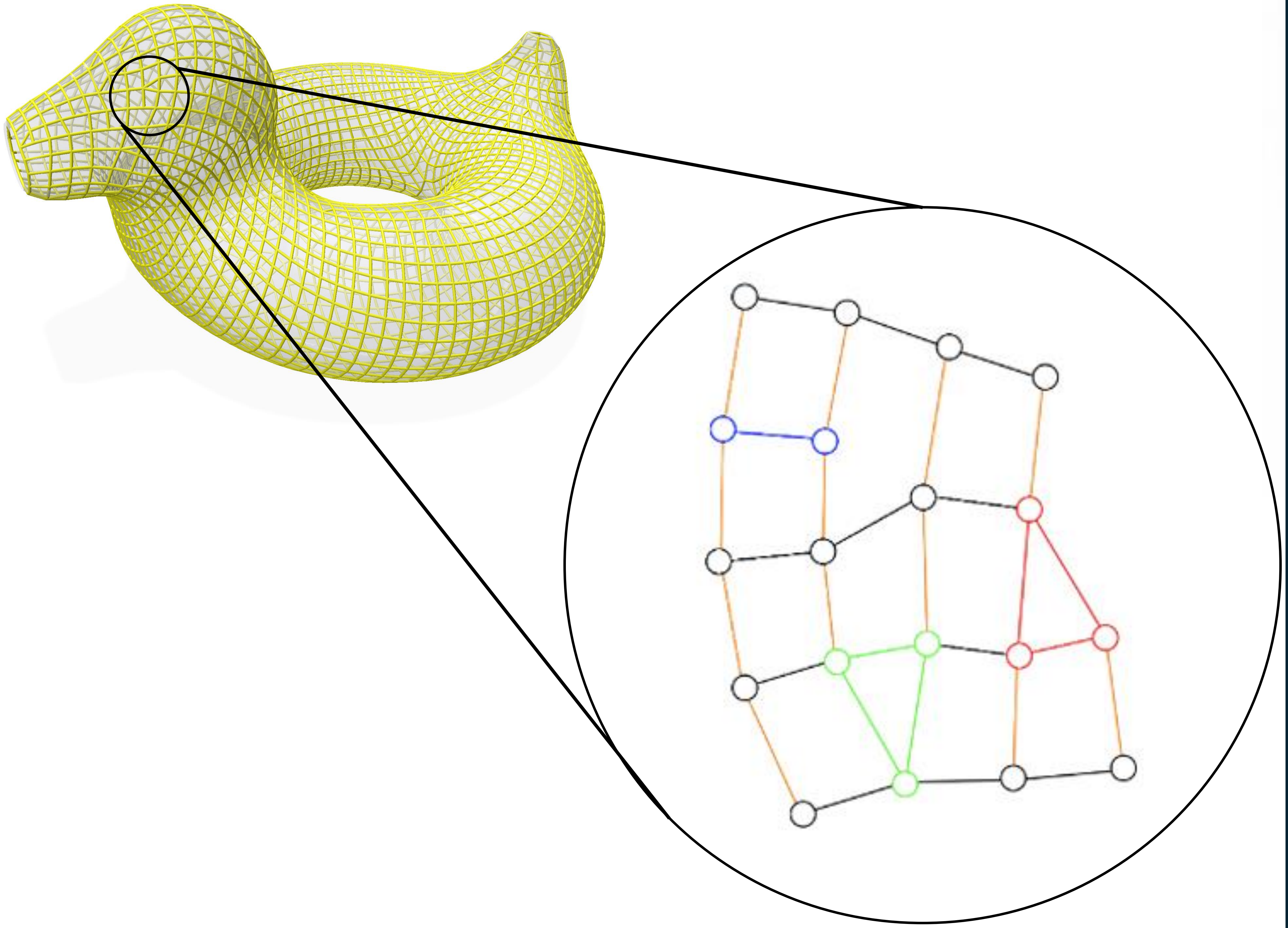
Bad underlying guidance direction

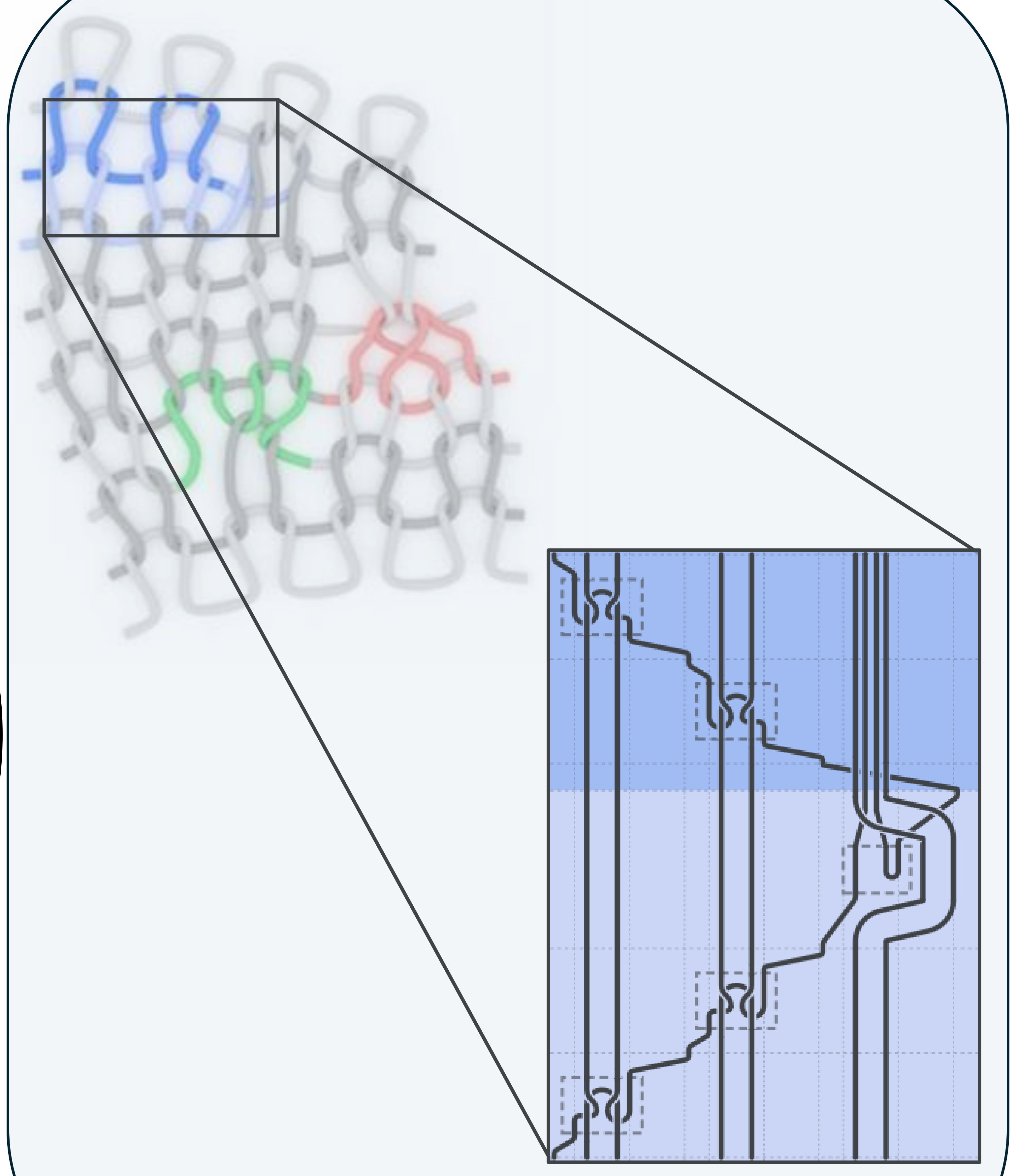
$$\overline{\nabla h} := \frac{\nabla h}{||\nabla h||}$$

Maybe allow yet another variable: the guiding vector field $\overline{\nabla h}$

Knitting Levels of Abstraction







Fabric **Stitch** **Yarn**

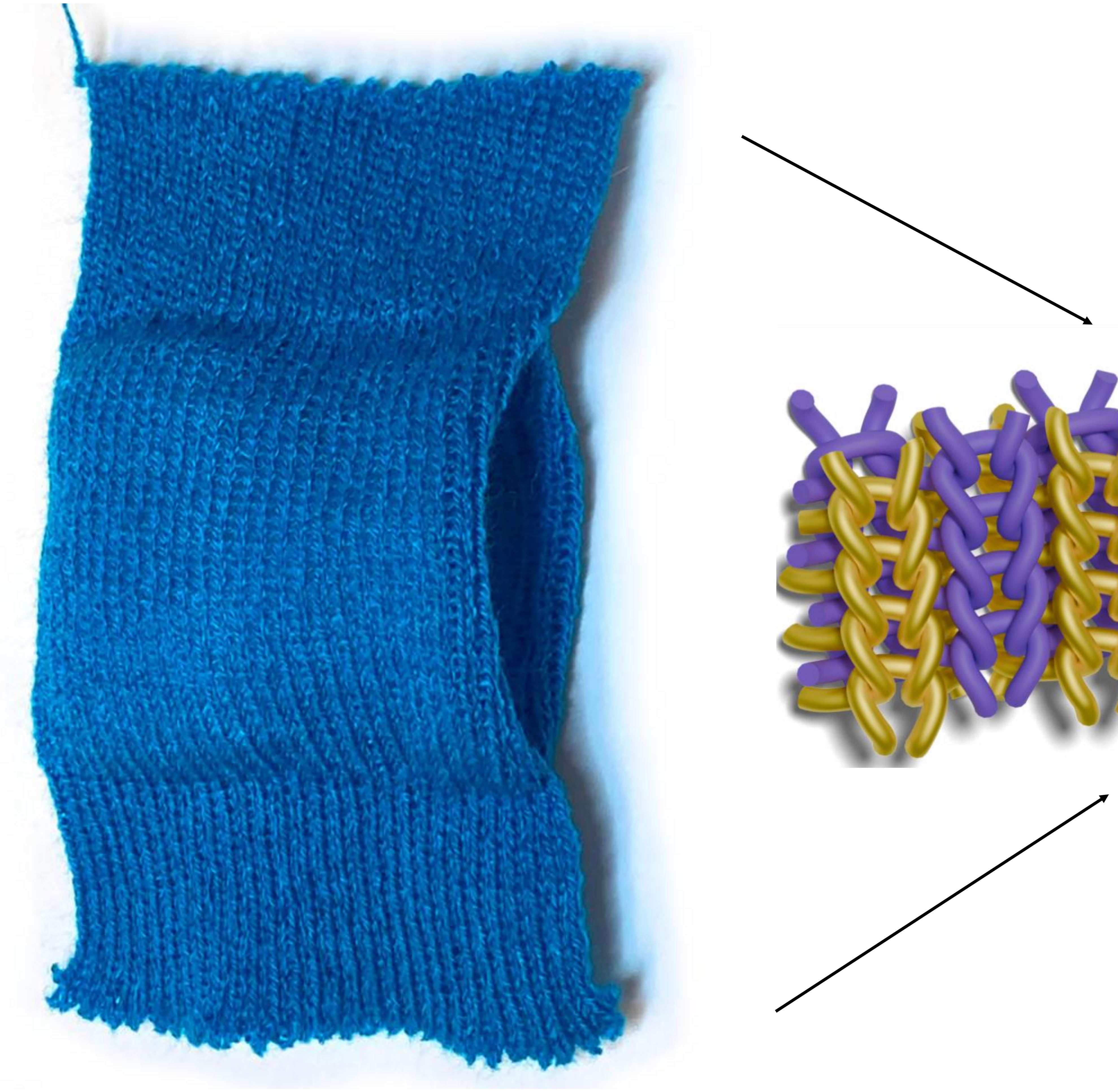
Non-Manifold Knits

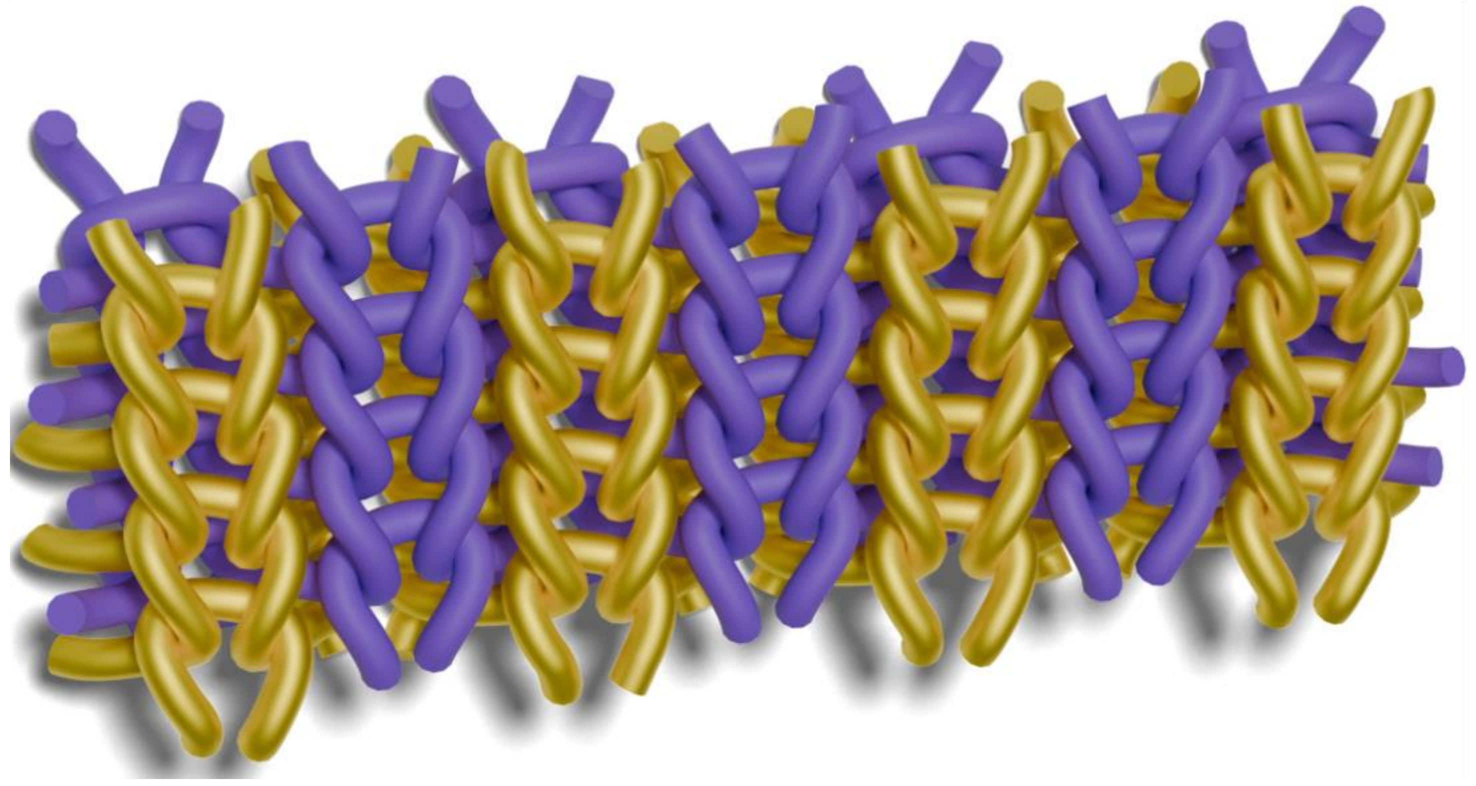
Albaugh et al. 2019

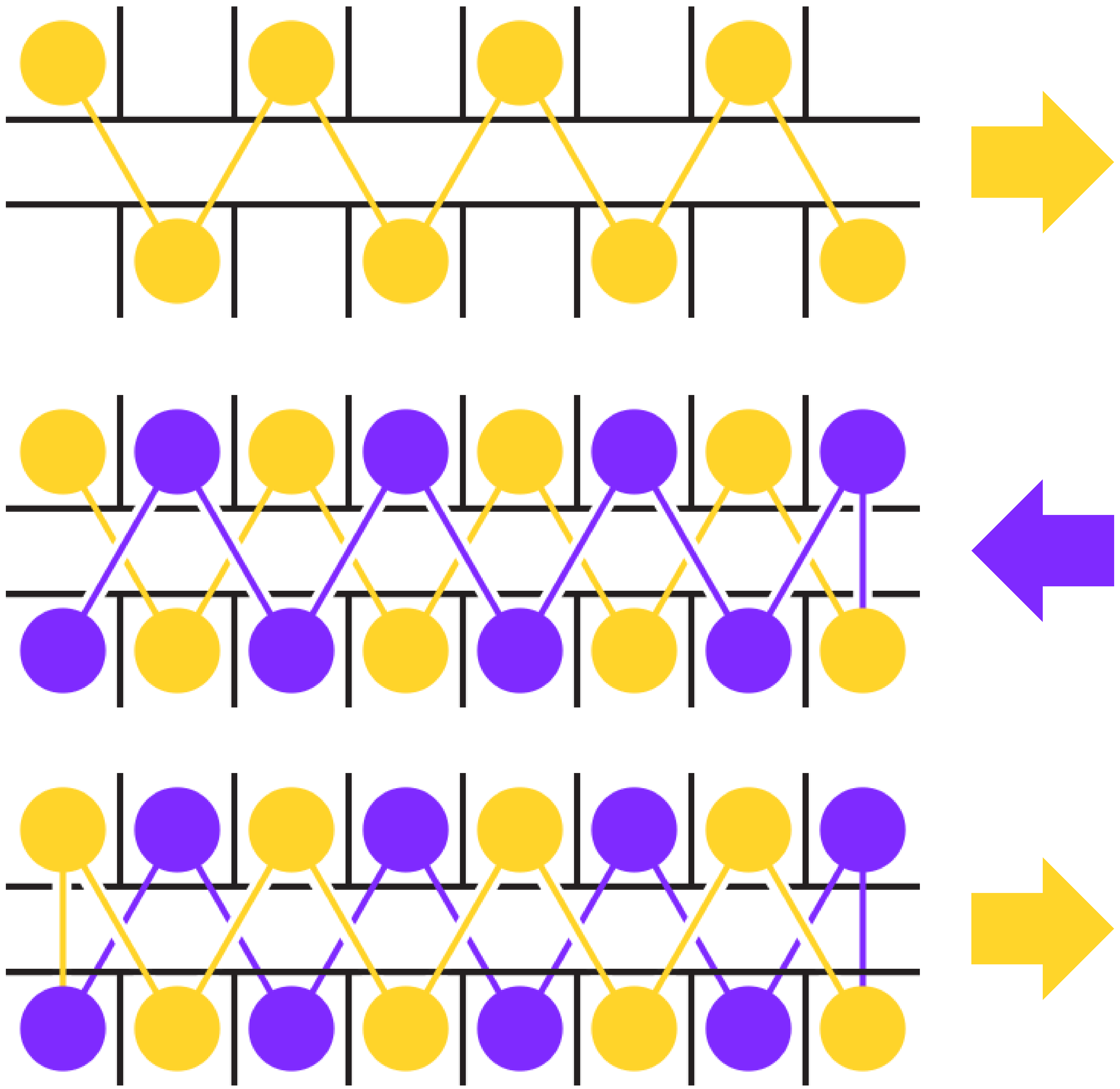
1. Locomotion on a rough surface

Kim et al. 2022

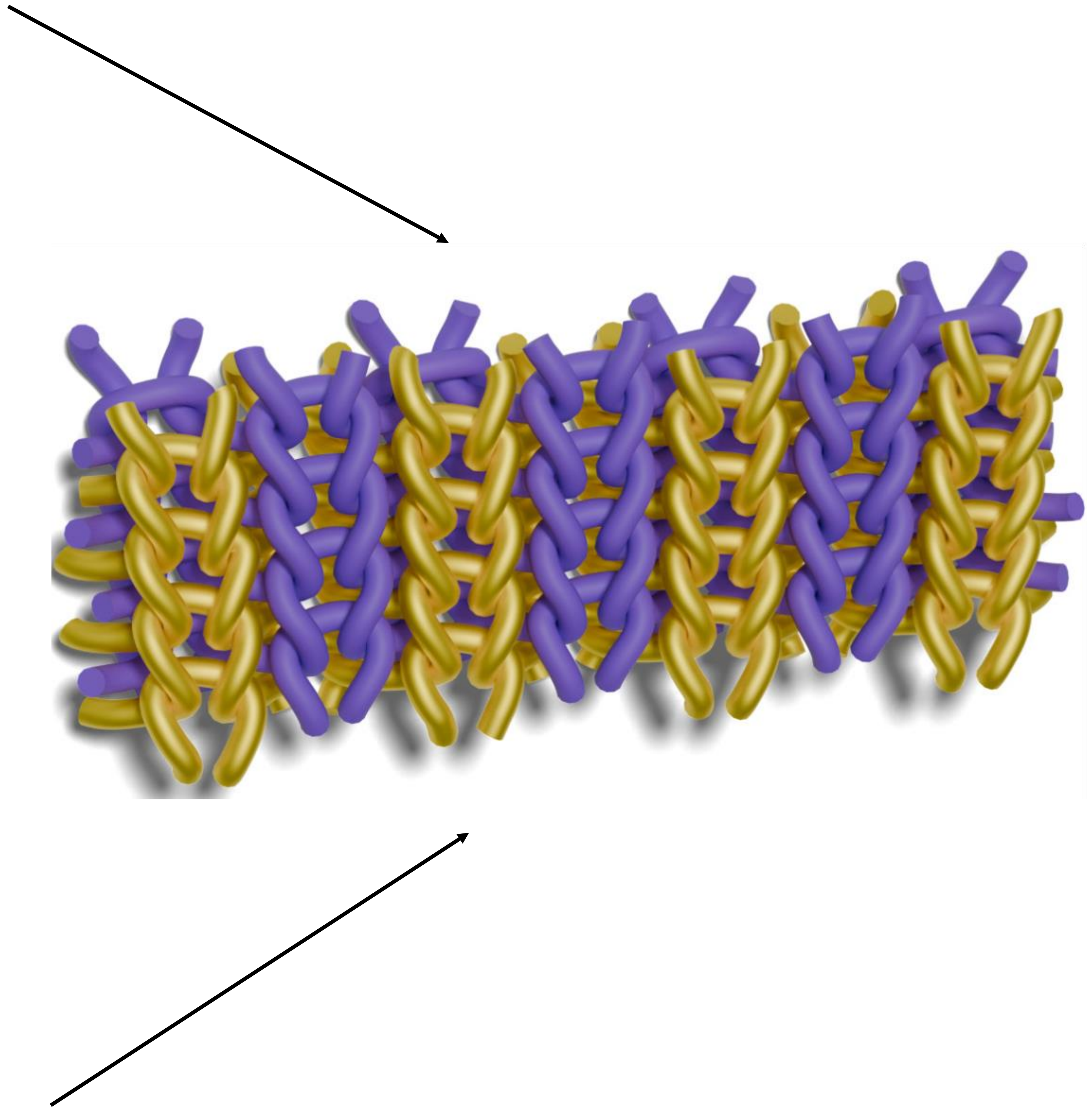
Non-Manifold Knits

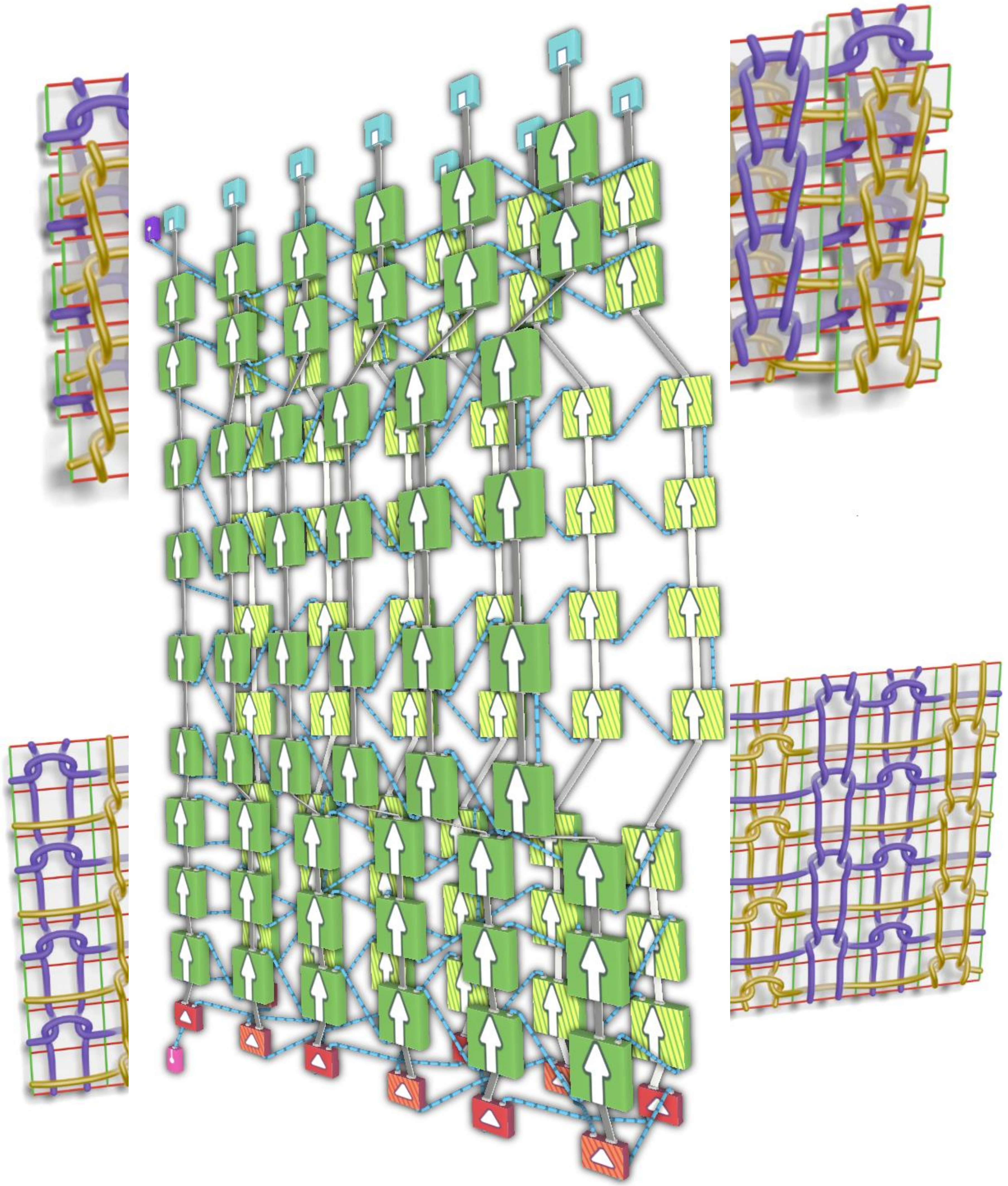




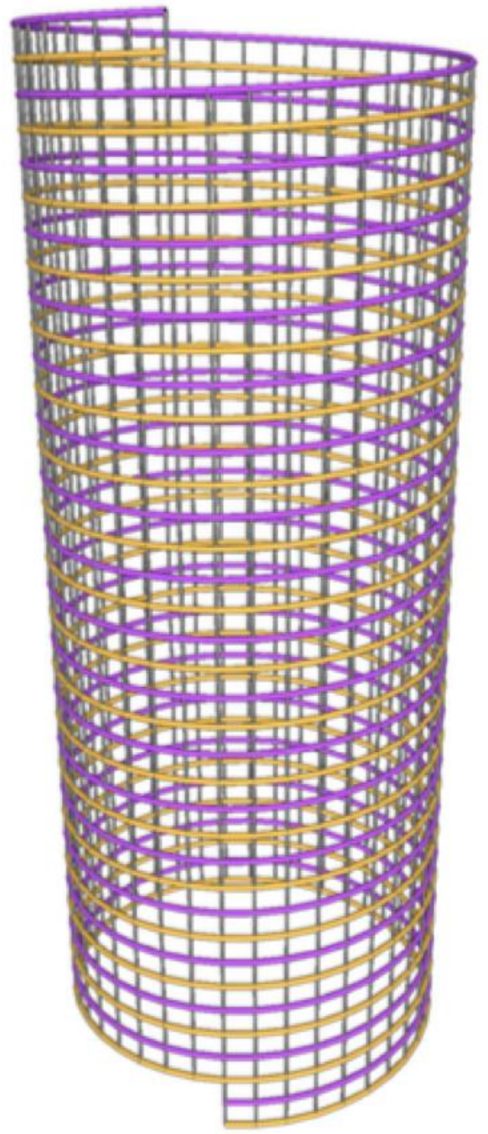


Non-Manifold Knits

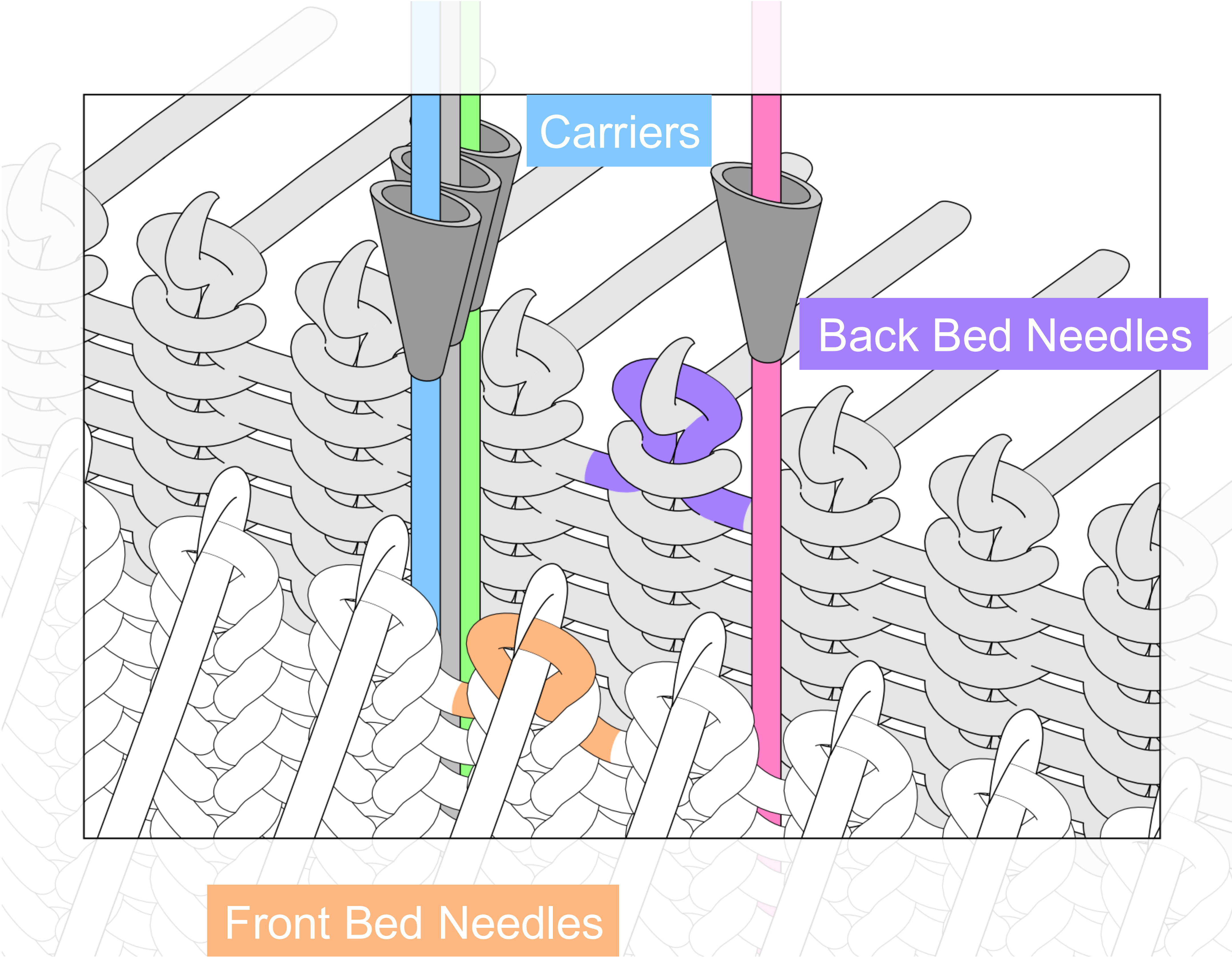




Knitting Machine Constraints

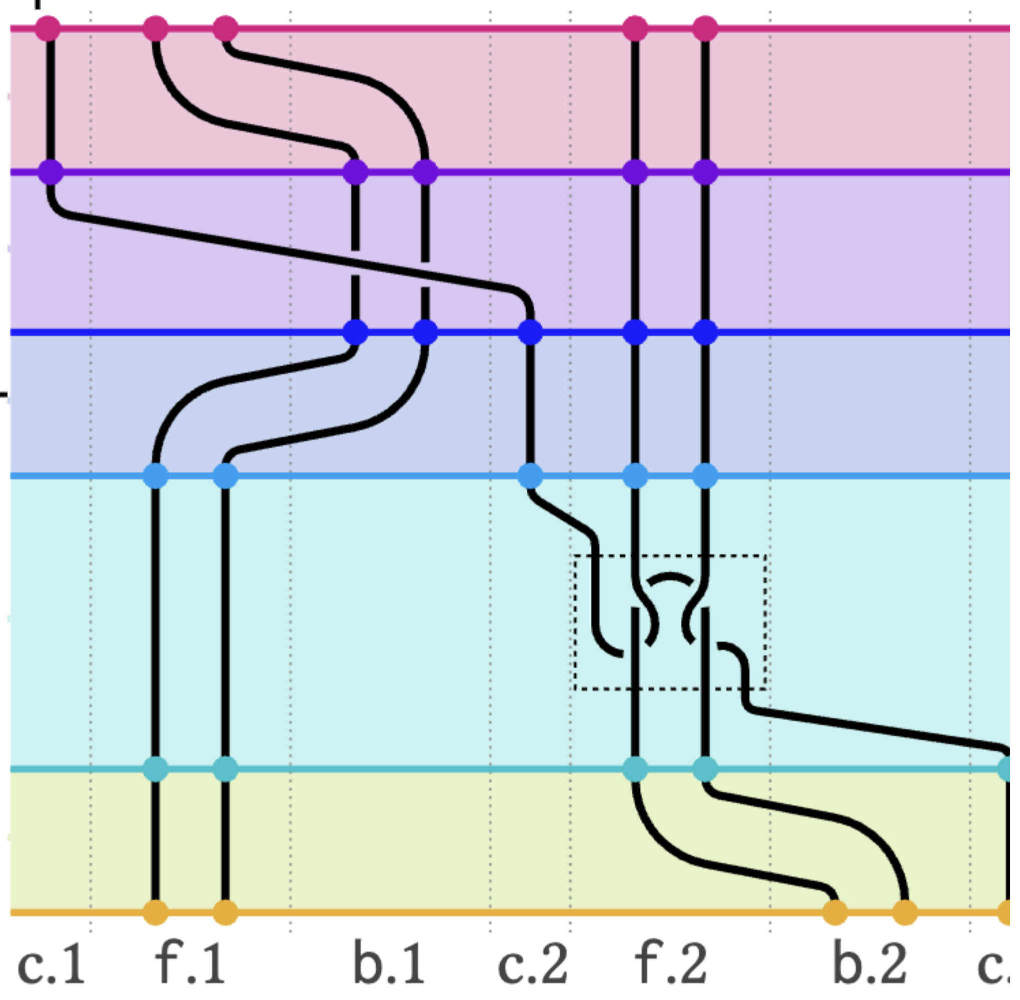


Narayanan et al. 2018



Topological Semantics for Knitting

```
xfer b.2 f.2;
knit - f.2 3.0 (2, 1.0);
xfer f.1 b.1;
miss - f.1 2;
xfer b.1 f.1;
```

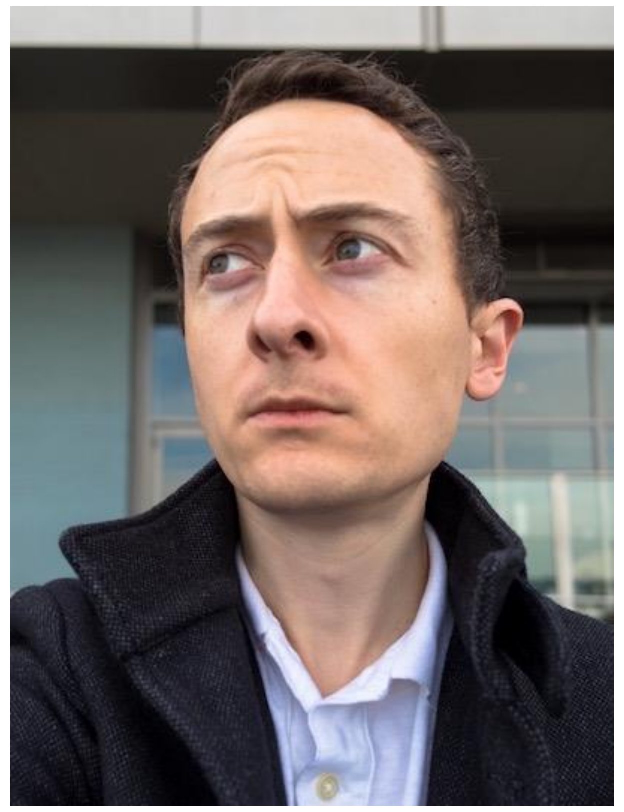


James McCann

Vidya Narayanan

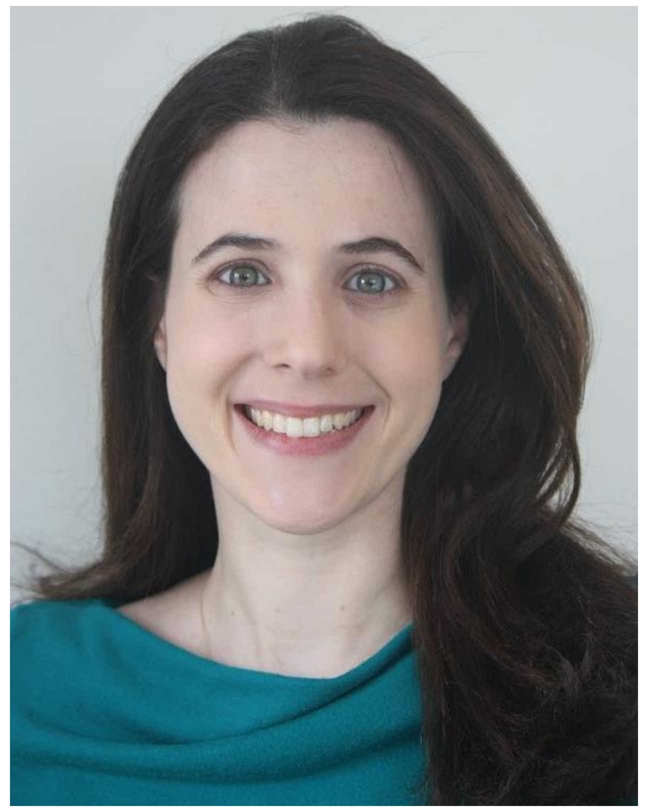
Tom Price

Gilbert Bernstein



Jonathan Ragan-Kelley

Yuka Ikarashi



Adriana Schulz

Nat Hurtig

Important Material Properties

High-level material properties
depend on low-level stitch topology

Stockinette stitch
Only front knits

Rib stitch
Alternating columns of front and back knits

Seed stitch
A checkerboard of front and back knits

Knit Transformations

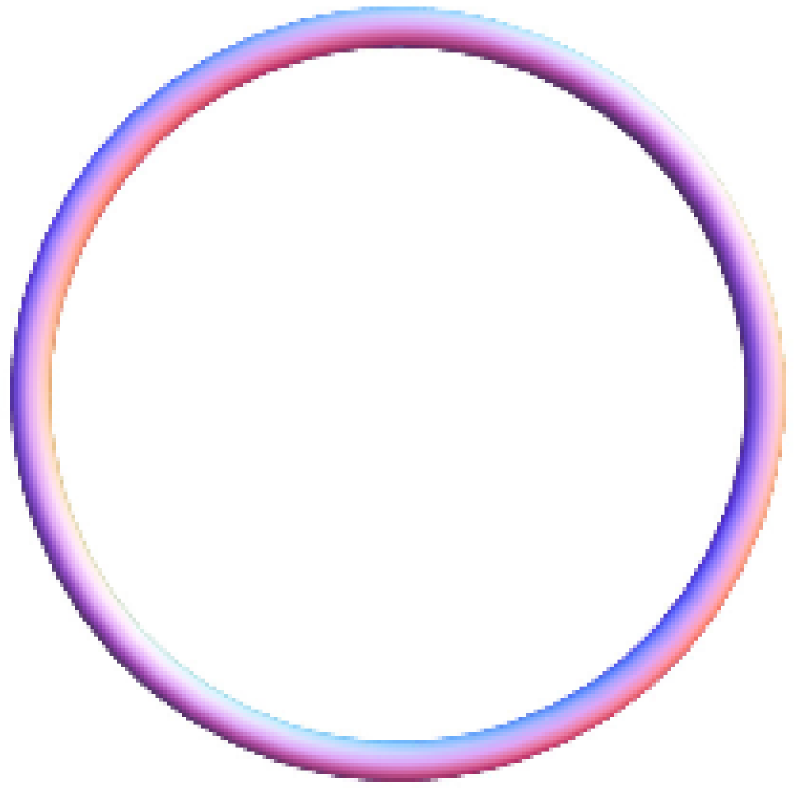
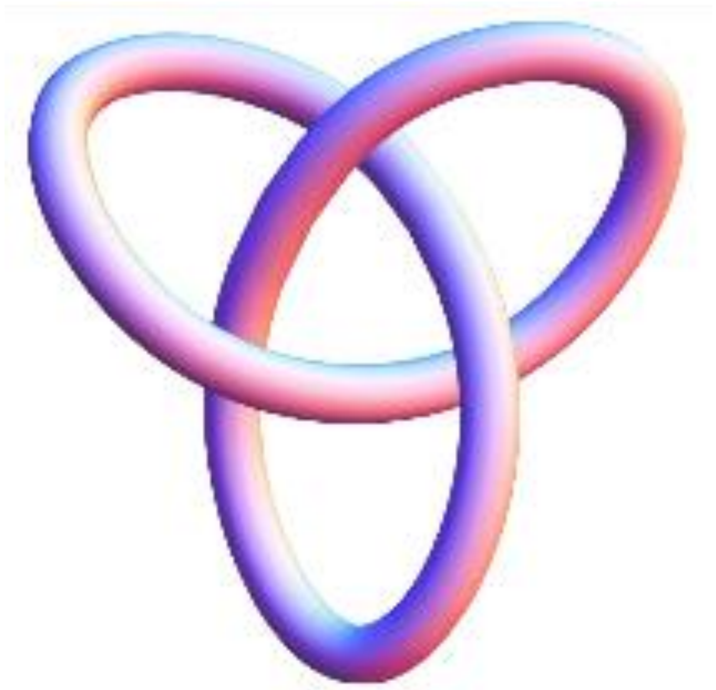
$\cong$

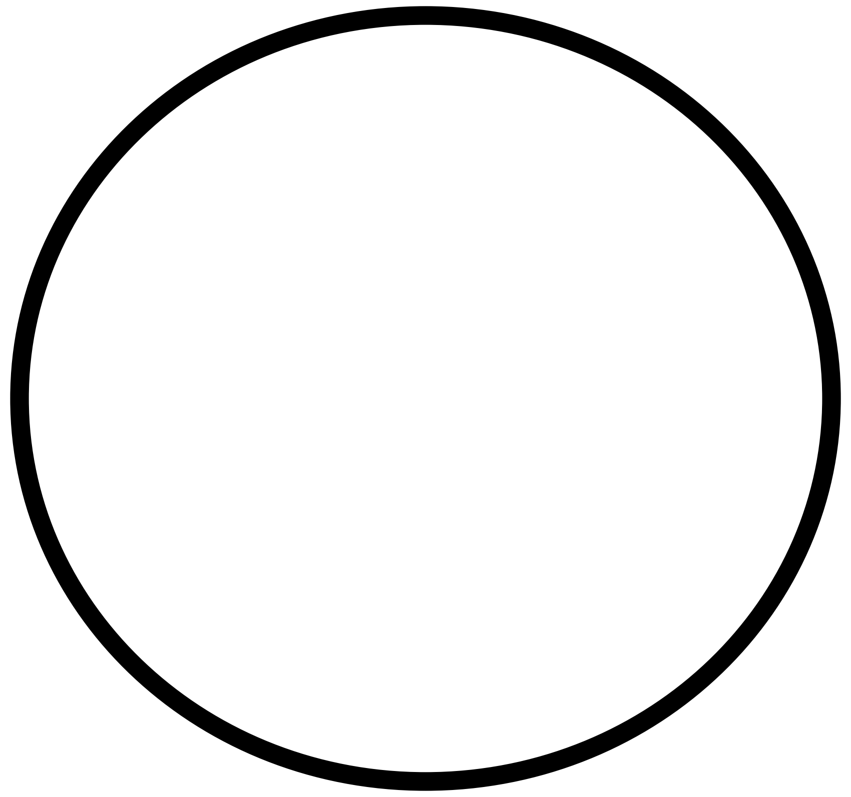
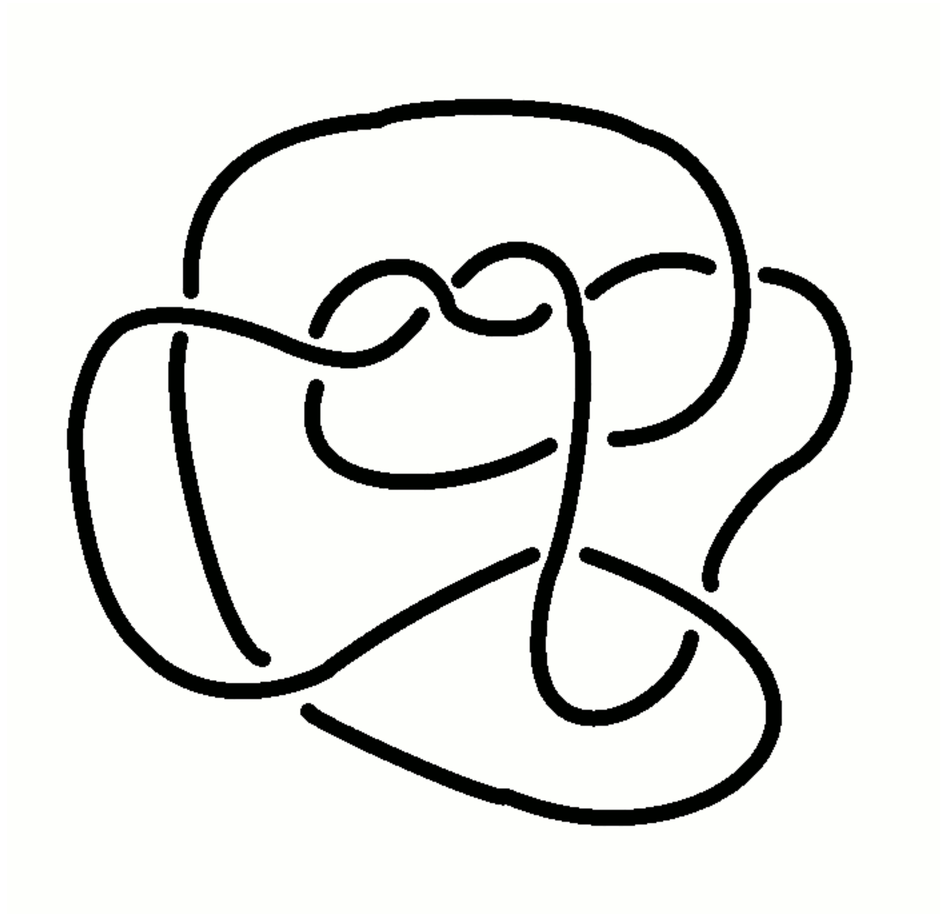
$\ncong$

Elastic: we can stretch and squash knitting

Some deformations are valid, but others are destructive

Knot Theory [Thompson, 1869]

 $\ncong$ 

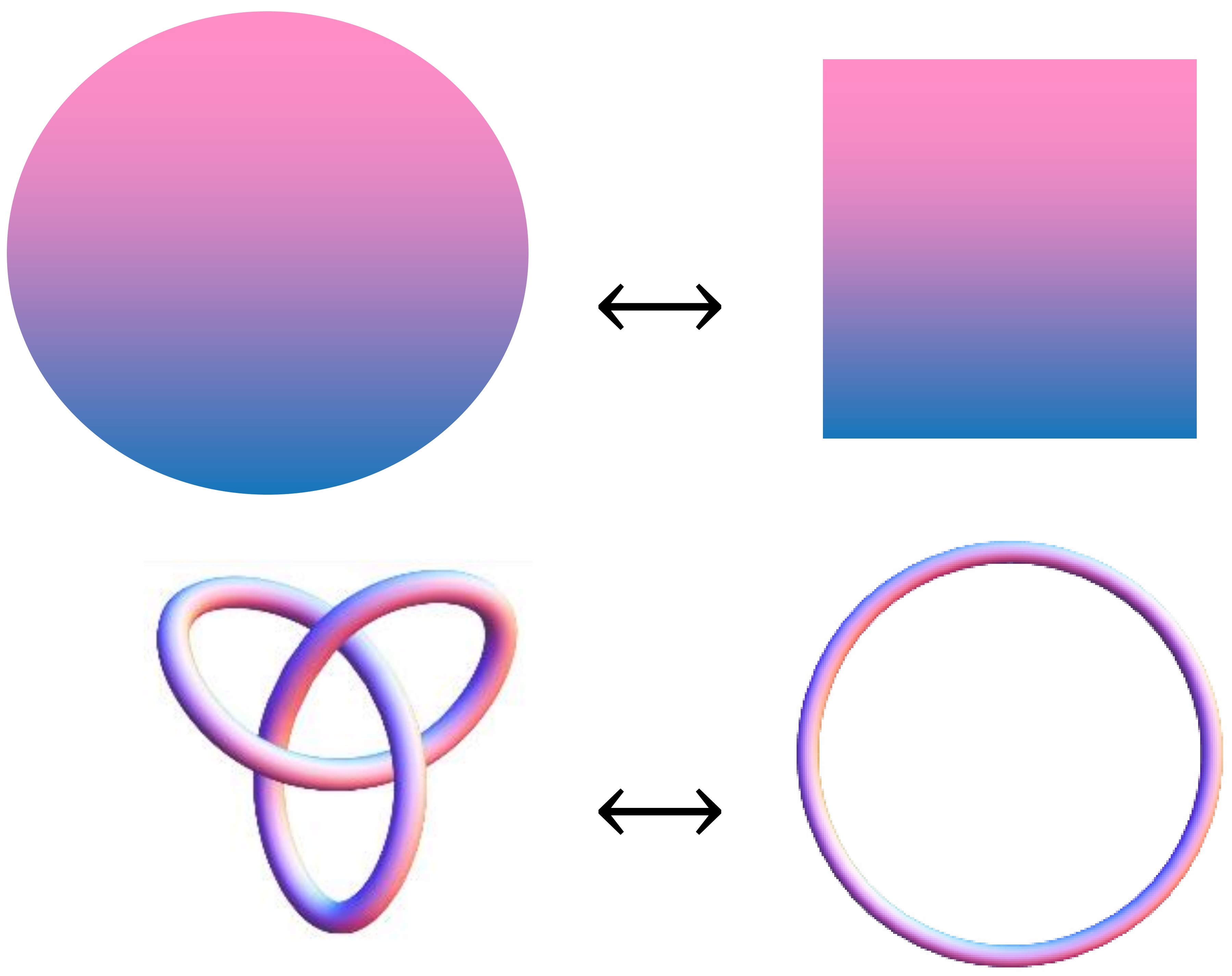
 $\cong$ 

Study of loops embedded in $\mathbb{R}^3$

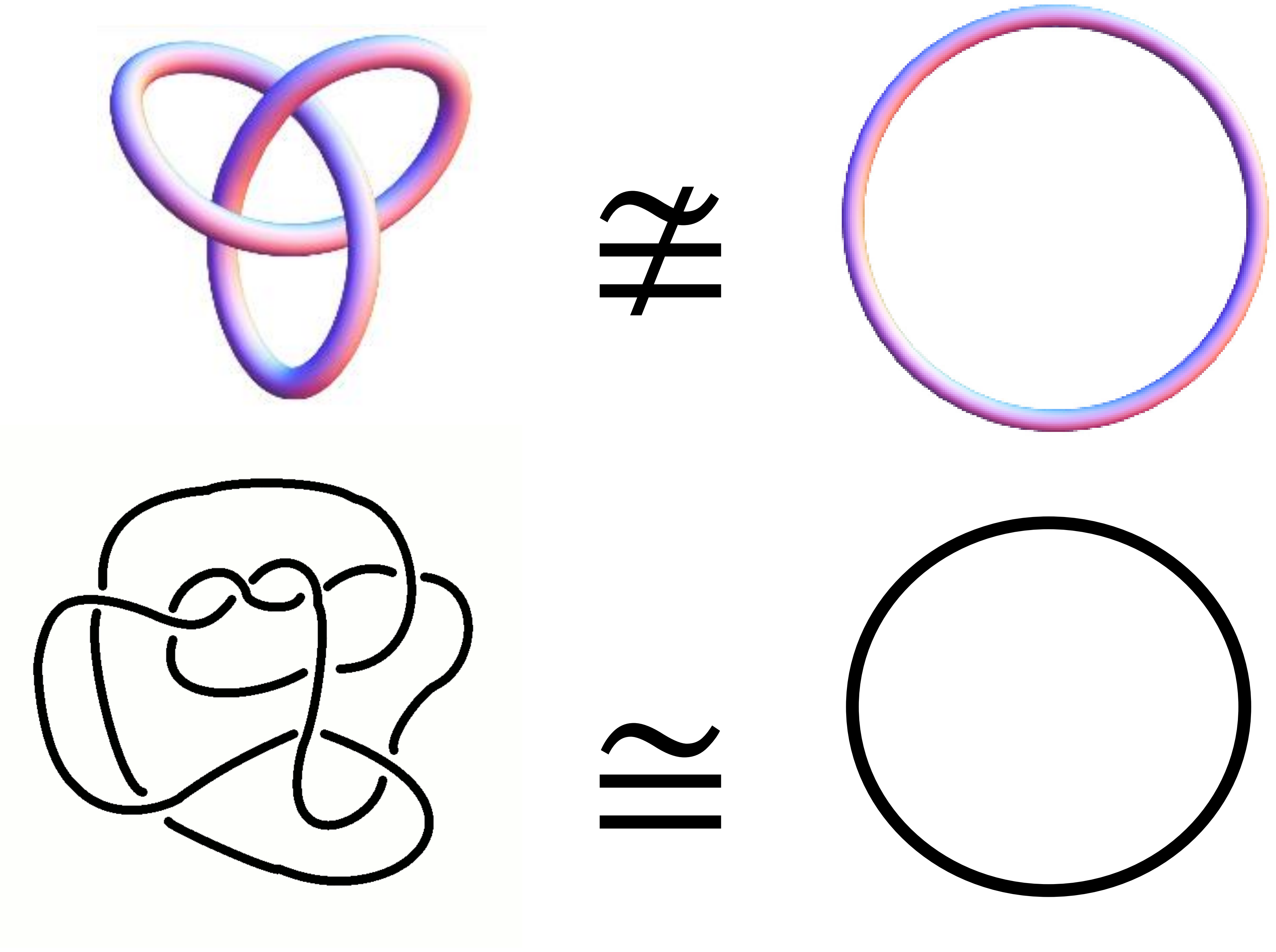
Two knots are equivalent under ambient isotopy

Problem: Loops have no loose ends

Topological Mappings and “Deformation”

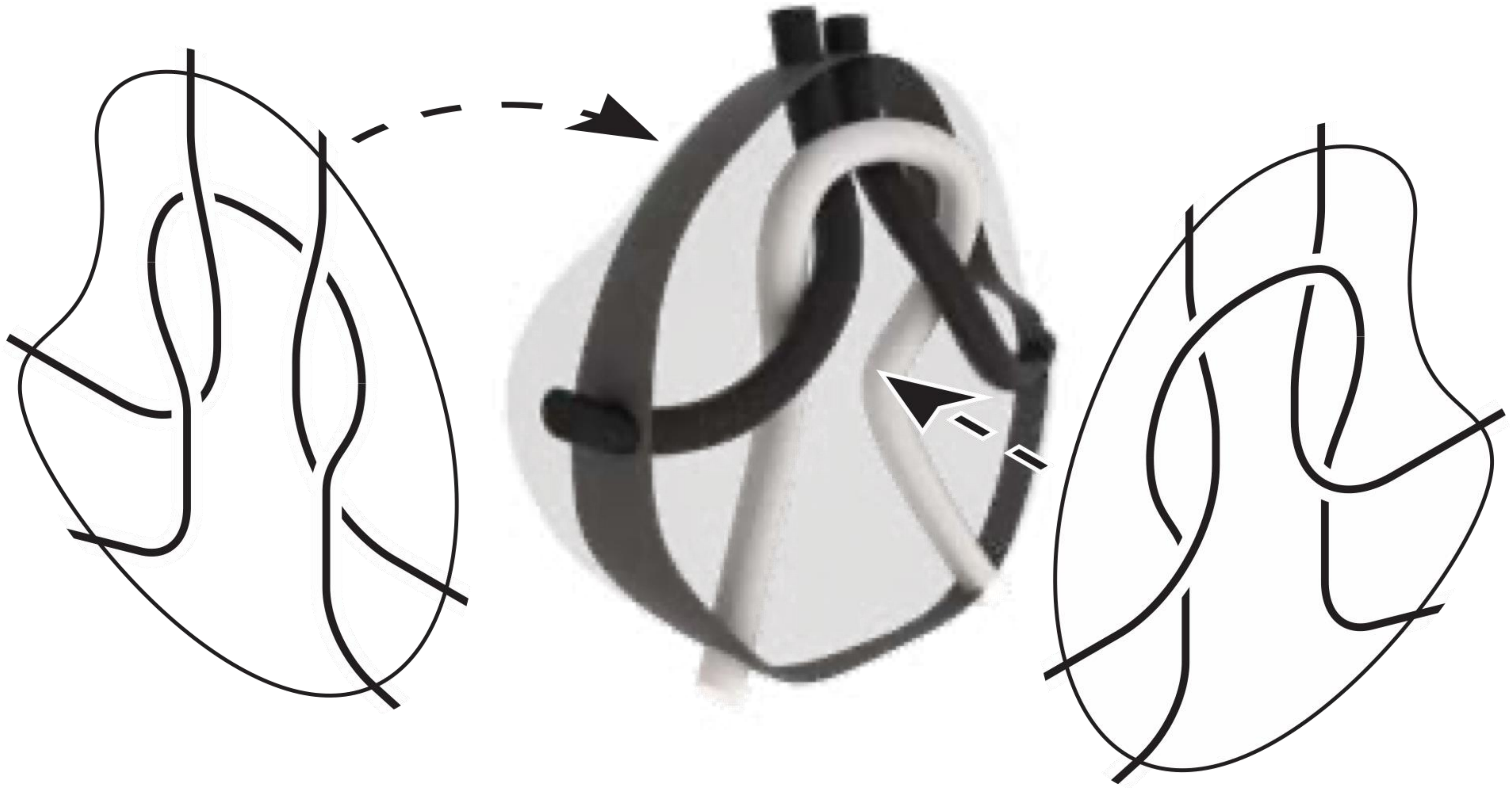


Homeomorphic: given objects A and B, there’s a continuous, invertible mapping between them



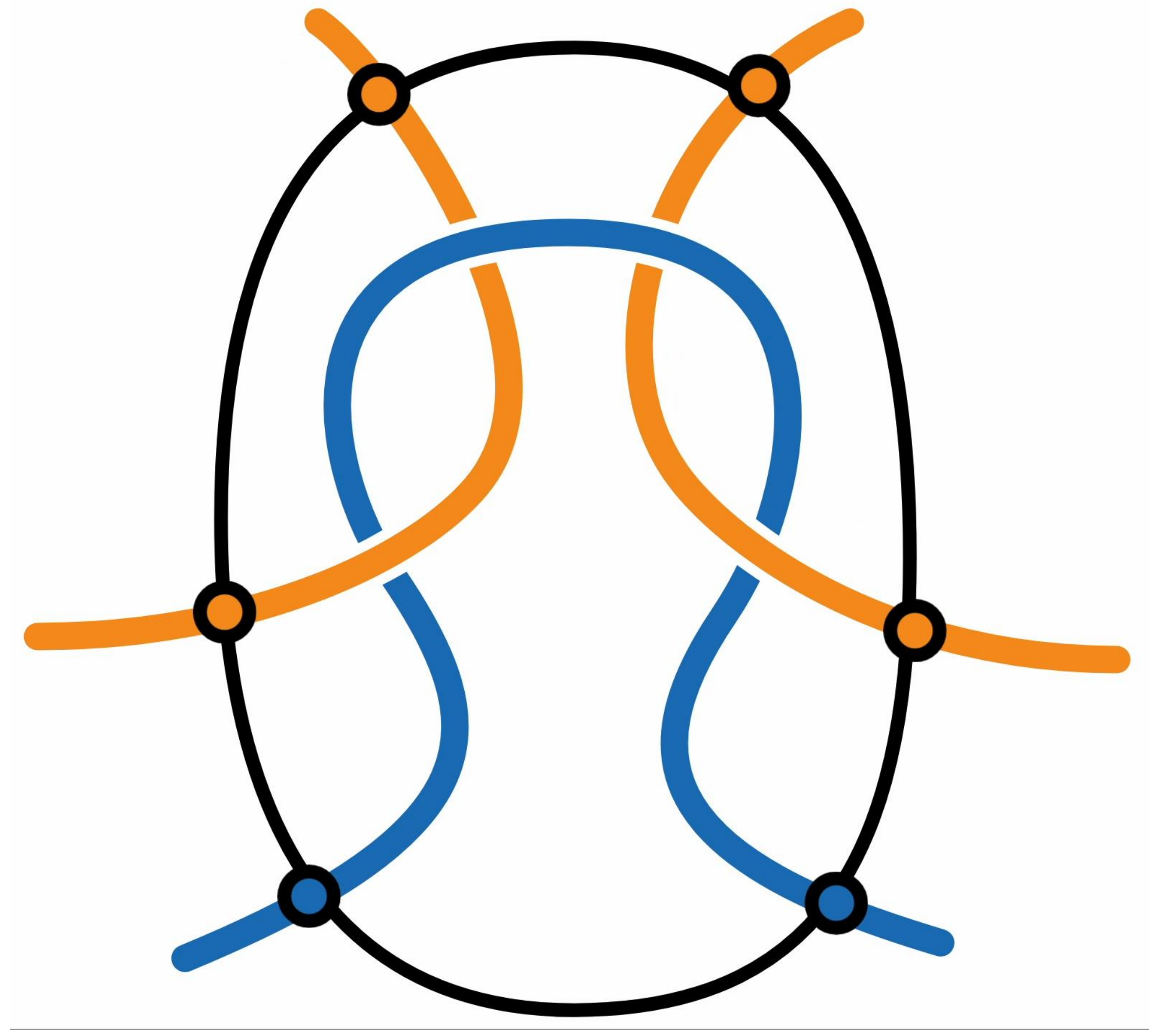
Ambient Isotopic: objects A and B are embedded in ambient space N_A and N_B. There’s a sequence of homeomorphisms from N_A to N_B that takes A to B

Tangles [Conway, 1970]

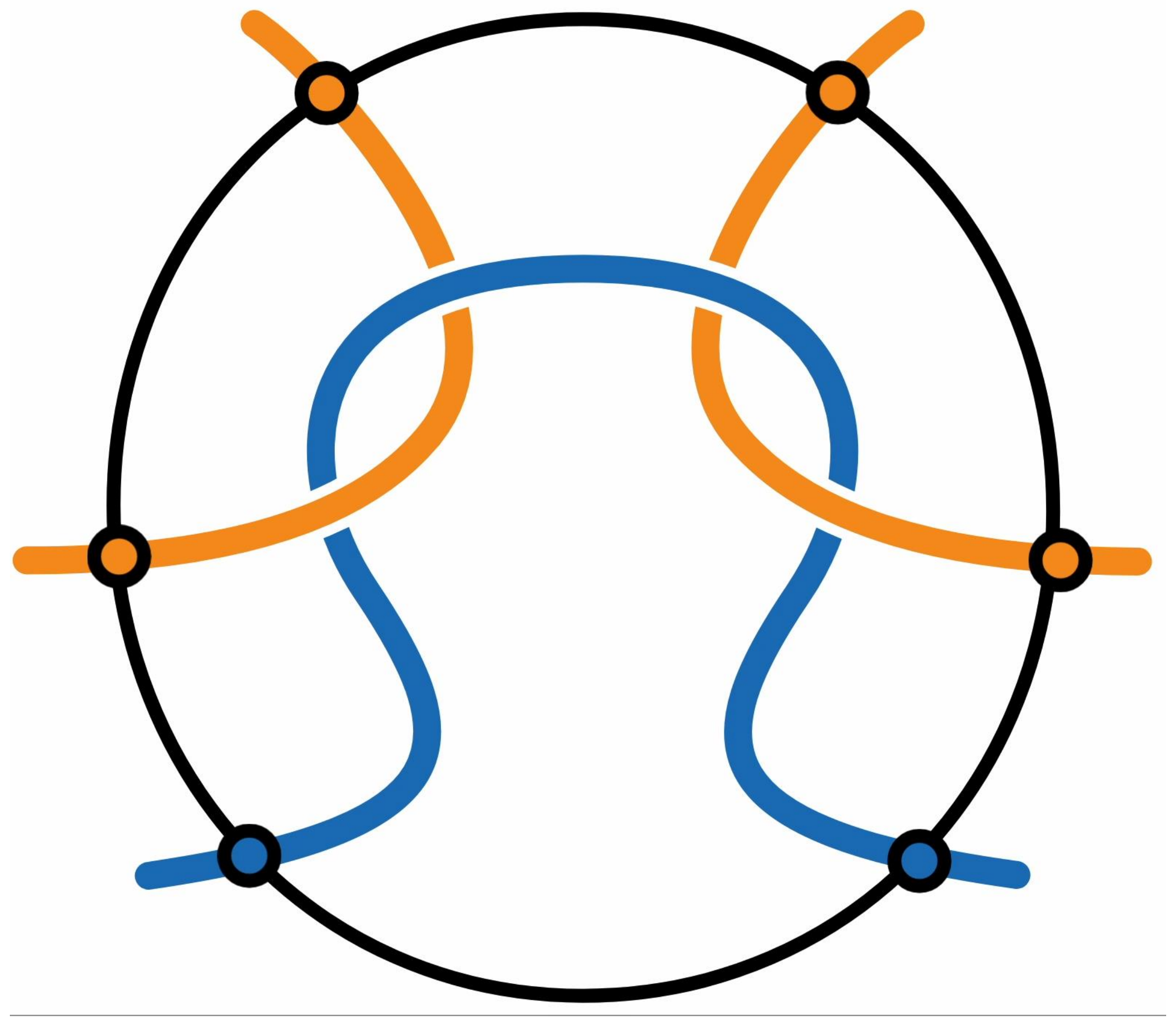


Take a portion of a knot, and embed it in a ball instead

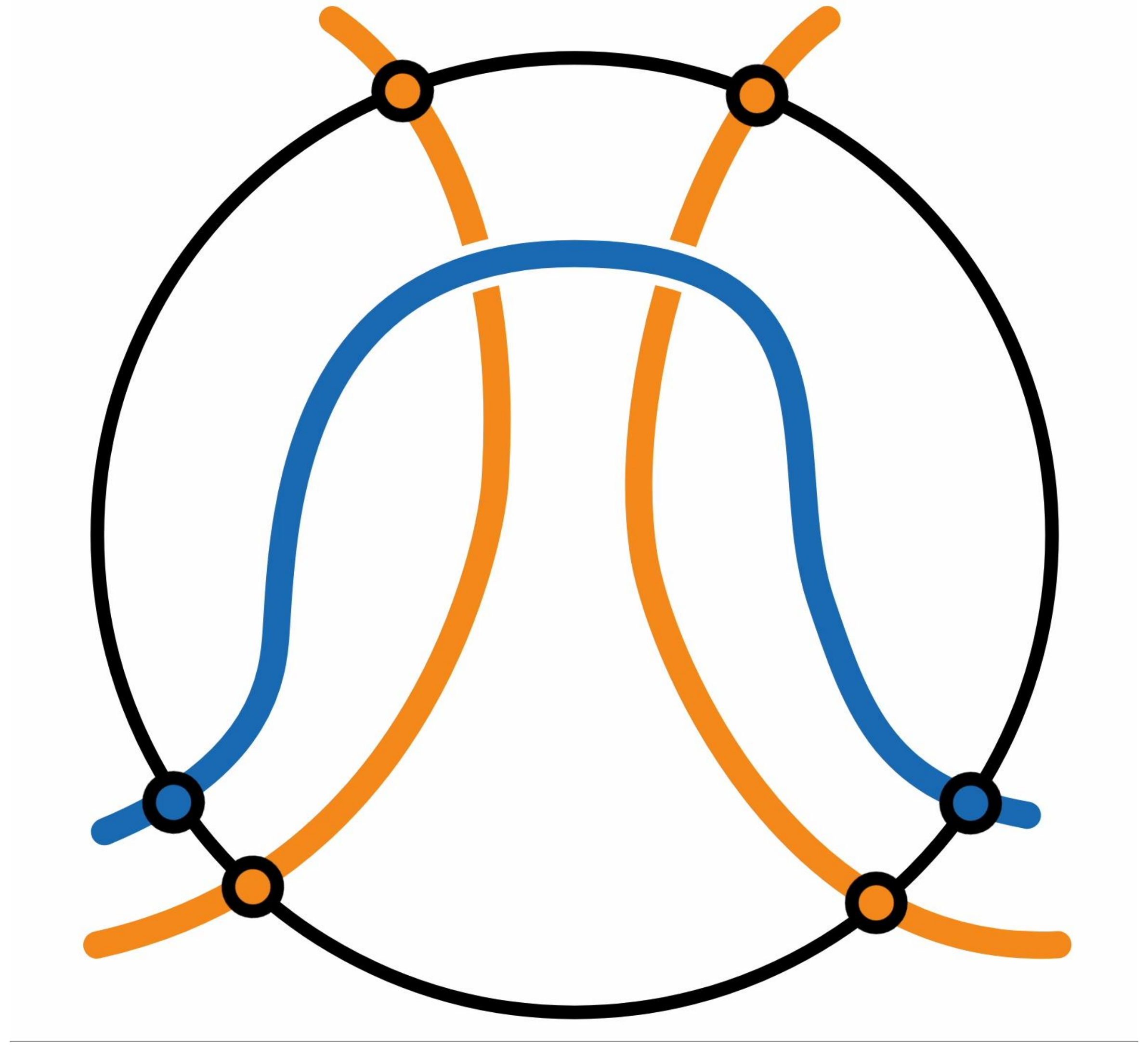
Equivalence on Tangles



$\cong$



$\not\cong$



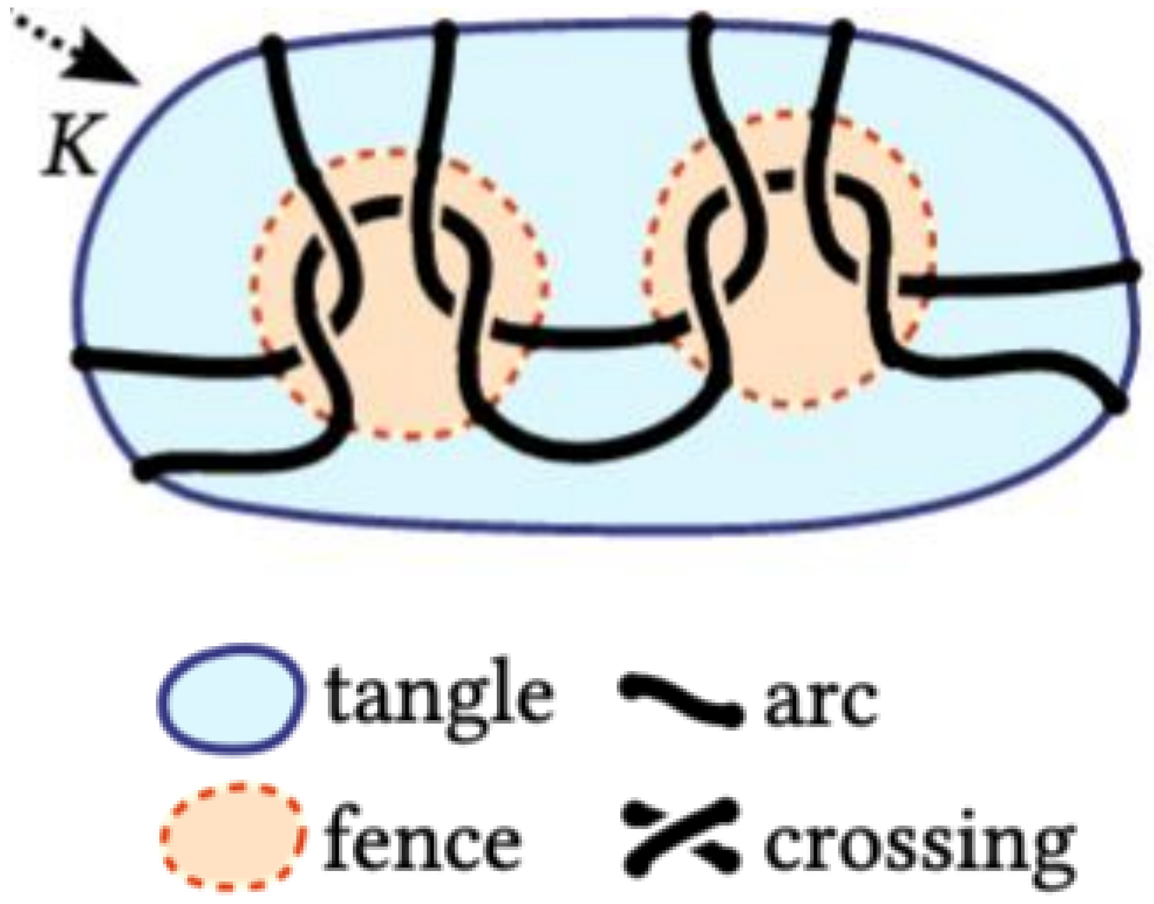
Fine to deform embedding space

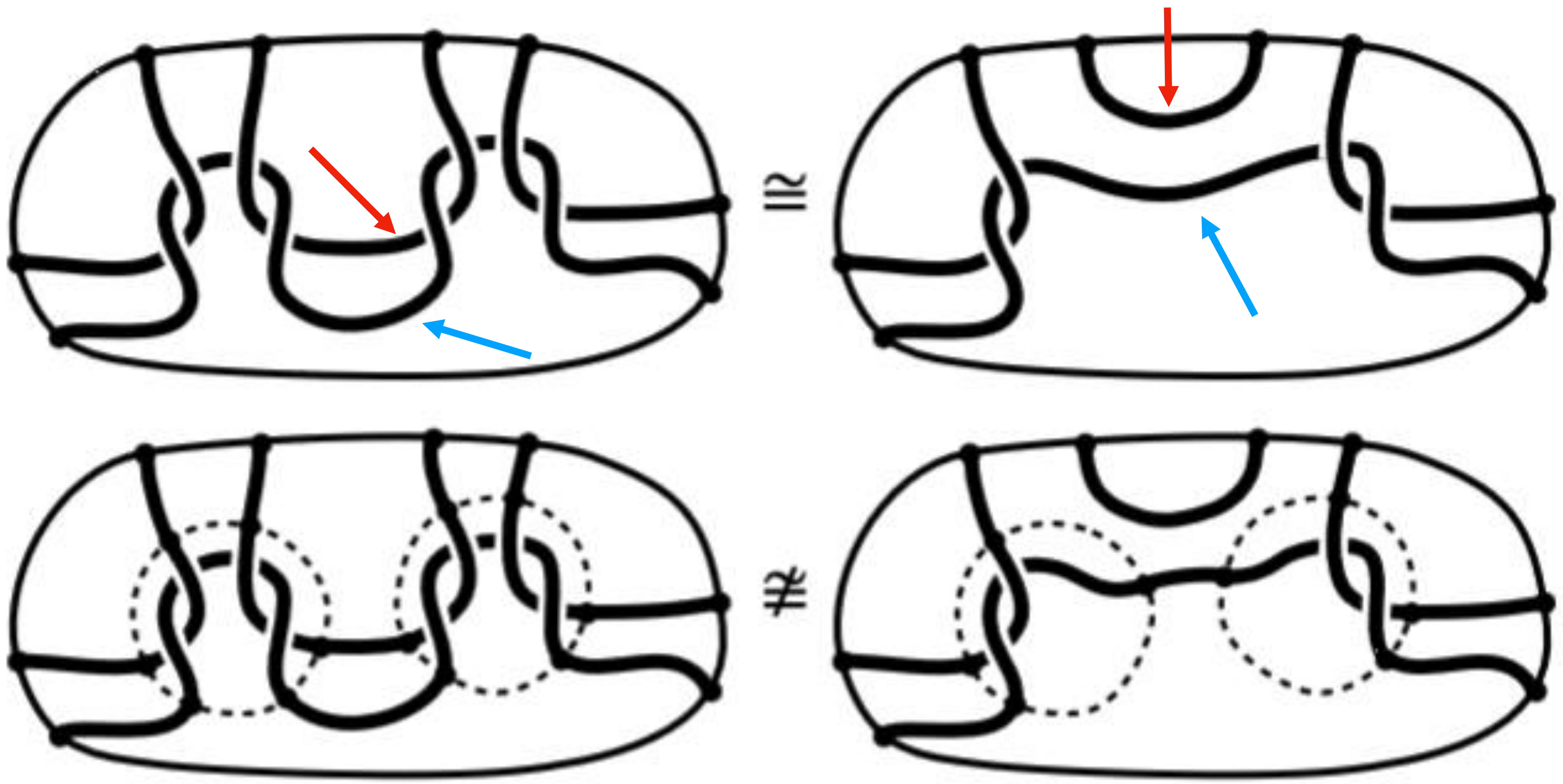
Order of endpoints on boundary must match!

Problem: Tangles are both over and under constrained

Fenced Tangles

SIGGRAPH ‘23

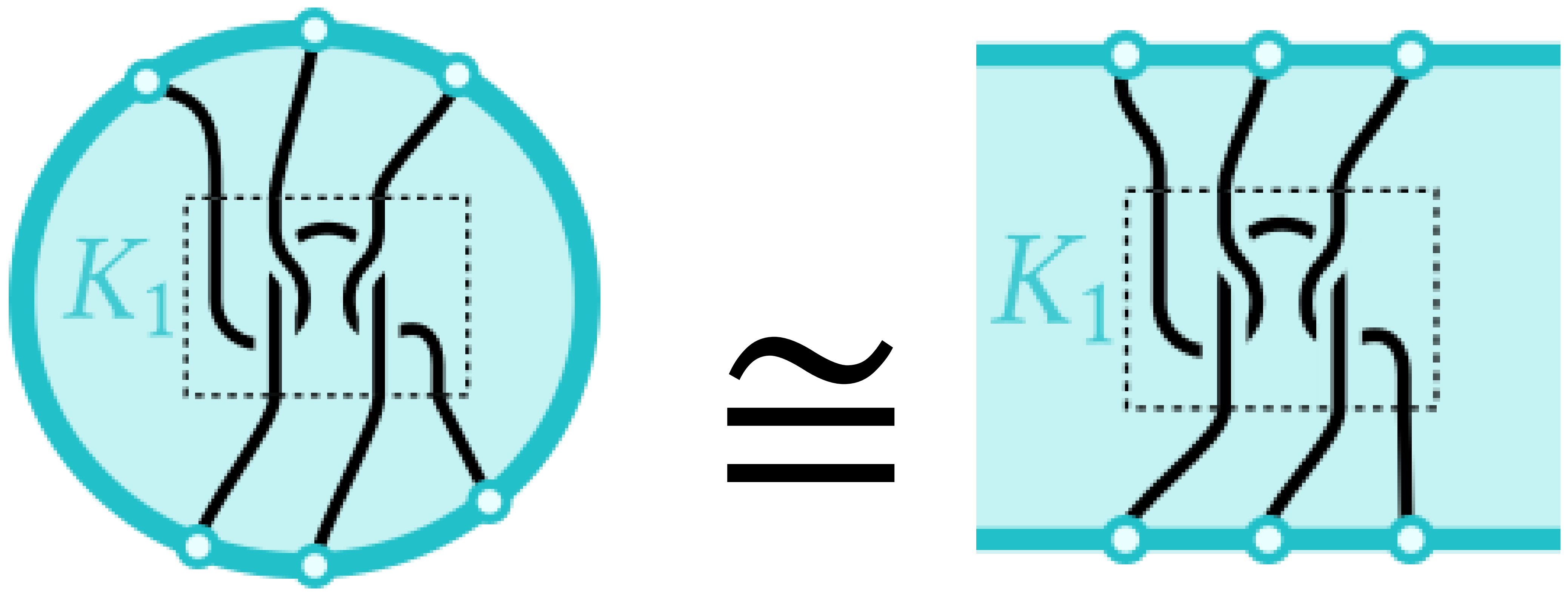


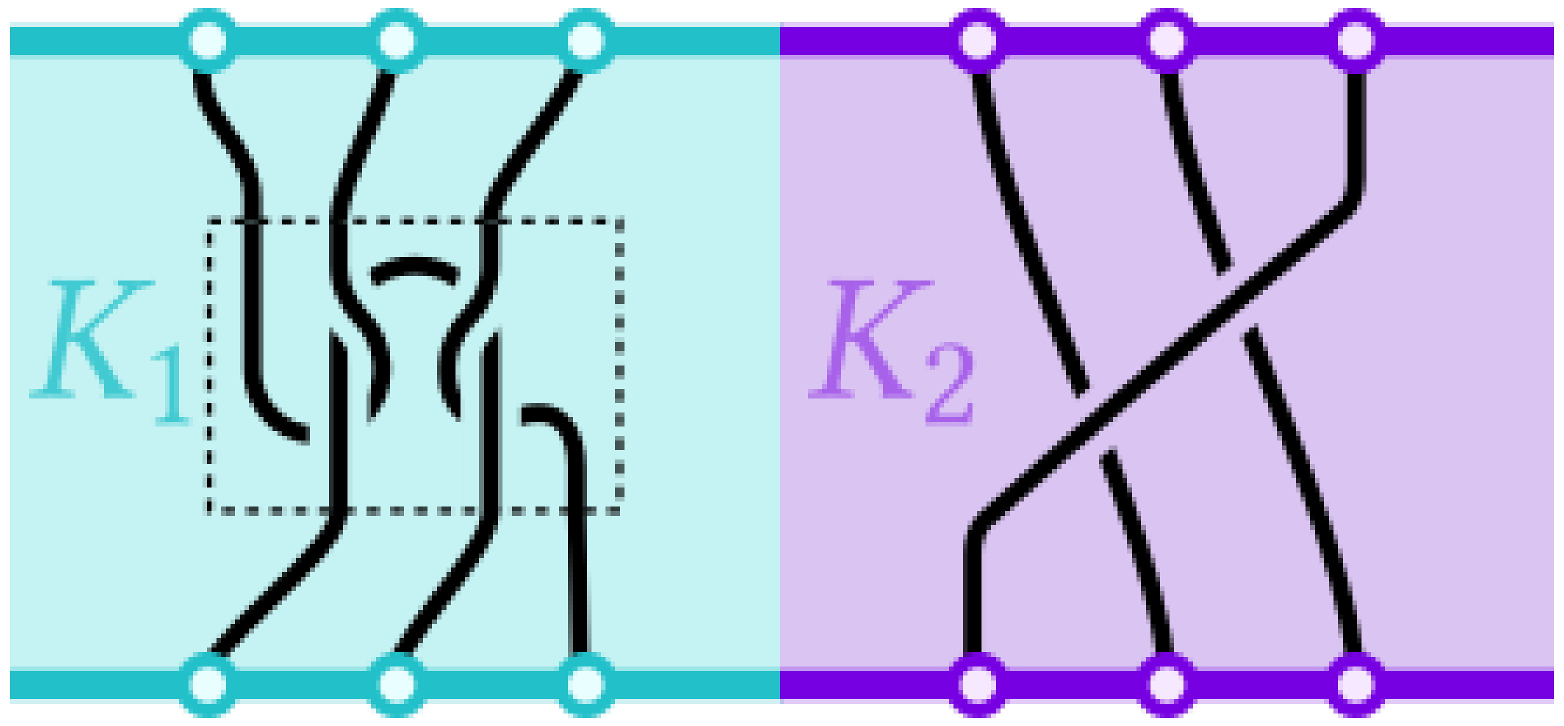


Embed additional “fences” that constrain a portion of the tangle

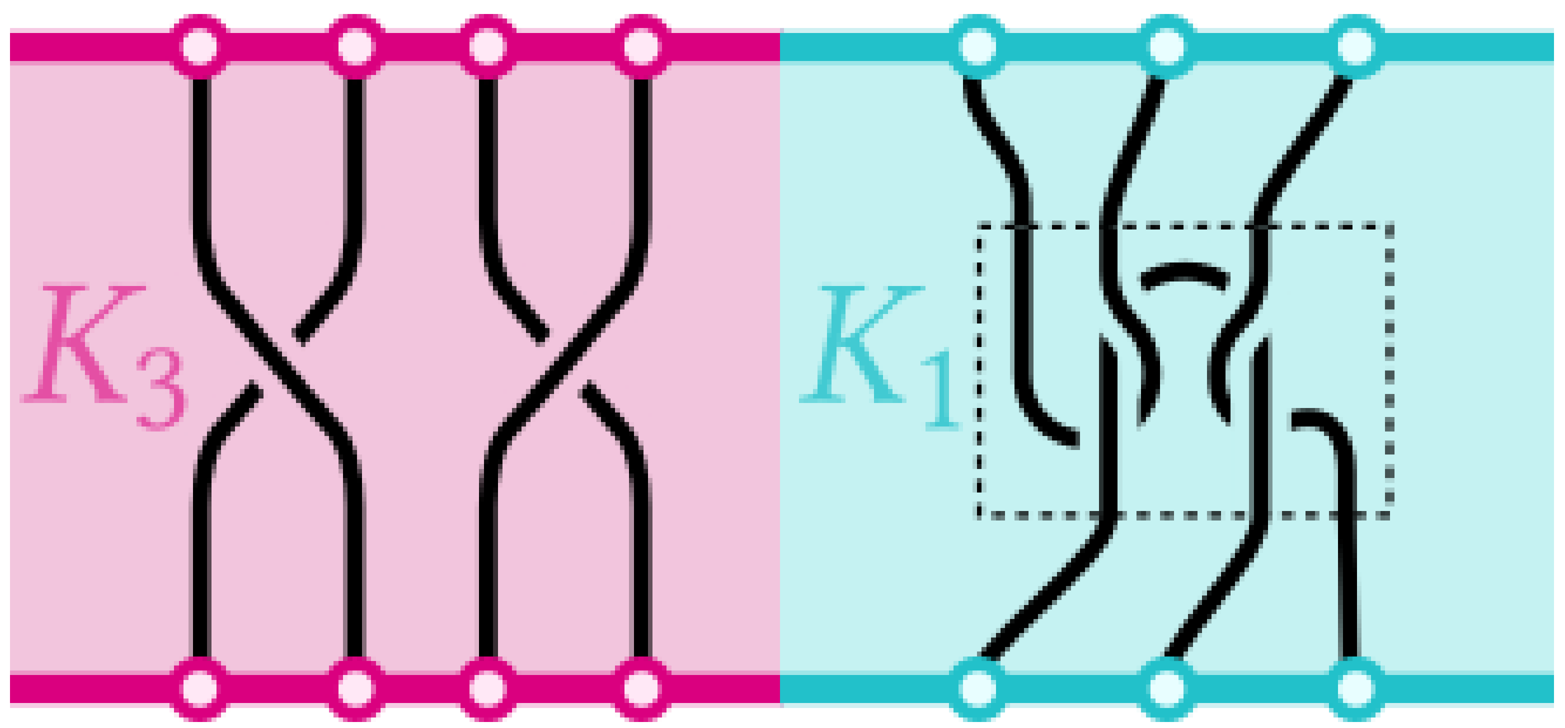
Horizontal Composition

Can present every tangle as a rectangle with an “input” (bottom) and “output” (top)





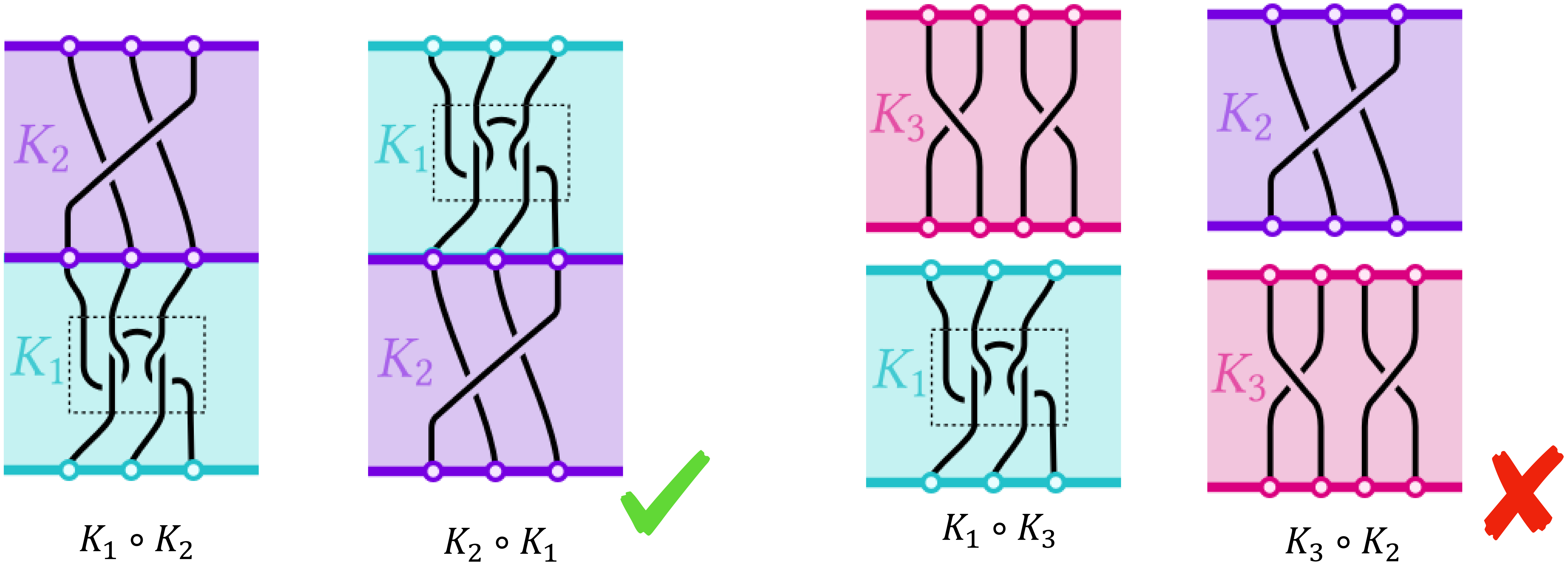
$K_1 \otimes K_2$



$K_3 \otimes K_1$

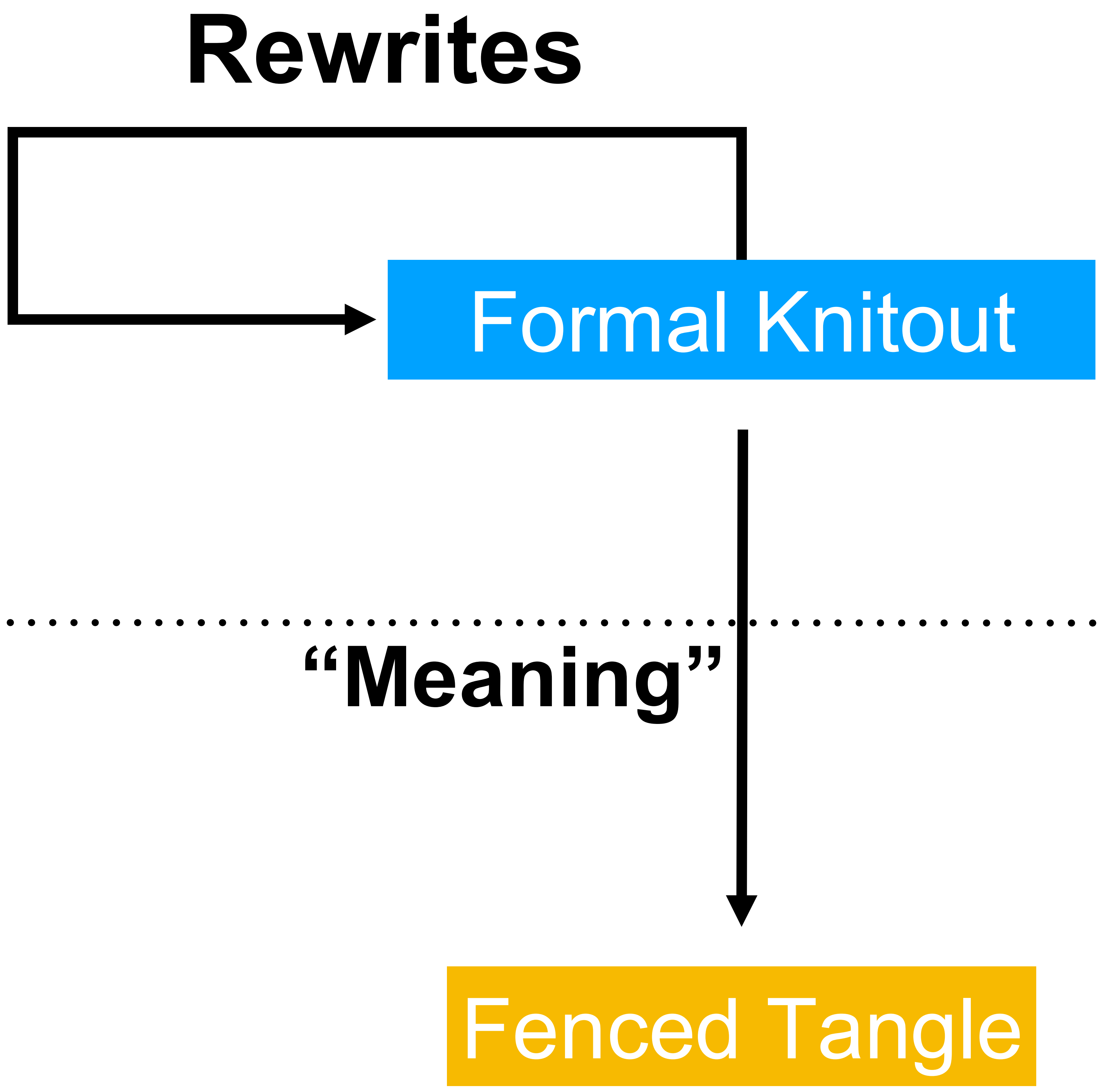
Horizontal composition always makes a fenced tangle

Vertical Composition



Vertical composition requires compatible boundaries

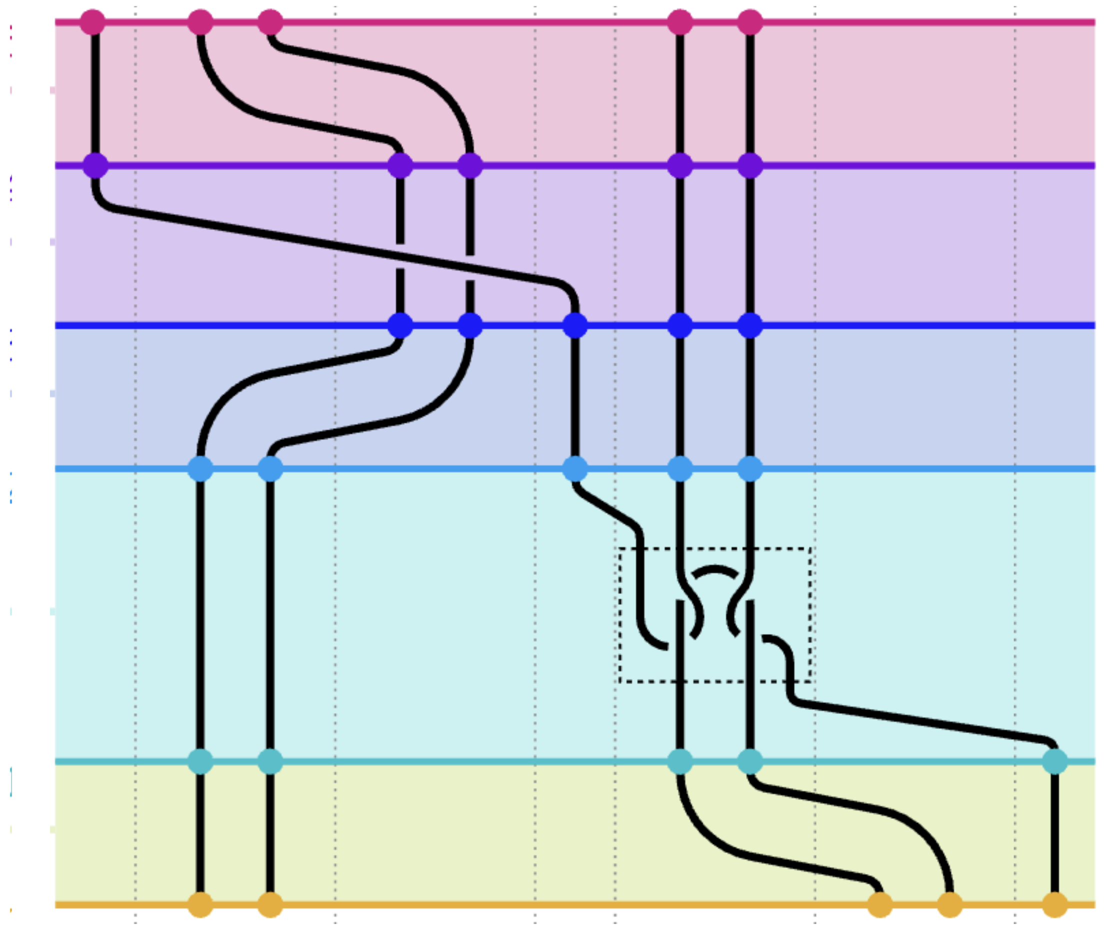
A Formal Semantics for Machine Knitting Programs



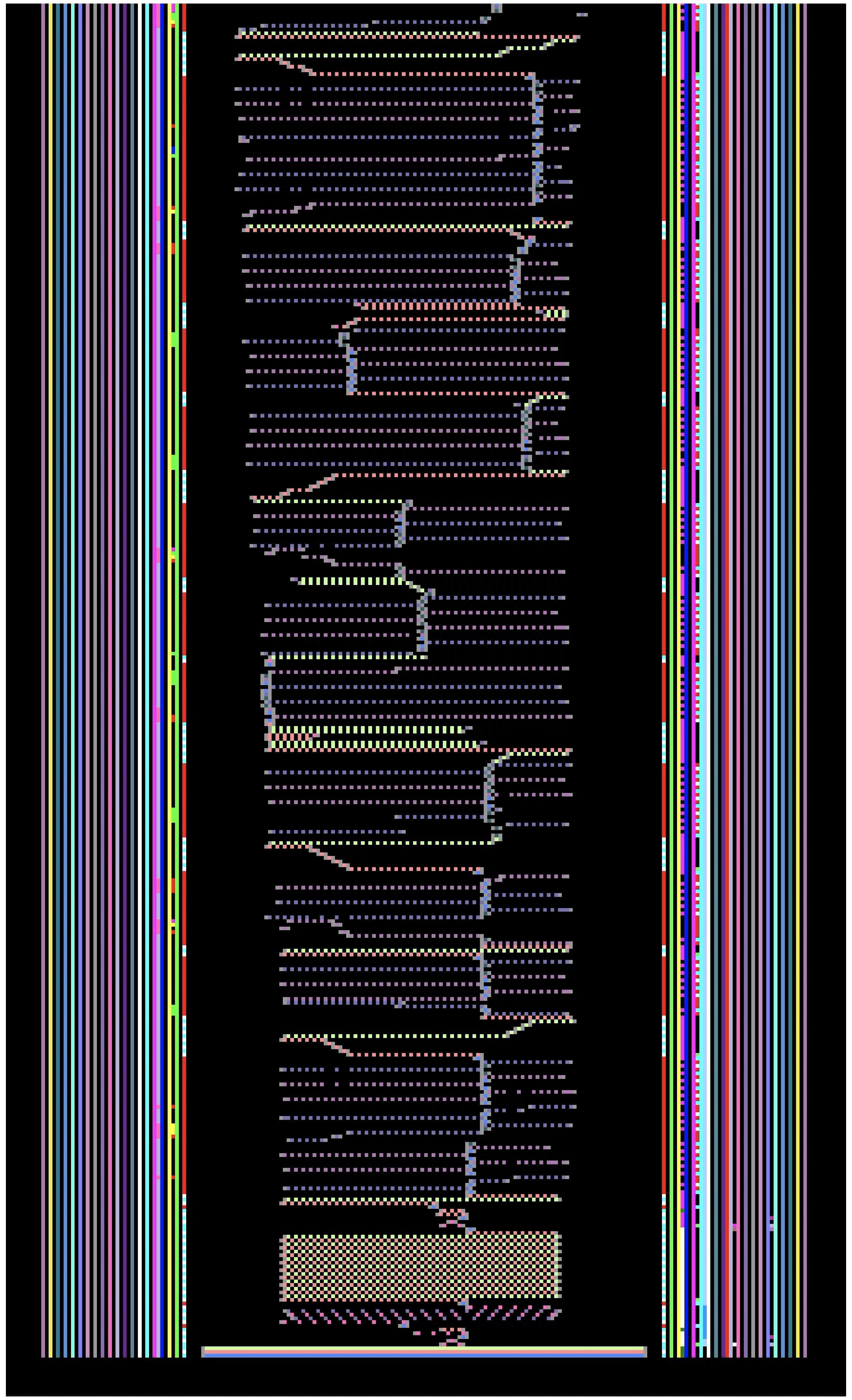

```
xfer b.2 f.2;
knit − f.2 3.0 (2, 1.0);
xfer f.1 b.1;
miss − f.1 2;
xfer b.1 f.1;
```

Computing

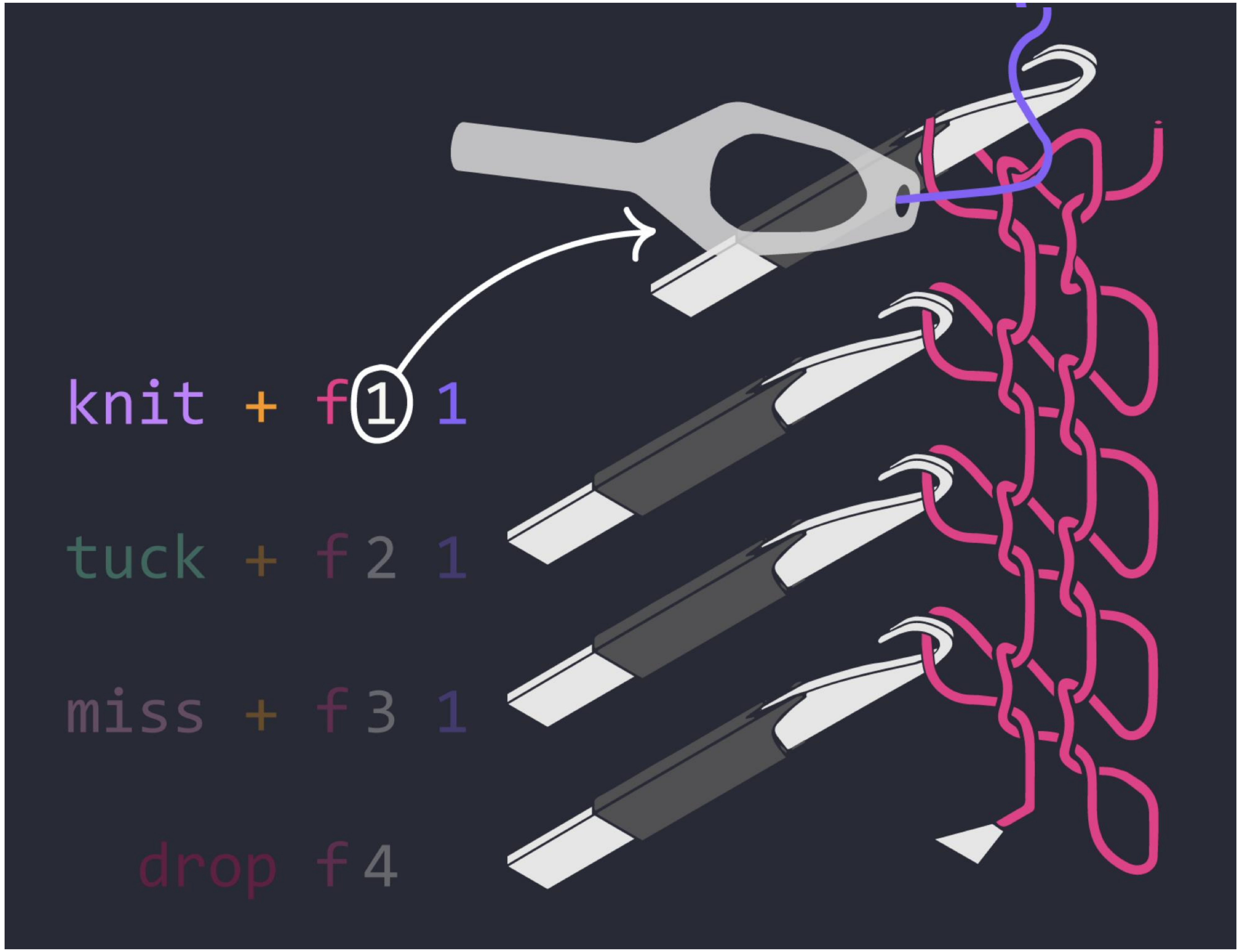
Math



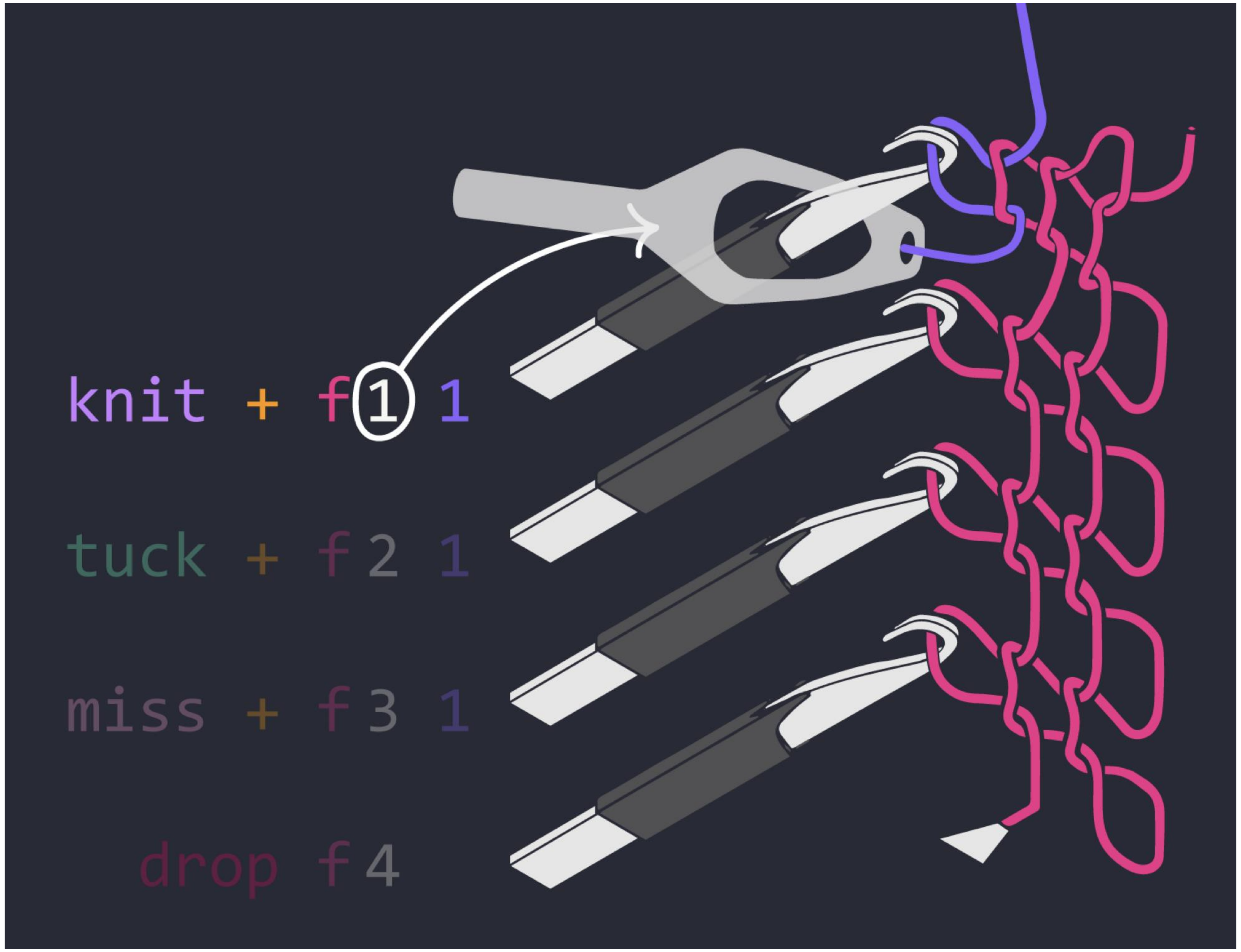
Knitting Programs are Hard to Read



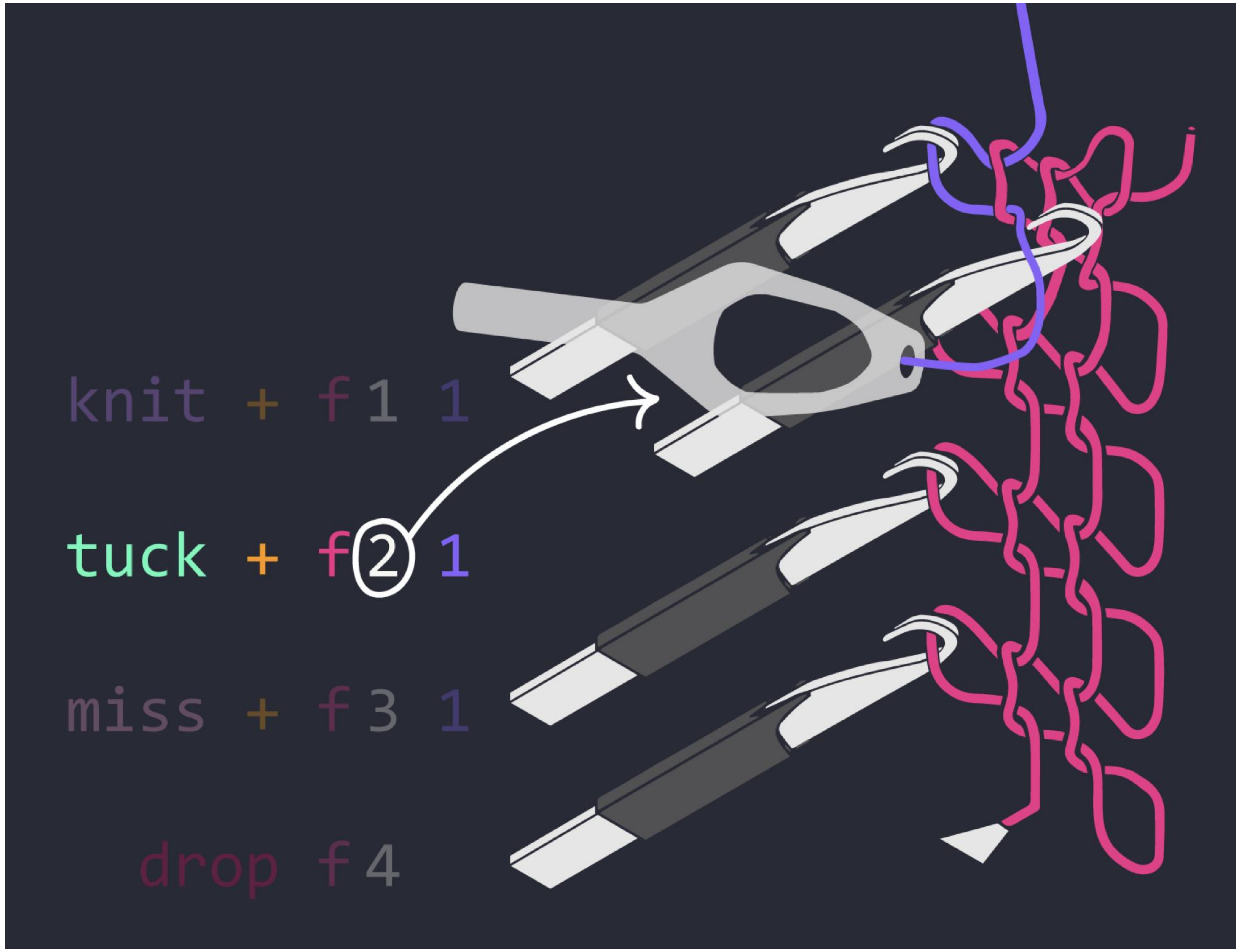
A Small Knitout Program



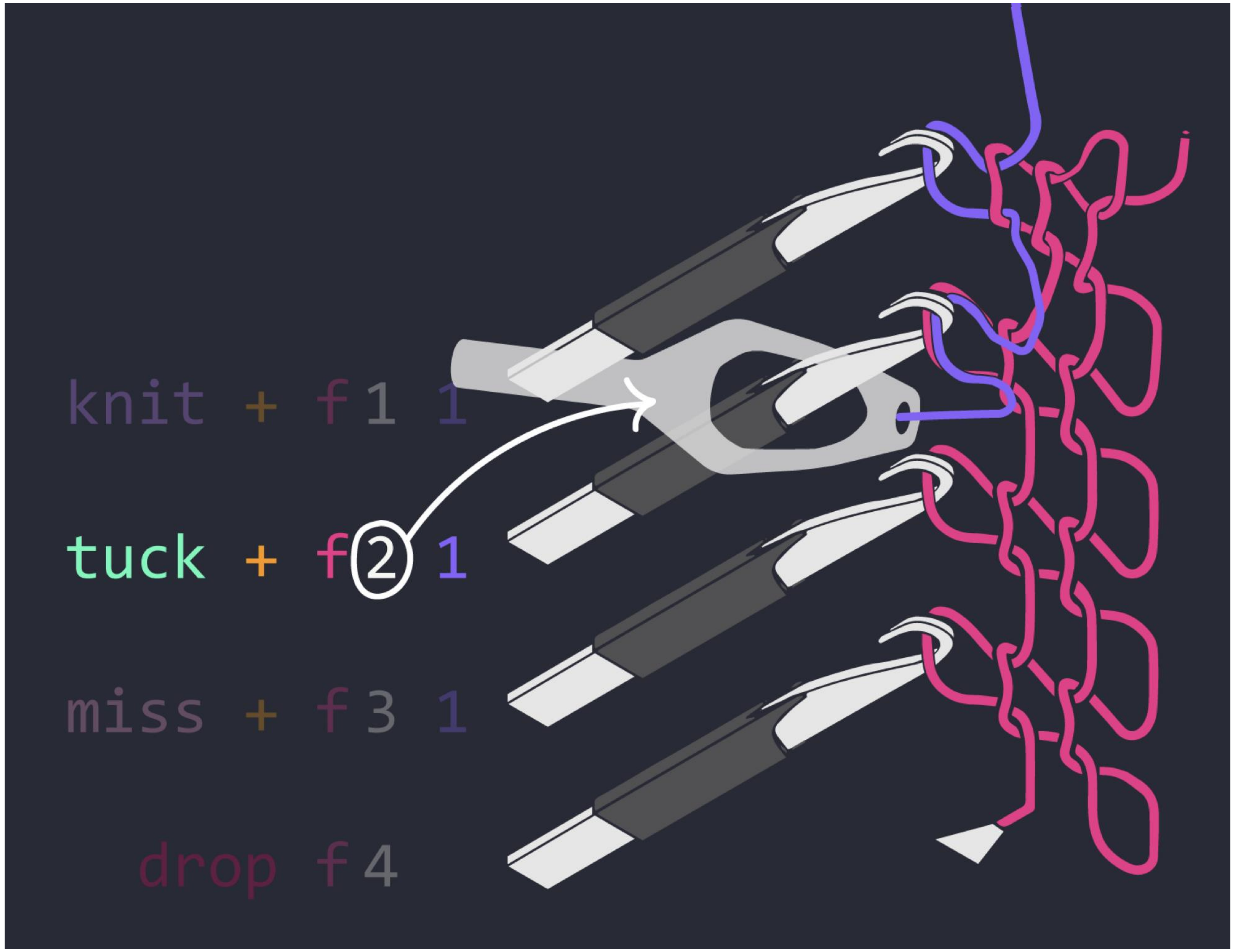
A Small Knitout Program



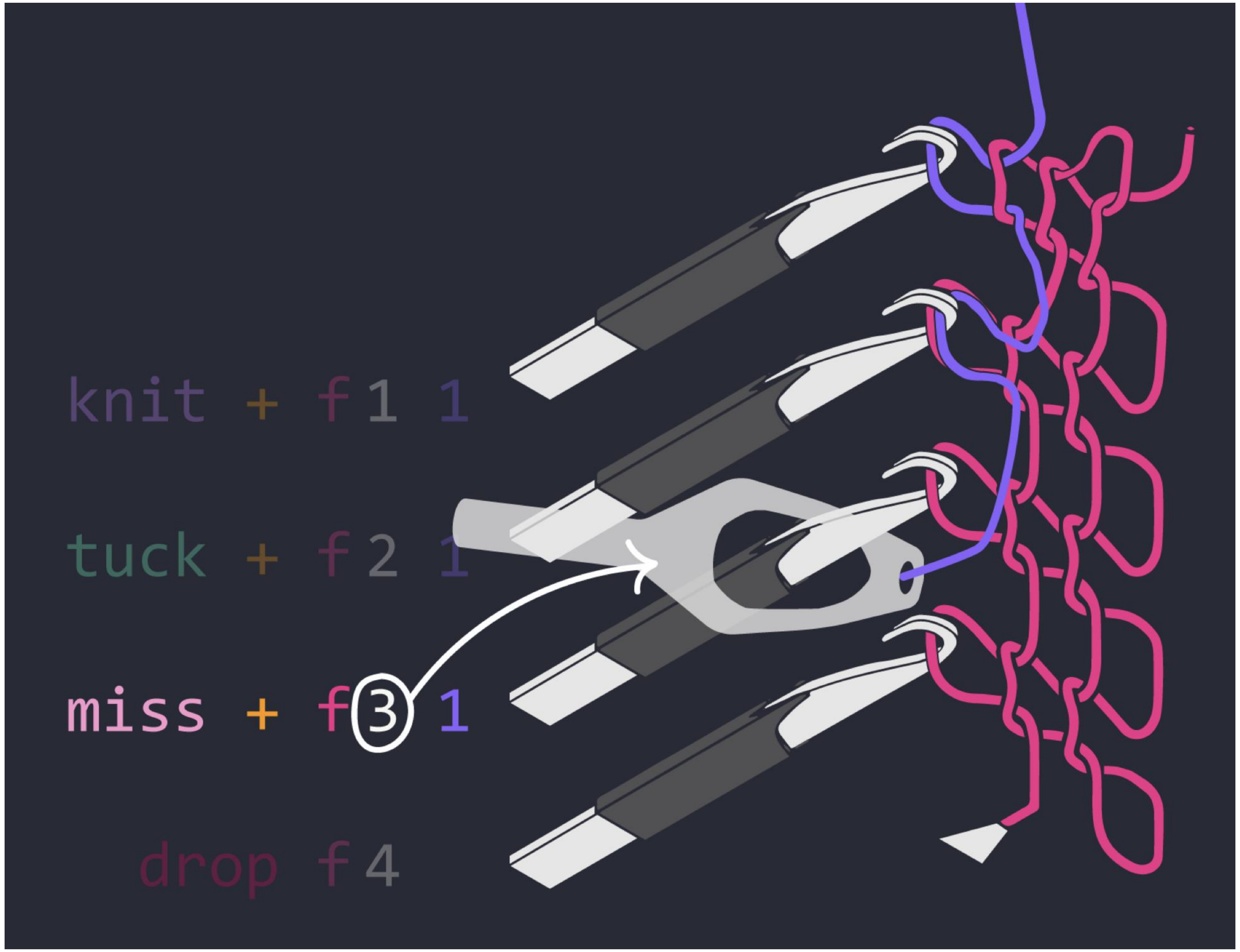
A Small Knitout Program



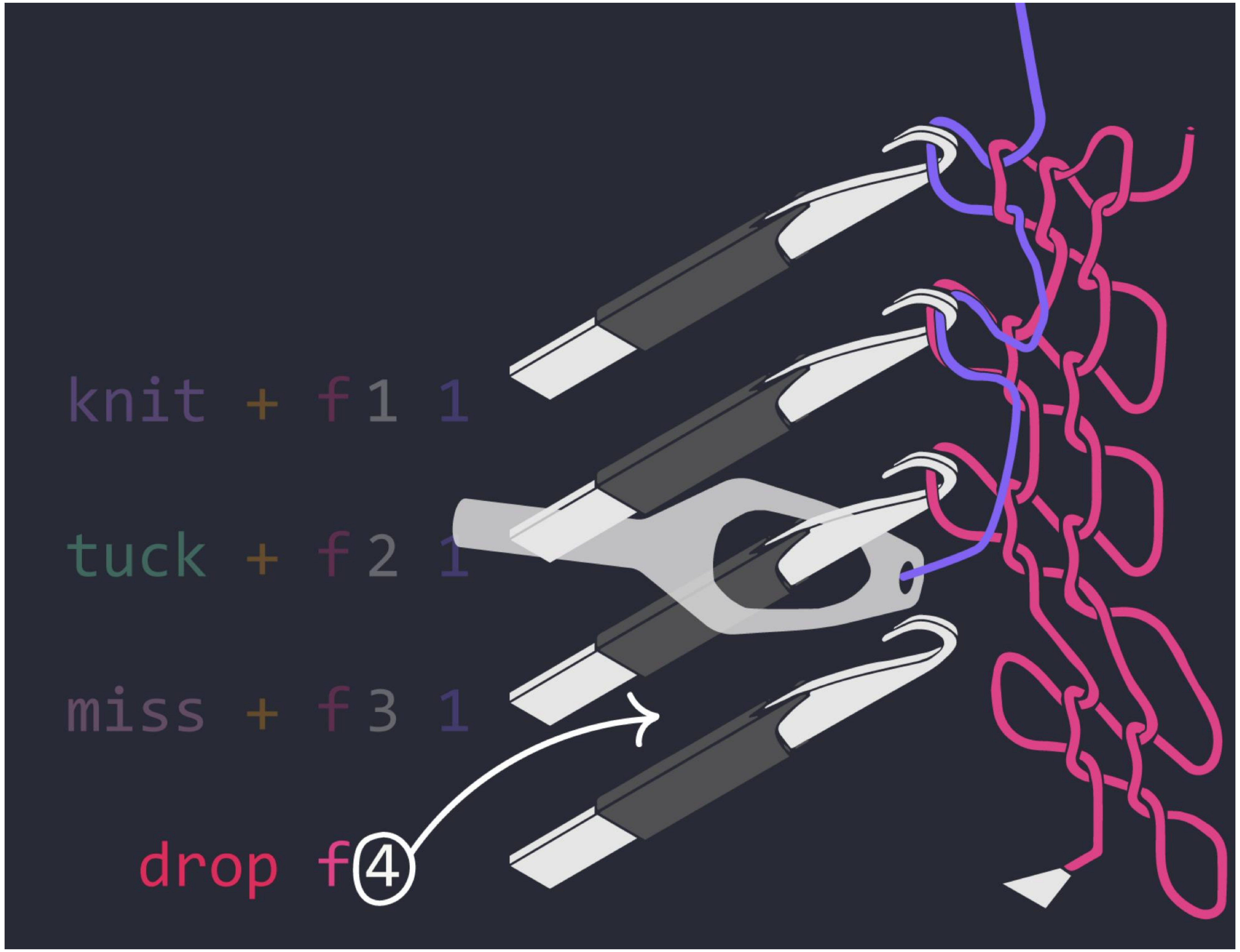
A Small Knitout Program



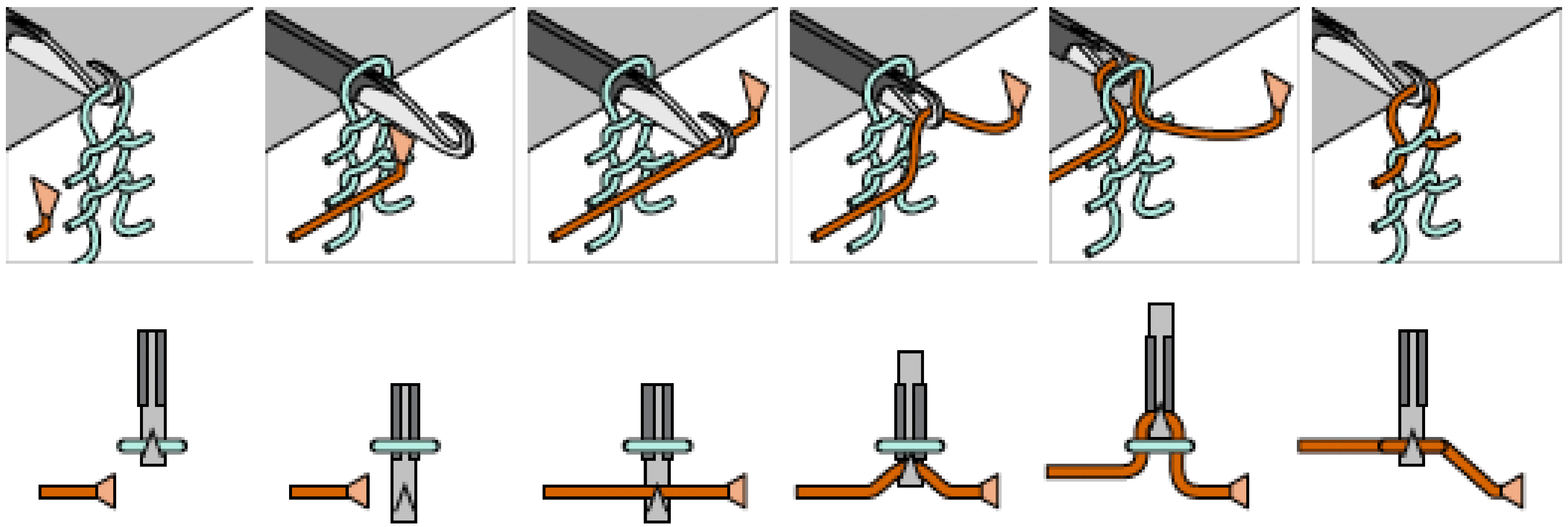
A Small Knitout Program



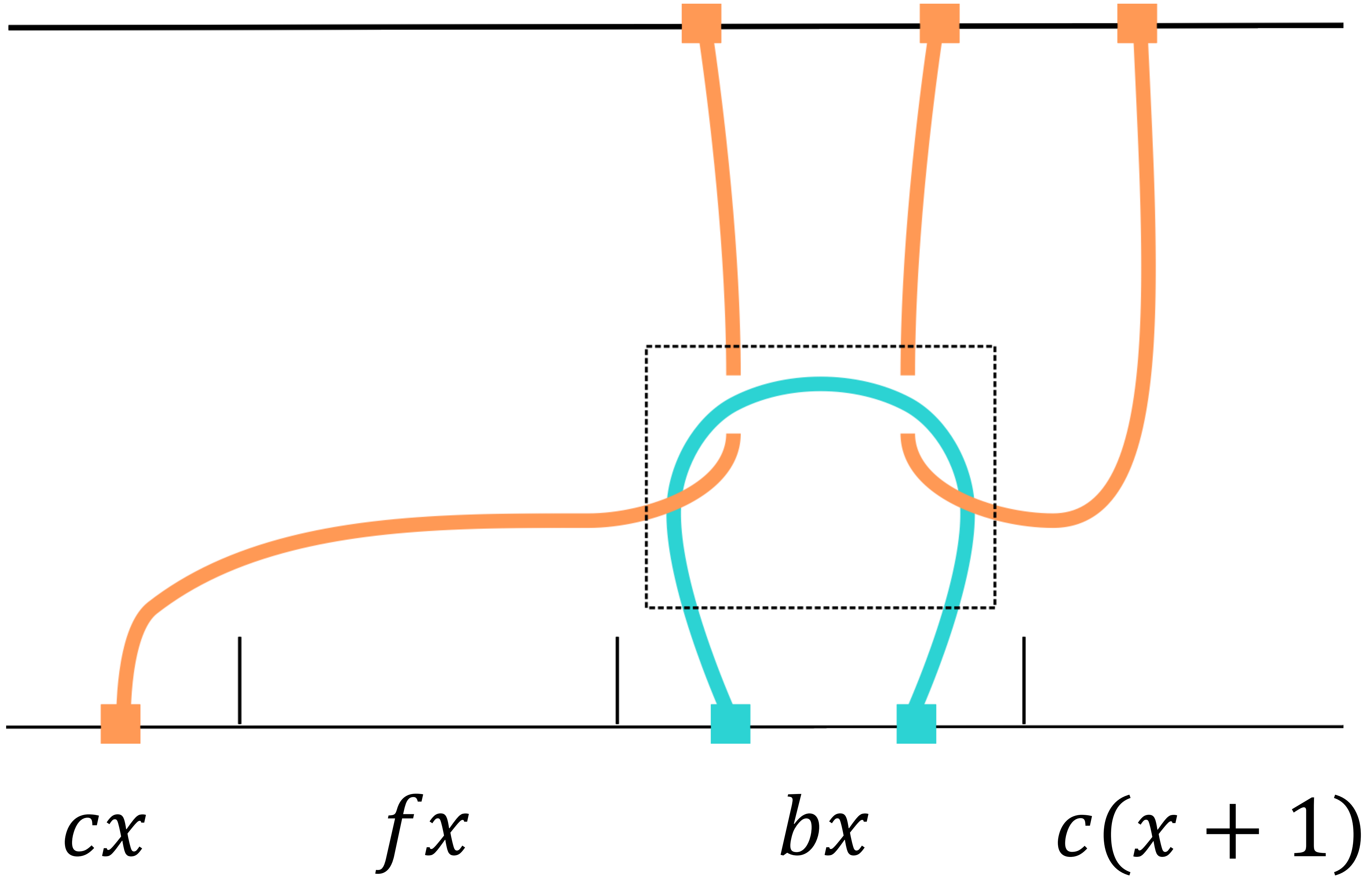
A Small Knitout Program



State Dependence of Machine Knitting

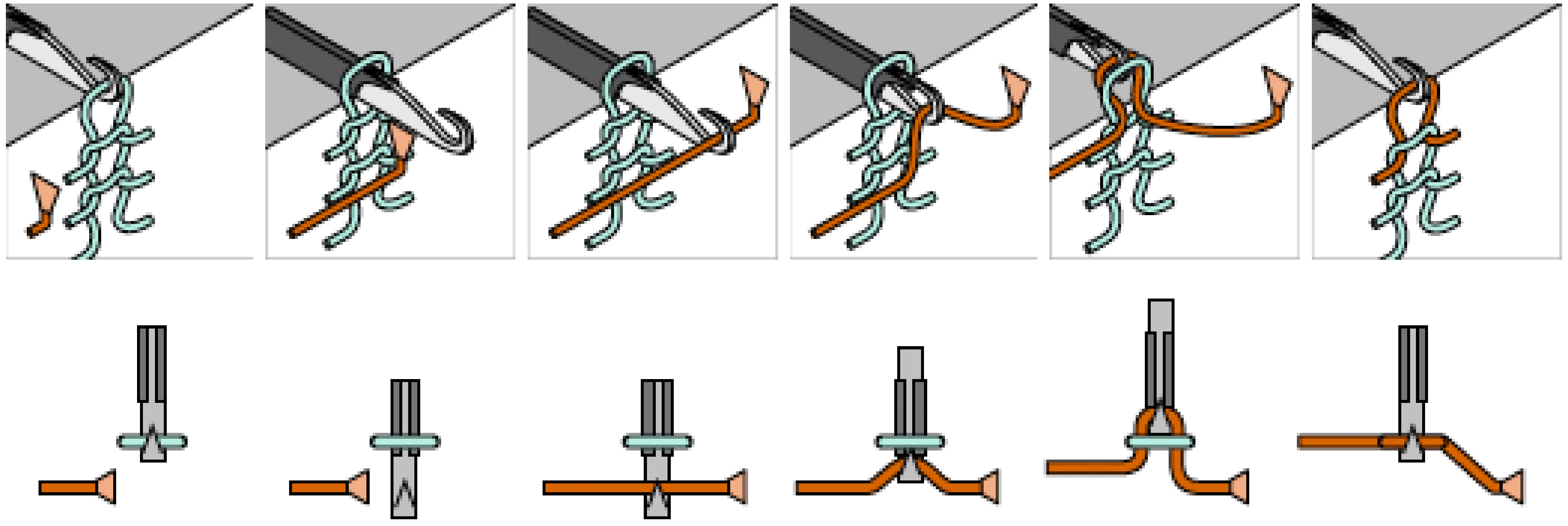


knit + bx $yarn$

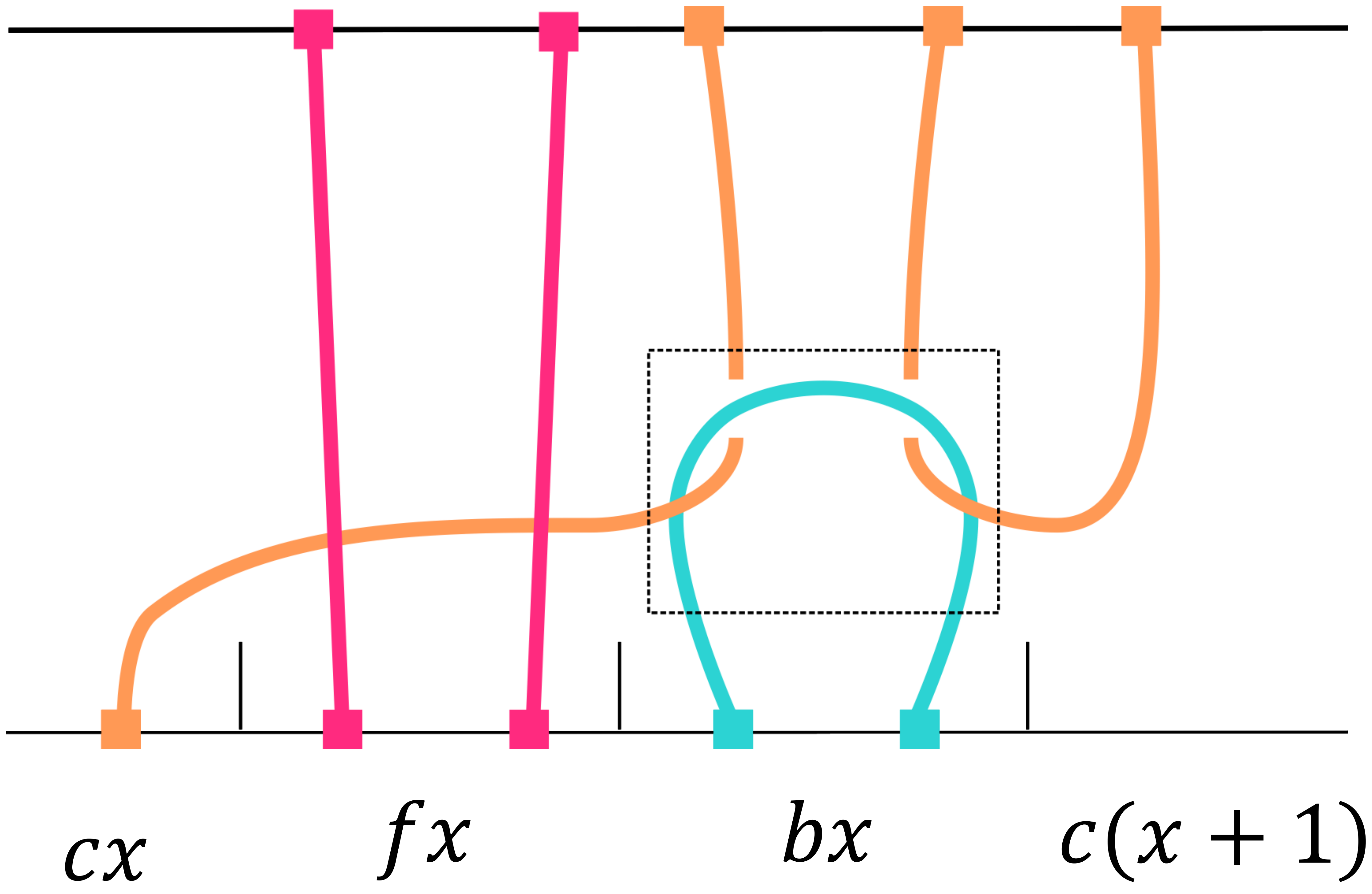


Most knitting machine operations are very local, so there's only small amount of "interesting" topology…

State Dependence of Machine Knitting



knit + bx $yarn$



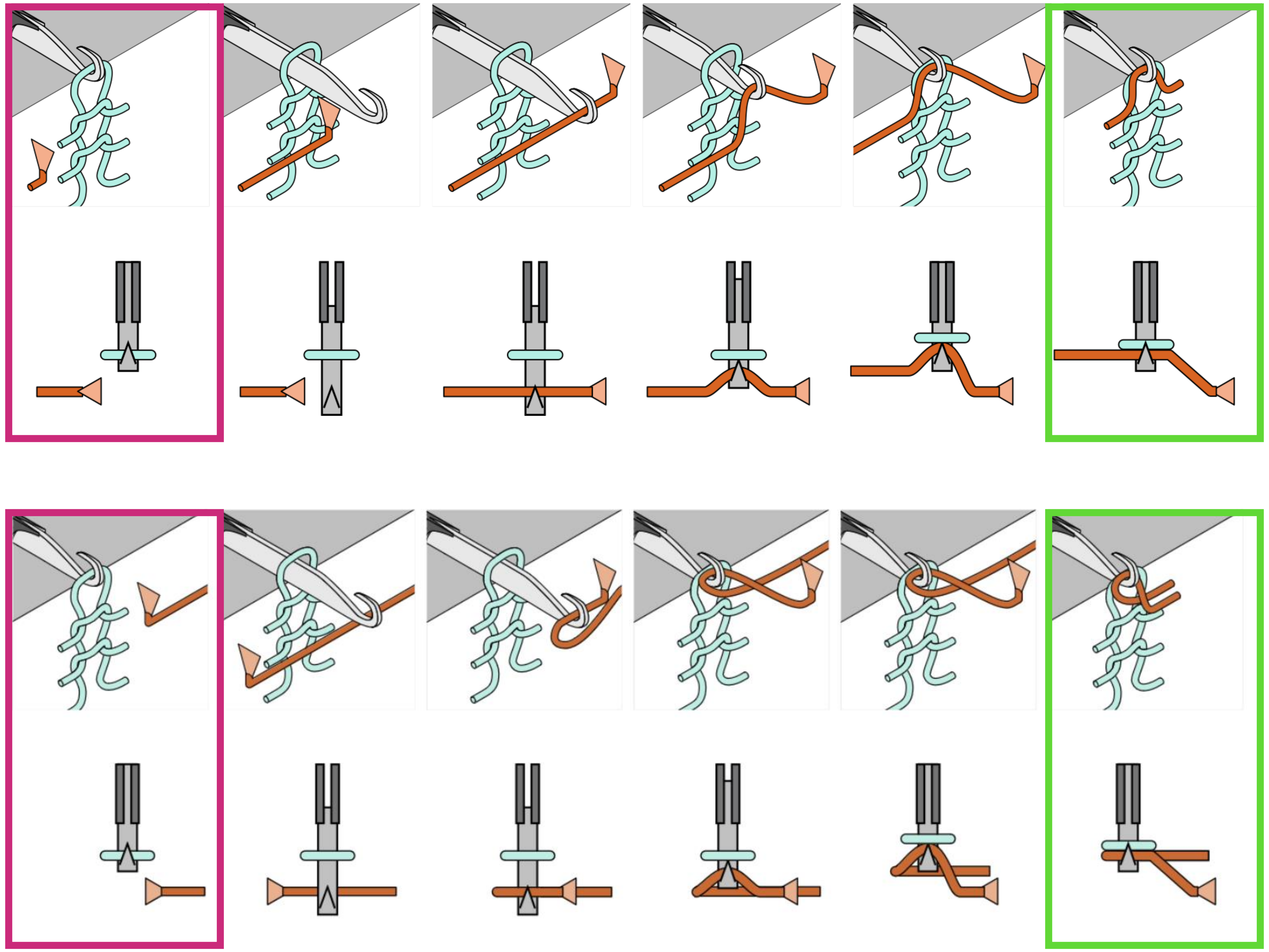
…but all the “uninteresting” topology is also important to our semantics!

State Dependence of Machine Knitting

Different machine states

$$\texttt{tuck} + \texttt{b1}\ y$$

"Needle $\texttt{b1}$ performs the $\texttt{tuck}$ operation while the carrier y is moving in the $+$ direction"



(McCann et al. 2016)

Different yarn topology

Formal Knitout Validity

All carriers used have to start in the correct location relative to the needle used

There has to be at least one loop on the needle to knit through

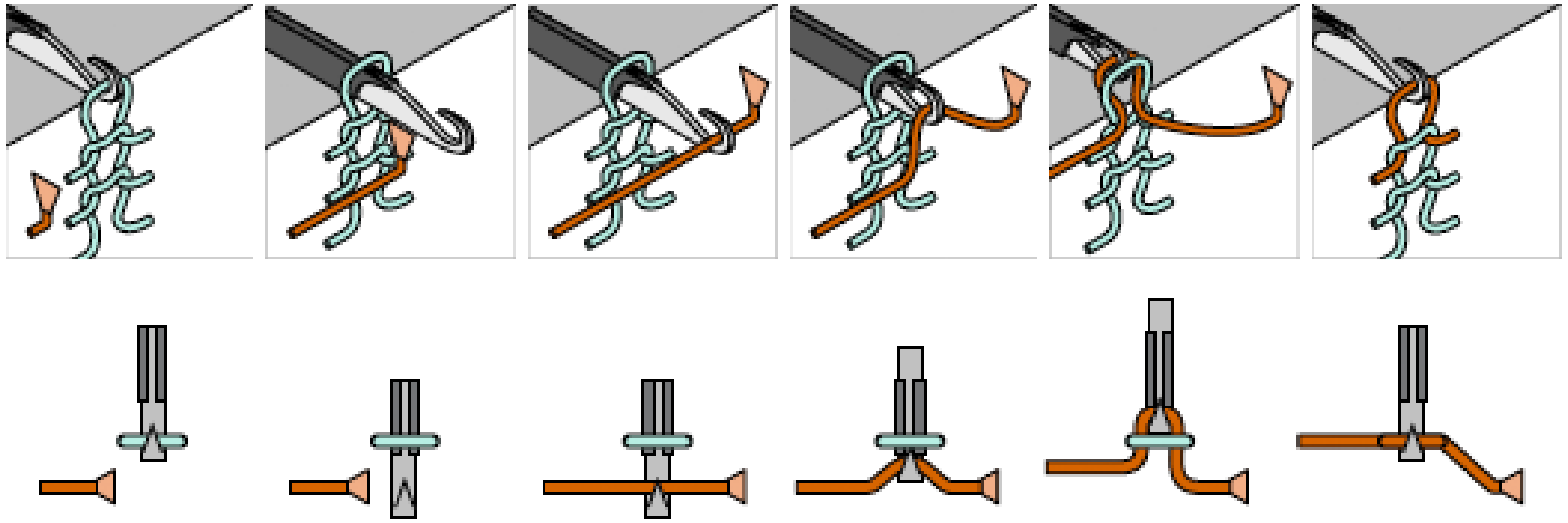
All carriers will stop in a fixed location relative to the needle used

$$\frac{\boxed{(Y, yarns) =_r (n.x, \neg dir)} \quad \boxed{L(n.x) > 0} \quad \boxed{\begin{array}{l} Y' = Y[yarns \mapsto \lfloor n.x, dir \rfloor_r] \\ A' = A[yarns \mapsto n.x] \end{array}}}{\boxed{(r, L, Y, A) \xrightarrow{\texttt{knit}\ dir\ n.x\ l\ yarns} (r, \boxed{L[n.x \mapsto \#yarns]}, Y', A')}} \ \textbf{V-knit}$$

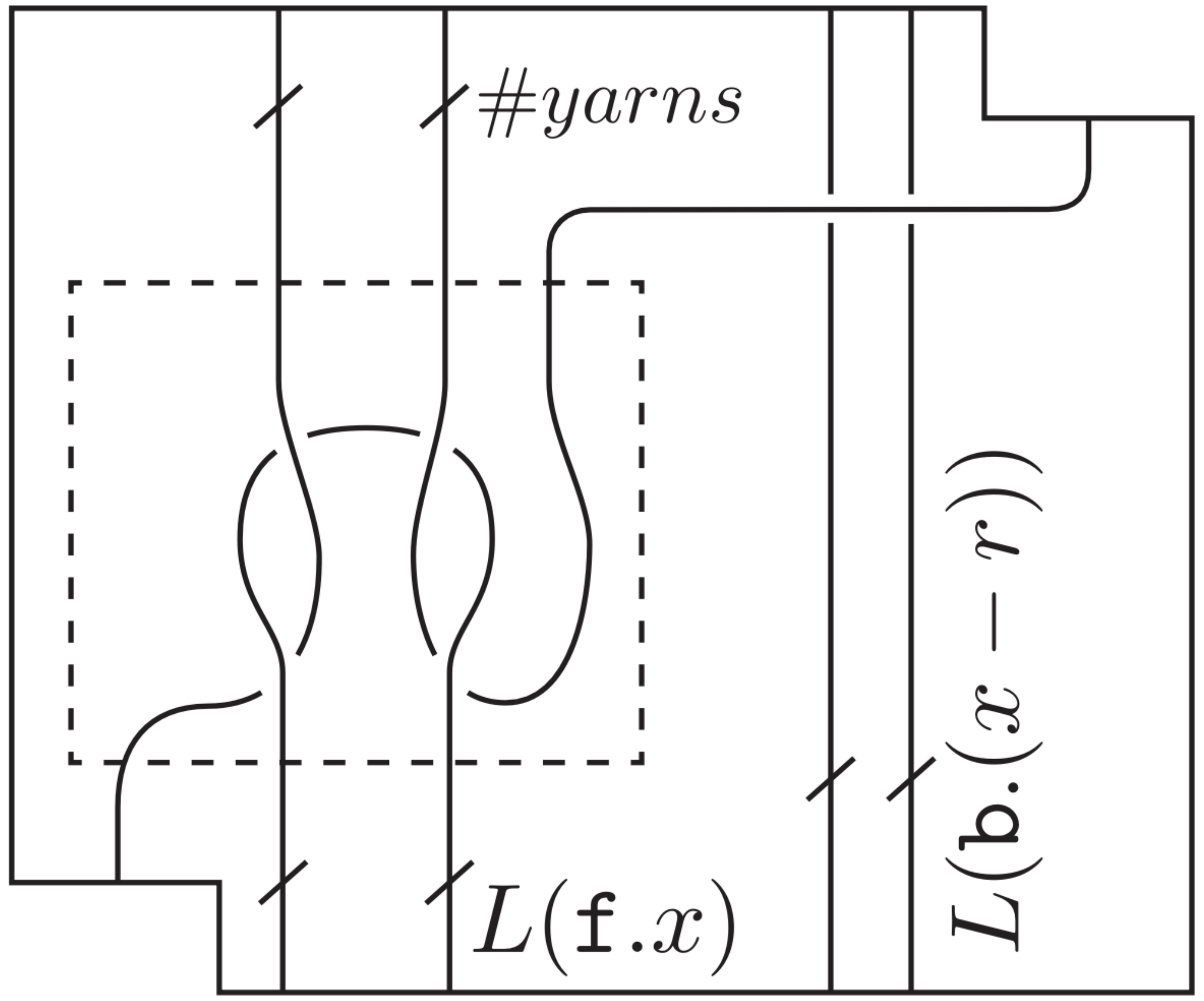
Type relation is an abstract execution of a program on an initial state

The new number of loops on the needle will be the number of carriers used

Formal Knitout Semantics



knit + f. x l $yarns$

$id_m \otimes$  $\otimes\ id_n$

Templated diagrams define the class of fenced tangles a knitout operation can make

If two program traces
are valid and have an
intermediate state

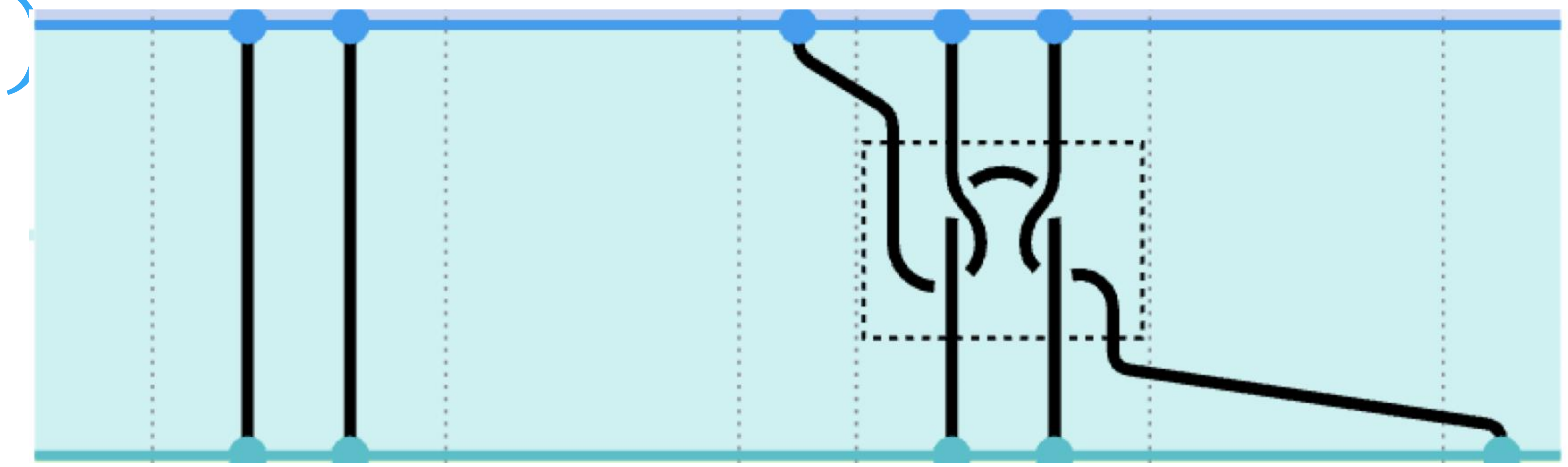
$$\frac{\boxed{S \xrightarrow{ks_1} S'} \qquad \boxed{S' \xrightarrow{ks_2} S''}}{\boxed{S \xrightarrow{ks_1;ks_2} S''}} \ \textbf{V-seq}$$

The concatenated
program is also valid

Formal Knitout Semantics

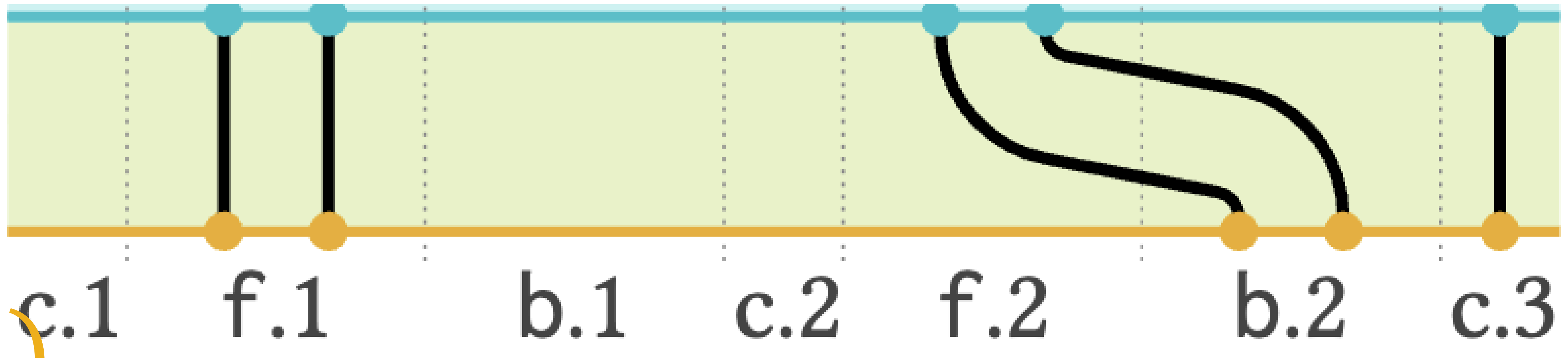
$S_2 = ([\mathrm{f.}1 \mapsto 1][\mathrm{f.}2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c.}2])$

$\uparrow$ knit − f. 2 yarn



$S_1 = ([\mathrm{f.}1 \mapsto 1][\mathrm{f.}2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c.}3])$

$\uparrow$ xfer b. 2 f. 2



$S_0 = ([\mathrm{f.}1 \mapsto 1][\mathrm{b.}2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c.}3])$

Formal Knitout Semantics

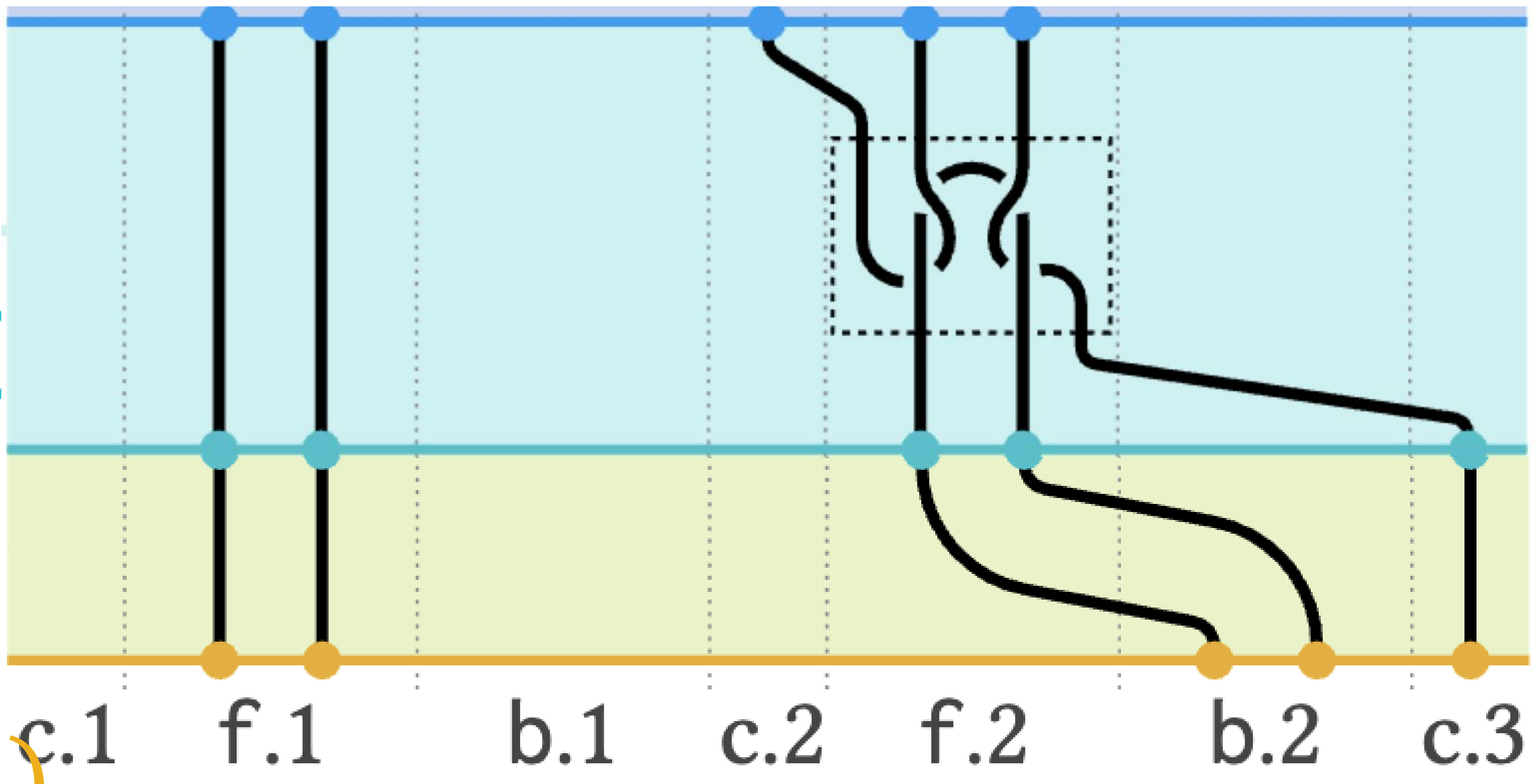
$$S_2 = ([\mathrm{f}.1 \mapsto 1][\mathrm{f}.2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c}.2])$$

$\uparrow$ knit $-$ f. 2 yarn

$$S_1 = ([\mathrm{f}.1 \mapsto 1][\mathrm{f}.2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c}.3])$$

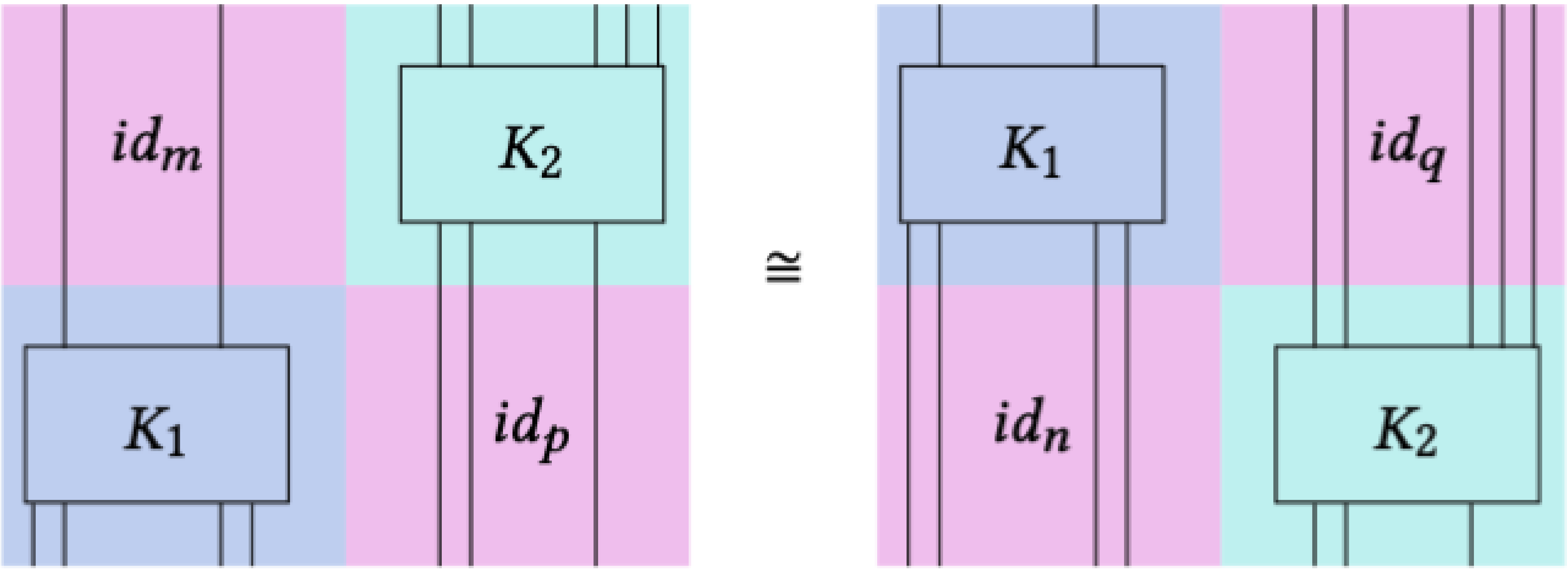
$\uparrow$ xfer b. 2 f. 2

$$S_0 = ([\mathrm{f}.1 \mapsto 1][\mathrm{b}.2 \mapsto 1], [\mathrm{yarn} \mapsto \mathrm{c}.3])$$



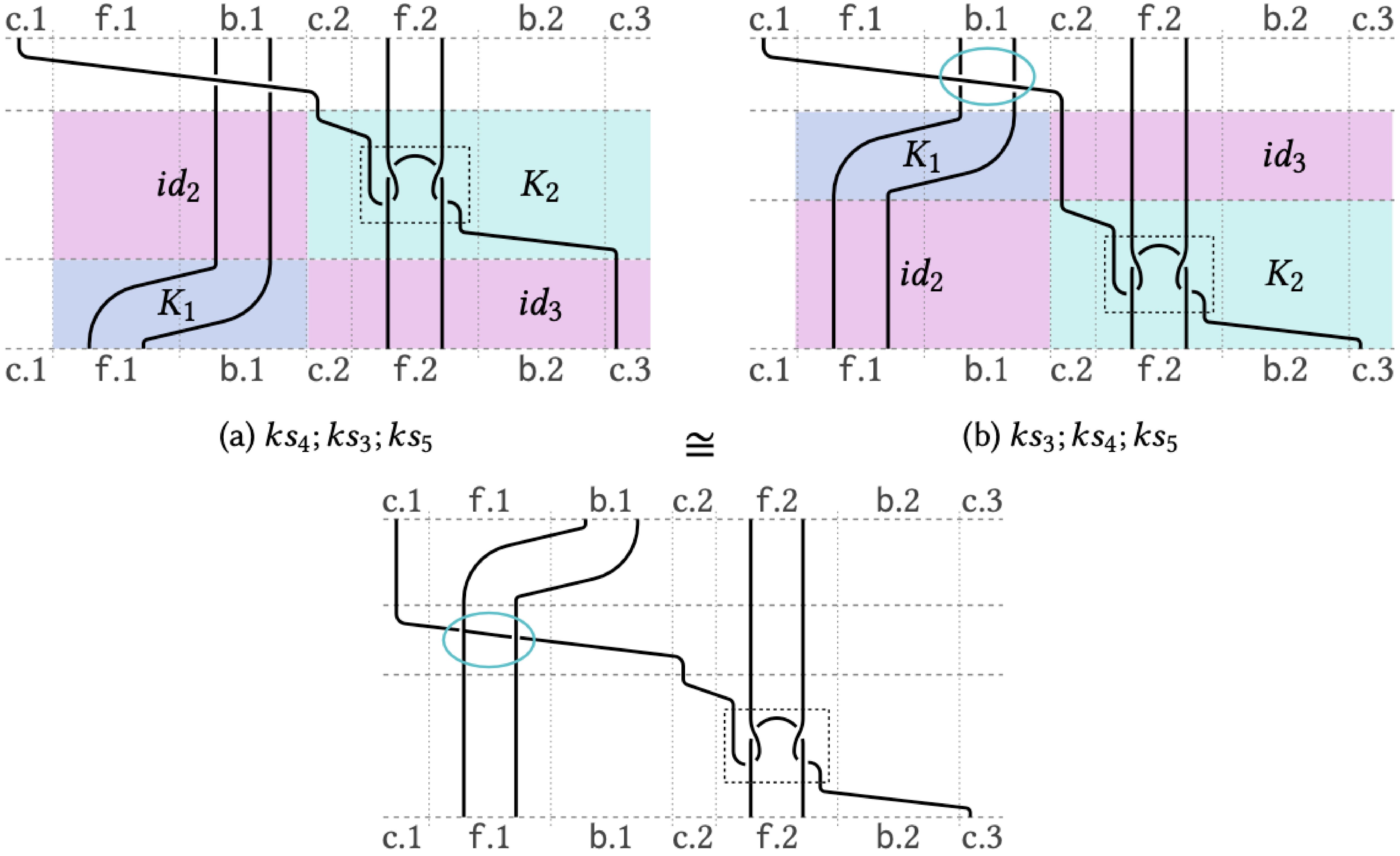
Program concatenation maps to fenced tangle vertical composition

Rewrite Rules



$(K_1 \otimes id) \circ (id \otimes K_2)$ $\qquad$ $(id \otimes K_2) \circ (K_1 \otimes id)$

We can prove general lemmas about fenced tangle equivalence



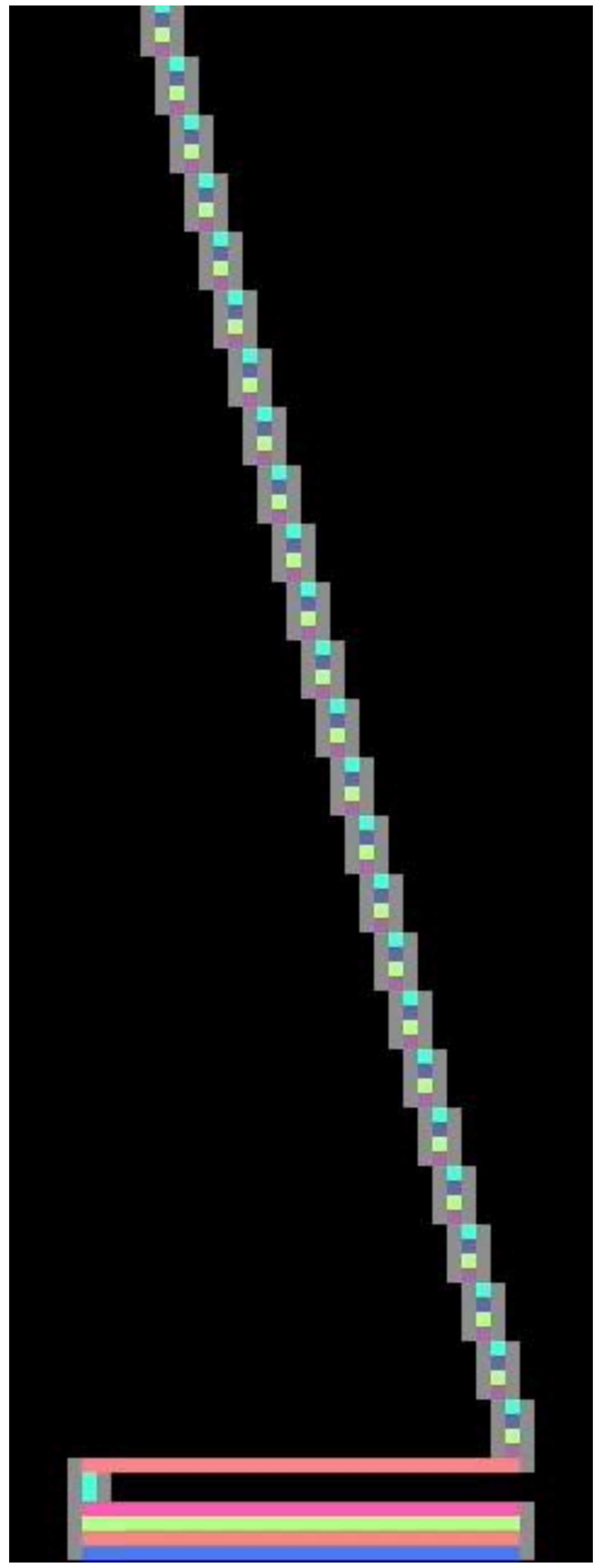
(a) $ks_4; ks_3; ks_5$

(b) $ks_3; ks_4; ks_5$

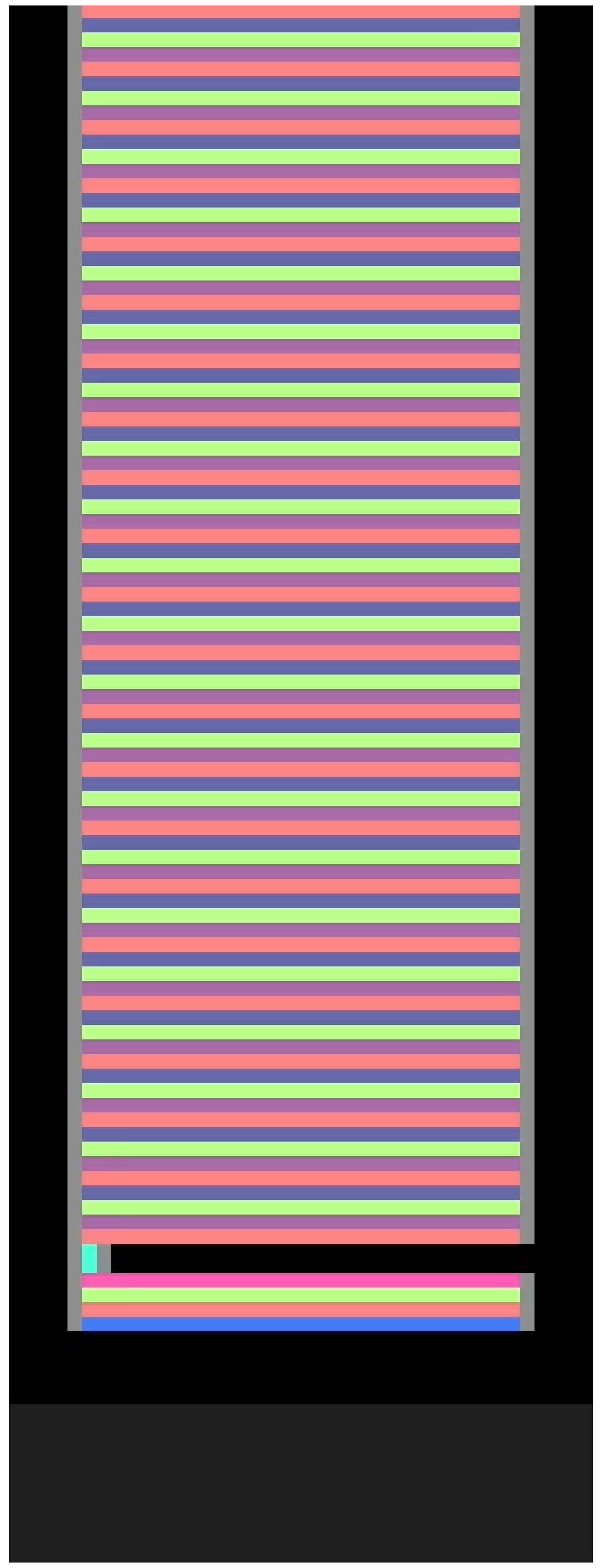
(c) $ks_3; ks_5; ks_4$

These lemmas can than be used to validate program rewrites

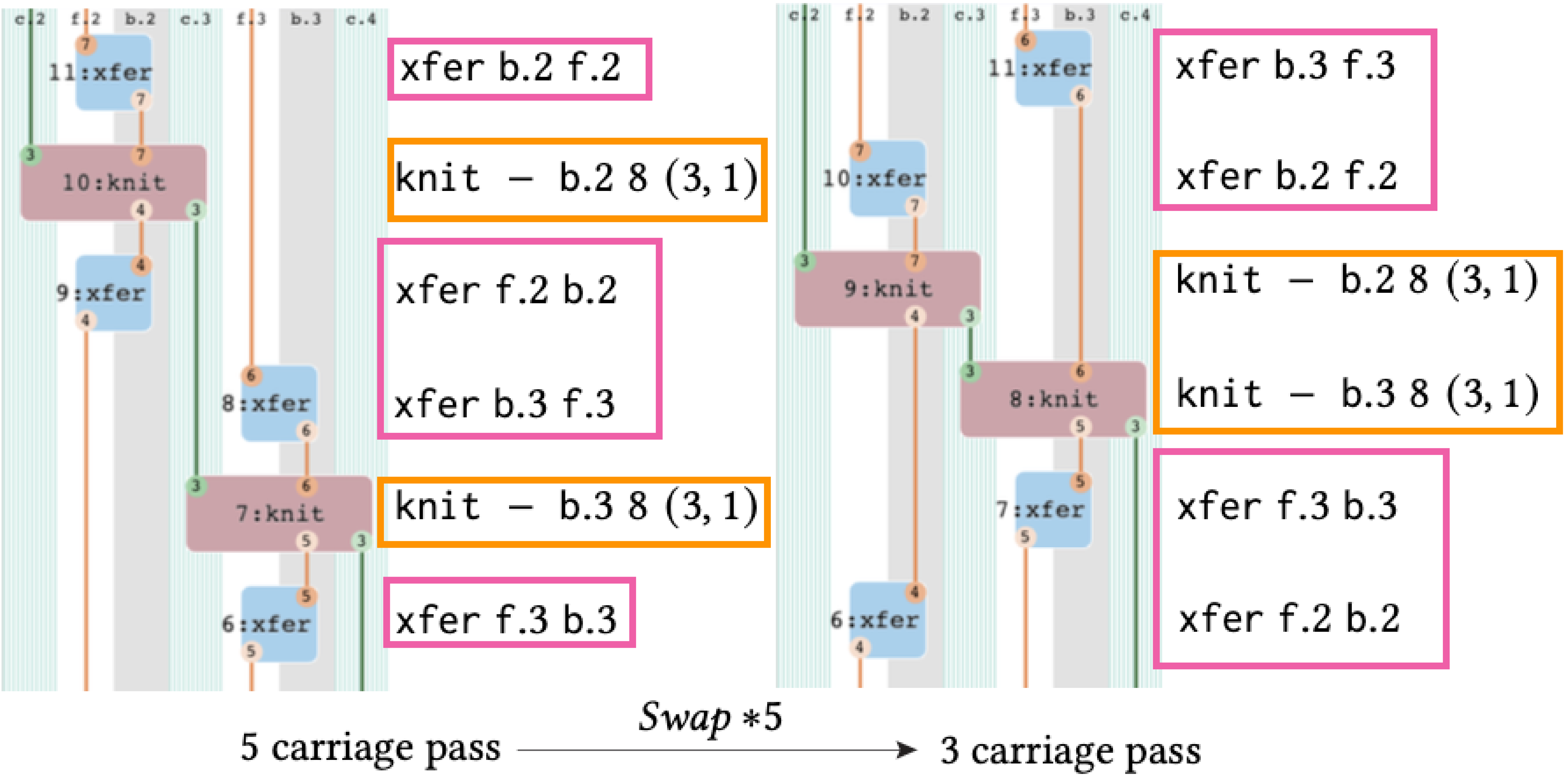
Improve Fabrication Time



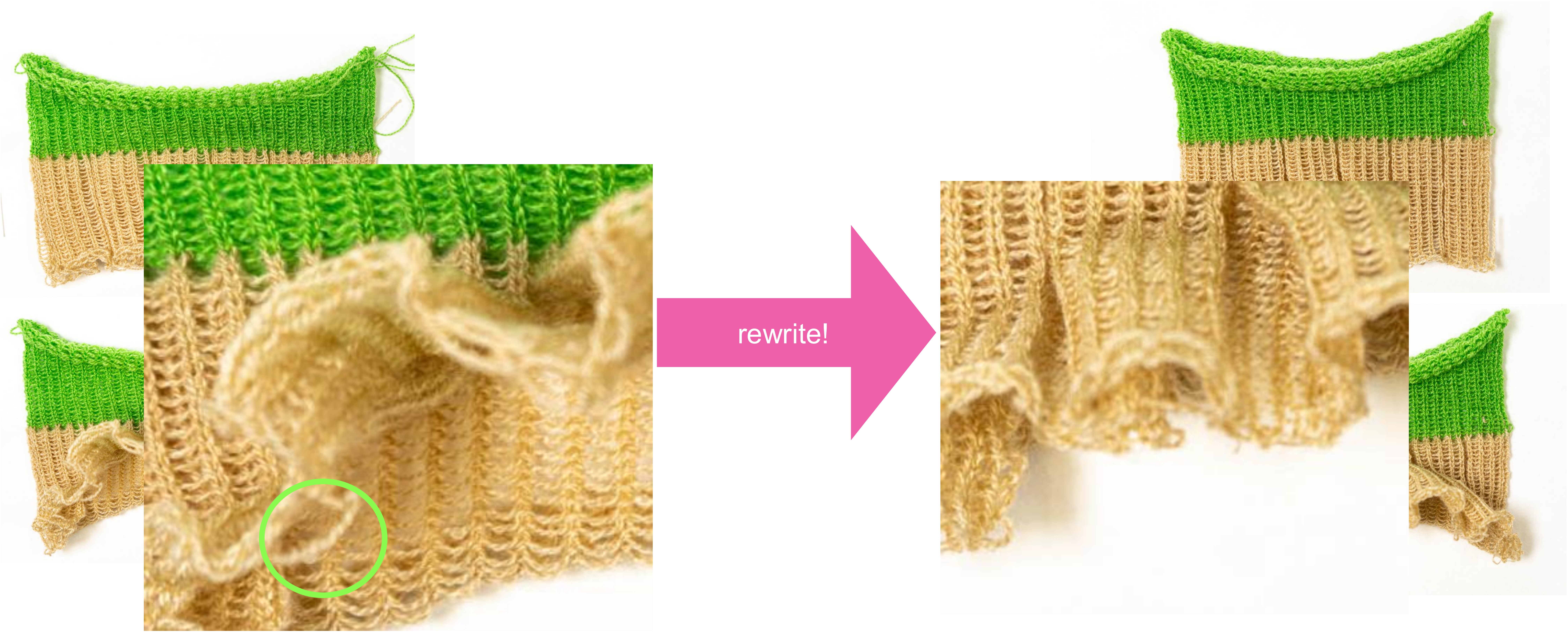
Novice



Expert



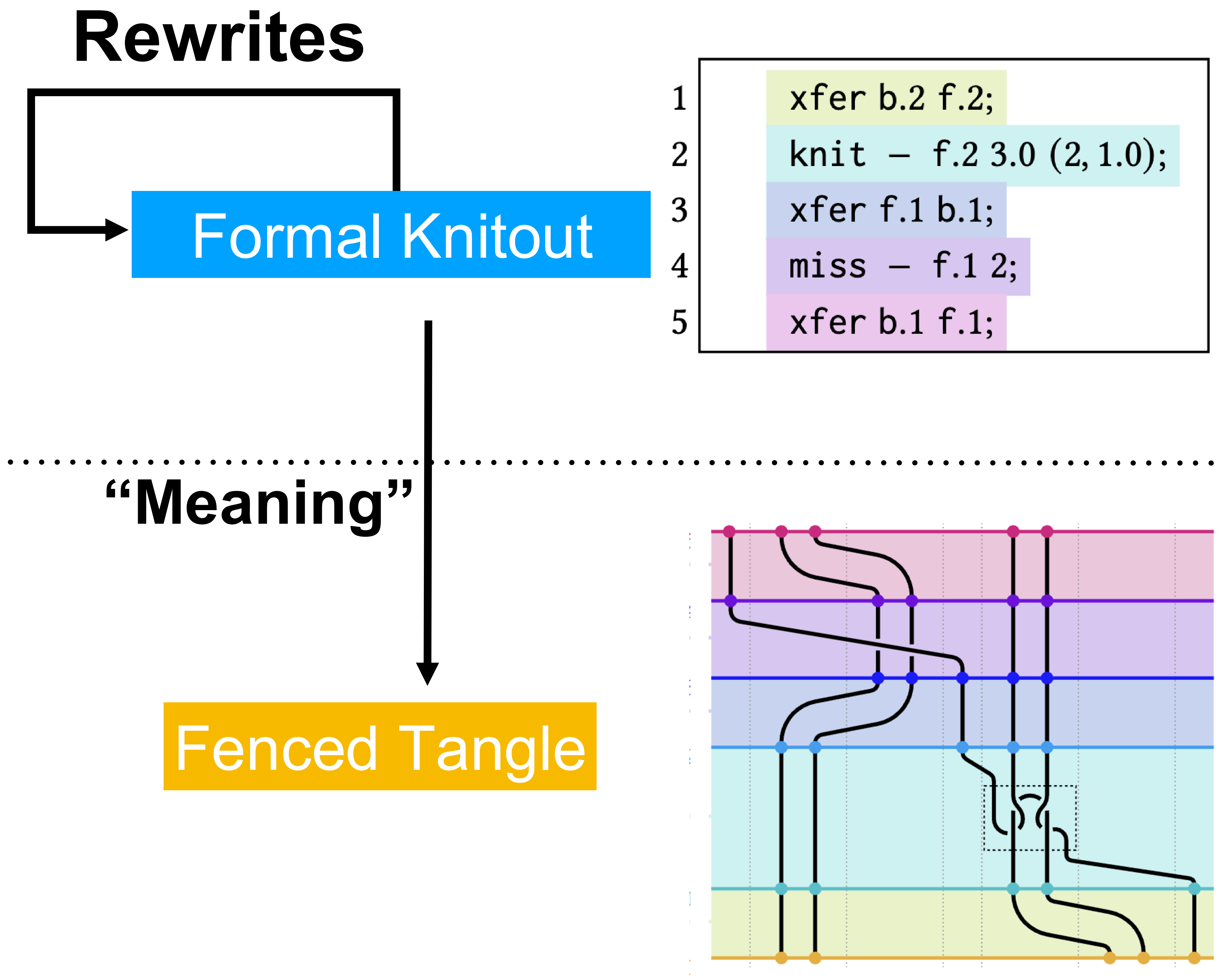
Program Optimizations



120 needles

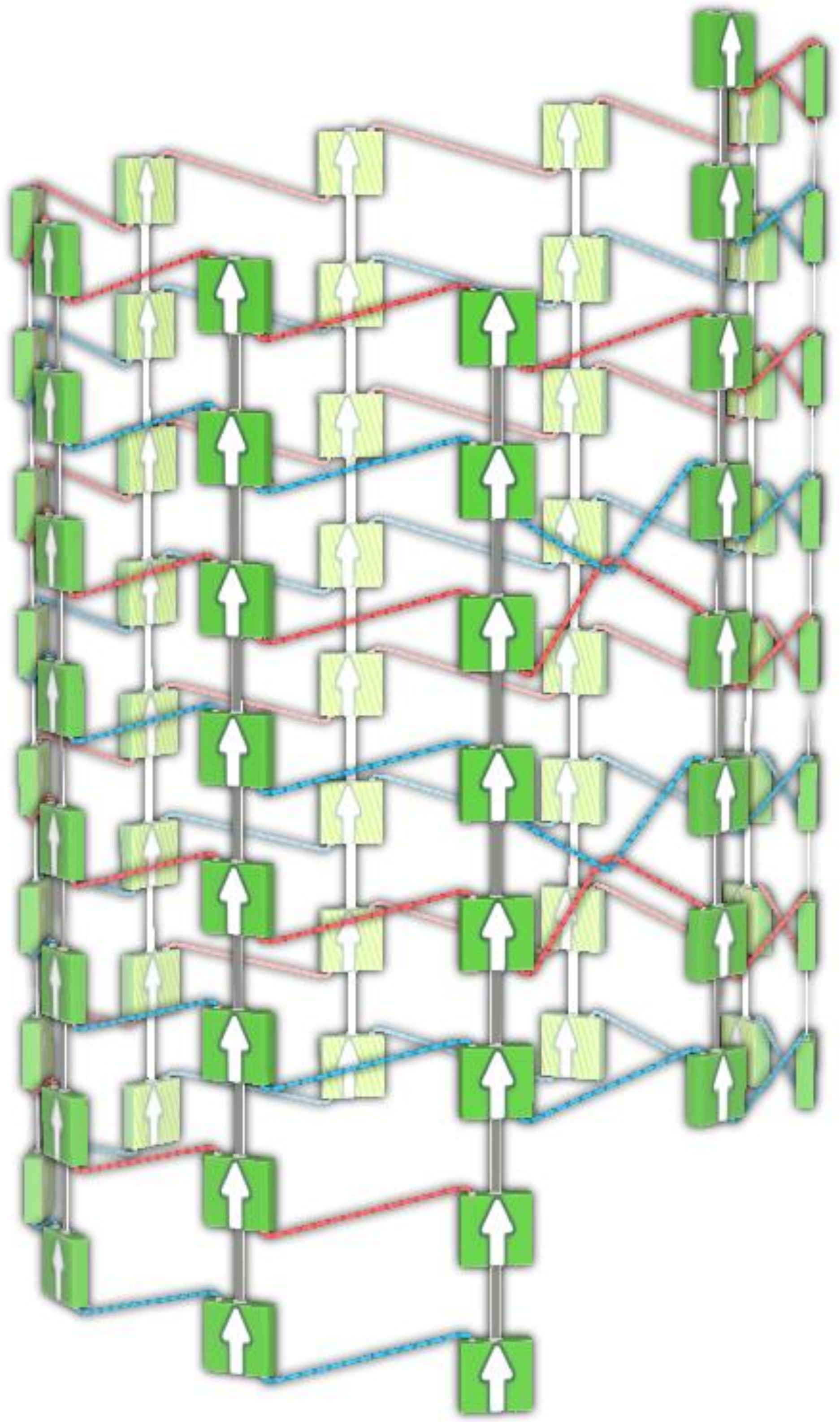
90 needles

A Formal Semantics for Machine Knitting Programs

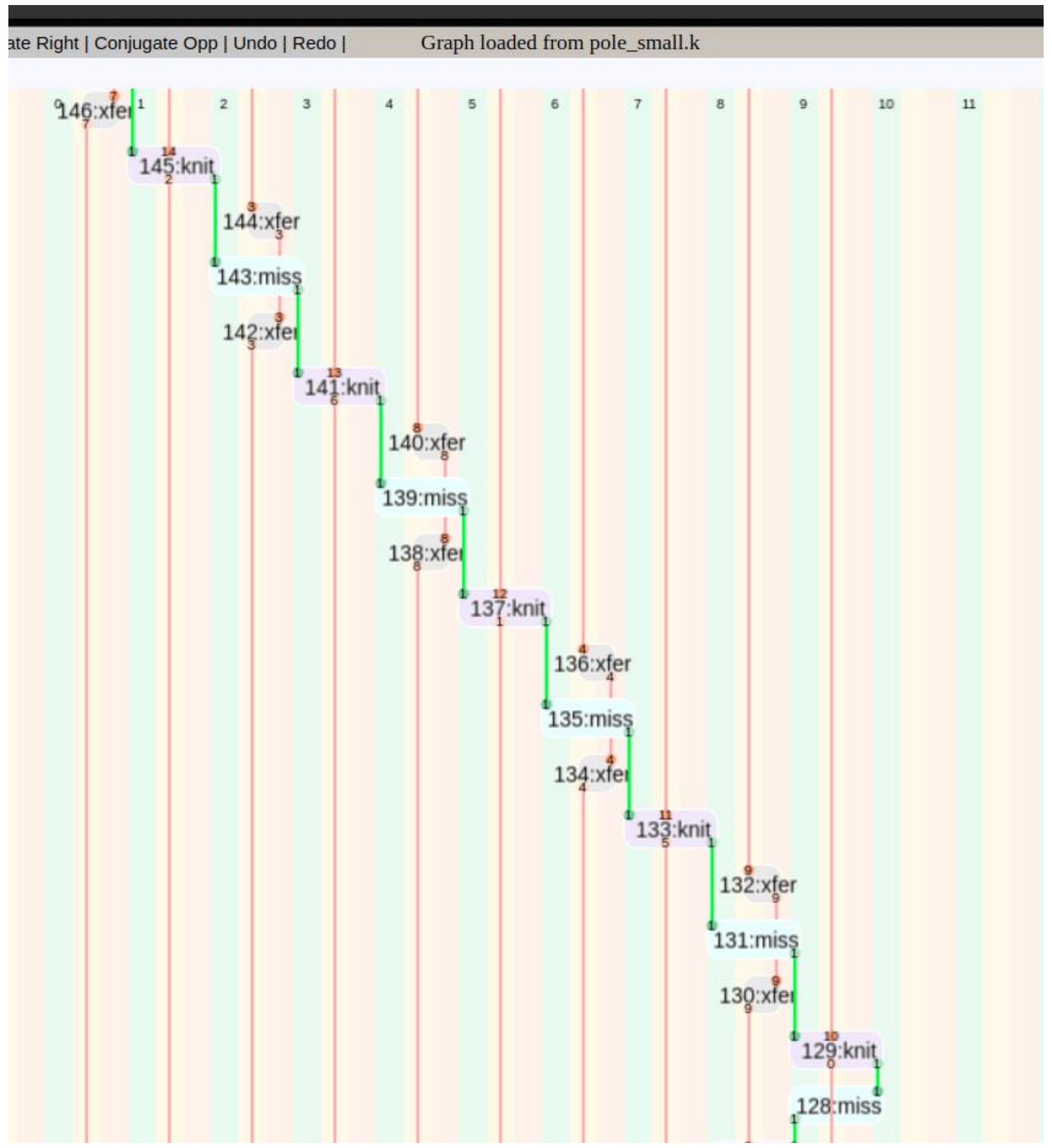


- First denotational semantics that applies to all machine knitting programs
- Provably correct rewrite rules enable general program optimizations

Compilation Process



Compile



Rewrites

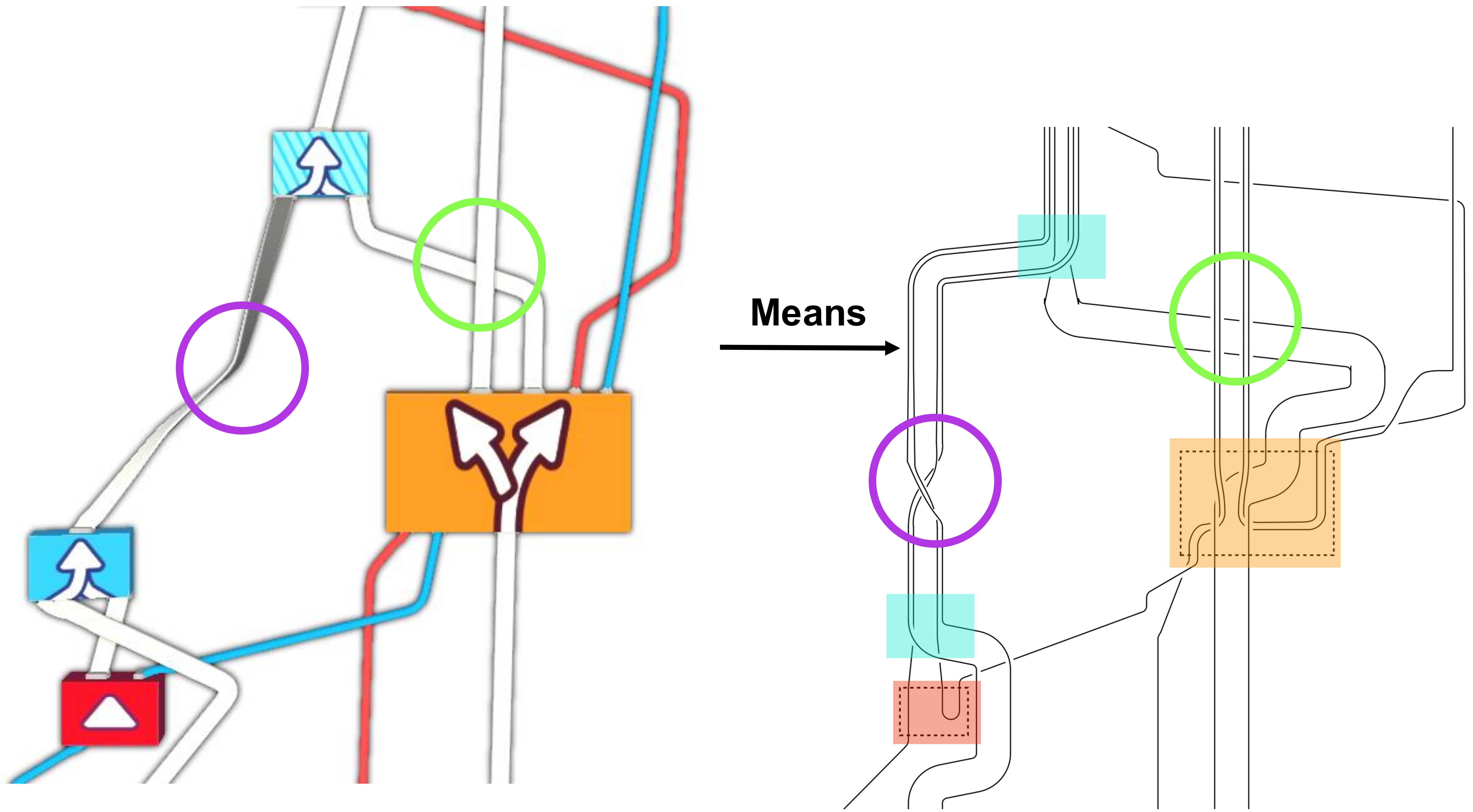
Result!

Instruction Graph

Formal Knitout

Instruction Graphs

SIGGRAPH Asia ‘24



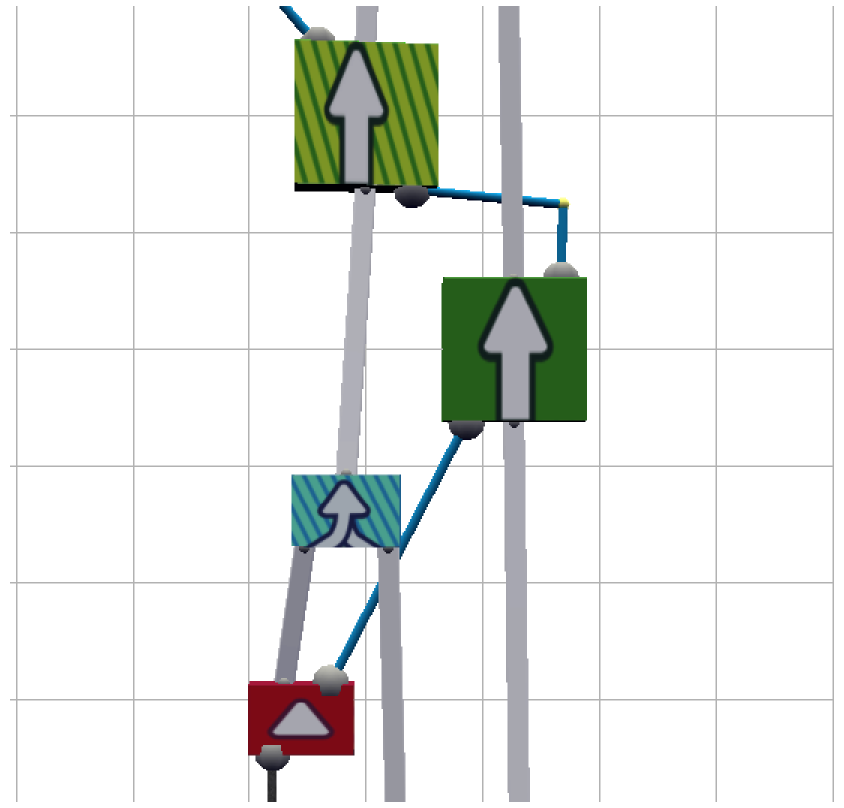
- Nodes are oriented boxes with input and output faces
- Two types of directed edges: arcs map to single paths and ribbons map to parallel bundles
- Embedding of graph in 3D space is important!

Knitting Semantics

```
xfer b.2 f.2;
knit − f.2 3.0 (2, 1.0);
xfer f.1 b.1;
miss − f.1 2;
xfer b.1 f.1;
```

Formal Knitout

Instruction Graphs



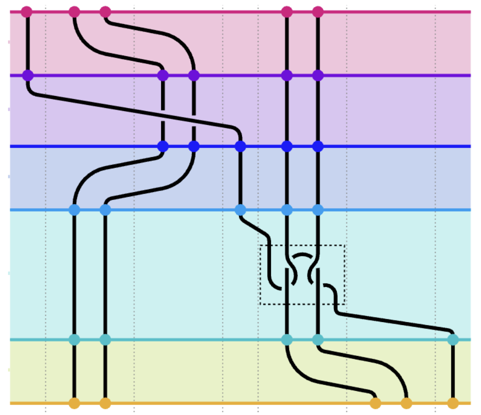
“Meaning”

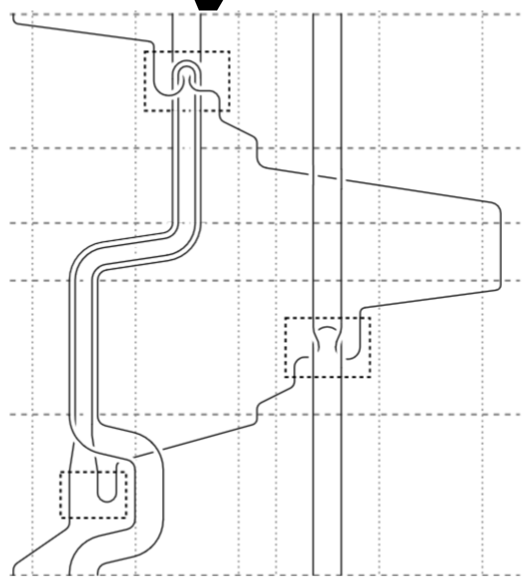
Computing

Math

Fenced Tangle

Machine Knittable



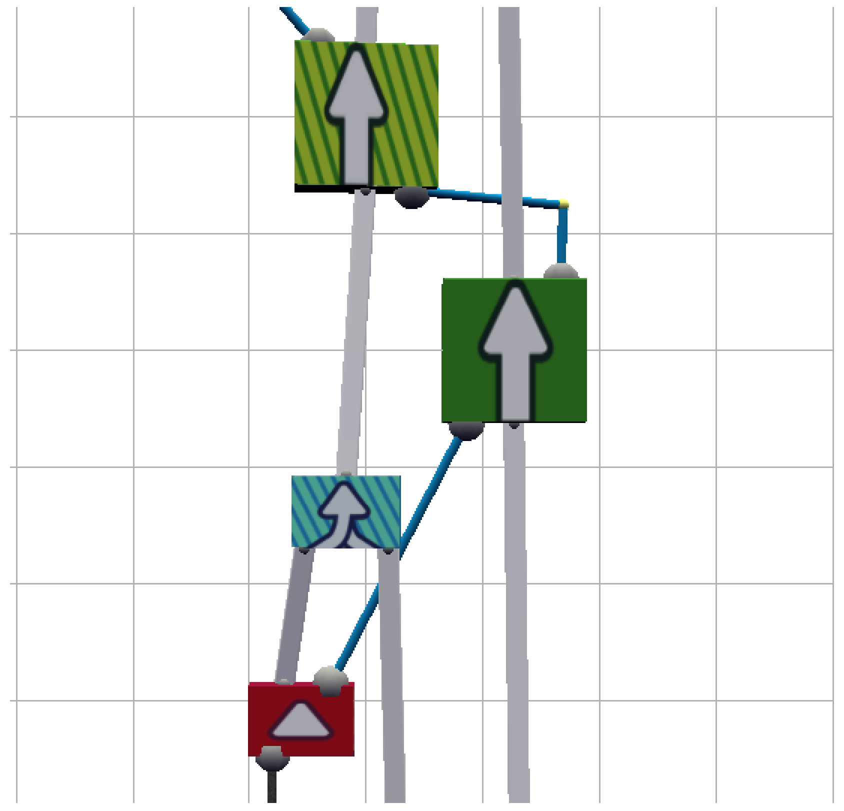


Instruction Graphable

Formal Knitout to Instruction Graphs

```
tuck + f.0 3.0 (2,1)
knit + f.1 3.0 (2,1)
xfer f.0 b.0
miss - f.1 2
knit - b.0 3.0 (2,1)
```

Formal Knitout $\xrightarrow{\phi}$ Instruction Graphs



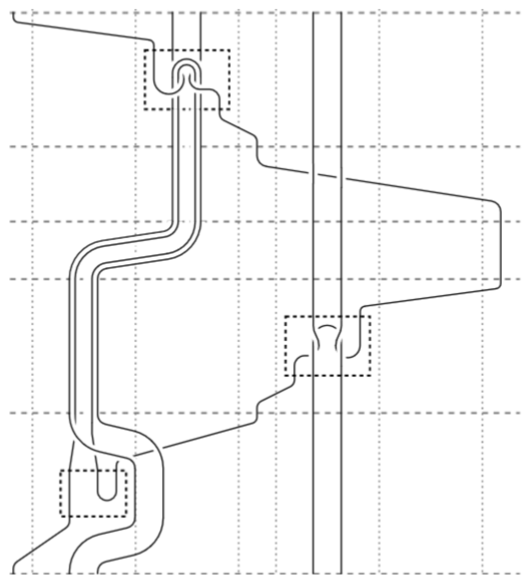
“Meaning”

Computing

Math

Fenced Tangle

Machine Knittable

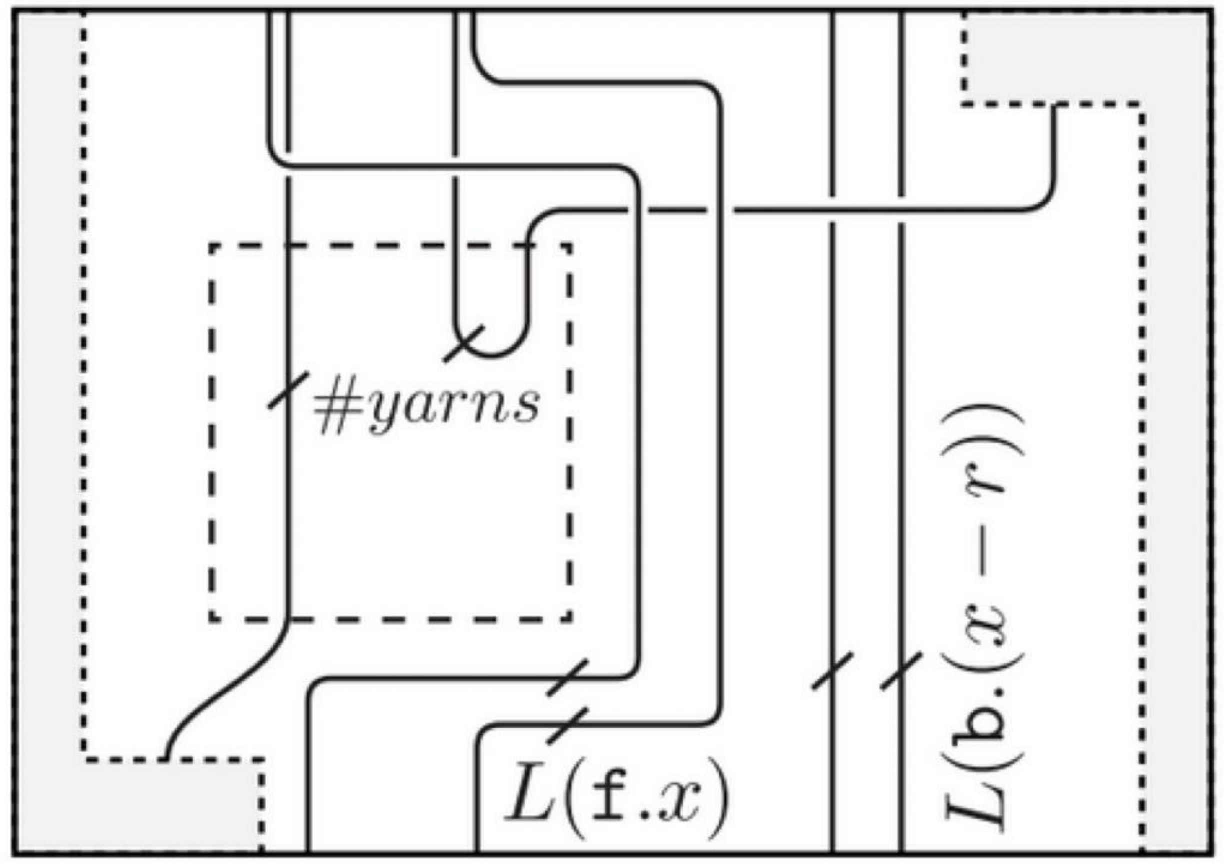


Every valid machine knitting program has an equivalent Instruction Graph

Lifting Formal Knitout to Instruction Graphs

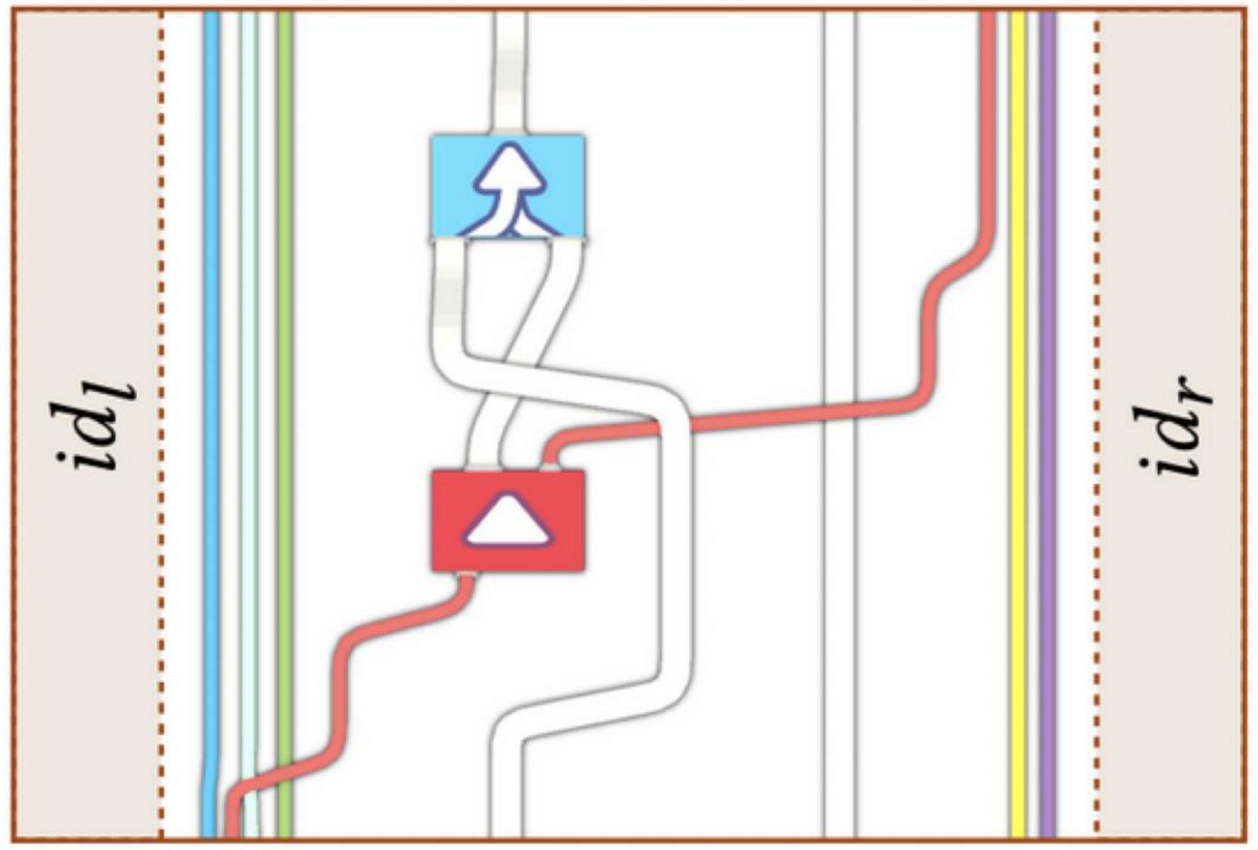
$\texttt{tuck} \; + \; \texttt{f}.x \; \texttt{b}.(x-r) \; l \; (y, s)$

$\mathcal{E}_K$



$\mathcal{E}_K(ks)$

ϕ



$\phi(ks)$

Instruction Graphs to Formal Knitout

```
tuck + f.0 3.0 (2,1)
knit + f.1 3.0 (2,1)
xfer f.0 b.0
miss - f.1 2
knit - b.0 3.0 (2,1)
```

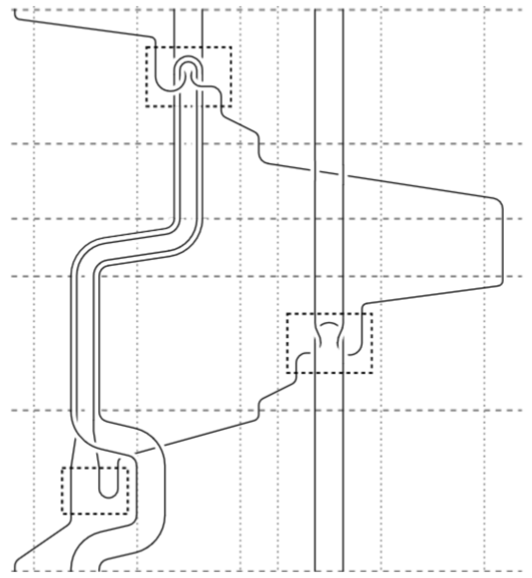
Formal Knitout ← ??? ← Instruction Graphs

Computing

Math

Fenced Tangle

Machine Knittable



???

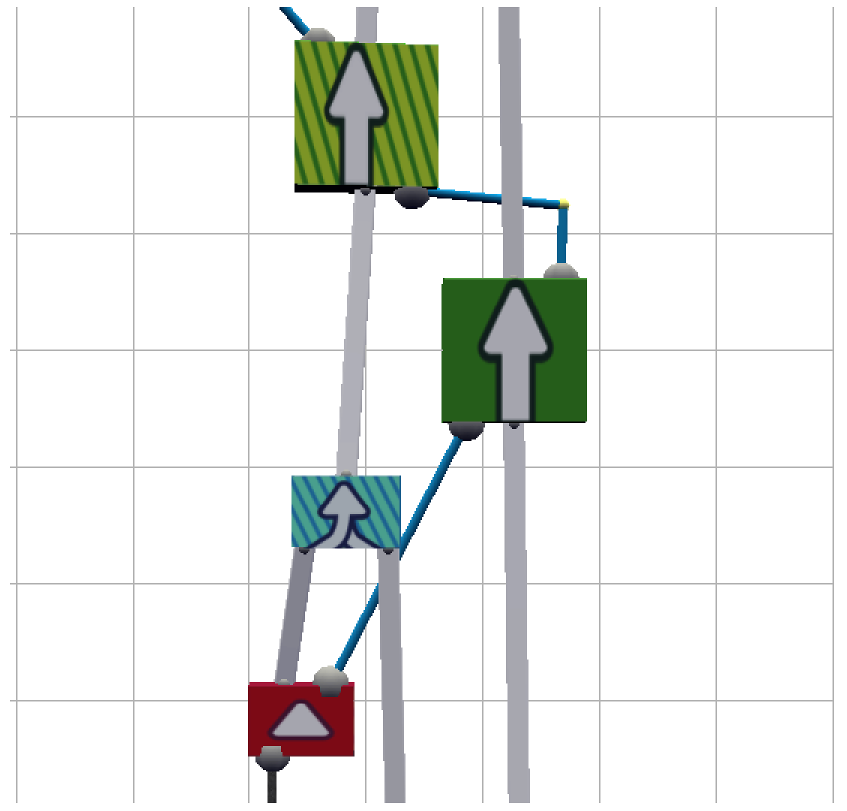
Instruction Graphs to Formal Knitout

```
tuck + f.0 3.0 (2,1)
knit + f.1 3.0 (2,1)
xfer f.0 b.0
miss - f.1 2
knit - b.0 3.0 (2,1)
```

Formal Knitout

$\mathcal{L}$

UFO Instruction Graphs

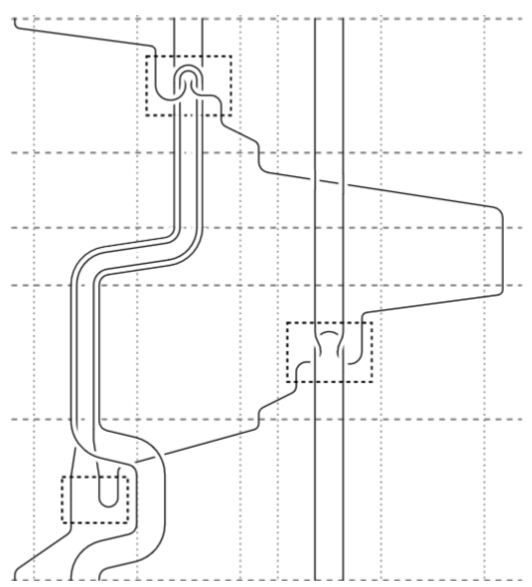


Computing

Math

Fenced Tangle

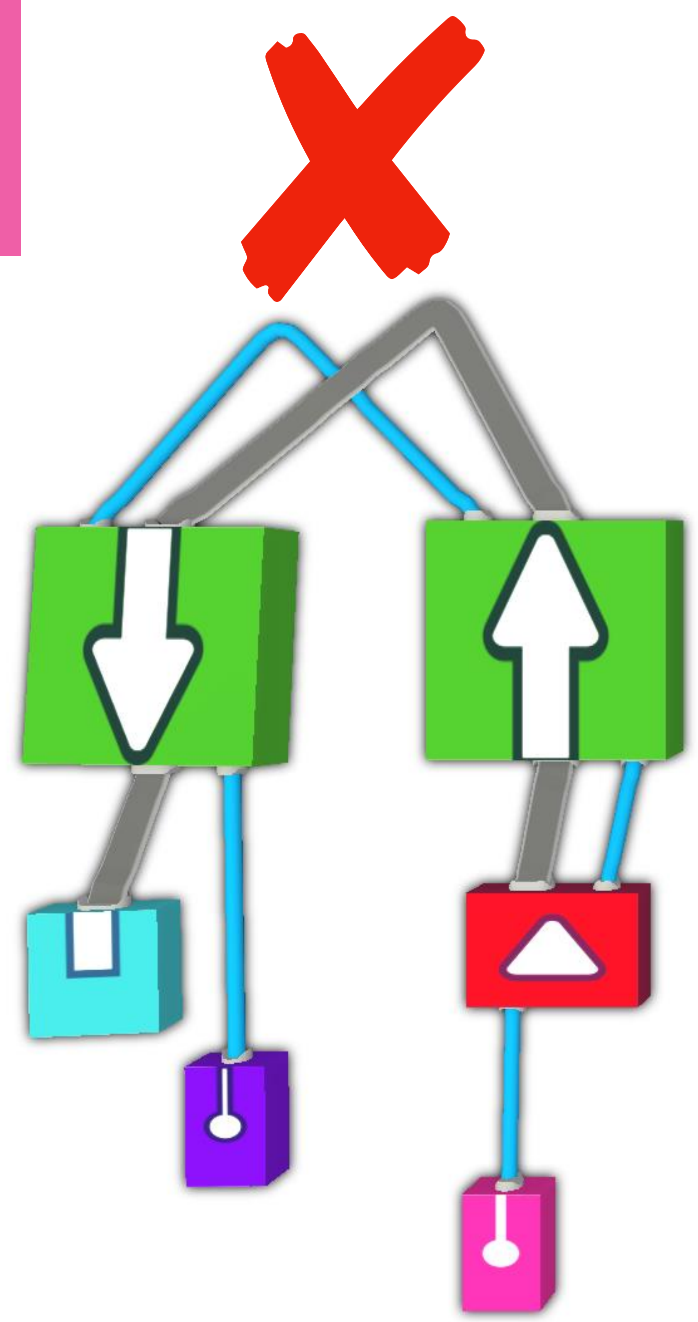
Machine Knittable

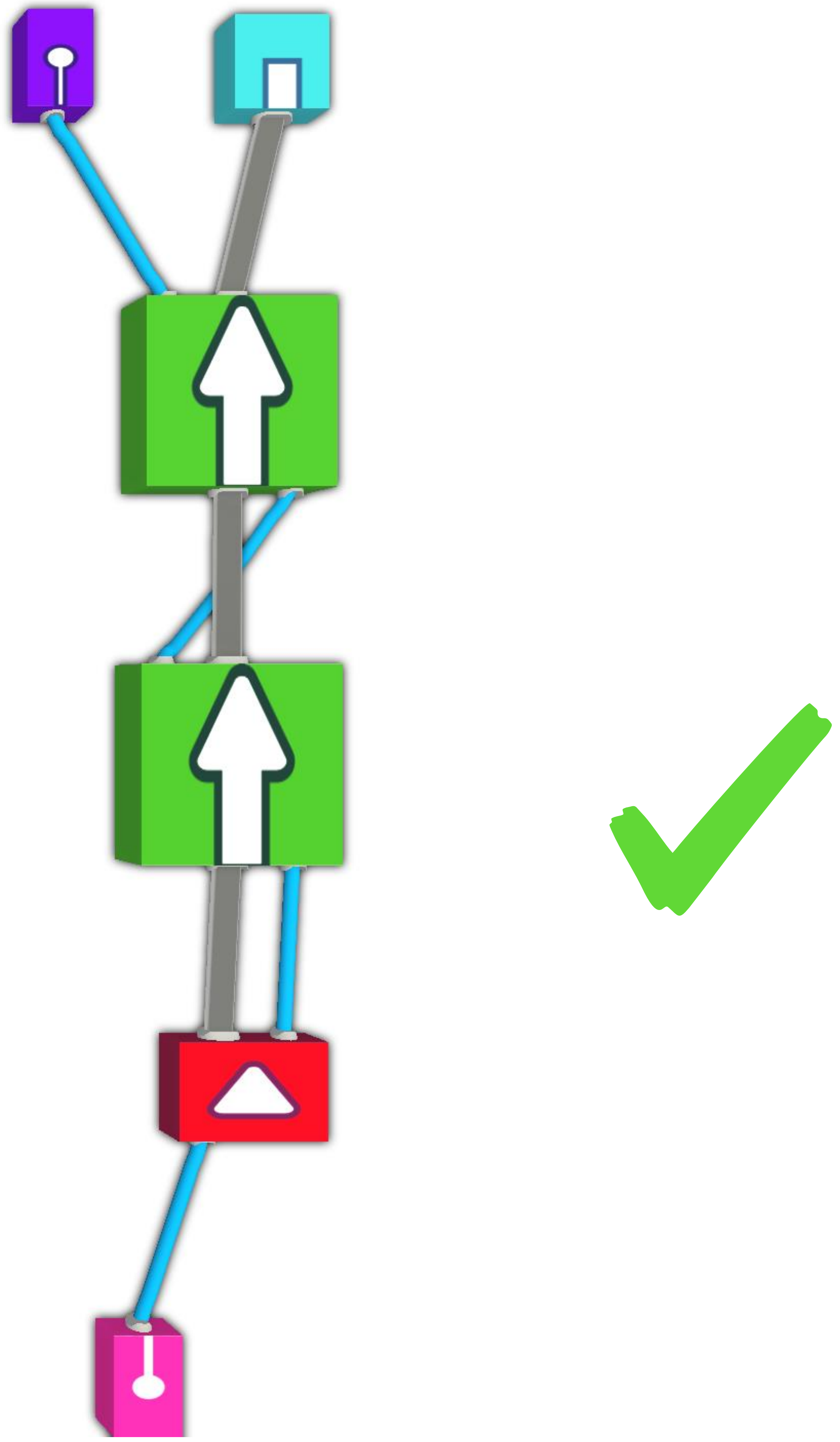


Every UFO Instruction Graph has an equivalent formal knitout program

What is Machine Knitable?

Knitting machine operations are performed sequentially



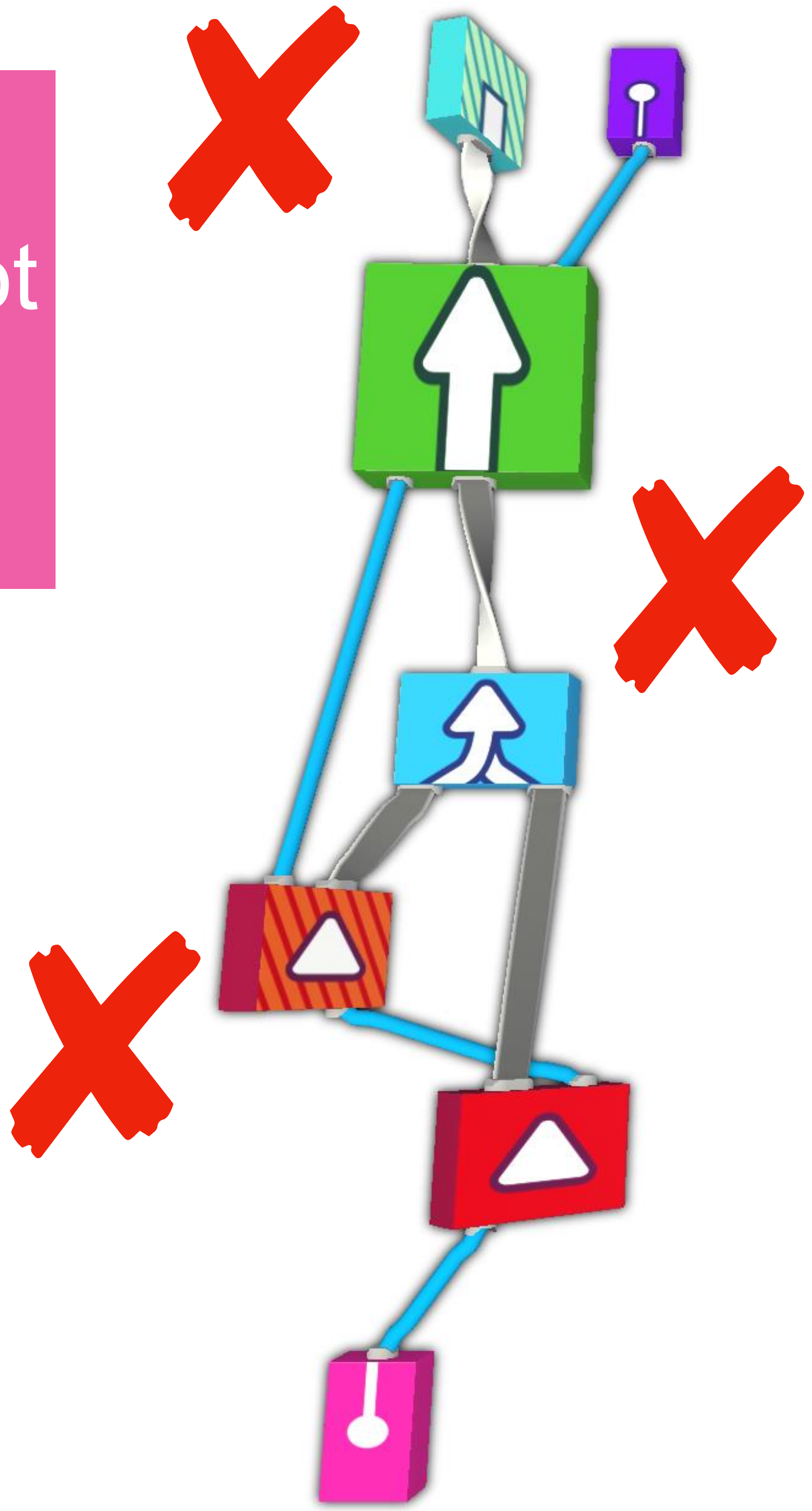


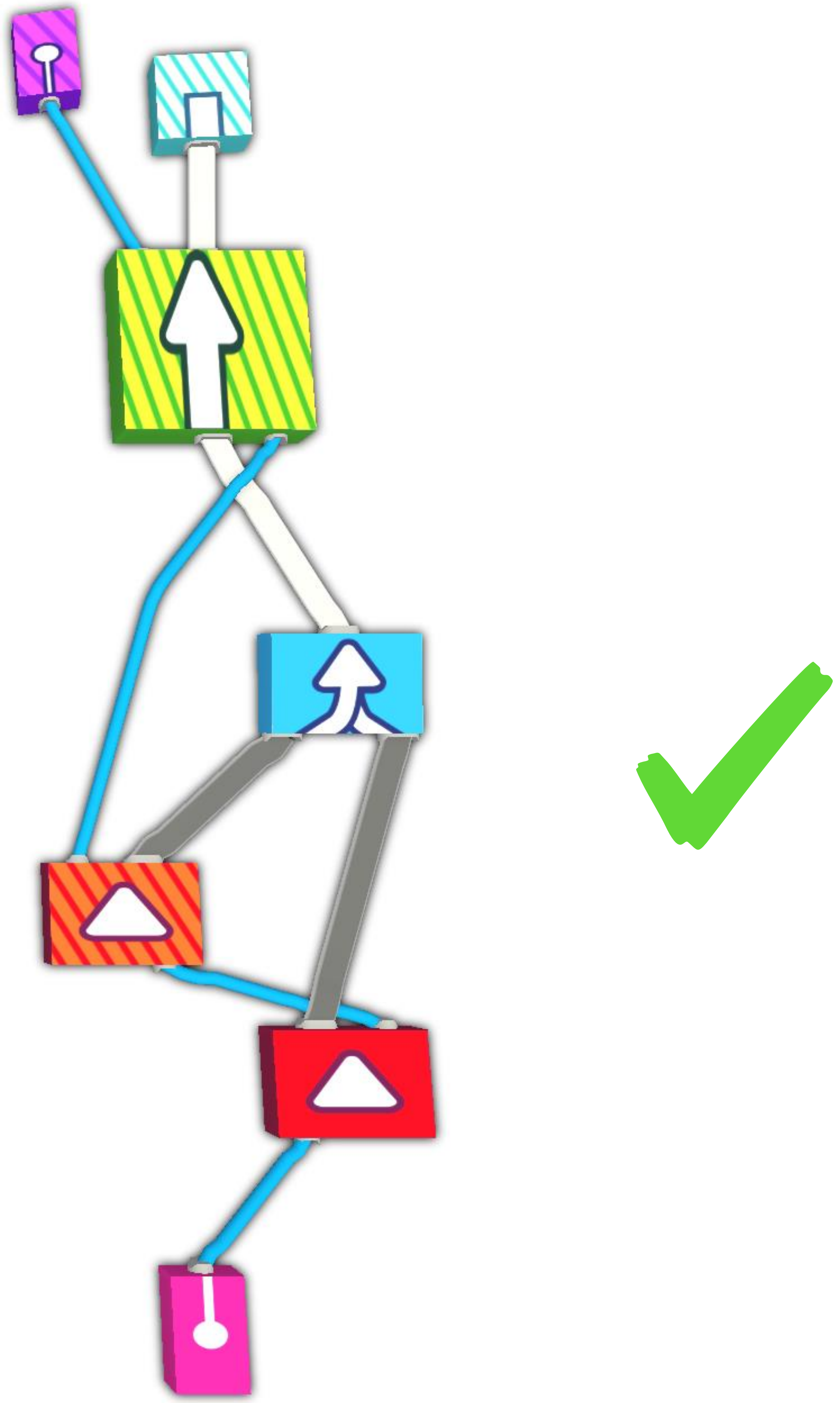
Upward

- **All arcs and ribbons only move up**
- **Each node is pointing up**

What is Machine Knitable?

Knitting machines cannot rotate loops



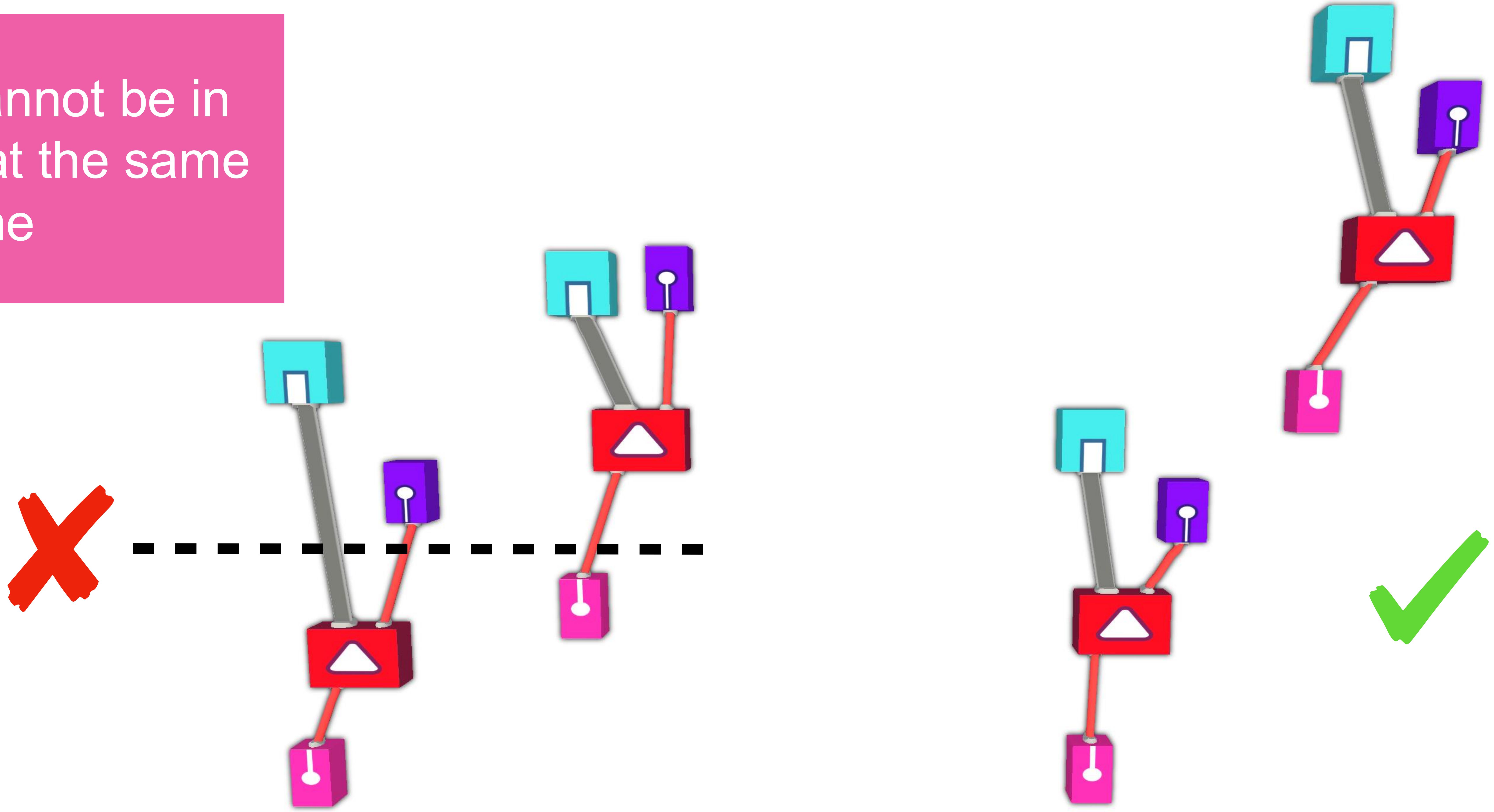


Forward

- **We only see the front or back of nodes**
- **We only see the front or back of ribbons**

What is Machine Knitable?

A carrier cannot be in two places at the same time

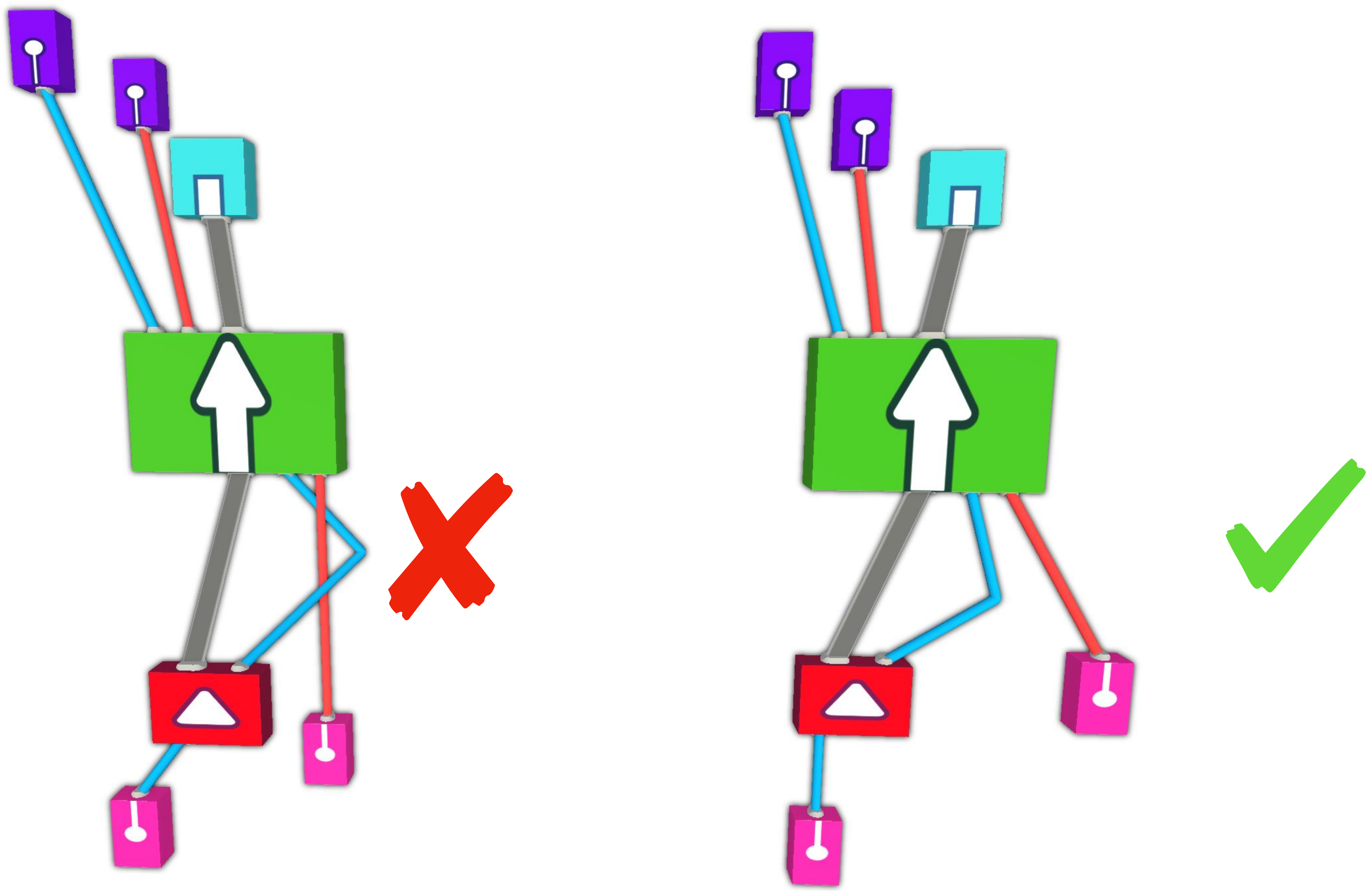


Ordered

- **Carrier ids are unique along the vertical axis**

What is Machine Knitable?

Carriers run on fixed rails and cannot twist around each other



Ordered

- **If two arcs cross each other, the smaller carrier id goes in front**

Lowering UFO IG to knitout

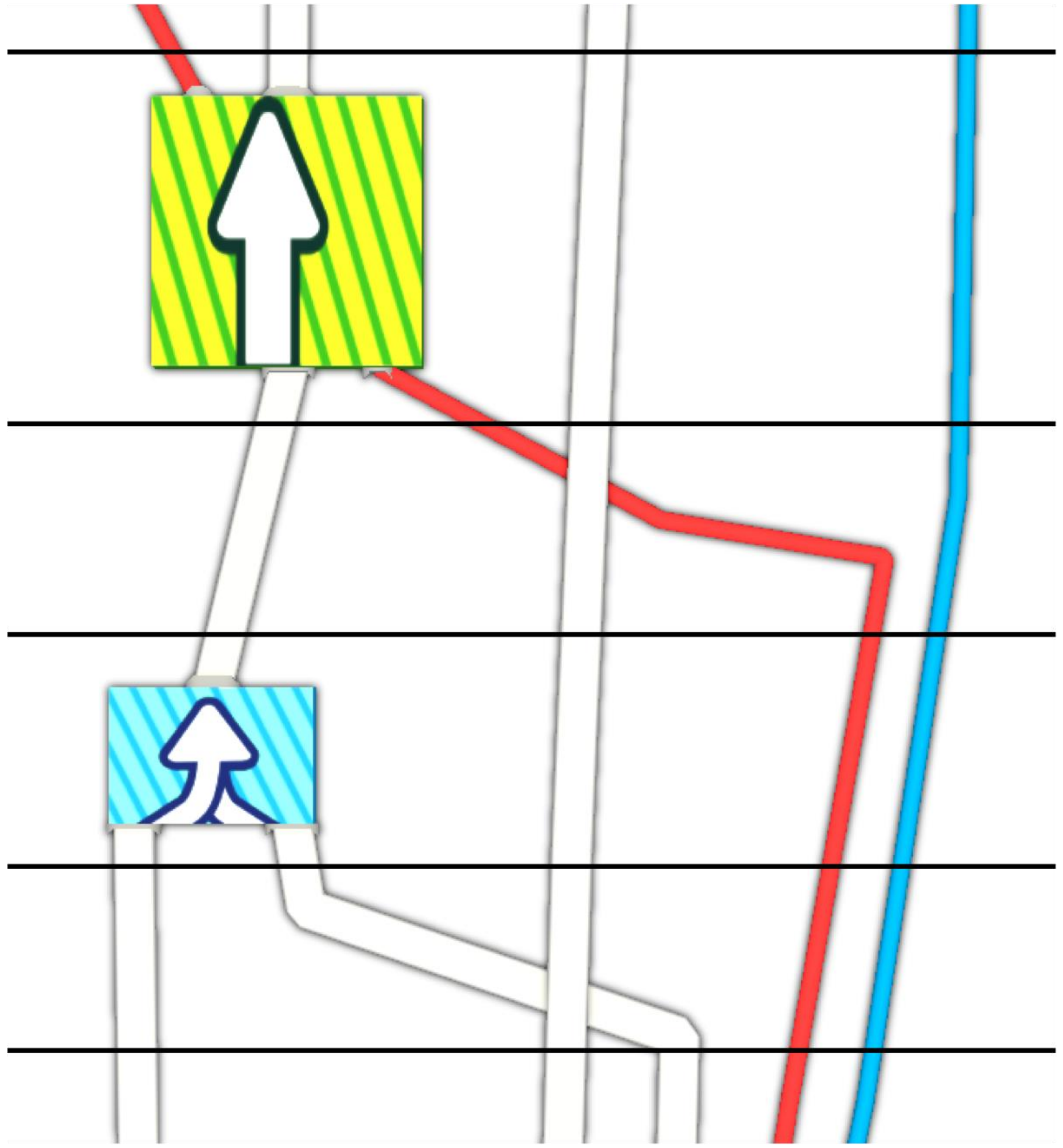
Single event partitions

G_7

G_6

G_5

G_4



Per-event compilation

$\xrightarrow{\mathcal{L}}$

$\mathcal{L}(G_4)$

```
xfer f.2 b.2
xfer f.1 b.1
rack -1
xfer b.2 f.1
rack 0
rack 1
xfer f.1 b.1
rack 0
```

$\mathcal{L}(G_5)$

```
xfer f.1 b.1
rack -1
xfer b.1 f.0
rack 0
xfer f.2 b.2
rack -1
xfer b.2 f.1
rack 0
```

$\mathcal{L}(G_6)$

```
miss - f.1 2
```

$\mathcal{L}(G_7)$

```
xfer f.0 b.0
knit - b.0 1 (2,1)
xfer b.0 f.0
```

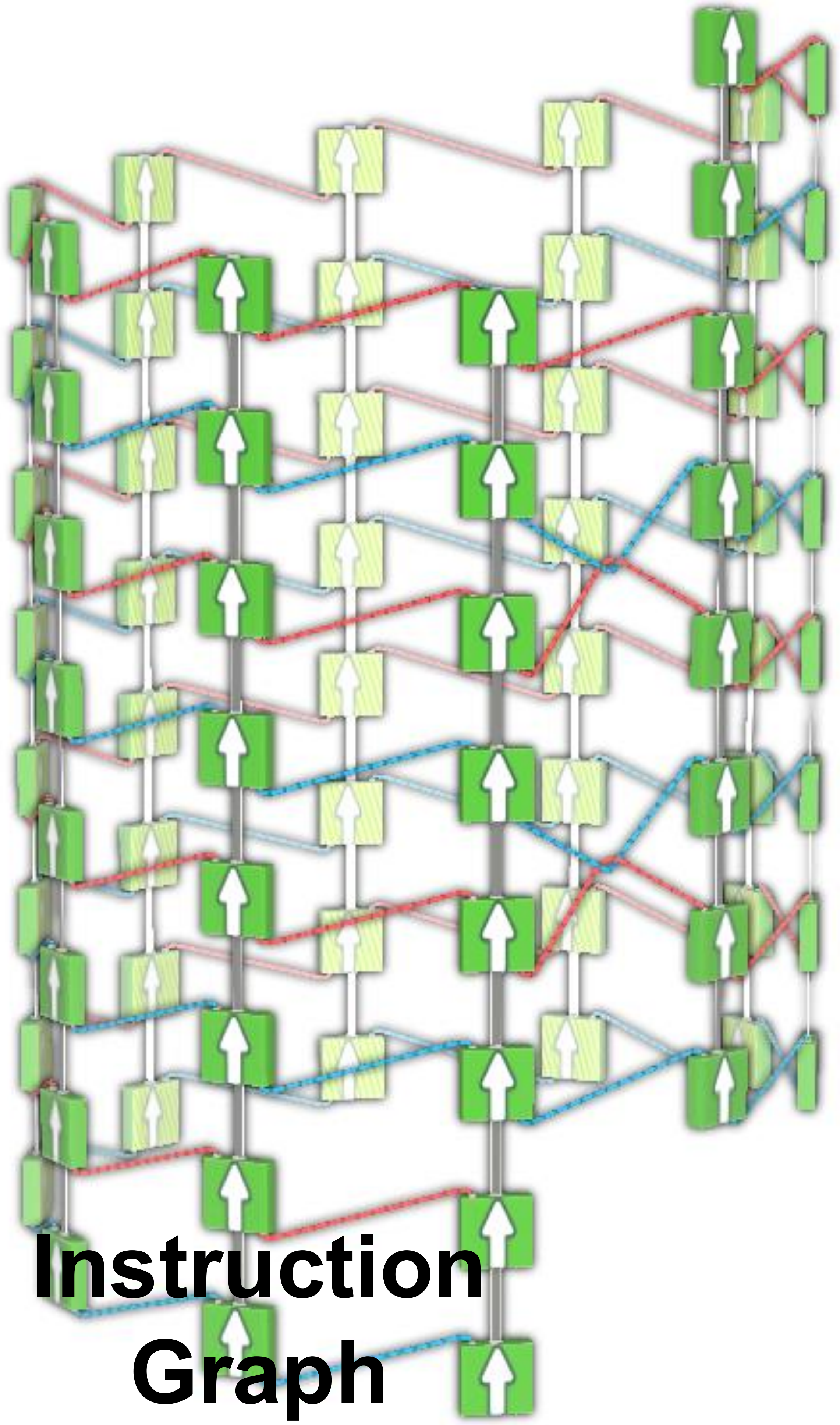
Per-event compilation preserves topological equivalence

$$\mathcal{E}_G[G] \cong \mathcal{E}_K[\mathcal{L}(G)]$$

Instruction graph composition implies knitout composition

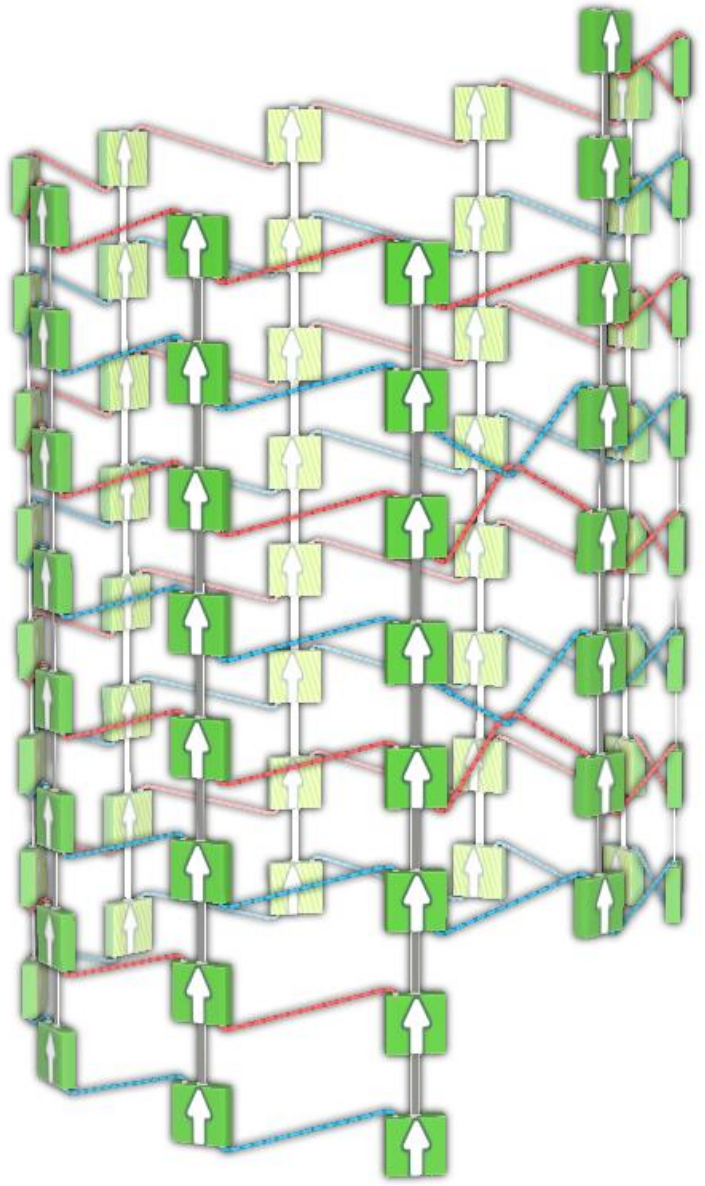
$$G_1 \circ G_2 \rightarrow \mathcal{L}(G_1); \mathcal{L}(G_2)$$

Compilation Pipeline



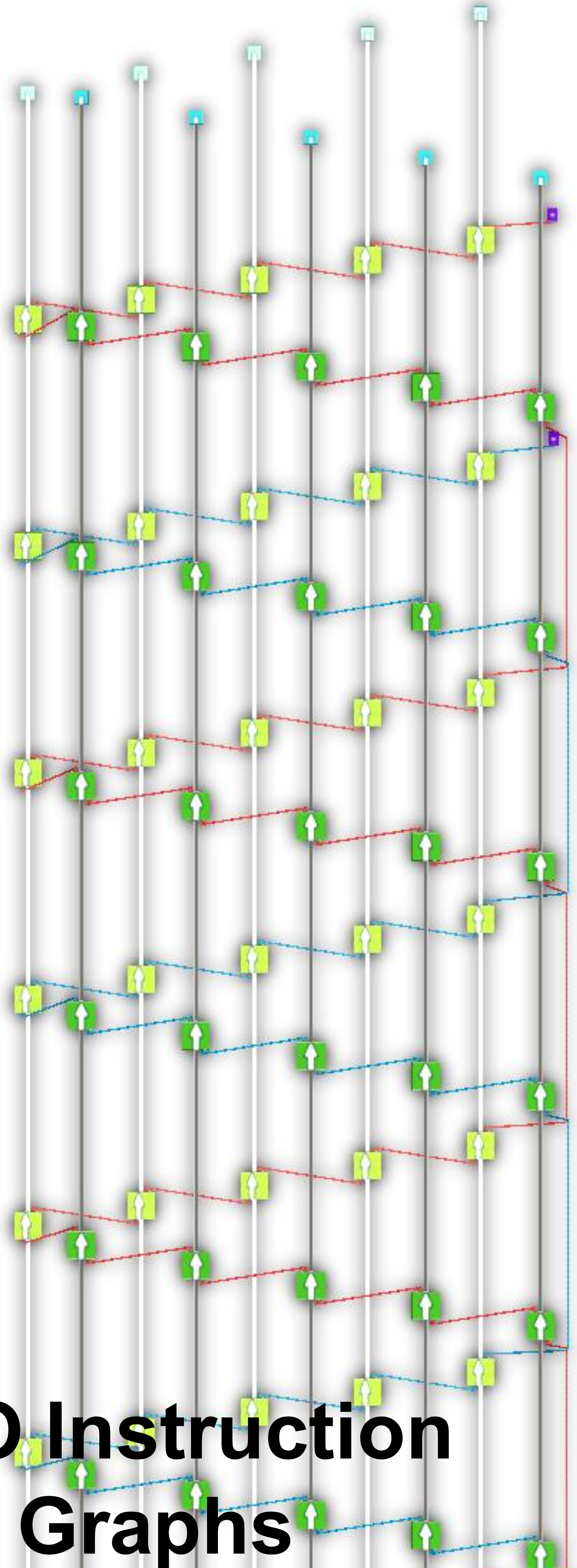
Intermediate representation that precisely describes knit topology

Compilation Pipeline



Instruction Graph

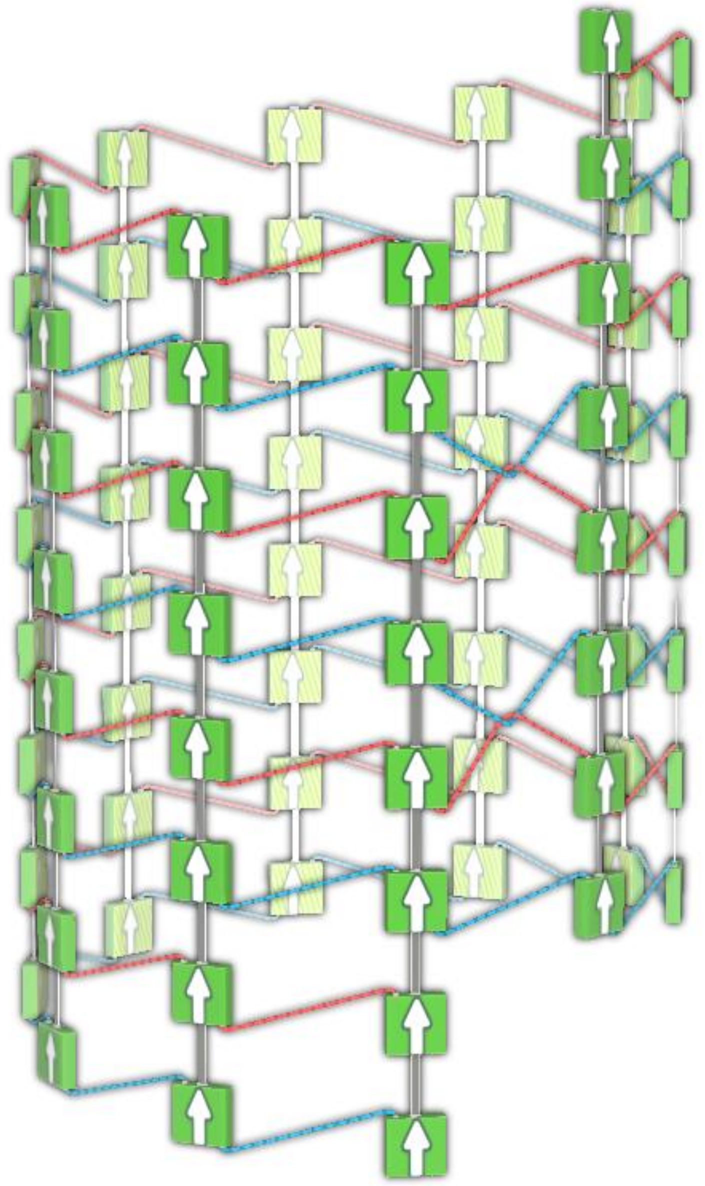
Ambient Isotopy



UFO Instruction Graphs

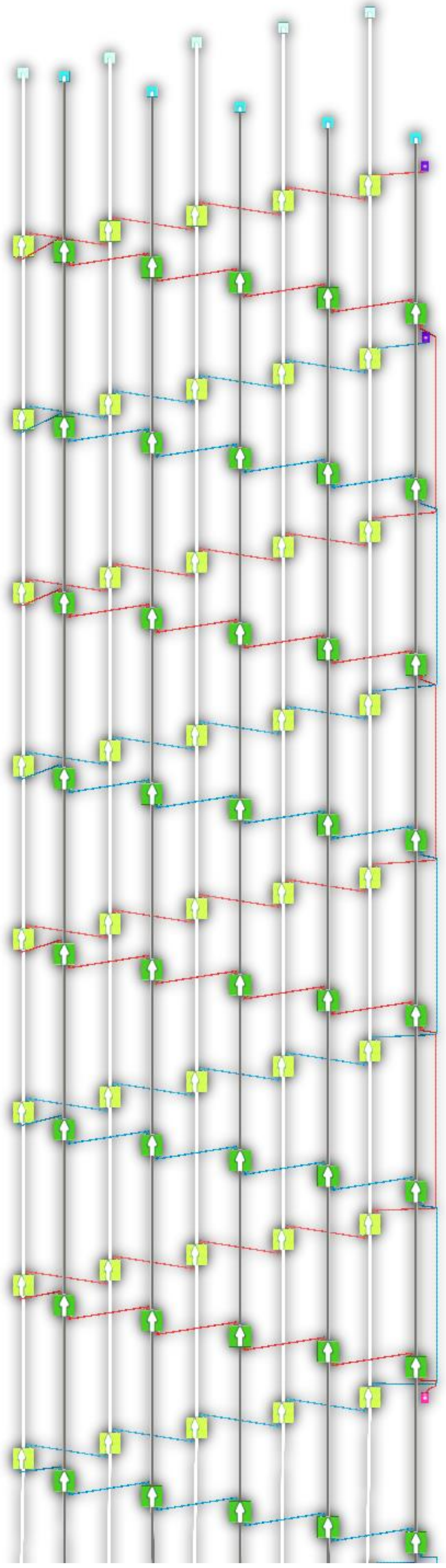
Upward, Forward, Ordered presentation guarantees a topologically equivalent program exists

Compilation Pipeline



Instruction Graph

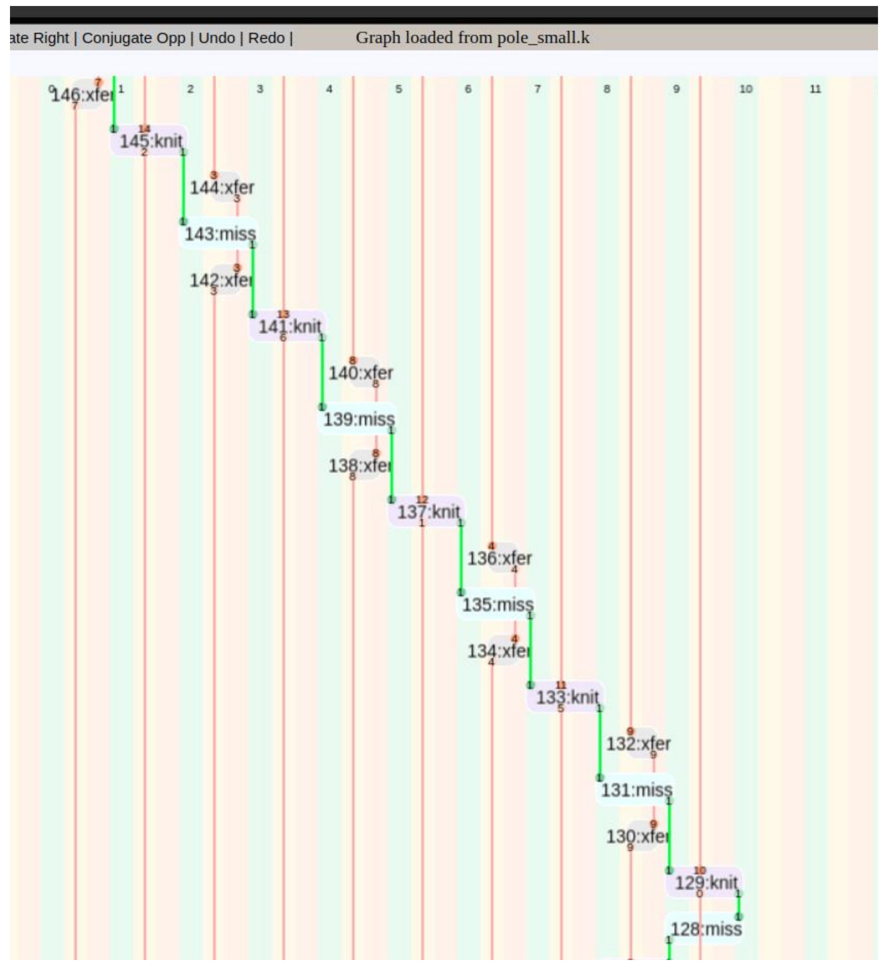
Ambient Isotopy



UFO Instruction Graphs

$\mathcal{L}$

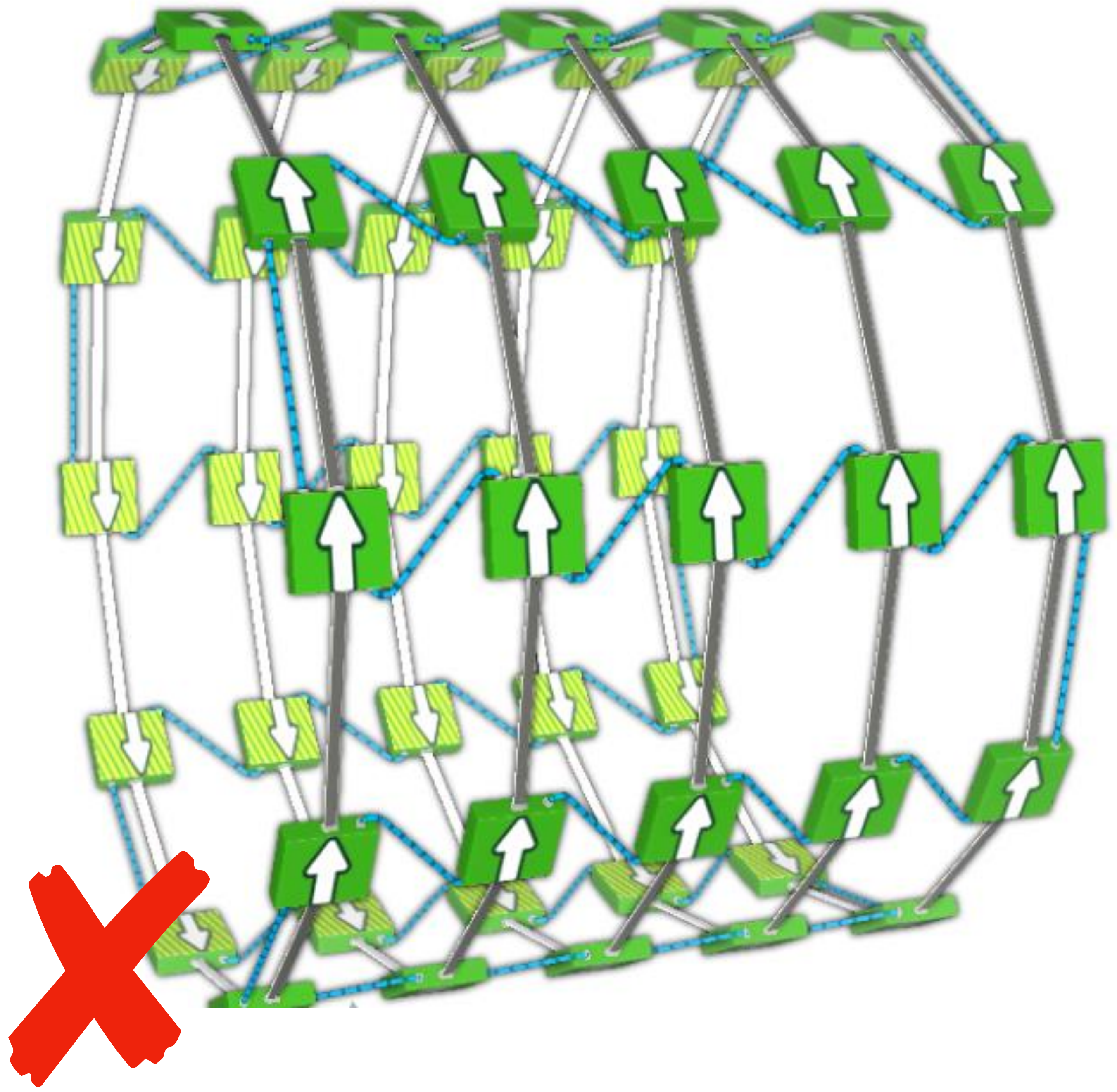
Lowering function always produces topologically equivalent program



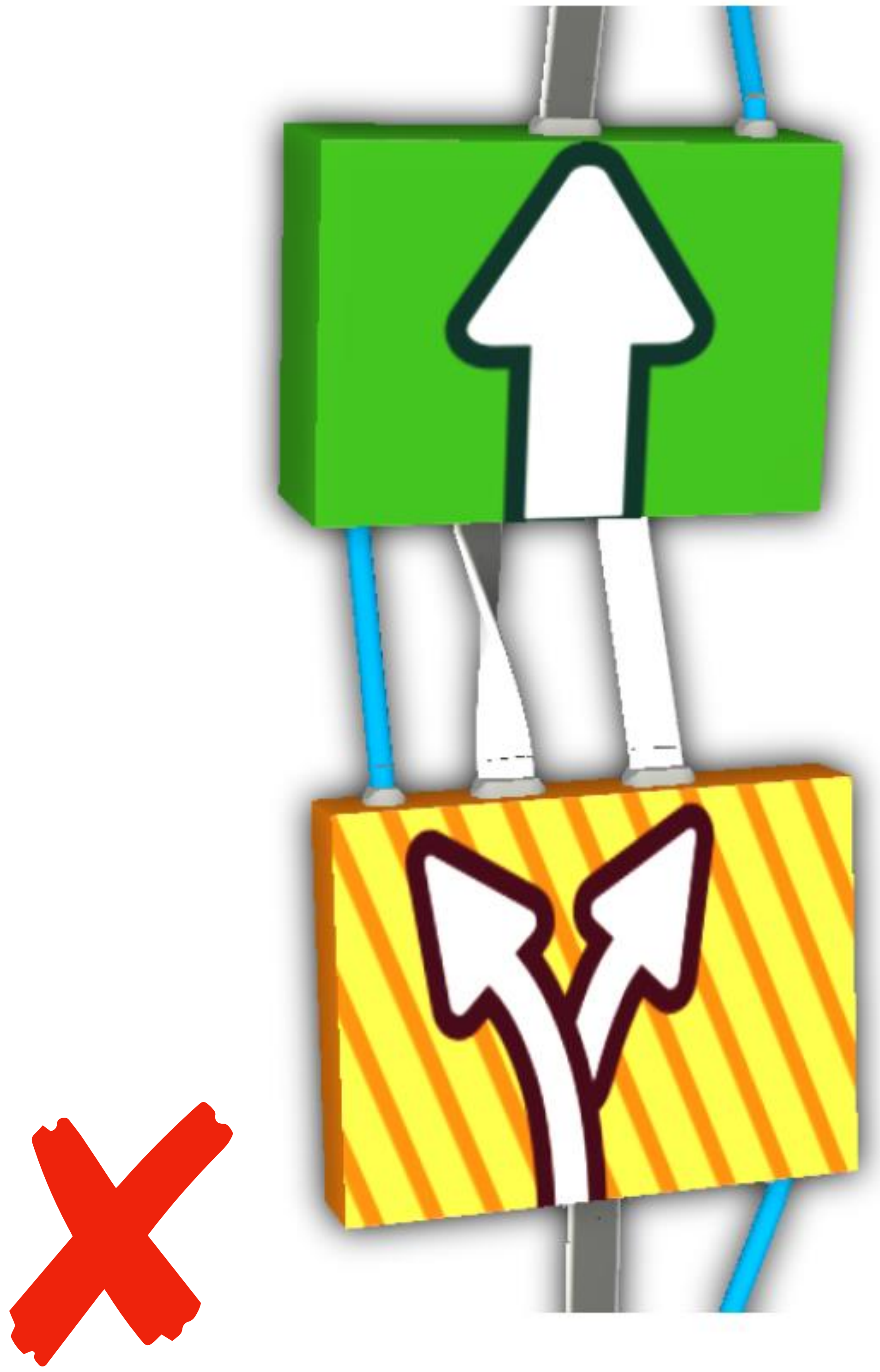
Machine Instructions

Result!

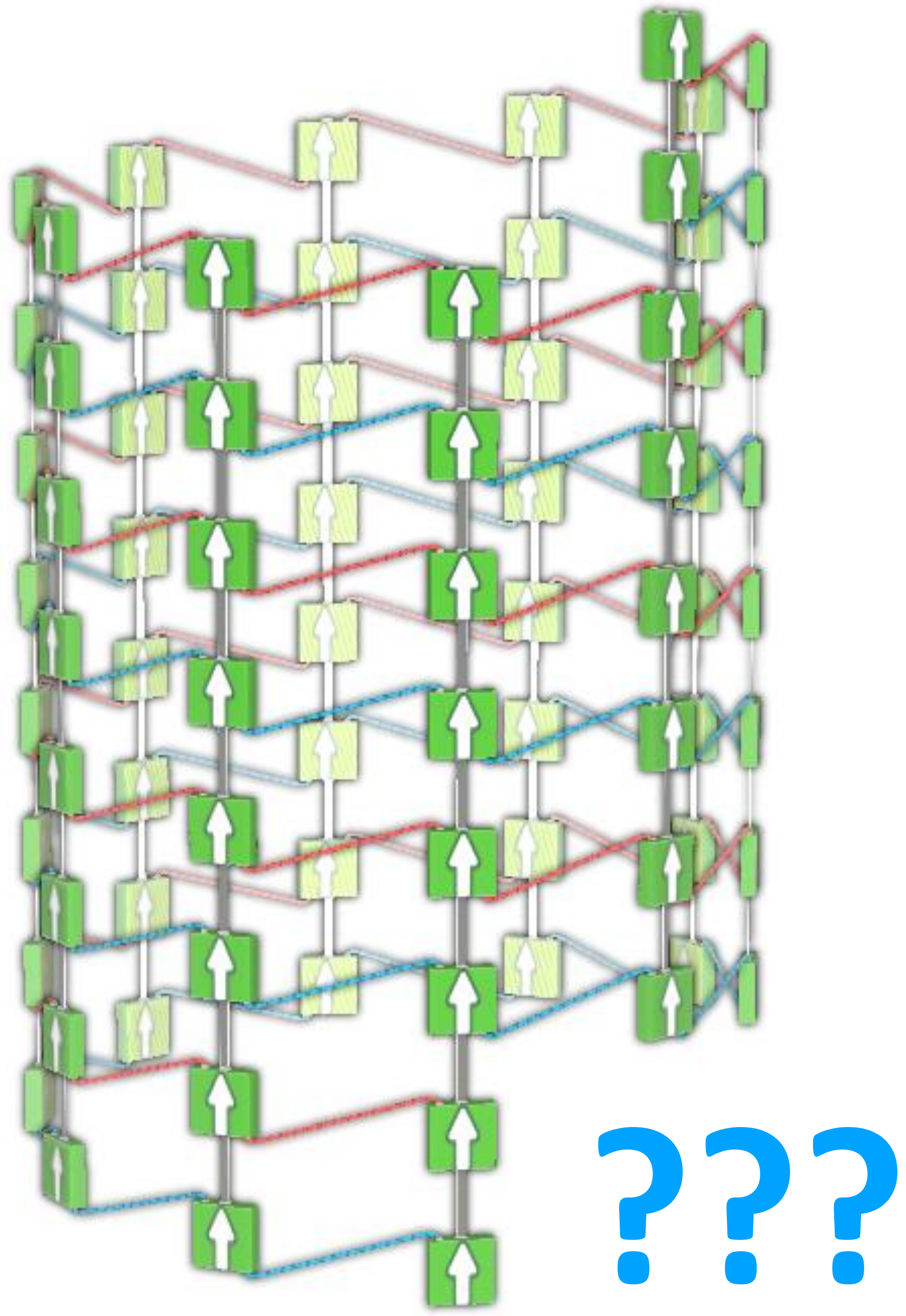
What isn't Machine Knitable?



Never Upward



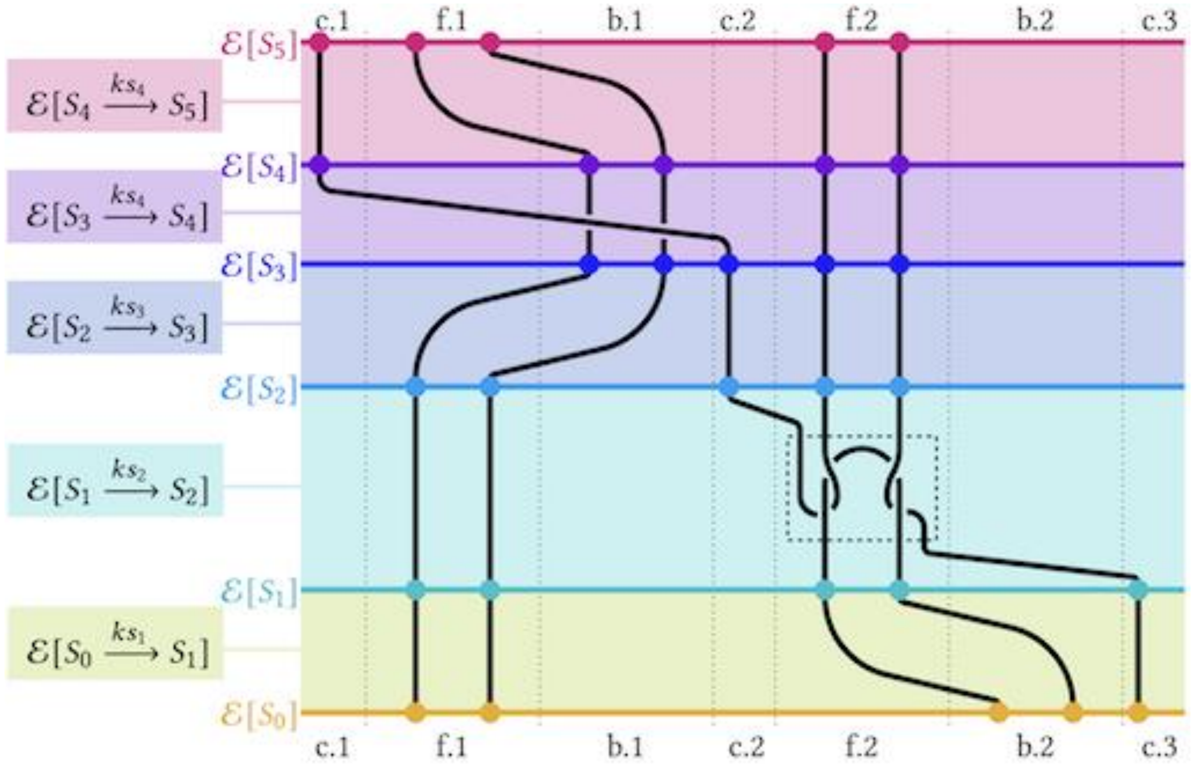
Never Forward



Never Ordered?

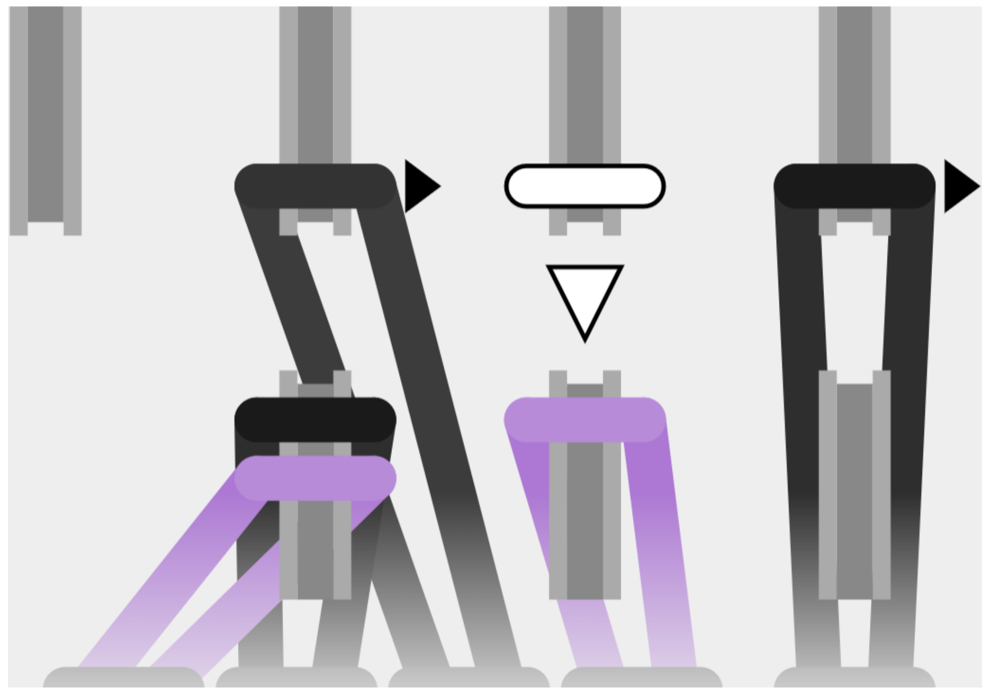
Alternative Program Semantics

Fenced Tangles



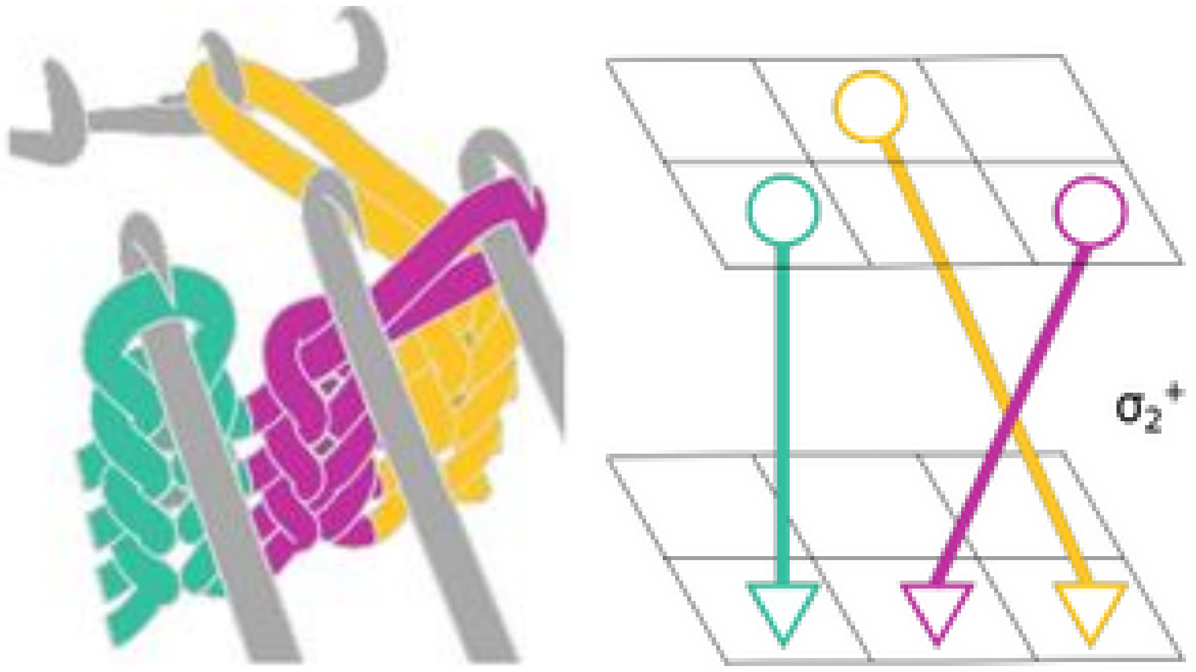
Lin, Narayanan, Ikarashi, Ragan-Kelly, Bernstein, McCann SIGGRAPH ’23

Discrete Offsets



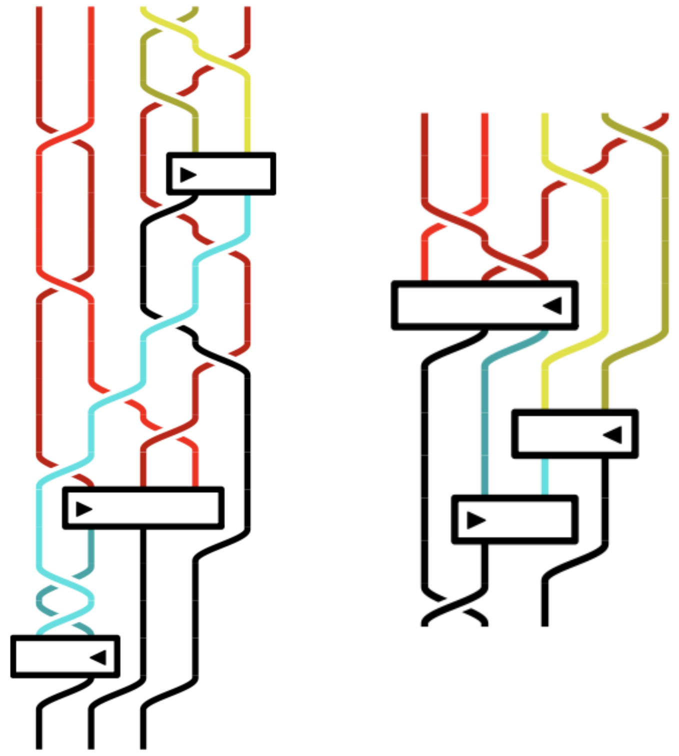
Lin, Narayanan, McCann SCF ’18

Artin Braids



Lin and McCann ICRA ’21

Monoidal Category



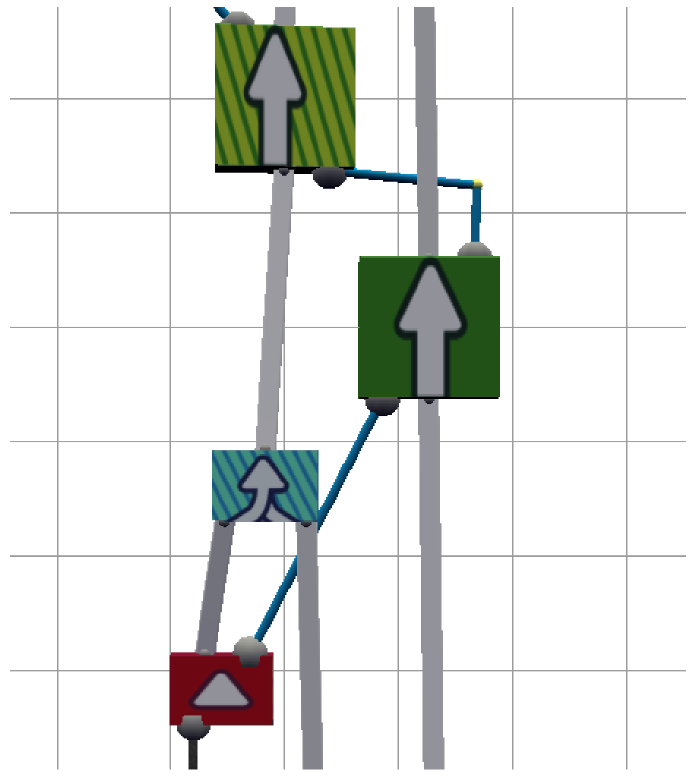
Hurtig, Lin, Price, Schulz, McCann, Bernstein ICFP '25

All of knitting, but computationally hard

Some of knitting, but computationally tractable

Can we find something fast and complete?

Compilation from Instruction Graphs to Formal Knitout



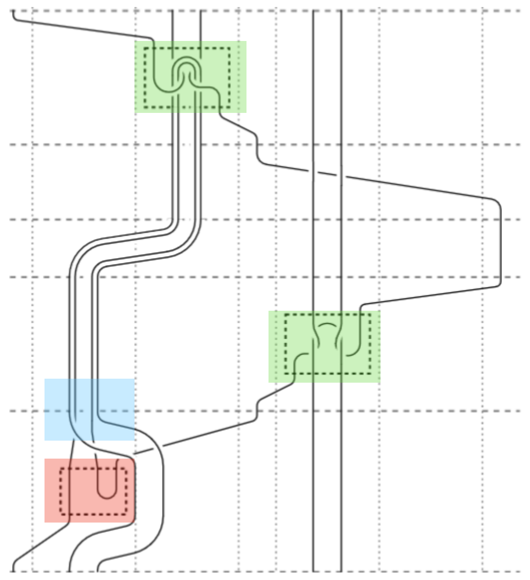
```
tuck + f.0 3.0 (2,1)
knit + f.1 3.0 (2,1)
xfer f.0 b.0
miss - f.1 2
knit - b.0 3.0 (2,1)
```

Formal Knitout ⟷ UFO Instruction Graphs

“Meaning”

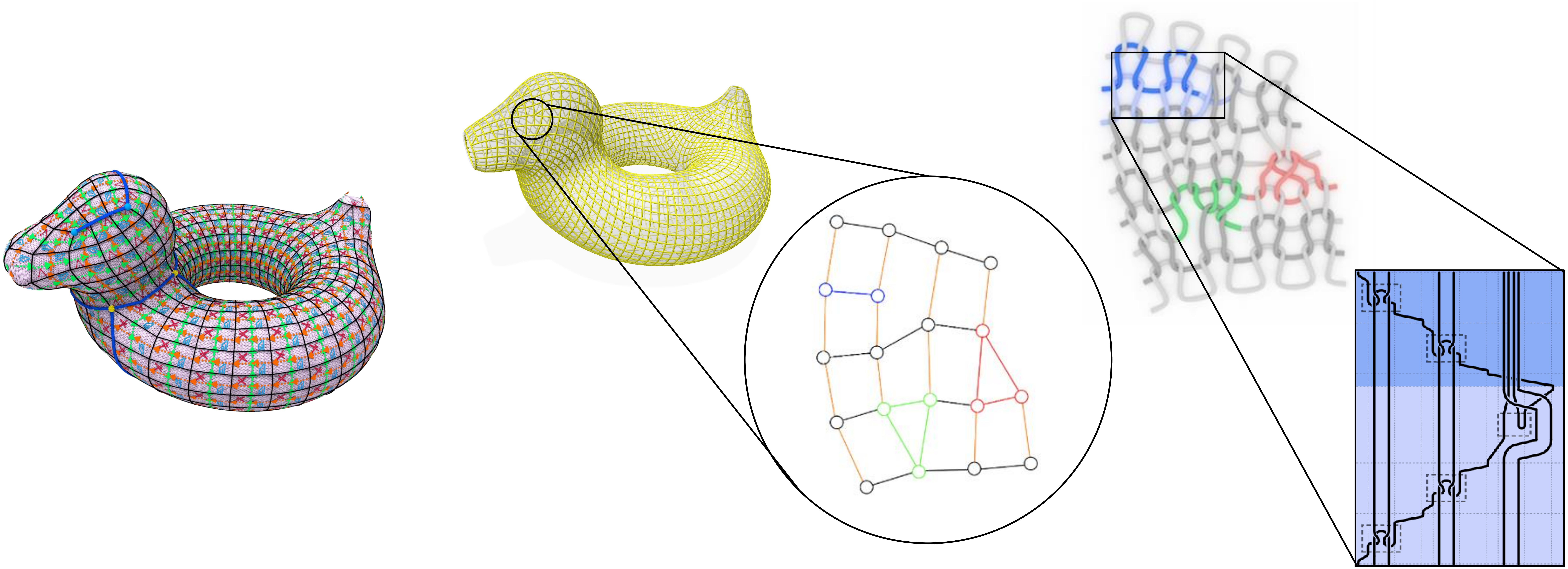
Fenced Tangle

Machine Knittable



- Instructions graphs are an intermediate representation with a compatible semantics with knitout
- UFO definition of machine knittability lays the groundwork for automatic compilation on all of machine knitting

Knitting Levels of Abstraction

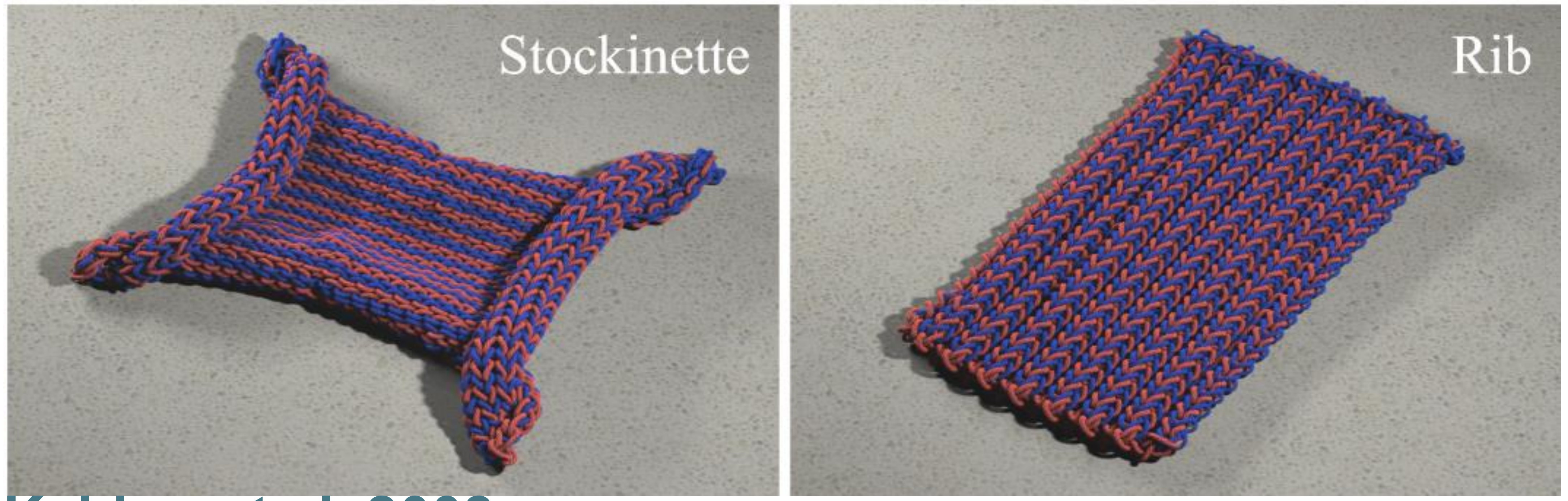


Fabric **Stitch** **Yarn**

The Fabrication Workflow

Interesting Future Work & Open Problems

Yarn-Level Simulation

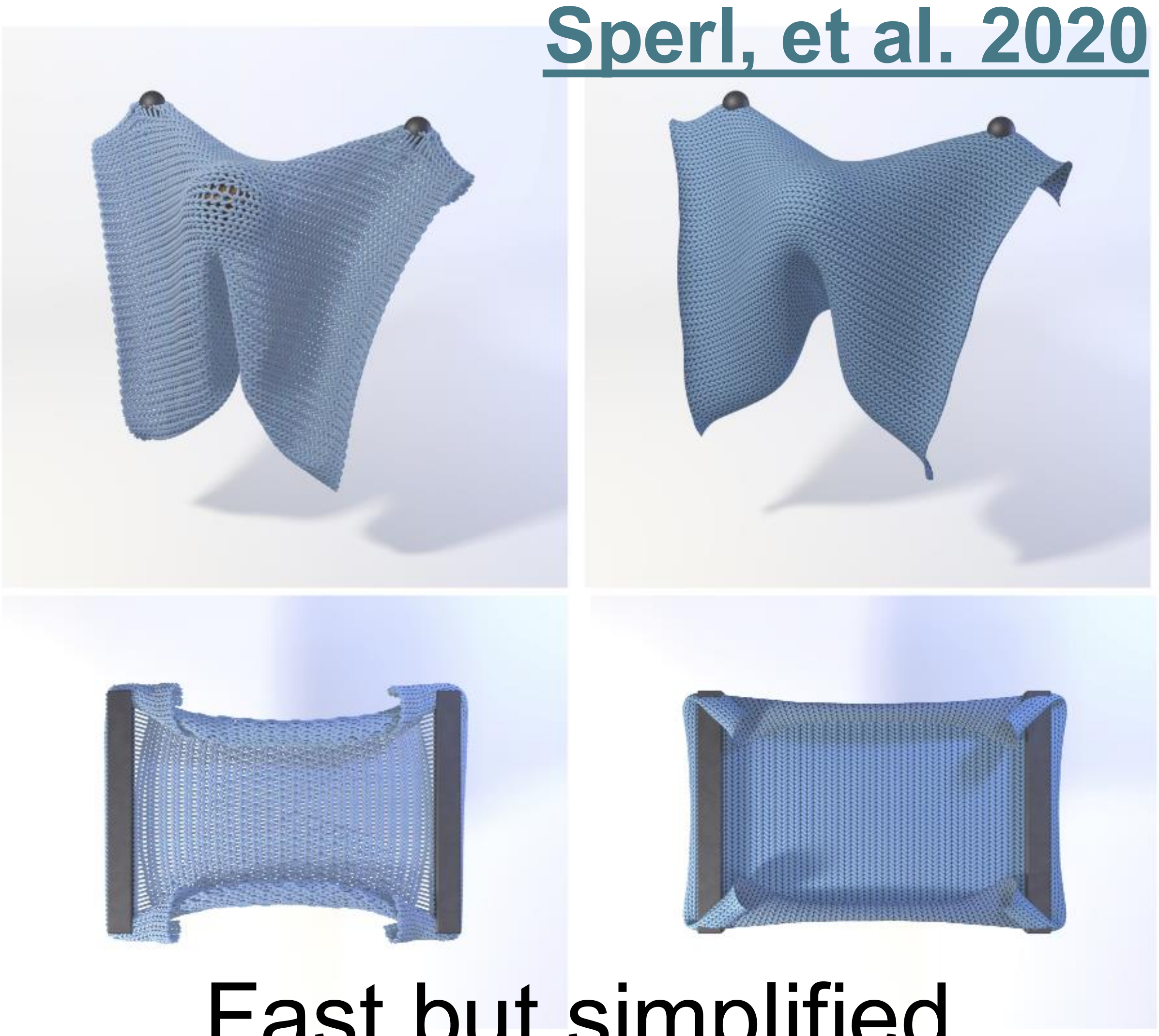


Kaldor, et al. 2008

Detailed but slow

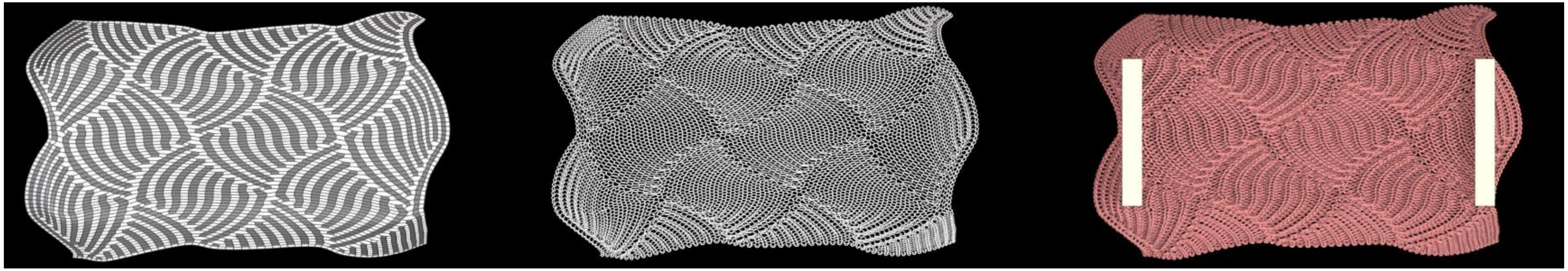
Thin-Sheet Simulation

Sperl, et al. 2020



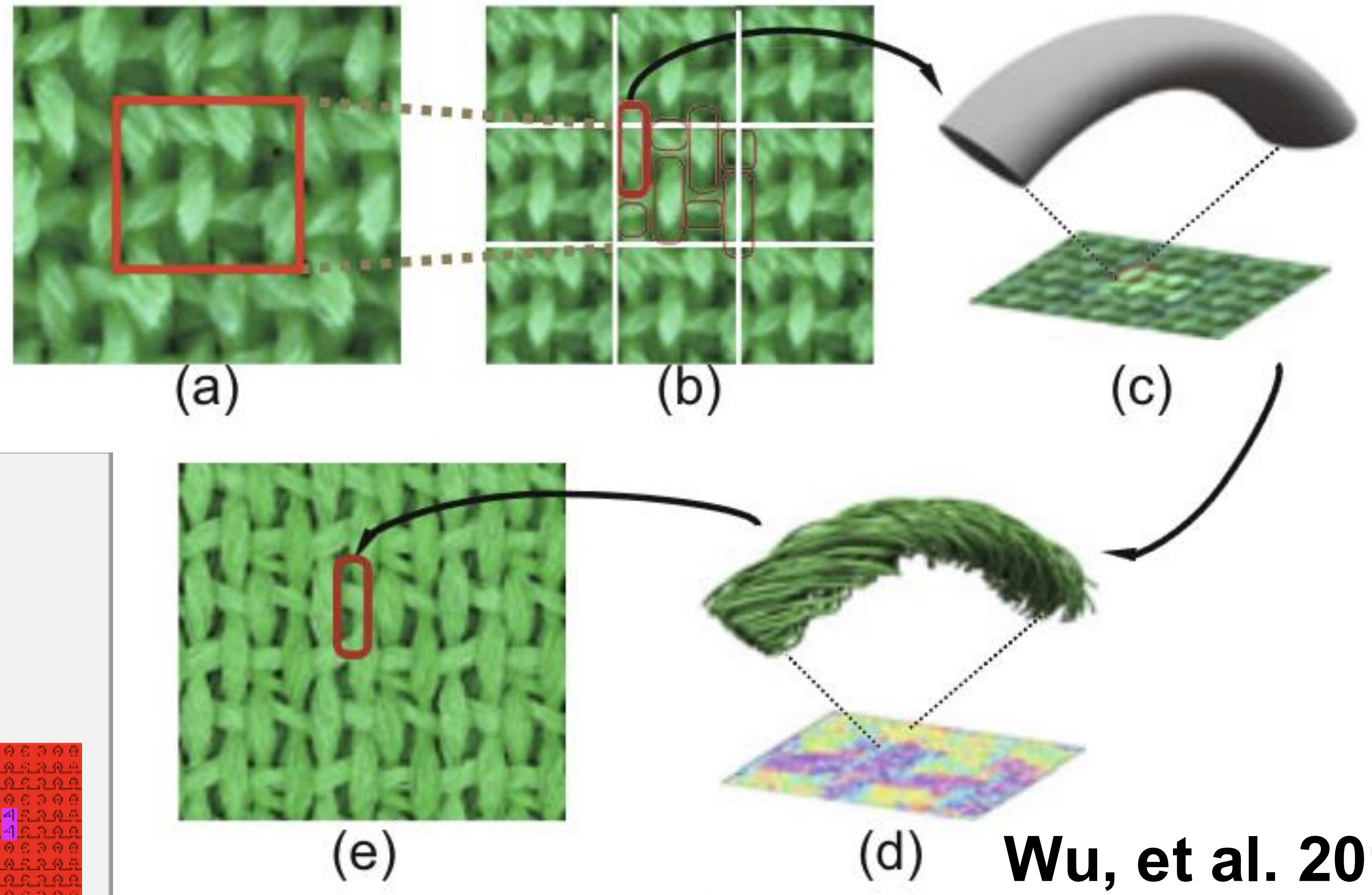
Fast but simplified

Stitch-Level Simulation

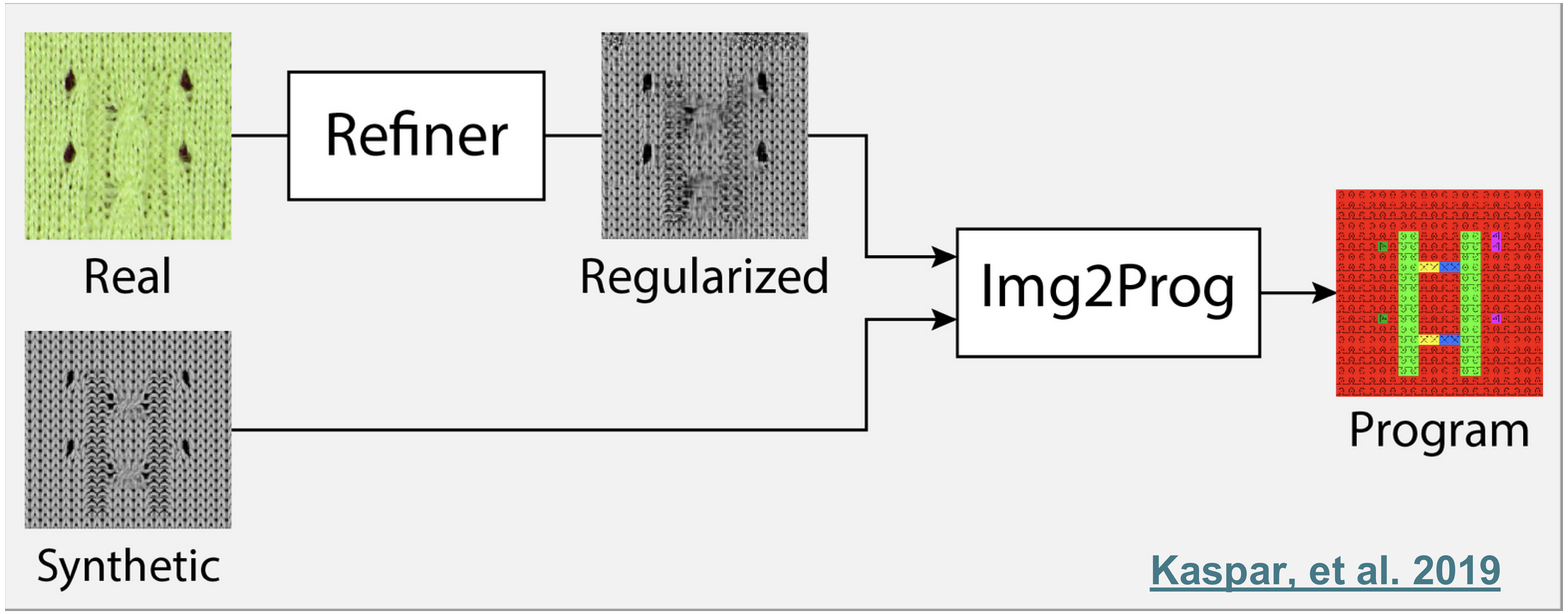


Wu, et al. 2025

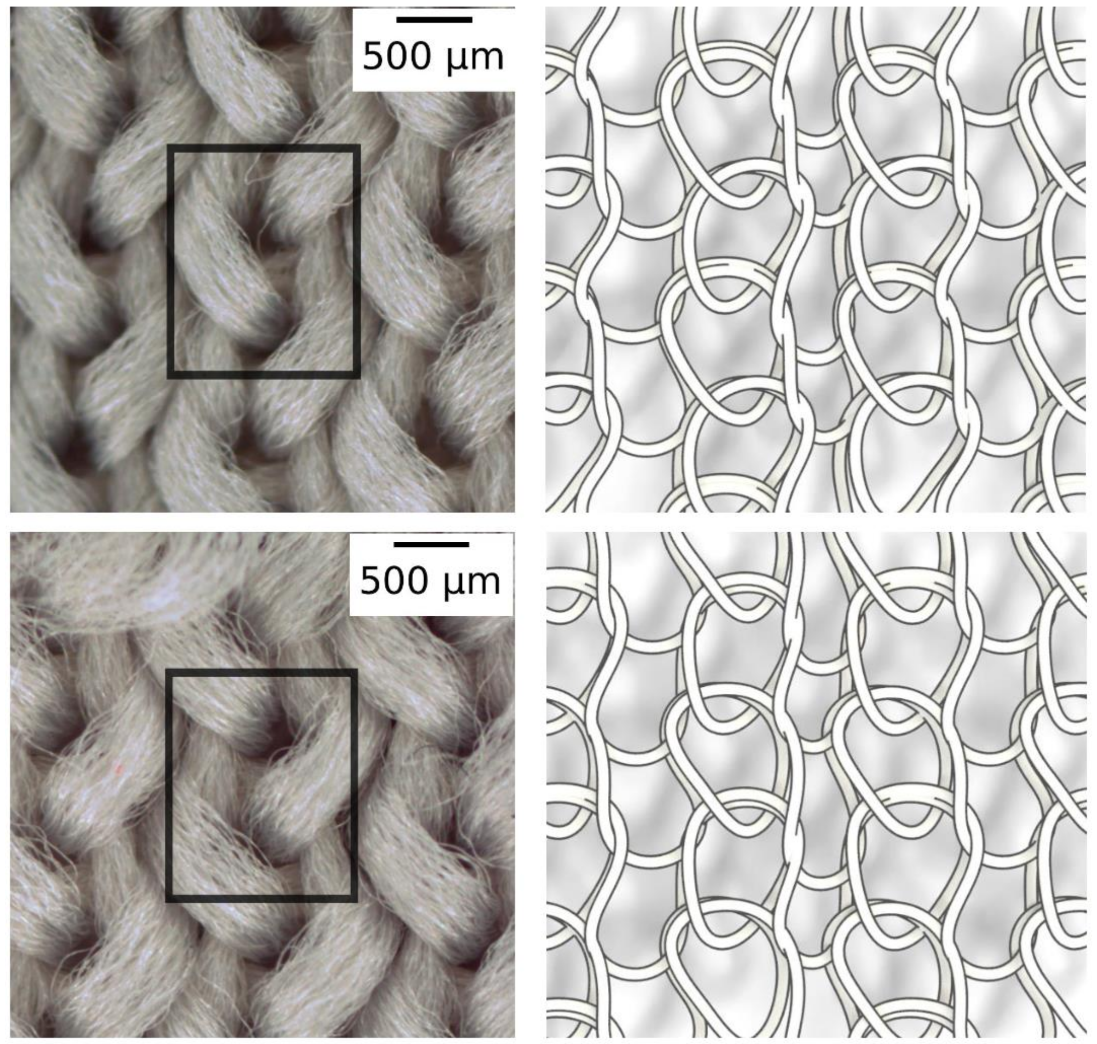
Interesting Future Work & Open Problems



Wu, et al. 2019



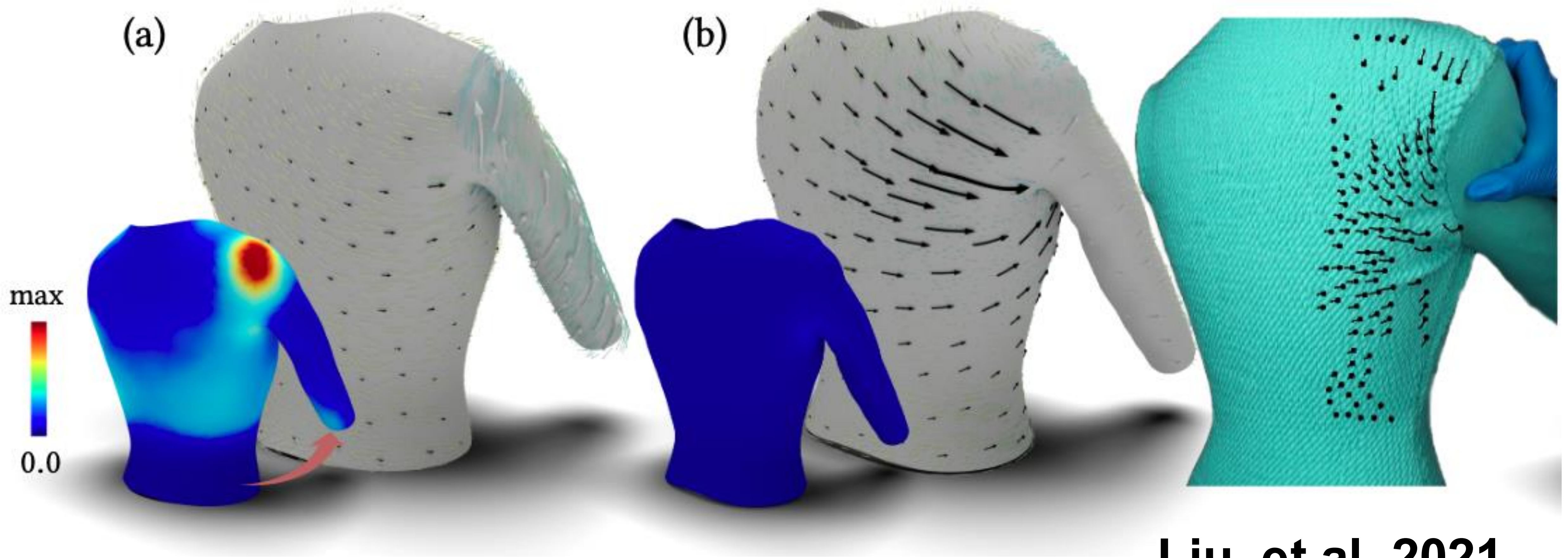
[Kaspar, et al. 2019]



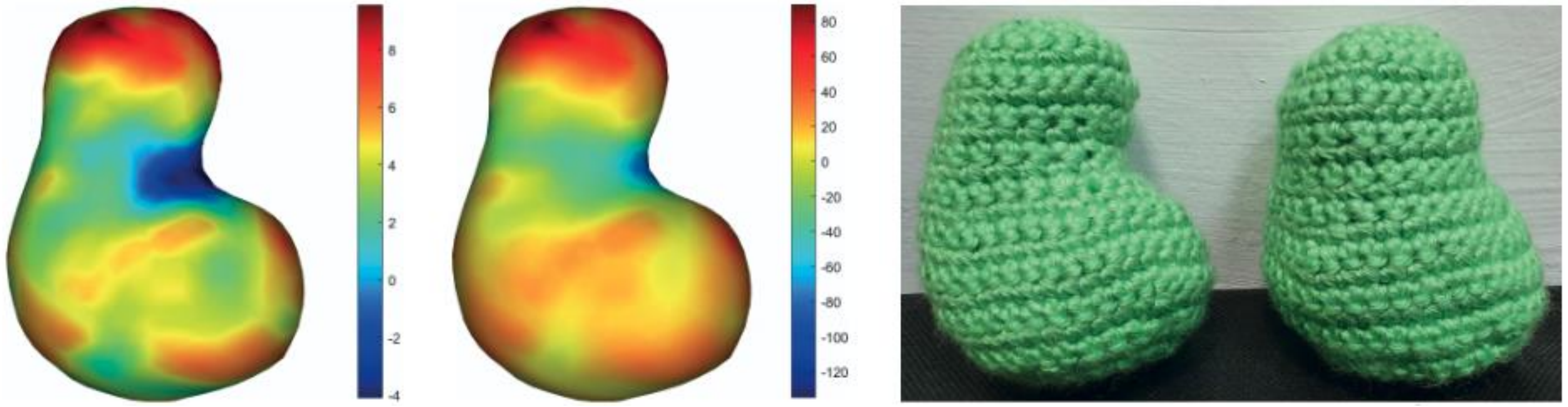
[Sperl, et al. 2022]

Existing knit capture pipelines are very constrained

Interesting Future Work & Open Problems



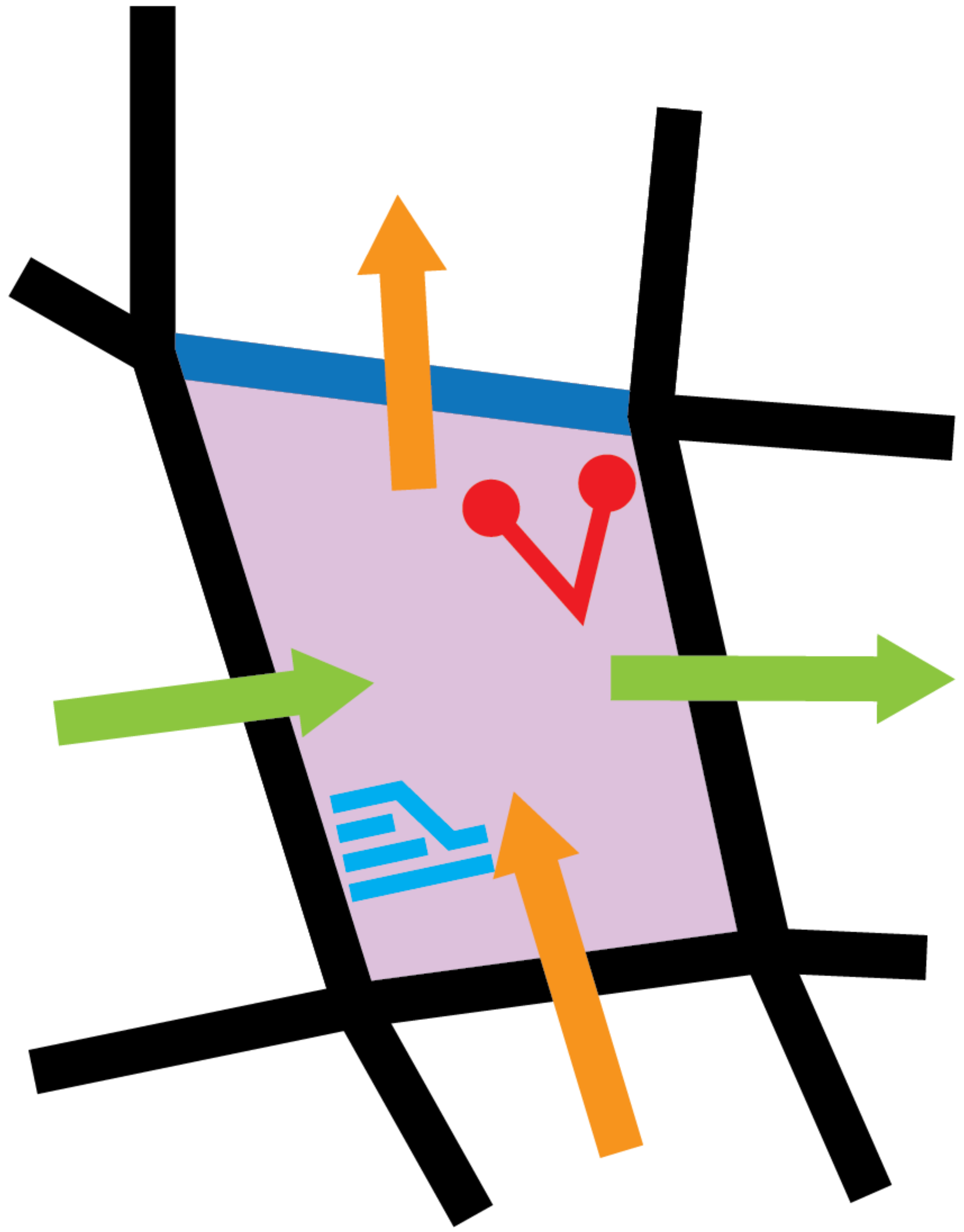
Liu, et al. 2021

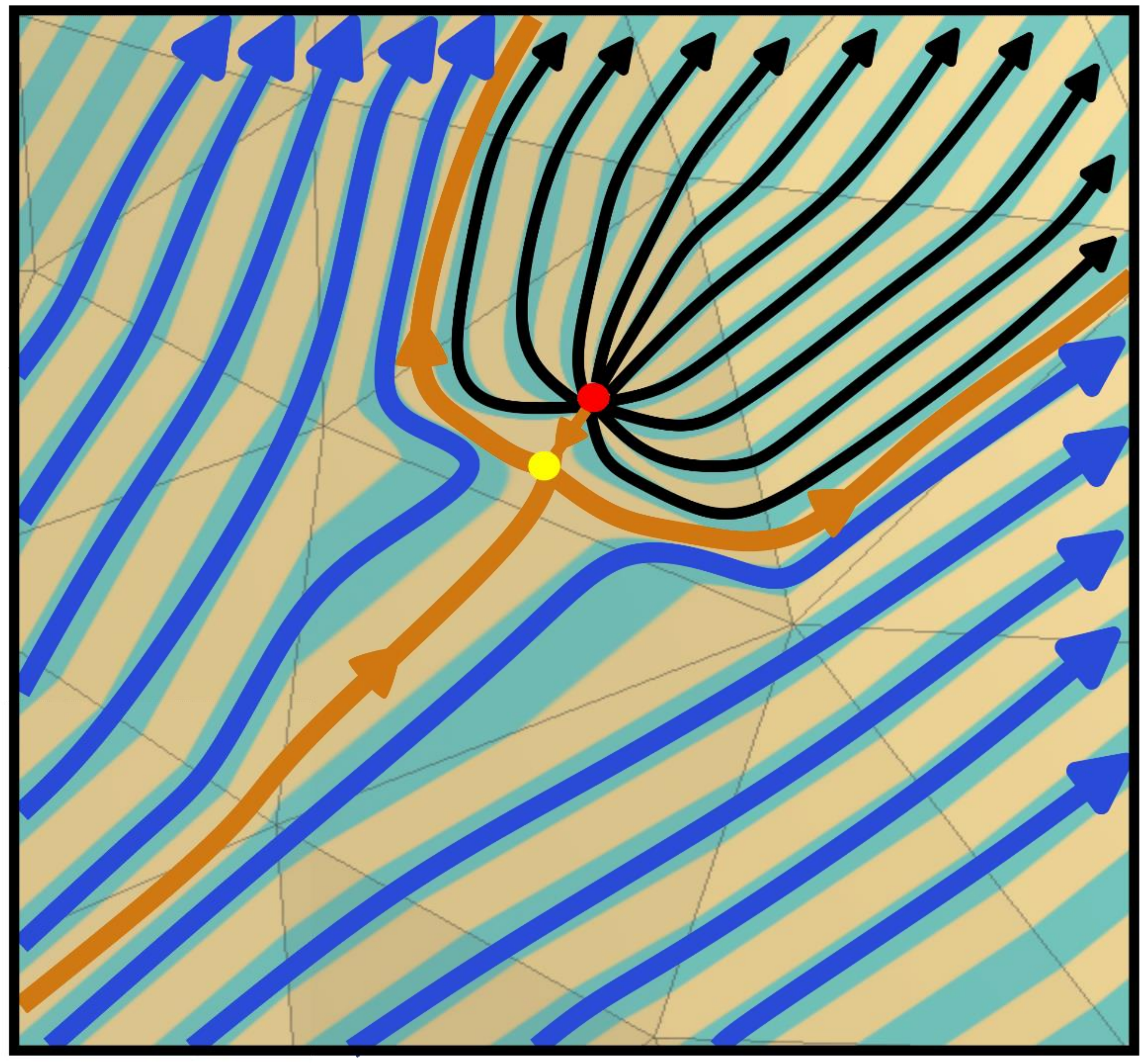


Edelstein, et al. 2022

How do we account for elasticity in the design process?

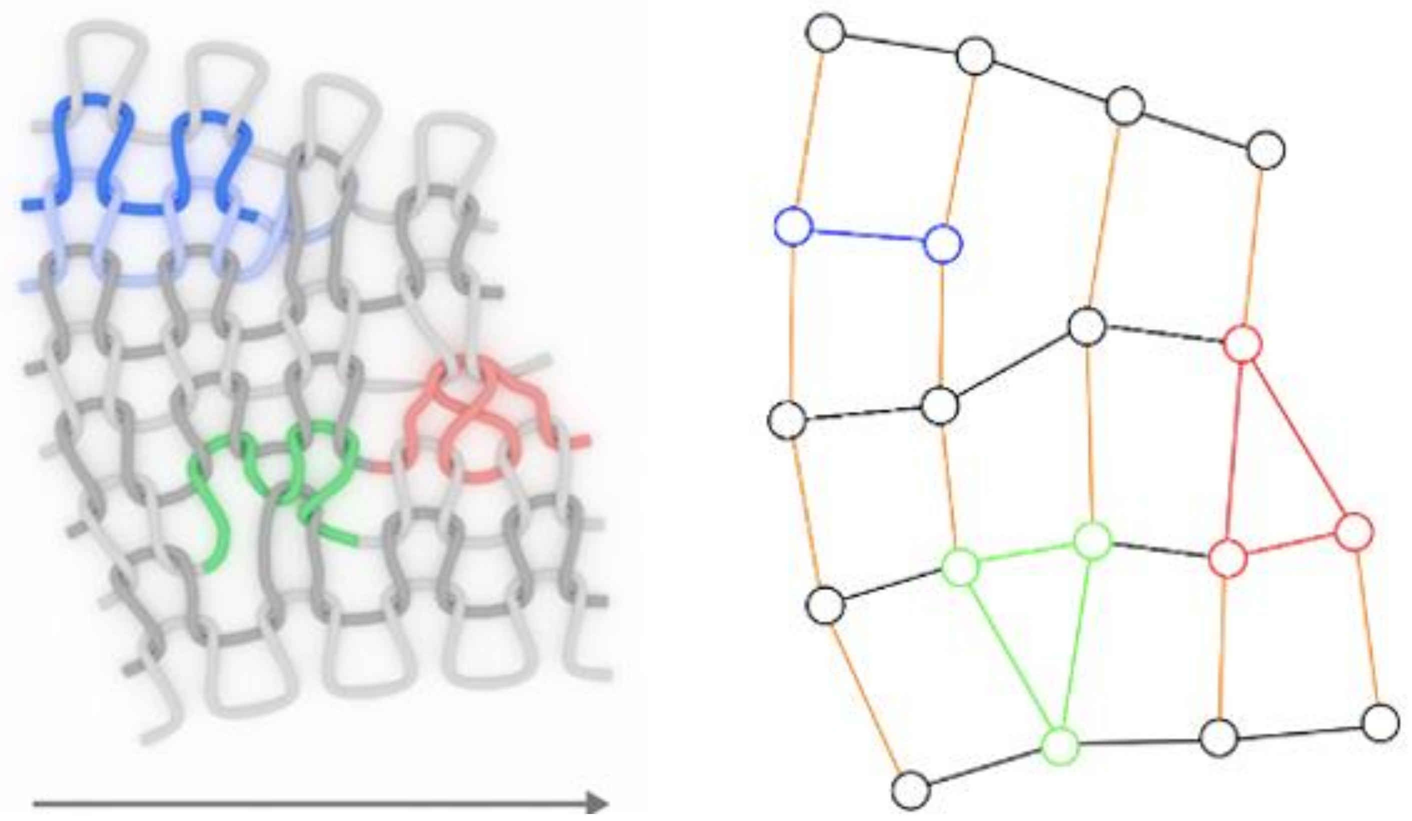
Interesting Future Work & Open Problems

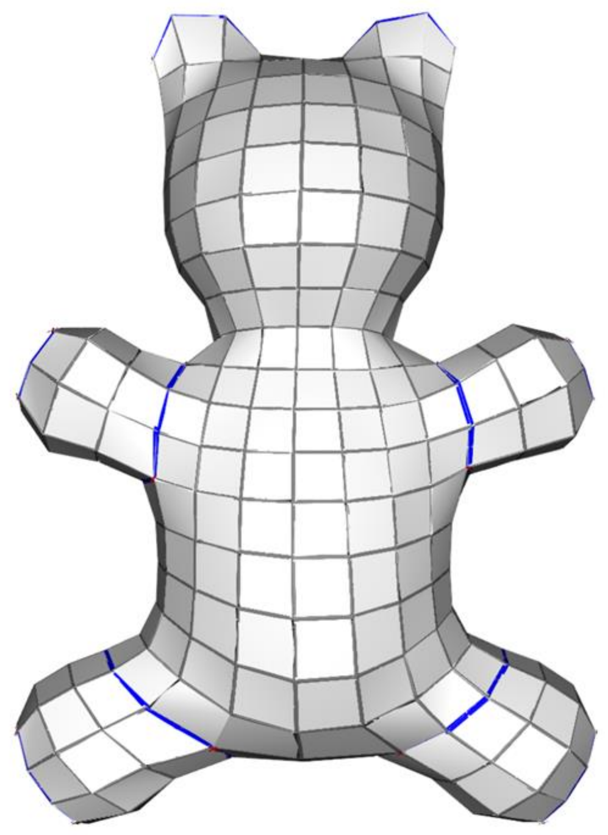


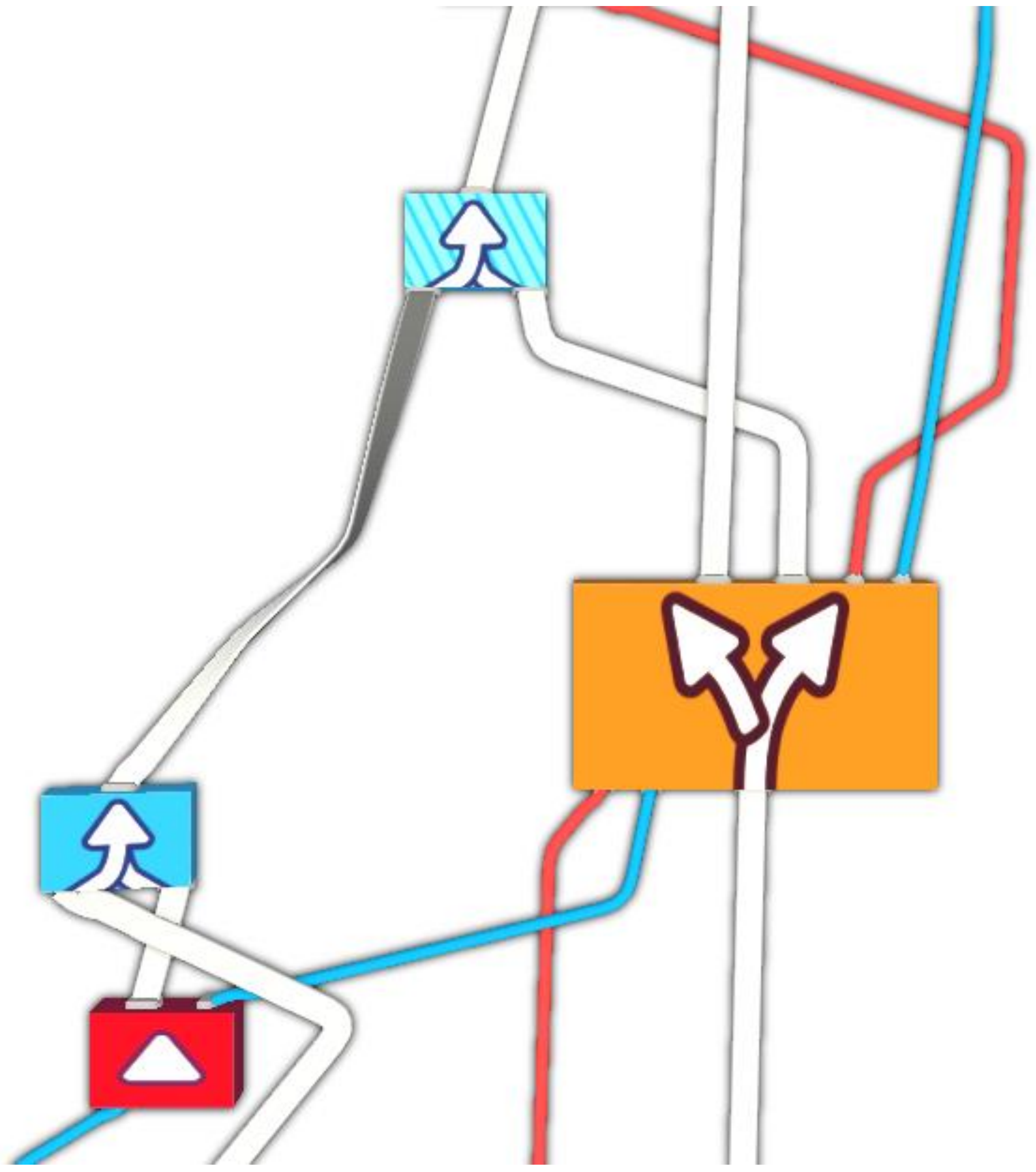


Combine high-level quad layout, and low-level singularity placement perspectives

Interesting Future Work & Open Problems

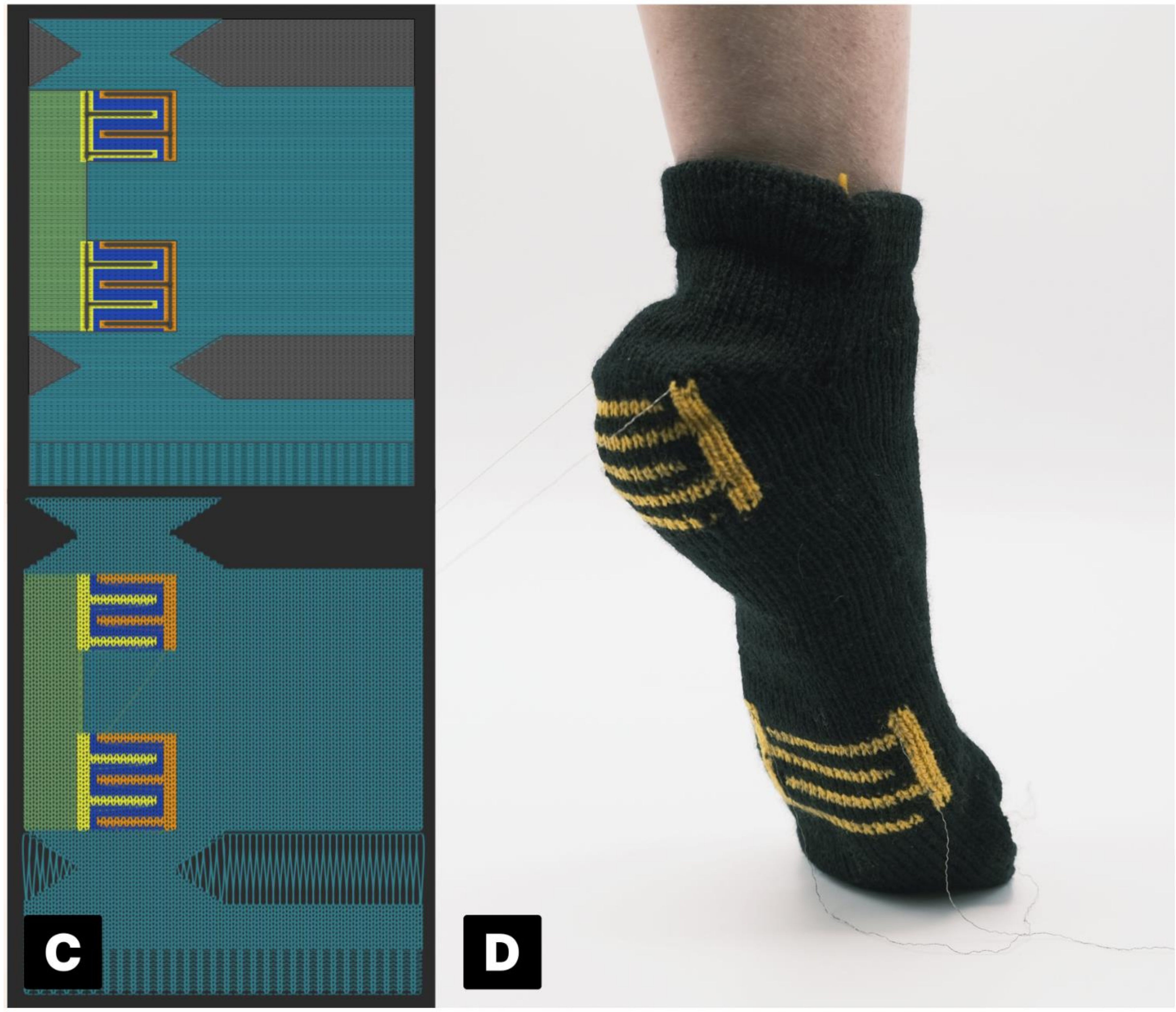




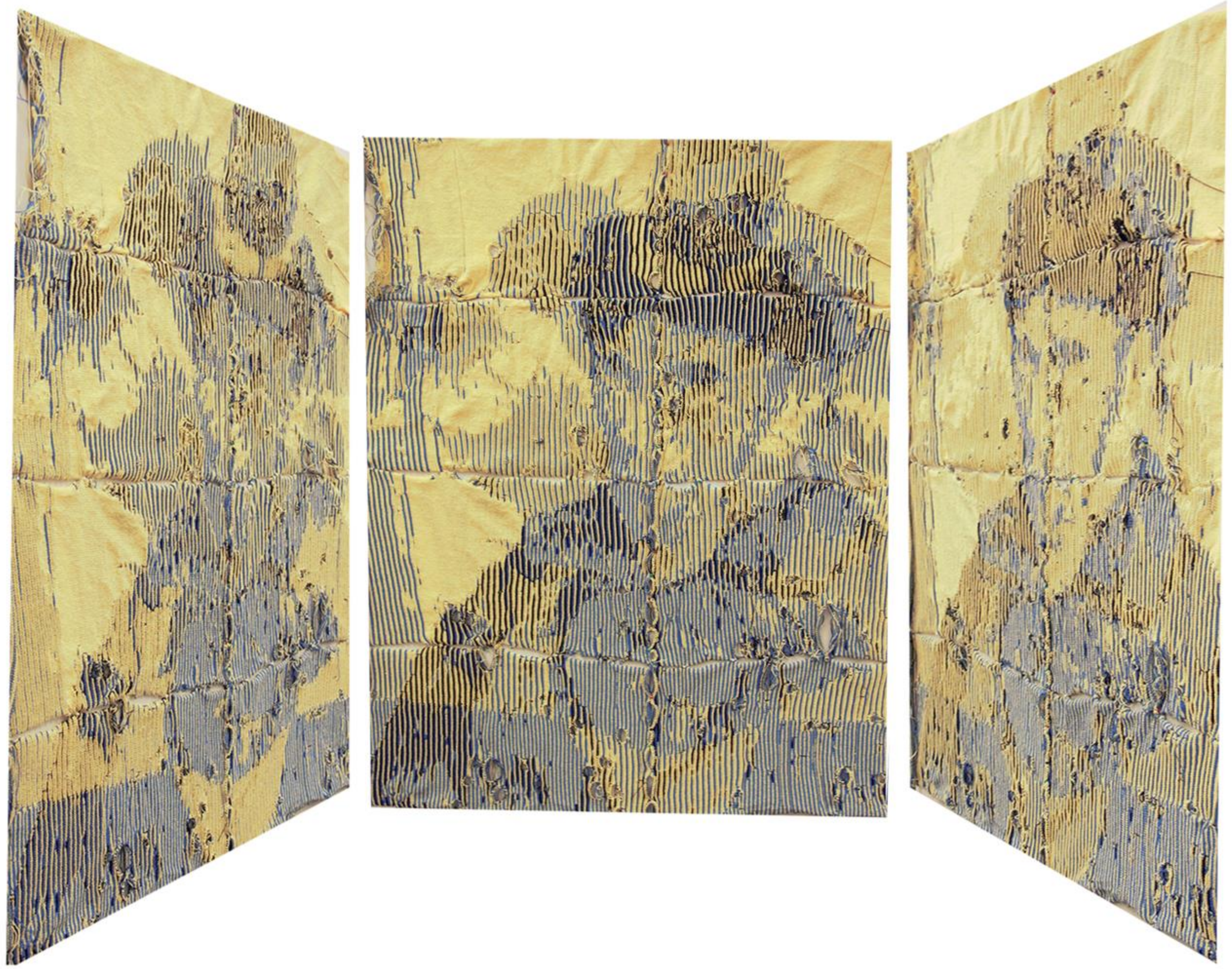


Transfer design criteria and fabrication constraints between different levels of abstraction

Interesting Future Work & Open Problems



Twigg-Smith, et al. 2024



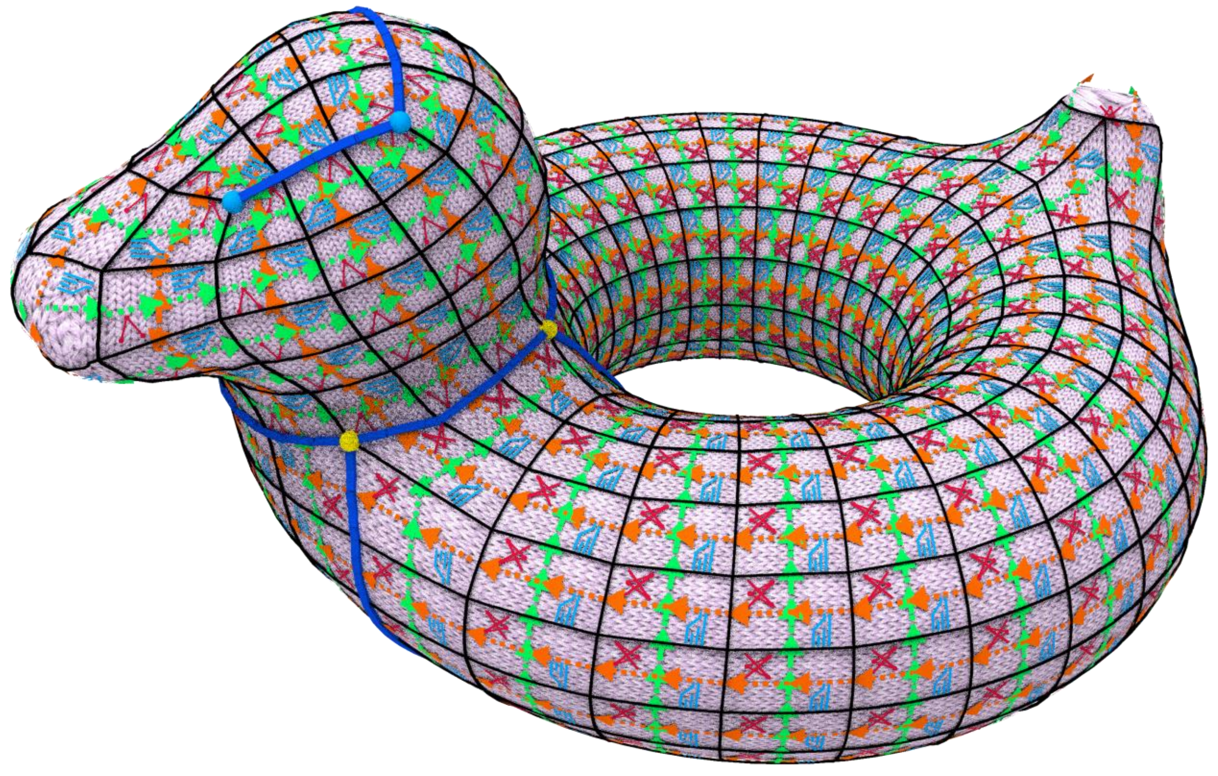
Zhu, et al. 2024

General computational tools for all of knitting

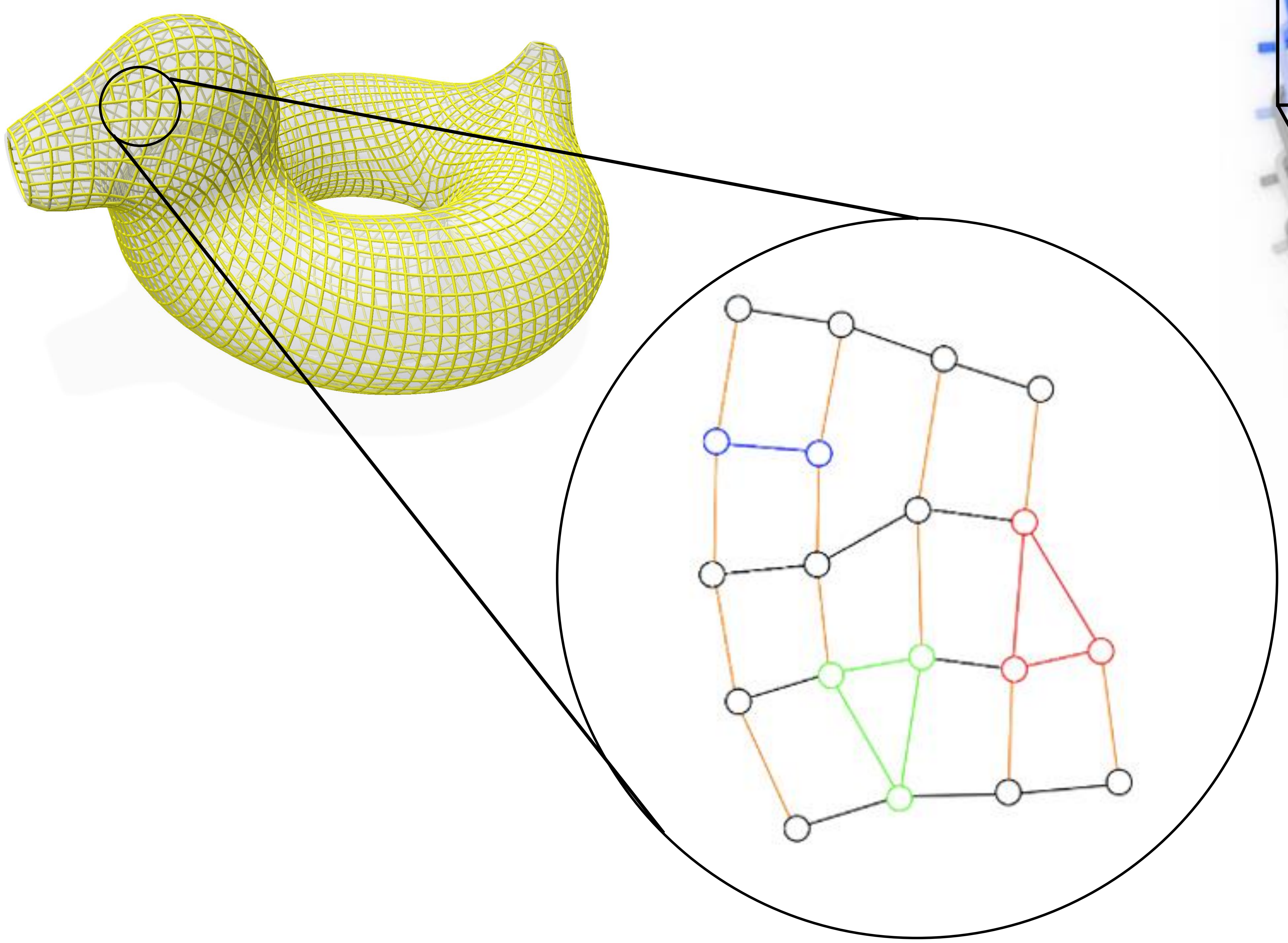
References

- Benjamin Jones, Yuxuan Mei, Haisen Zhao, Taylor Gotfrid, Jennifer Mankoff, and Adriana Schulz. 2021. Computational Design of Knit Templates. ACM Trans. Graph. 41, 2, Article 16 (April 2022), 16 pages. https://doi.org/10.1145/3488006
- Rahul Mitra, Liane Makatura, Emily Whiting, and Edward Chien. 2023. Helix-Free Stripes for Knit Graph Design. In ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23). Association for Computing Machinery, New York, NY, USA, Article 75, 1–9. https://doi.org/10.1145/3588432.3591564
- Rahul Mitra, Erick Jimenez Berumen, Megan Hofmann, and Edward Chien. 2024. Singular Foliations for Knit Graph Design. In ACM SIGGRAPH 2024 Conference Papers (SIGGRAPH '24). Association for Computing Machinery, New York, NY, USA, Article 38, 1–11. https://doi.org/10.1145/3641519.3657487
- Jenny Lin, Vidya Narayanan, Yuka Ikarashi, Jonathan Ragan-Kelley, Gilbert Bernstein, and James Mccann. 2023. Semantics and Scheduling for Machine Knitting Compilers. ACM Trans. Graph. 42, 4, Article 143 (August 2023), 26 pages. https://doi.org/10.1145/3592449
- Jenny Han Lin, Yuka Ikarashi, Gilbert Louis Bernstein, and James McCann. 2024. UFO Instruction Graphs Are Machine Knittable. ACM Trans. Graph. 43, 6, Article 206 (December 2024), 22 pages. https://doi.org/10.1145/3687948

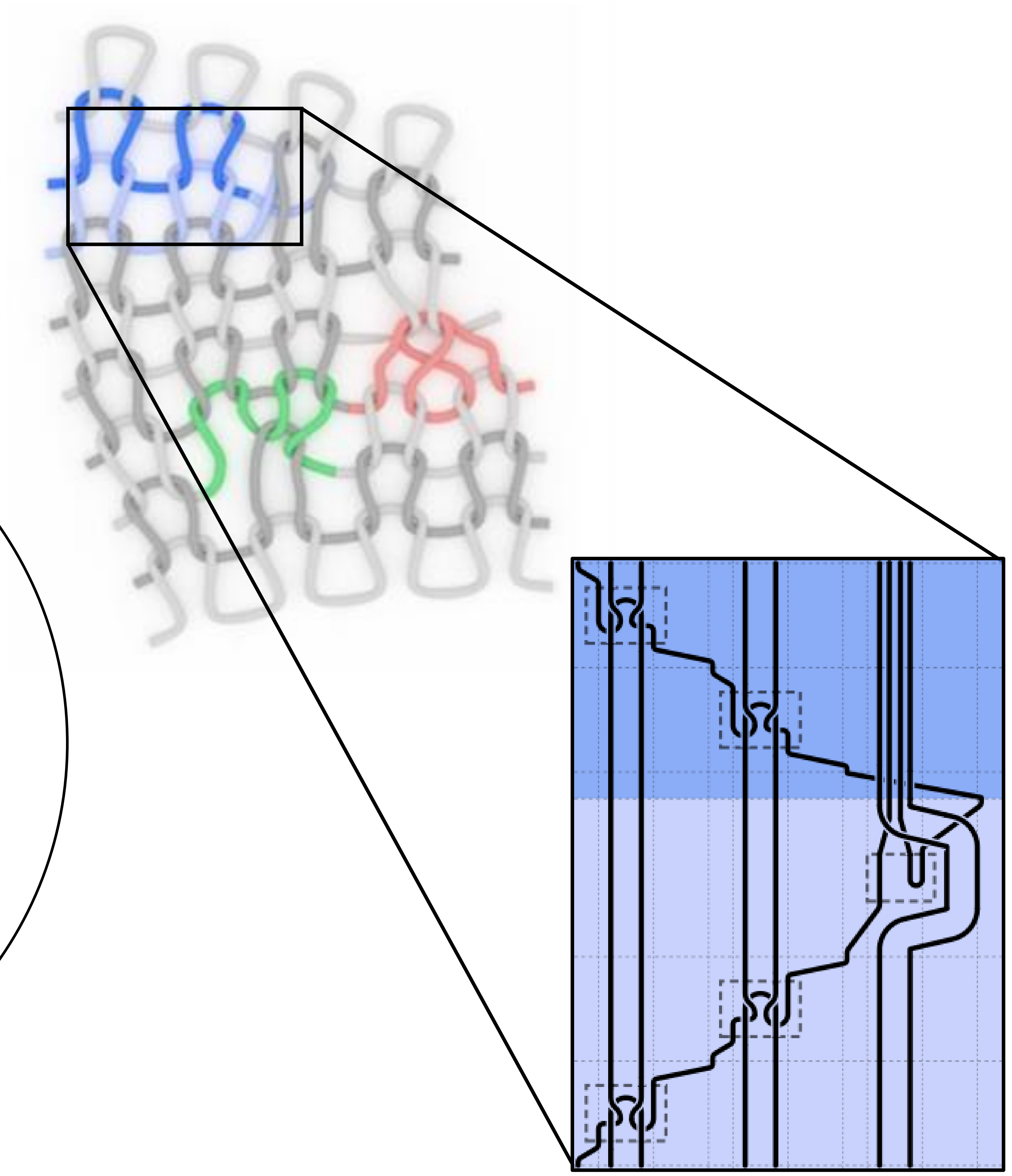
Thanks & Questions!



Fabric



Stitch



Yarn