Physically Motivated Safety Guarantees for Machine Knitting

Jenny Lin (jennylin@cs.cmu.edu)

1 Abstract

Industrial knitting machines are capable of creating complex, three-dimensional surfaces with a wide variety physical properties. In order to shape these surfaces, the machine must rearrange the position of stitches. When moving stitches, however, it's possible to move the stitches too far from each other, which produces breaks and other errors in the fabric. Thus when generating transfer plans to move stitches around on the machine, it's necessary to keep the physical constraints between stitches in mind. In this paper, I describe models for the knitting machine and for knit stitches as well as how the knitting machine's control can use knowledge about the knit stitch's model to ensure the stitch configuration is safe.

2 Background

2.1 Machine Knitting Background

A knitting machine holds stitches on hook shaped *needles* that are arranged into rows called *beds* [fig 1]. The *yarn carrier* holds onto the free end of the yarn and releases more yarn for forming new stitches. The amount of yarn that goes into a new stitch can be controlled. A standard v-bed knitting machine consists of two beds of needles which face each other. Aside from creating new stitches, a v-bed machine can also move existing stitches by combining two operations: *transfer* and *rack*. Any given needle can transfer the stitches it holds to the needle directly across from it. The two beds can also rack, or slide parallel to each other, introducing an offset that changes which needles are across from each other. By combining transfer and rack operations, it is possible to move the stitches on one needle to some other needle on the machine [fig 3].

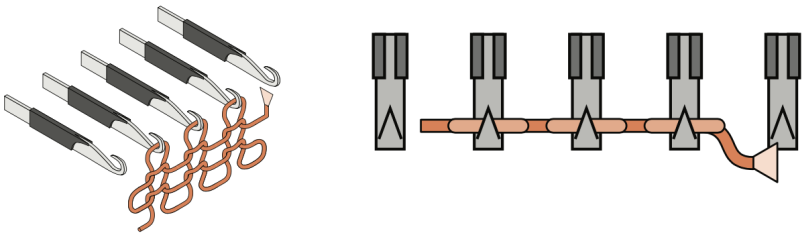


Figure 1: An isometric (left) and top down (right) view of five machine needles holding a fabric that is three stitches wide. The triangle represents the carrier, which releases more yarn for forming new stitches.

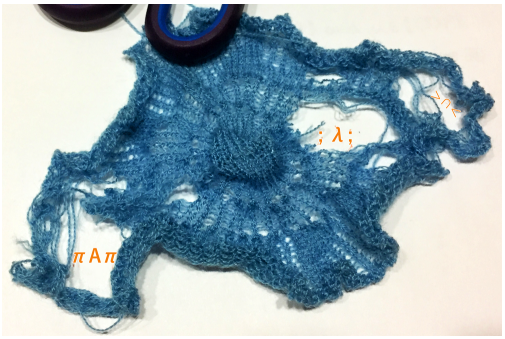


Figure 2: A knit circle with several broken stitches that unraveled into larger holes.

Transferring without taking into account physical constraints on the stitches can result in massive errors in the final product [fig 2]. Existing research on generating transfer plans assume that the target arrangement of stitches is valid on the machine and then use a discrete needle-based notion of distance between stitches to define upper bounds on how far stitches move from each other during the course of the plan [1, 2]. However, they provide no definition of what constitutes a valid arrangement of stitches. Meanwhile, commercial design tools for machines do come with assisting software that provide warnings about dangerous operations in a given pattern. In Shima Seiki's Knit Assist program, the "racking wearing out yarn" warning appears when two connected stitches knit on neighboring needles are moved more than four needles apart. The limit becomes eight needles if the connected stitches were knit every other needle. This value is hard coded in the system, however, which means it can't be adjusted to account for yarn characteristics or stitch size, which is a variable that can be controlled in knitting machines. To my knowledge, this is the first model of stitch safety that takes into consideration dynamic physical effects. In addition, I formalize a machine control model that independent of the stitch model, guarantees safety of the stitches. While there currently exists no commercial knitting machine that has stitch level feedback for the machine, this opens up a control mechanism for such a machine if it ever were to exist.

2.2 Modeling Background

My initial approach attempted to use component based modeling to represent the system. In component based modeling, the system is broken into smaller, repetitive components which can be individually proven safe. A component consists of a behavior model, which describes the dynamic behavior of the component, and the interaction model, which defines guarantees on the output of information from the component and any constraints on input of information from other components. So long as the output guarantee of a component implies the input constraint of another component, those components may be connected. Component based modeling has been looked at for proving traffic safety and roller coasters [3, 4]. Here we provide a brief description of the component structure. While this approach was useful for conceptualizing the interaction between the machine model and the stitch model, I couldn't define the stitch model such that it was a valid composable component with other stitches, as their internal behavior model was too tightly coupled with the behavior model of other stitches. While I could have modeled this with a lossy, delayed communication between stitches, finding a reasonable bound on drift in values was difficult.

What proved useful was looking at distributed hybrid logic [5] which has been in used in distributed systems such as distributed cruise control [6]. Similar to a distributed network of cars, each stitch receives some information from a global controller (the knitting machine) and then has its own continuous dynamics involving its own local interaction with its neighbors.

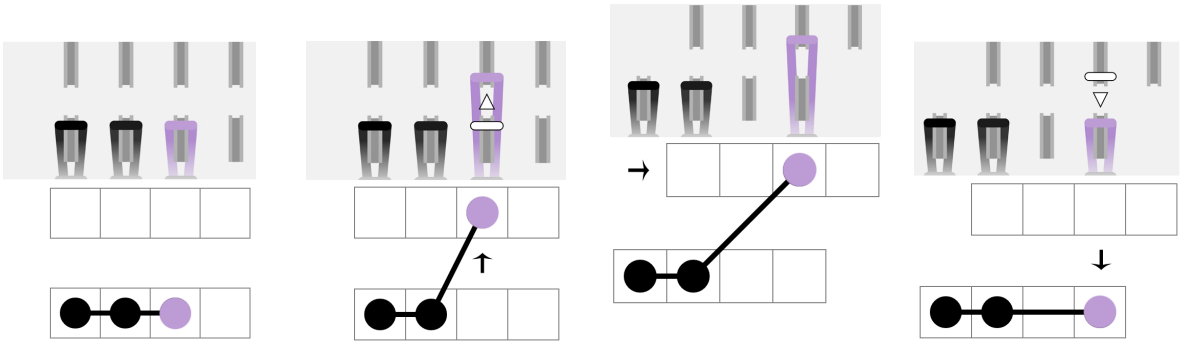


Figure 3: An example of combining transfer and rack operations to add spacing between stitches. The top shows a needle bed and stitch view while the bottom shows a more abstract dot and line view that emphasizes the presence of yarn connections. (a) All three stitches start on adjacent front bed needles 1, 2, and 3. (b) Front bed needle 3, which holds the purple stitch, transfers to the back bed. (c) The back bed is racked one needle to the right (+1). (d) Back bed needle 3 transfers to the front bed, placing the purple stitch on front bed needle 4 and creating a one needle gap between the purple stitch and its nearest neighbor.

3 Model

3.1 Stitch Model

When describing the stitch model, we make the observation that while single discrete stitches are normally thought of as the loop that is held by the needle, that is not the only way to divide up the knit structure. For our stitch model, we instead define a stitch as the length of yarn that lies between two needles [fig 4]. This stitch contains some *Length* of yarn that must cover some *Distance* between the two needles. Intuitively, if the *Distance* is very large, but the *Length* available in the stitch is small, then the stitch is under high tension and is at risk of breaking. In other words, there is some $maxTension$ that the stitch is not allowed to exceed. In order to determine the tension the stitch is under, we can model it as a simple spring with some caveats: most yarn is flexible and will buckle before it will compress. Thus when $Distance <= RestLength$, the tension in the stitch is zero. When $Distance > RestLength$, the internal tension is $k * (Distance - RestLength)$, where k is the stitch's spring constant. A stitch is thus safe when it satisfies the condition $k * (Distance - RestLength) \leq maxTension$

While intuitively, *RestLength* should depend on *Length*, the exact relationship is difficult to determine. A knit stitch holds its characteristic shape is in part because of the previous row of stitches holding it in place, so their structure would also have an effect on *RestLength*. For the rest of the paper, we make the simplifying assumption that $c * Length = RestLength$ for some constant c in order to more easily model the continuous aspect of the stitch under tension.

As a single continuous piece of yarn can be used to create multiple stitches, high tension in one segment of the yarn should affect the rest of the yarn. This is in fact one of the reasons why the model of the stitch spans between needles; the end points are both the region of overlap with the neighboring stitch and a place where the forces change. The needle is actually holding on to the stitch and is thus a source of *Friction*. If the stitch's internal tension overcomes . Consider the two stitch case with rest lengths L_1 and L_2 where T_1 and T_2 are their respective tensions. Their continuous interaction would be as follows:

Figure 4: A single stitch is the region between the two needles. Anything outside would belong to a different stitch.

$$\begin{aligned}&\{L_1'' = T_1 - T_2 - Friction, L_2'' = T_2 - T_1 + Friction \& T_1 \geq T_2 \wedge T_2 \geq Friction\}\\ &\quad \cup \{L_1'' = T_1 - T_2 + Friction, L_2'' = T_2 - T_1 - Friction \& T_2 \geq T_1 \wedge T_1 \geq Friction\}\\ &\quad \cup \{L_1'' = 0, L_2'' = 0 \& T_1 \leq Friction \cup T_2 \leq Friction\}\end{aligned} \tag{1}$$

The split in domains is to account for how the friction force always opposes the direction of motion and exactly equals tension forces when static. Assuming both initial tensions respect the stitch's safety condition, they will still be safe after running the differential equation. Note that in the case where $Friction > maxTension$, $RestLength$ remains constant and we recover the heuristic used in Shima Seiki's Knit Assist program, namely that there is some constant maximum distance two stitches are allowed to move from each other.

3.2 Machine Model

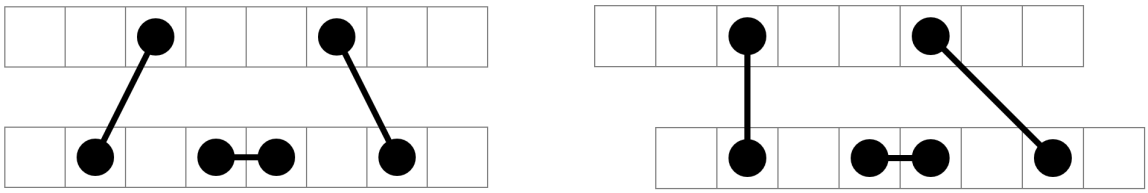
At a high level, what the machine does is it changes the *distance* value of stitches. How those values change depends on the arrangement of stitches. In figure 4, we see that despite all three stitches starting with the same distance and receiving the same rack value, their final distances is different. This is because of the relative placements of the stitch endpoints. Stitches that lie entirely on one bed do not change distance when racking, while those that cross the bed either increase or decrease by the racking value depending on whether the right edge is on the front bed. We define the machine transfer operation as assigning a *cross* value to a stitch from range $[-1, 0, 1]$. The cross value is then multiplied by the new rack value to determine how much the stitch distance changes by. The machine control instructions are thus as follows for all stitches:

$xferctrl := cross := 1; \vee cross := 0 \vee; cross := -1;$
$rackctrl := rack := *; distance := distance + cross * rack;$

3.3 System Model

As of now, when choosing a racking value, the machine makes an unguarded nondeterministic assignment. Thus there are no guarantees that the racking is a valid one for the configuration of stitches. The obvious strategy is to turn the assignment into a guarded one such that the racking value is safe for all stitches on the machine.

Each stitch has a local understanding of its maximum distance and its current distance. From that, it can calculate its own maximum acceptable racking values in both the positive and negative directions. If the machine has knowledge of each stitch's max values, it determine a safe range by taking the minimum of the stitches' max values in both directions and choosing nondeterministically within that range. A proof of this included and was solved using loop induction. An important



(a) (b)

Figure 5: The machine racking three different stitches that started with the same distance by the same amount. From left to right, their distance values decreased, stayed the same, and increased.

part of the loop invariant is not only that the rack value falls in the valid range for all stitch, but also that it's possible for the machine to choose a valid racking value. Given a $maxRack$ and $minRack$ value on the machine, where $rack \leq maxRack \land rack \geq minRack$, it must be the case that $maxRack \geq 0 \land minRack \leq 0$.

4 Discussion

In this paper, I describe models for the knitting machine and for knit stitches as well as a safe control for the machine. One think learned is how modeling problems don't necessarily fit into one framework or another. I initially was sure that a component-based approach was the right way to tackle the problem, and while learning about it certainly helped, in the end it didn't quite fit. While I initially wasn't too interested by the machine model and its controls, there ended up being a lot of subtleties there as well that helped me with formalizing the stitch model. It would be nice to come back to work and formally prove the full systems model as a distributed hybrid system for any number of stitches. More physically accurate stitch models would also be interesting.

5 Deliverables

Deliverables are one .kya and one .kyx file. The .kya is a proof of the system control for a machine with two stitches on it. The .kyx has the continuous dynamics for two neighboring stitches.

References

[1] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica Hodgins. A compiler for 3d machine knitting. *ACM Trans. Graph.*, 35(4):49:1–49:11, July 2016.

[2] Jenny Lin, Vidya Narayanan, and James McCann. Efficient transfer planning for flat knitting. In *Proceedings of the 2Nd ACM Symposium on Computational Fabrication*, SCF '18, pages 1:1–1:7, New York, NY, USA, 2018. ACM.

[3] Andreas Müller, Stefan Mitsch, and André Platzer. Verified traffic networks: Component-based verification of cyber-physical flow systems. In *ITSC*, pages 757–764, 2015.

[4] Brandon Bohrer, Adriel Luo, Xue An Chuang, and André Platzer. CoasterX: A case study in component-driven hybrid systems proof automation. *IFAC-PapersOnLine*, 2018. Analysis and Design of Hybrid Systems ADHS.

[5] André Platzer. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Logical Methods in Computer Science*, 8(4):1–44, 2012. Special issue for selected papers from CSL'10.

[6] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In Michael Butler and Wolfram Schulte, editors, *FM*, volume 6664 of *LNCS*, pages 42–56. Springer, 2011.