

Digital Crochet: Toward a Visual Language for Pattern Description

Klara Seitz

Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
klara.seitz@hpi.uni-potsdam.de

Jens Lincke

Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
jens.lincke@hpi.uni-potsdam.de

Patrick Rein

Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
patrick.rein@hpi.uni-potsdam.de

Robert Hirschfeld

Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
robert.hirschfeld@uni-potsdam.de

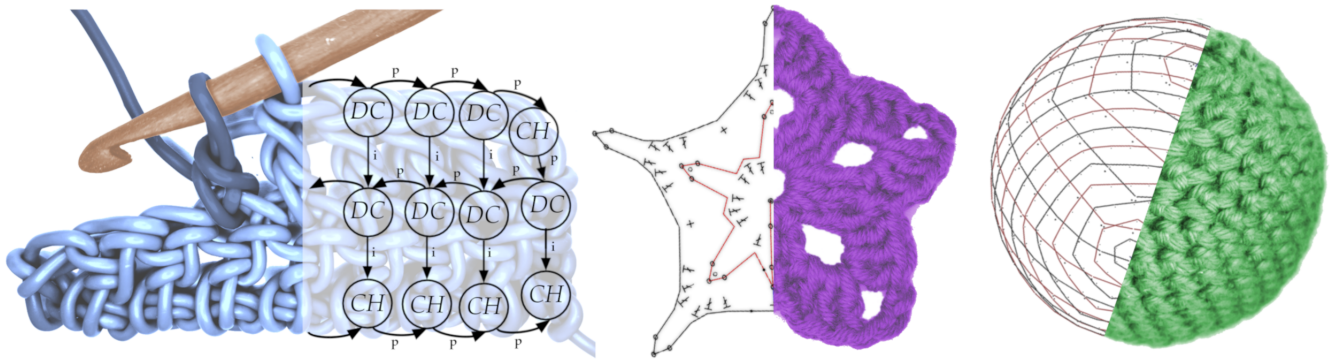


Figure 1. Providing digital tools for crochet: A graph-based language for representing the structure of crochet fabric serves as the foundation. 2D/3D edit views based on the crochet chart standard provide an impression of the shape of crocheted items.

Abstract

Crochet is still purely manual. While other crafts such as knitting or weaving have received technical support, the process of creating instructions for new crochet patterns is barely supported by domain-specific, digital tools. Those tools are constrained by their underlying crochet pattern languages that are either ambiguous or limited in their expressiveness. As a result, creating crochet instructions requires substantial manual effort and can result in incomplete or ambiguous instructions after all.

We propose a first visual, domain-specific, graph-based language for crochet pattern representation. We show how

this language can be leveraged to provide domain-specific tool support by a prototypical implementation of an editor for creating patterns in 2D and viewing them in 3D. In a user study, we demonstrate that the proposed language allows pattern designers to express both 2D and 3D patterns and removes ambiguities observed in current standard notations.

CCS Concepts: • **Applied computing** → **Computer-aided manufacturing**; • **Software and its engineering** → *Visual languages*.

Keywords: crochet, visual language, domain-specific language, fabrication, tools, crafts, computer-aided design, user study



This work is licensed under a Creative Commons Attribution 4.0 International License.

Onward! '22, December 8–10, 2022, Auckland, New Zealand

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9909-8/22/12.

<https://doi.org/10.1145/3563835.3567657>

ACM Reference Format:

Klara Seitz, Patrick Rein, Jens Lincke, and Robert Hirschfeld. 2022. Digital Crochet: Toward a Visual Language for Pattern Description. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '22)*, December 8–10, 2022, Auckland, New Zealand. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3563835.3567657>

1 Introduction

Designers of knitting patterns benefit from a range of digital tools, from low-level instruction languages for knitting machines, to high-level languages for describing whole garments, and corresponding interactive design tools [20, 23].

In contrast, crochet designers do not yet benefit from such tools, neither from explicit, formal languages for describing crochet patterns nor from sophisticated design tools. There are graphical notations for crochet patterns, such as *crochet charts* [6]. However, these notations lack a formal, structured representation of relations between individual stitches in a fabric, and consequently, the support through domain-specific design tools remains limited.

The lack of rich tool support leads to increased manual effort for pattern designers and challenges for crocheters due to erroneous, incomplete, or ambiguous patterns. Even when using the existing tools, designers have to manually arrange graphical stitch symbols as if working in a general-purpose graphics editor. As the tool cannot reflect on the structure of the pattern, it cannot check whether the pattern can actually be crocheted and automatic translations to other instruction formats, such as a textual description, are not possible. As a consequence, creating instructions remains time-consuming and error-prone.

In this paper, we propose a digital representation of crocheting patterns as a foundation for rich tool support in the future (see Figure 1). These tools should support crochet pattern designers in creating patterns for manual crochet. In detail, our contributions are:

- A visual, graph-based, domain-specific language for describing crochet patterns in an unambiguous way
- A prototype of an editor illustrating the potential features build around the language: creating crochet patterns in 2D, viewing them in 3D, and basic auto-completion of rows
- A qualitative user evaluation of the language and the editor with regard to expressiveness

2 Background and Related Work

Crochet is a very flexible craft that enables crafters to create all kinds of objects, including 3D objects such as the small toys known as amigurumi. Crochet is used as a rich medium of expression from democratic participation through crocheted memes, to the exploration of tactile interfaces, and the visualization of chaotic systems [10, 14, 25, 27, 32].

Since crocheting is not as widely known as knitting or weaving, we briefly introduce the basic vocabulary of crocheting. Then, we briefly present related, advanced tools and languages for knitting which shows the potential for digital tools supporting crafts. Finally, we discuss existing representations and tools for crochet pattern instructions in order to illustrate the existing workflows and resulting challenges for crocheters.

2.1 Crochet Stitches and Methods

Crochet is the manual process of creating fabric with a crochet hook and yarn. The hook is used to get hold of the thread and to pull it back through the fabric. Generally, pulling the yarn through the fabric and potentially additional loops on the hook results in a *stitch*. The way the stitches are arranged and which stitch types are used influence the shape and texture of the fabric.

Chain stitches only result in a single line of stitches. *Slip stitches* are a common utility stitch to connect two stitches in the fabric or to advance through the fabric without widening it. Other stitches such as the *single crochet* widen the fabric. New stitches are typically inserted into all types of stitches except for slip stitches. These stitches are started by inserting the hook into the previously created fabric and pulling through a loop of yarn. Then, depending on the stitch type this process can be repeated multiple times to create more loops on the hook. To finish the stitch, the yarn is pulled through some or all of the loops on the hook until only one loop remains.

Depending on the desired shape and design, stitches can be crocheted in rows or in rounds. A pattern can switch from one to the other method. The row method works stitches one by one up to the end of the row where the fabric is turned and stitches are worked again in the opposite direction. When working in rounds the fabric does not have to get turned and stitches are worked continuously in the same direction going around the fabric.

To create three-dimensional pieces and shapes more complex than rectangles, crocheters typically use two methods: *increase* and *decrease*. When using the increase method rows or rounds widen and when decreasing they narrow. When using the increase method multiple stitches are worked into the same existing stitch. When using the decrease method, the yarn is pulled through multiple stitches before finishing the current stitch.

2.2 Knitting and Crochet

The varying tool support for pattern creation in crochet and knitting results from the difference in the degree to which the techniques themselves are mechanized. Knitting is supported by machines on an industrial scale. Meanwhile, there are no machines that are able to industrially produce fabric based on the technique of crochet. Crocheted clothing, figures, or other items are still hand-made. The few existing machine types for crocheting either only imitate crochet or are limited to specific stitch types, which makes them unsuitable for most patterns [5, 9, 11, 19, 28, 36].

¹video instruction 'How to crochet star applique': <https://www.youtube.com/watch?v=v5JVHjHp6kM>, textual pattern 'jip the owl': <https://www.ravelry.com/patterns/library/jip-the-owl>, crochet chart: <http://www.avystore.com/wp-content/uploads/2017/12/Crochet-Star-Pattern.jpg>

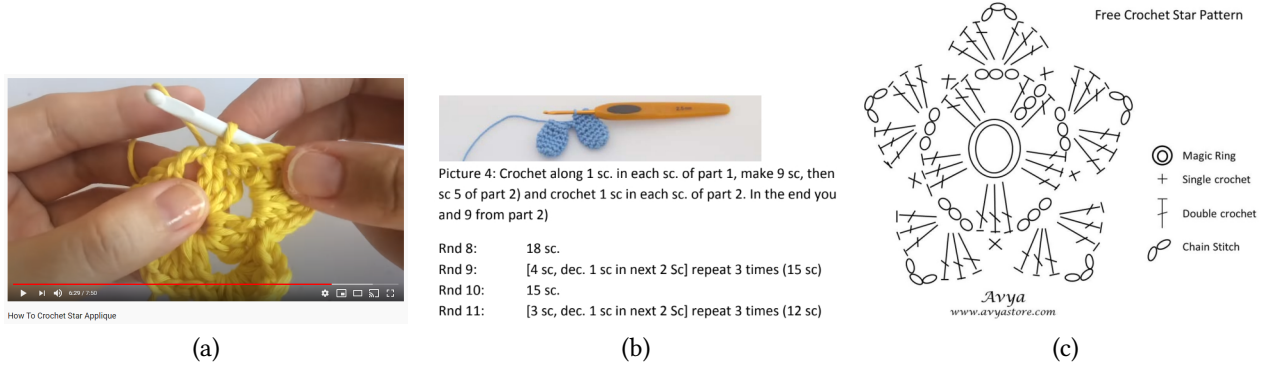


Figure 2. Crochet instruction types: (a) video of the crochet process (b) written instructions with the help of images (c) crochet chart¹

Due to the high degree of mechanization, there are formal instruction formats for knitting machines which can be used to describe knitted garments digitally [31]. Beyond these low-level instruction formats, high-level languages have been proposed to define knitted garments [20, 23]. In these languages, designers can express single components of a garment, the composition of these components, and the application of patterns to parts of the garment. The corresponding compilers translate these high-level specifications into instructions for knitting machines.

Further, based on the instruction formats and the high-level languages, design tools exist that provide designers with features such as composing garments graphically, specifying shapes as well as patterns in one tools, or seeing a three-dimensional rendering of the garment [20]. Recent approaches even enable designers to automatically translate a 3D mesh model to machine knitting instructions [18, 24]. Lately, machine learning techniques were added to design tools in order to support designers in choosing patterns to achieve certain properties in the resulting fabric or to extract pattern instructions from images [21, 33].

In comparison, crochet does not yet benefit from such rich tool support. So far, crocheting even lacks a formal language that could be used to build such tools upon.

Representing Knitting Patterns. The language we propose is based on the idea of modeling the relations between stitches in the fabric as a directed graph. This general approach has also already been applied to knitting [7, 15, 24]. For example, one approach uses a distinction between row and column edges which are similar to our distinction between previous and insertion edges [24]. This approach also includes the notion of machine knittability that defines properties a knit graph has to adhere to, so that it can be produced by a knitting machine. Other approaches represent patterns as meshes of stitches [26, 34].

While both row/column-based graphs and meshes are suitable representations for knitting, they are not suitable

for representing crochet patterns, as they lack the flexibility to express that crochet stitches might be inserted anywhere in the fabric [13, Fig. 13].

The main reason why there are no industrial crochet machines is rooted in the multitude of stitch insertion points [12]. In crochet, stitches can be inserted in various points, for each stitch exist multiple places where the hook could be inserted. Any other spaces between stitches can also be used. To build a machine, which supports arbitrary insertion points, either many place holders have to be positioned which might get in each others ways, or the manual crocheting technique could be imitated. The latter would require for example a robotic arm which has the ability to 'see' or 'feel' the insertion points similar to a human [9, 12]. Current knitting machines, on the other hand, can take advantage of a single loop per knitting stitch where the insertion options are limited. Koch shows how knitting can be easily represented through basic code or even binary [22]. Due to the lack of crochet machines, there has not been any need for exact sets of instructions as required to run a machine and thus no digital and formal way to describe crochet patterns has been developed so far.

2.3 Crochet Pattern Instructions and Notations

Crochet designers create instructions for patterns in various forms (see Figure 2), such as how-to videos, textual descriptions, crochet diagrams or charts, and combinations of these forms. The different types of instructions differ in the degree to which they use standardized notations or vocabulary, whether they can describe 2D or 3D patterns, whether they can express all kinds of stitches, and whether the resulting instructions can include errors or ambiguities.

Instruction videos (see Figure 2 (a)) are mostly repetitive and everyone follows a different structure. At the same time, designers can demonstrate any technique and pattern and the videos are not inherently ambiguous. Thus, they might

be well suited for beginners or for explaining unusual techniques. At the same time, they might not be ideal for experienced crocheters due to the inherent repetition and the fact that they have to be consumed linearly.

Textual instructions are another presentation of a pattern (see Figure 2 (b)). They often consist of a mixture of free form text and semi-formal representations of the actual pattern (see lower part of Figure 2 (b)). These semi-formal representations are not standardized but every designer comes up with their own notation. The instruction texts are written manually and often include accidental errors such as skipping steps or incorrect sums of stitch number. As there are several schools of terminology, some patterns might require readers to readjust (for example, “double crochet” can refer to two different stitches). Also, non-native speakers who use English terminology may mix different terminologies by accident. All of these issues can unnecessarily confuse crocheters trying to reproduce the pattern.

A first, more formal, visual representation of a pattern is a *crochet graph* that shows the pattern as a grid (see Figure 3). Hence, crochet graphs can only be used for patterns that are flat and whose arrangement of stitches matches a grid. The cells of the grid often encode stitch types through symbols or letters and the color of the yarn through the cell color. Again the notation is not standardized. While they leave little room for ambiguities, they are also very limited with regard to the types of patterns that they can represent.

Crochet charts are a standardized way to represent patterns (see Figure 2 (c)). Such a chart shows stitches that are represented by schematic crochet symbols [6]. These charts can express any two-dimensional pattern and some three-dimensional patterns. The charts show which stitch is worked into which by rotating the stitch symbols to point to the inserted stitch. Common stitches are represented by standardized symbols. Rounds or rows are distinguished visually either by adding numbers or by alternating the color of stitches. Symbols are typically arranged to mirror the resulting shape of the pattern. When all symbols are well arranged, the chart can represent a pattern very precisely. However, as the insertion points are only defined through the visual cue of the direction of the stitch symbol, many charts include ambiguities (even the example in Figure 2 (c)). We demonstrate several ambiguities as part of the user study in subsection 4.2. Many designers draw the charts by hand and incorporate a photo of it into their pattern instructions. If any programs are used, they are mostly generic drawing programs or text editors. There exist also some crochet specific programs which try to tackle the issue of positioning the stitch symbols. They support the designers by supplying the stitch symbols and snapping guidelines to facilitate placement. Yet, the stitches are never aware of any connections between them and thus are rather a more specialized drawing program.

An even more formal approach to specifying a pattern is the so-called “Berliner Häkelschrift” [29]. It was particularly designed to represent how stitches of 3D objects are connected in a 2D coordinate system. This system supports the representation of rows and rounds but is limited to using only one stitch type - single crochets. While it is formally defined it is not well-known.

2.4 Tools for Creating Crochet Patterns

Most digital tools for creating crochet instructions are based on crochet charts.

A very basic tool are *stitch fonts*, which have been designed to allow designers to use graphics or text editors to create crochet charts [1, 17]. They supply a variety of symbols used for crochet charts and designers can arrange them using general graphics editors.

Beyond basic stitch fonts, there are also crochet chart editors, such as Stitch Fiddle, CrochetCharts, and others [8, 16, 30]. All of these existing tools have in common that they merely support the graphical arrangement of stitch symbols (for example, see Figure 4). Some provide additional layout tools, for example circular snapping grids to ease the creation of rounds. Nevertheless, none of these tools allows designers to explicitly define the relation between stitches. As a result, the process of creating crochet charts using these editors remains just as manual as the usage of stitch font symbols in a graphics editor. Also, because of the missing semantic information on the structure of the pattern, these tools can not convert patterns to other formats, visualize the end-result, or analyze the pattern in any way. Analogously, while there are serialization formats for crochet charts, these charts do also not represent the actual relation of stitches but only the arrangement and orientation of stitch symbols [35].

Recently, a powerful tool set has been proposed based on the approach to represent crochet patterns in the Stitch Mesh model, which was originally created for knit objects [13]. By representing crochet patterns as stitch meshes, the resulting tooling can provide high-quality 3D previews of patterns and automatically derive crochet patterns from 3D models. However, the tooling is limited to crochet patterns corresponding to the capabilities of knitting. Crochet patterns may insert stitches at earlier points in the fabric and can thus not be expressed with stitch meshes. At the same time they are common for creating 3D objects (for example, in case of a handle made of chain stitches) or sophisticated textures (for example, the front post double crochet) [13, Fig. 13].

3 Visual Crochet Pattern Language

When designers currently digitize their patterns they type text with a text editor or draw a chart using a generic graphics program. While these representations are digital and editable, the used editors are not aware of the crochet domain. To provide a foundation for future, domain-specific

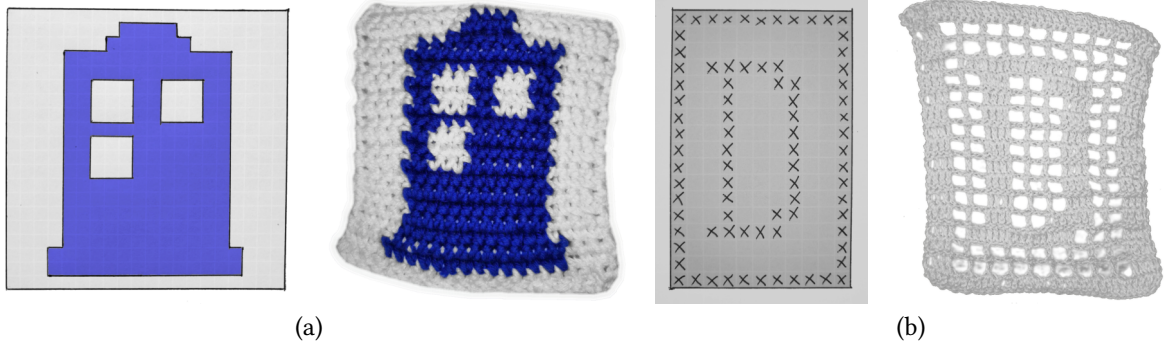


Figure 3. Examples of crochet graphs as pattern description and their results. (a) The color of the cells indicate which yarn color to use, all cells are worked using the same stitch type (b) Filled cells indicate to use a different stitch type than empty cells

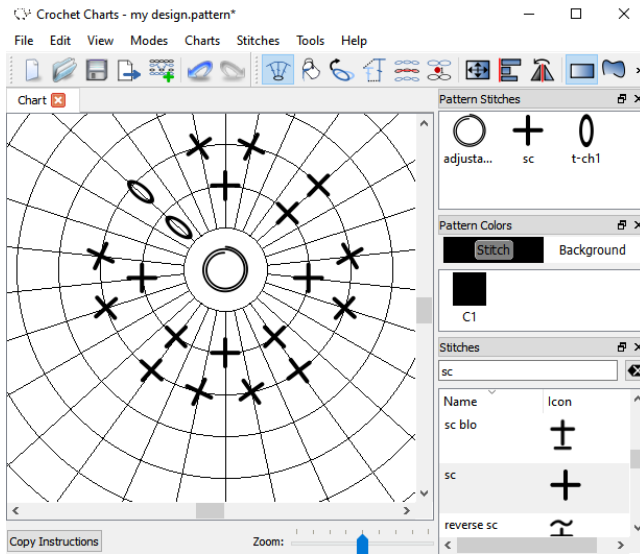


Figure 4. A screenshot of CrochetCharts, an application for creating crochet charts. The arrangement of stitch symbols is purely graphical and the editor has no understanding of the structure of the pattern.

tool support and to enable unambiguous instructions for crochet patterns, we designed a language to represent patterns as a graph. In the following, we first illustrate the principal mapping of crochet to a graph which already covers basic crochet methods such as different stitch types and insertion points. Based on these methods, we show how the language represents more advanced crochet techniques that are needed to provide complete instructions, such as different starting methods. Finally, we describe the 2D syntax and the 3D visualization of patterns, and the editor prototype.

3.1 Core Principles of the Language

The structure of the graph is designed to be able to represent all of the main crochet characteristics and methods. We chose to use a graph to represent crochet patterns as it allows us to

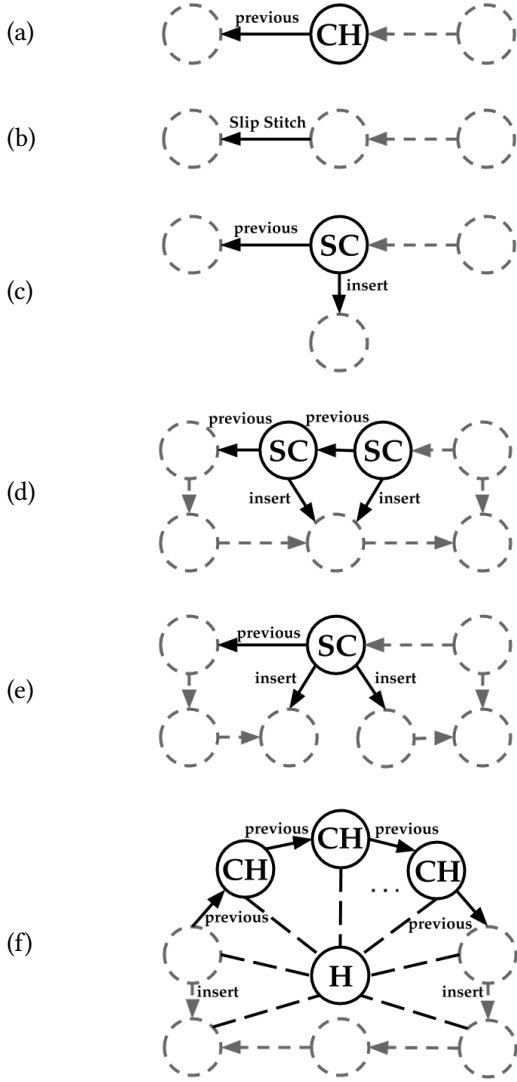
express the interconnectivity of crochet patterns and allows rich tool support through the explicit encoding of all aspects of the pattern.

The basic idea of the language is to directly represent the structure of a pattern, not a linear sequence of stitches to produce this structure. The graph representing the structure also includes the dependency information of how one stitch in the structure depends on the other parts of the structure.

Insertion Points and Transitions. Crochet fabric consists of stitches that tie other loops in the fabric. These loops can be other stitches or different structures such as a hole resulting from a string of chain stitches. Thus, to express a stitch, we need to know where the yarn came from and which other parts of the fabric the stitch connects to. For example, a single crochet stitch uses yarn from a previous stitch and inserts into one or more other loops in the fabric to tie them together in the actual stitch.

In our graph, these loops that are connected through stitches are called *insertion points*. Our graph includes such *insertion points* as *nodes*. Note, that these insertion points do not correspond to stitches directly. A stitch ties loops together, but the stitch itself may not always result in a loop that can serve as an insertion point in the pattern (for example in the case of a slip stitch). Also, not all insertion points in a pattern are the result of a single stitch, but may be purely virtual (for example the string of chain stitches from above).

Edges can represent three different relations between such insertion points: the previous insertion point, an insertion, or a slip stitch connection (see Figure 6). A *previous* edge points to the insertion point that was previously created or used and thus also points into the direction from which the yarn originates. As the yarn will always originate from one previous insertion point, each node has *exactly one* outgoing *previous* edge. *Insertion* edges point to the insertion points that are part of the current stitch. The *slip stitch* edge denotes a slip stitch as discussed below.



chain stitch (CH), single crochet stitch (SC), hole node (H)

Figure 5. Examples of how the graph language can be used to describe different crochet techniques. For example, for the increase method (d) the graph encodes the following: Coming from the stitch on the upper left pointed to by the previous edge, we do a single crochet stitch (sc) into the stitch pointed to by the insert edge. We then follow the previous edge backwards which gets us to a node that tells us to do another single crochet stitch into the same stitch as before. The figure shows the graph-representation of common crochet techniques: (a) the chain stitch (CH) only gains height and does not insert into any stitch, (b) the slip stitch connects to stitches, (c) other stitches, represented by the single crochet stitch, gain height and insert into another stitch, (d) the increase method based on multiple incoming insert edges (e) the decrease method based on multiple outgoing insert edges, (f) a hole node (H) in the graph structure

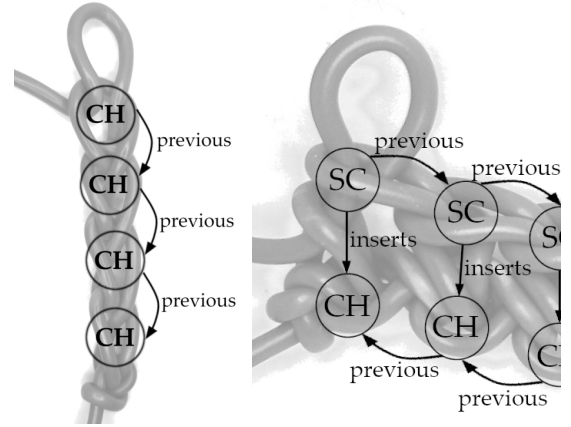


Figure 6. Example graph structure mapped onto crochet fabric. The pattern on the left consists of a simple string of chain stitches (CH), which only depend on the previous chain stitch. The pattern on the right is a two-row pattern, with the first row consisting of chain stitches (H) and the second row of single crochets (sc), which are created by inserting into corresponding chain stitches from the first row.

Insertion edges are ordered among each other, as well as previous and slip stitch edges, which are ordered in one group.

Representing Stitches. Based on this generic perspective, we distinguish between three different groups of stitches and represent each one differently.

Chain stitches only raise the current height. A chain stitch does not connect to any insertion point but allows insertion in or under that stitch for other upcoming stitches. It is represented by a single node representing the created insertion point and a *previous* edge (Figure 5 (a)).

Other stitch types, such as the single crochet, can change the height by different amounts depending on the type of stitch. They insert into an existing insertion point and create a new one. This whole group of stitches is modeled by one node, one *previous* edge, and one or more *insertion* edges (Figure 5 (c)). The insertion edge represents the connection between the current stitch, in this case the single crochet, and the insertion point.

For all stitches that result in an insertion point, the corresponding node stores the type of the stitch as a property.

Slip stitches merely connect two points while not gaining any height or creating any new insertion points. Hence, a slip stitch is represented by a single *slip stitch* edge (Figure 5 (b)). As they can lead the track of the yarn back to a previous insertion point so that this insertion point becomes the previous insertion point for a new insertion point, we have to take special care of slip stitch edges when determining how to crochet the pattern.

Track of the Yarn. For graphs described with the elements above to become patterns that crocheters can produce, crocheters also need an ordering in which they should work through the pattern. Crochet patterns typically are created along one yarn. By following the yarn through the pattern we can deduce a sequence of stitches that crocheters should make to follow the pattern.

The basic rule for tracking the yarn is to simply follow the previous edges backwards from the initial insertion point (one with an incoming but no outgoing previous edge) to the last insertion point (one that has an outgoing previous edge but no incoming one). In case there is no previous edge but only a slip stitch edge, one follows the slip stitch edge forwards. Due to the presence of slip stitches, correctly tracking the yarn is non-trivial, as an insertion point may be the starting point for multiple sub-patterns, with some of these sub-patterns using insertion points of other patterns. A typical situation in which this occurs is when multiple slip stitches lead to the same insertion point, which then serves as the starting point for multiple sub-patterns and thus has multiple *incoming* previous edges.

In general these “intersection nodes” have multiple incoming previous edges or outgoing slip stitch edges or a combination of the two. At such “intersection nodes” we use the ordering between previous and slip stitch edges to determine which one to follow next. As there is only a single yarn, and thus only a single track, an intersection means that all except for one of the outgoing edges start a path that leads back to the intersection. Thus, the next time we encounter the same intersection, we have to choose the next edge according to the order. Also, as “intersection nodes” are the only place at which ambiguities can arise, the rules above ensure that there is always only one path of the yarn through the pattern and that no ambiguities arise.

Crochetability. In our editor (see subsection 3.4) designers create patterns step-wise as if they were crocheting the pattern. As a result of this restriction, designers can only create crochetable patterns. However, for more advanced features, such as the insertion of reusable parts, a notion of crochetability becomes relevant.

For a pattern to be crochetable, each node needs to be crochetable. For a node N to be crochetable, there needs to be a track of yarn that leads from the initial node to this node N , so that none of the nodes along the path uses node N as an insertion point or slip stitch target. Intuitively, this represents the property that no step in the pattern uses an insertion point that is only created in the future.

For example, this means that in order for the pattern to be crochetable, the ordering of the edges needs to adhere to the following rule. When an edge A starts a loop that uses insertion points from a loop that was started by another edge B , the edge A needs to be ordered before B . Otherwise, the loop started by A would use future insertion points.

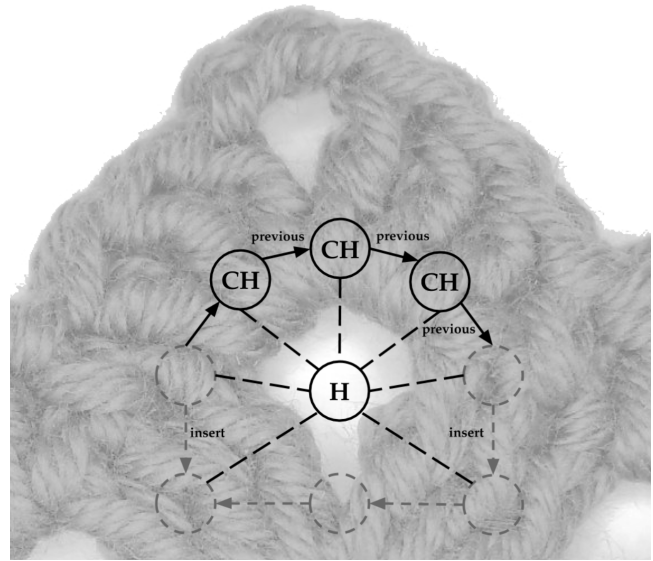


Figure 7. Example subgraph using a hole node mapped onto the fabric of the star shown in Figure 2. The chain stitches around the hole make up the hole. The double crochet stitches making up the ends of the star all stitch into the hole not into the chain stitches making up the hole.

3.2 Support for Particular Techniques

As the language should cover arbitrary patterns, it can express more advanced techniques either by design or through additional mechanisms.

Rows, Rounds. Typically, crochet patterns are structured by rows or rounds. While rows or rounds are not relevant for the reproduction of a pattern, they are a basic unit of many instructions. We unite these two terms into one: *layers*. In the graph, we do not need specific changes in structure to represent either a row or a round. Which method users use solely depends on where you continue inserting stitches after finishing one row or round. Therefore, we merged these terms into one since a pattern can also change anytime from row- to round-wise crochet. Depending on the connections made, you can imitate one or the other method. The graph contains the layer number of each stitch as a property in the node.

Increasing and Decreasing. Increasing and decreasing can be represented in the graph by multiple incoming insertion edges and multiple outgoing insertion edges for nodes (see Figure 5). Increasing happens when several stitches have an insertion edge to the same insertion point (see Figure 5 (d)). A decreasing stitch is a stitch that has more than one outgoing insertion edge (see Figure 5 (e)). The amount of stitches for that layer are reduced by the amount of outgoing insert edges subtracted by one.

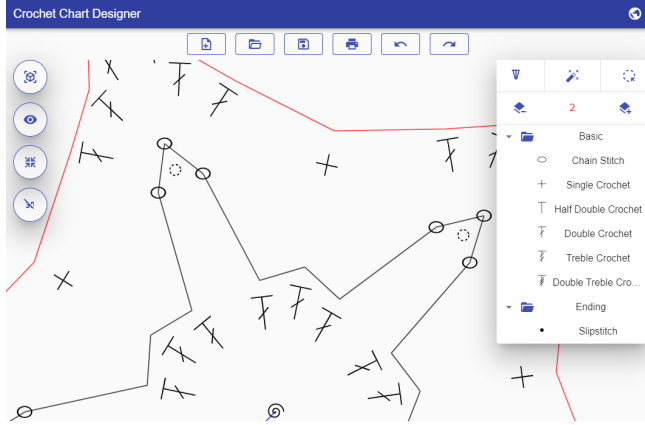


Figure 8. Screenshot of the editor prototype with the 2D view showing part of the pattern used in the user study (see Figure 10)

Insertions into Holes. So far the language assumes that all insertion points are created by stitches. However, for some kinds of patterns it is common to not insert into a stitch but into an opening under a stitch or a chain of stitches. For example, in a pattern a string of chain stitches may create an opening in the pattern. Now, crocheters often do not insert new stitches into the chain stitches creating the opening, but instead insert new stitches into the opening itself (see Figure 7). Standard crochet charts cannot represent this technique (see Figure 11 (b)).

In the proposed language, we represent such *holes* explicitly as nodes and thereby make them available as insertion points (see Figure 5 (f)). The hole nodes are connected to the rest of the graph through directed *surroundingNode* edges pointing to all nodes making up the border of the hole.

Detailed Insertion Points. In order to create specific textures in the resulting fabric, stitches are not always inserted under a whole stitch. Generally, any place in between strands of yarn of the fabric where the crochet hook can pass through are valid insertion points. To represent these variants *insertion* edges have a property denoting the detailed insertion point, such as *both loops*, *back loop only*, *front post*, and others.

Starting Techniques. Patterns can be started in three different ways: a line of chain stitches, a round of chain stitches, or a special loop called “magic ring”. Lines and rounds of chain stitches are not represented in a particular way. A magic ring is represented as a node with the *magic ring* node type. Regardless of the type of start, the first node in a pattern will not have any outgoing edges, as there are no previous nodes to connect to.

3.3 Surface Syntax in 2D and 3D

To make it convenient for crochet designers to express patterns in the semantics defined above, we decided to base our surface syntax on the principles and symbols of crochet charts (see Figure 8) [6]. Our proposed surface syntax is only a very thin presentation layer on top of the graph structure. Essentially, the visualization renders graph nodes as the crochet symbol representing the stitch type of the node and makes use of edges to arrange these symbols. Insertion edges are not rendered as lines but instead we denote visualize insertions through proximity and orientation of symbols, as it is typically done in crochet charts. Different layers are distinguished by different colors. In addition, our surface syntax also explicitly shows the *previous* relation, to allow readers to trace the yarn. Similarly, the syntax includes the slip stitch as an edge with a small symbol on top. The slip stitch is rendered as an edge as readers need to follow it in order to trace the yarn. Further, as we support techniques that are not part of the standard set of symbols, we added symbols for the magic ring technique and holes.

Due to the direct mapping between the graph and the surface syntax, modifications on the graphical representation in the editor directly translate to modifications in the graph, for example removing a double crochet symbol corresponds to removing the corresponding node and its outgoing edges.

Currently, crochet charts are only used to represent two-dimensional patterns, but our language also supports three-dimensional shapes. To support viewing these 3D shapes, we also provide a basic 3D visualization of the pattern. It uses the same general notation of stitch symbols that are oriented towards their insertion points. Further, they are rotated so that they always remain legible for the viewer. Further, to pull the pattern into three dimensions, each type of stitch has a constant height that is used to position the individual stitches with regard to its related stitches. The varying distances between nodes also hint the texture of the fabric.

3.4 Domain-Specific Tool Support

To enable interactive editing of the visual language, we created an editor prototype for designing crochet patterns (see Figure 8). It also serves as a proof-of-concept for tooling based on the language. The editor visualizes patterns as crochet charts in 2D and 3D as described above and automatically adjusts the layout through a force-based algorithm. The visualizations in the editor are only a rendering of the underlying graph structure that fully corresponds to the structure described in subsection 3.1. For editing patterns it provides a point-and-click interface.

The editing features include adding stitches, as well as undo and redo. In the prototype, users have to create a pattern linearly, as if they were crocheting it. While a stitch type is selected, new stitches are added by clicking on the desired

insertion point. These are directly added to the chart and the layout is adjusted. More advanced language features, such as increasing or decreasing, are integrated into this interaction. The linear nature of the creation workflow also ensures that the pattern can be reproduced by a crocheter, as there is an explicitly defined step between each insertion point.

Both the 2D and the 3D view are implemented using force-based layouting as a heuristic. The implementation of the 3D view in particular is based on the 3D force-directed graph library [2] which uses Three.js and WebGL for rendering and a variant of D3 [3] as a physics engine [4]. Despite its general nature, a 3D force-based layout can serve as a first approximation of how a crochet pattern would look like in 3D, and thus is a convenient implementation strategy for our editor prototype. For a more exact model, physical properties of the stitches and the yarn need to be taken into account, as is done by 3D visualizations of knitting patterns [13].

The editor serves as the core environment that can also provide access to other tools we built, such as the *auto completion* and generation of *textual descriptions*.

Stitch Group Repeating. To demonstrate the potential of an explicit graph structure for more advanced tool support, we also implemented a basic tool to repeat a group of stitches. The repetition tool should reduce the effort of creating repetitive patterns. The feature allows designers to repeat a set of previously added stitches for a specified number of times.

Generating Textual Descriptions. Beyond the visualizations of the patterns, our editor can also provide a textual representation in a format similar to the one in Figure 2 (b). As the graph explicitly represents all properties of the pattern, we can easily generate other representations from the graph.

To generate the textual instructions, we follow the virtual “yarn” through the graph (as described in section 3.1). The instruction generation generates the instructions layer by layer. For each layer, the generation condenses repeating stitches of the same type into one token denoting the type and number of repetitions. Further, groups of stitches sharing the same insertion point are enclosed in brackets. Repetitions of larger groups of stitches, repetitions across layers, and absolute references to insertion points are not yet supported.

3.5 Example Shapes

In the following, we present example patterns created with the software to demonstrate the variety of patterns that can be represented (see Figure 9). Alongside each chart we also display the actual crocheted output of the same pattern. This shows that the created patterns are not only valid and can be crocheted but also allow a comparison between the automatic layout of the pattern in the system and the shape of the actual crocheted output.

The first four examples show the variety of shapes that can be created. The star, the heart and the oval (examples 1 to 3) are built using the round method. The triangular shawl (example 4) uses rows instead. All the patterns are displayed with an even but wide spread layout. The similarity to the actual crocheted output is not always exactly the same but the general shape is recognizable.

The other examples all show 3D patterns. The ball (example 5) uses the spiraling round method. The basket (example 6) is based on the oval shape continues with joined rounds and finished with rows to only raise half of the basket. Finally, the complete hat (example 7) is based on the instructions of the myBoshi hat called “Aomori”. The shape of the 3D patterns are similar to the actual output. Only the hat (example 7) and the basket (example 6) are more shallow than the actual crochet results but again the basic shape remains recognizable.

4 User Experience of Designers Creating Digital Patterns

In our evaluation, we aimed to determine (1) whether crochet designers can understand the model of our language well enough to express 2D and 3D patterns correctly and (2) the experience of the editing workflow in the prototype editor.

4.1 Procedure

We investigated these questions in a think-aloud user study with post-study interviews. In the post-study interviews we asked participants about notable situations that occurred while they worked on the tasks. Six professional crochet designers participated in the study. We recruited the designers from a list of 25 participants of a previously conducted, formative survey. In turn, the survey had been sent to a group of 200 crochet designers that offered their designs via the crochet pattern company *myboshi*. All sessions were conducted via online video conferencing. We sent participants the access link to the editor and participants shared their screen while working on the tasks. The study consists of three tasks.

For the first task, we gave participants a state-of-the-art crochet chart that includes ambiguities and missing information and then asked them to go through the pattern in their head and identify flaws and strengths of this chart. The identification of flaws serves as a confirmation of the assumed flaws of the chart. In addition, their dissemination of the pattern serves as the preparation for reproducing the pattern in our editor. We then presented a short introductory video showing the main concepts and interactions of the editor. Afterwards, for the second task the participants were asked to reproduce the pattern from the first task with the prototype using the 2D view. Participants were allowed to refer to the original chart for reproducing the pattern in the editor.

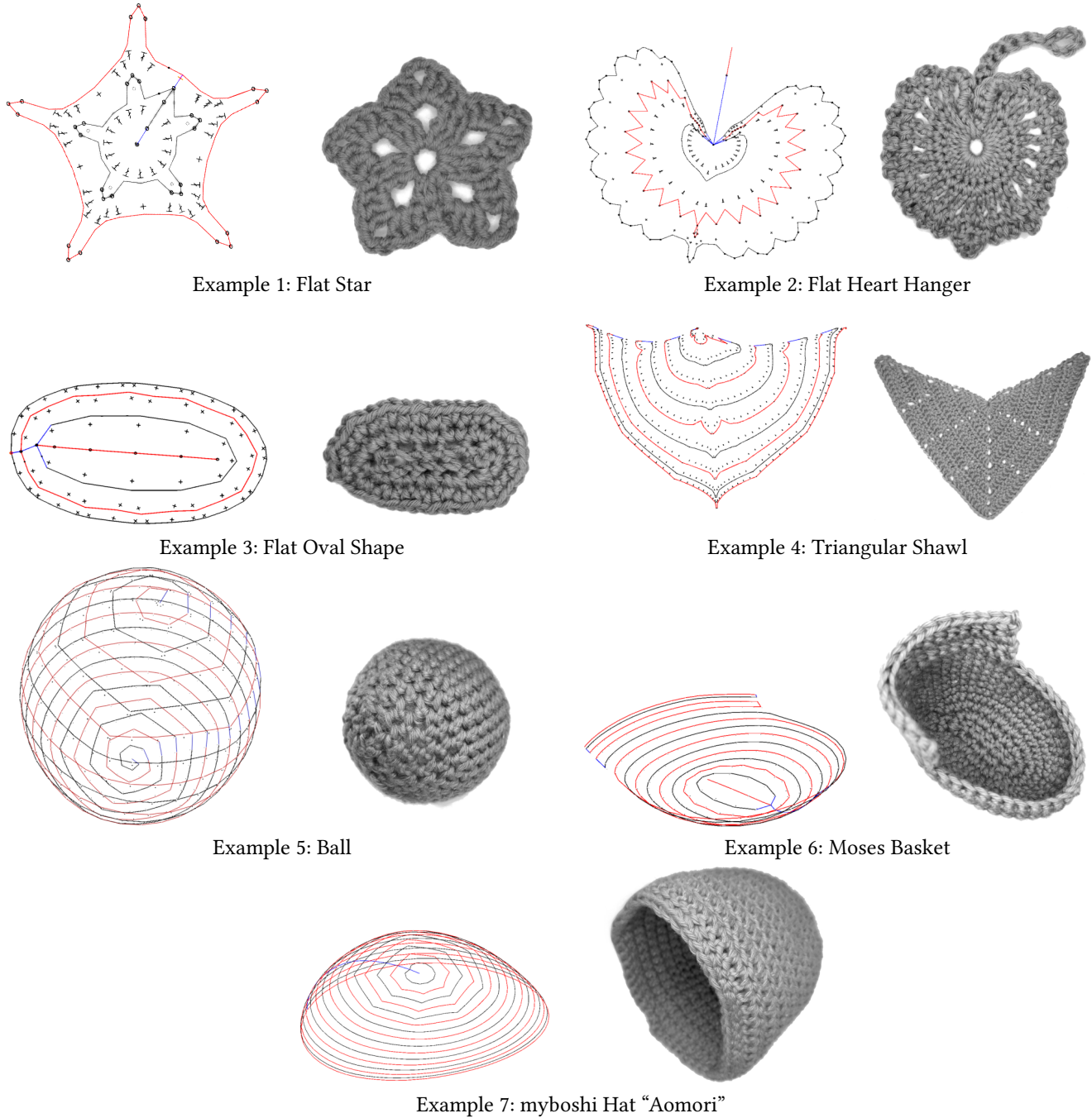


Figure 9. Seven example patterns demonstrating the use of the different crochet methods

Our goal for the third task was to determine the experience of the 3D perspective in the editor in greater detail. Therefore, we asked the designers to create a bowl-shaped object in the 3D perspective. We explicitly introduced the increase and decrease methods and the stitch group repeating tool again but did not constrain the designers in their approach to creating the pattern in any way.

4.2 Task 1: Evaluation of an Existing Chart

For this task, we showed the designers a crochet chart of a flat star (see Figure 10). We chose this pattern, as it includes a variety of techniques, we suspected it to be ambiguous and incomplete, and it was small enough to keep the discussion short. We assumed that, while the chart seems clear and well drawn, it has three weak points which are imprecisely notated and give unclear instructions (see Figure 11): the

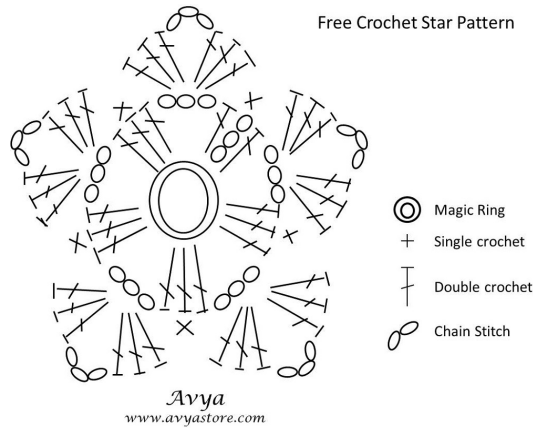


Figure 10. Crochet chart used in the user study

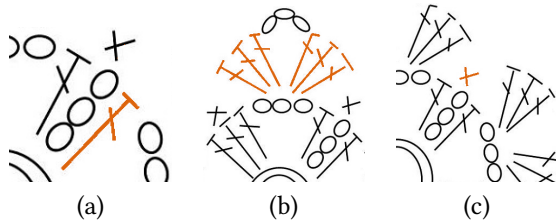


Figure 11. Three extracts from the chart showing expected problems: (a) unspecified closing method for the first round, (b) insertion point missing for the six double crochets on each point of the star, (c) unspecified closing method for the last round

closing method of the first round is unspecified, the insertion point of the double crochets for points of the star are missing, and the closing method for the second round is unspecified.

We first asked participants, whether the charts seems comprehensible. All participants observed that the given crochet chart was very well-arranged. At first sight, only two participants mentioned a missing slip stitch to finish the last round and one was missing an indicator for the beginning of each round.

We then asked participants to orally follow the chart stitch by stitch and comment on any encountered flaws. The participants identified the three issues we suspected to be present in the pattern (see Figure 11). This shows that even small charts can already be flawed even though they look well-arranged and correct at first glance.

4.3 Task 2: Creating the Star Pattern in 2D

The participants were asked to reproduce that pattern from the first task with the prototype using the 2D view (see Figure 12 for the results for each participant). All designers, except for participant A, were able to reproduce the whole design successfully. Participant A did not finish the star, as they were confused by the movement of the pattern that

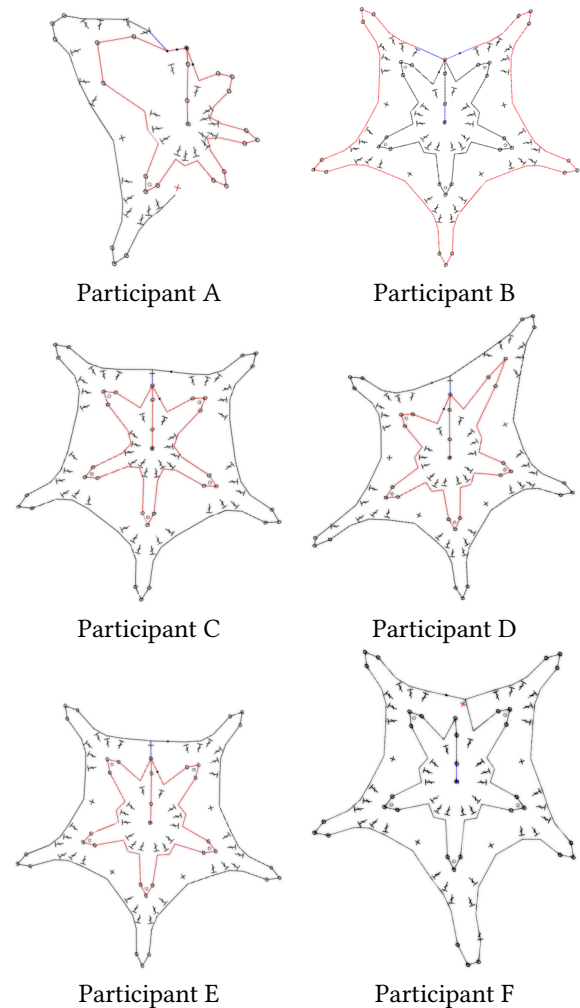


Figure 12. Resulting charts of the 2D patterns created by the participants

happens whenever a new node is added. Another notable issue with the participant solutions is the missing layer information in the pattern created by participant F who forgot to increase the layer counter. After finishing their patterns, four of the designers remarked that reproducing the pattern in the editor made the issues previously identified apparent. This may result from the linear editing workflow that corresponds to the crochet process and thereby requires crocheters to consider every detail of the pattern. The patterns vary, as the designers were free to choose how to handle the ambiguous instructions.

Four designers mentioned the shape preview of the editor as a beneficial feature, as it could, for example, show the effect that different methods have on the resulting form. At the same time, the designers still saw room for improvement with regard to the layout of the crochet chart. They noted that the space between some stitches is too large and that it impeded readability and broke symmetries.

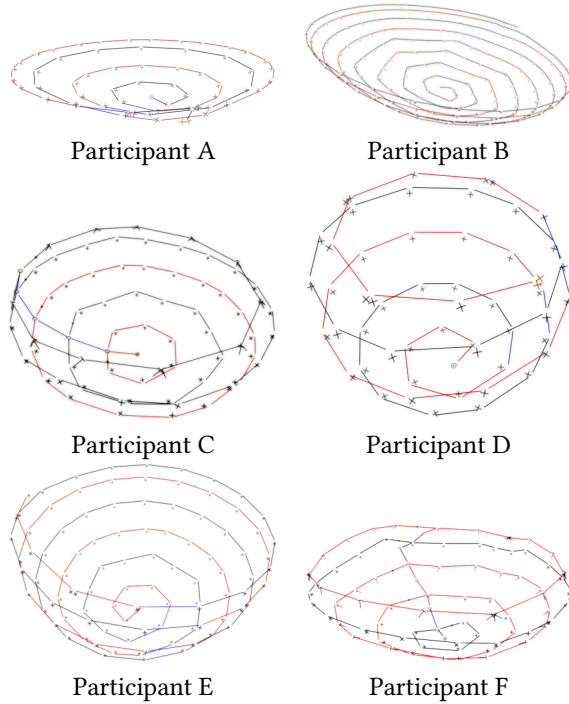


Figure 13. Resulting charts of the 3D patterns created by the participants

One designer noted that it was unusual for them to add stitches in the order they would be created. They were used to other crochet design tools and said that they were used to place stitches out-of-order to determine the general layout of the pattern beforehand. They also noted that they realized that they would not need that due to the automatic layout.

Other general feedback concerned the user interaction for applying the decrease method, and the confusion and inconvenience resulting from the rotation of the rendered chart while adding nodes. The designers enjoyed using the advanced tools such as the auto completion, as it could save time, and the capability to explicitly denote holes, as it is not supported by current charts.

4.4 Task 3: Creating a Bowl in 3D

For the last task, we asked the participants to design a bowl using the 3D view in the prototype editor (see Figure 13 for the results for each participant).

When they initially switched to the 3D perspective, five of the designers expressed amazement at the possibilities this might provide. Three participants particularly mentioned the benefits of the perspective for amigurumi patterns (crocheted figurines), for example for comparing the proportions of multiple parts.

At the same time, four of the six designers struggled with finding a good perspective to continuously work on the pattern. As a consequence, some of them regularly lost track

of their progress and current location in the current layer. The fact that the navigation in a 3D perspective was new to all of them probably contributed to the disorientation.

Overall, some designers also mentioned they were missing an overview of their progress, such as the count of stitches per layer and a clear marking of the layer numbers. Furthermore, the designers tended to forget to update the layer number and one suggested that the editor may remind them of the layer change. Additionally, two designers explicitly asked to build their own repeatable patterns, a set of stitches which can be chosen similar to the stitch types and added by a single click.

During the task, we also explicitly asked participants to use the auto completion feature. Two participants initially struggled with the user interface for defining the repetition to be applied. After successfully applying the auto completion for at least one layer, two designers continued using it for the rest of the pattern and all designers noted that this feature would save them a lot of time in describing patterns.

5 Discussion

Domain-specific tool support for crochet pattern design has the potential to reduce manual effort for designers, and support them in creating unambiguous and complete instructions for crocheters. We proposed a graph-based language for describing crochet patterns and illustrated the potential for tool building through a editor prototype. The evaluation based on this editor prototype shows that our language can be used by crochet designers, and their feedback indicates that they also expect to benefit from the resulting tools.

In the following, we discuss some of the limitations of the language and the current editor prototype, more advanced editing capabilities, and potential, future domain-specific tools.

5.1 Scope of Language and Editor

Generally, our proposed language aims to cover all techniques of crochet. Nevertheless, some aspects of crochet patterns are not fully supported yet. Notably, switching the yarn to a different one, for example to change the color, can currently not be expressed. While it could easily be represented as a property for nodes and edges, the design of a visualization remains a challenge, as the color of symbols typically denotes different layers.

With regard to the editor, a stable layout algorithm for the 2D and 3D visualization of crochet patterns is desirable. The movement of the pattern caused by the force-based layout made it difficult for users to orient themselves. Also, the designers liked the 3D visualization of their patterns but at the same time the current force-based layout is only an approximation of the actual three-dimensional form of a pattern. A more exact visualization would thus be an interesting avenue for future research.

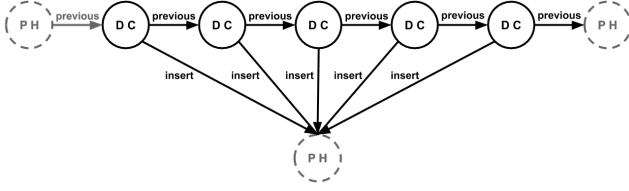


Figure 14. Representation of partial patterns in the graph structure. With the example of a shell stitch of five double crochet (dc) stitches. Place holder nodes (ph) mark open connection points

Further, another major limitation of the the editor is the linear editing workflow, which impedes the exploration of designs. In order to correct errors early in a pattern, designers would need to undo large parts of a pattern. We expect the mechanism of partial patterns as discussed below to support non-linear editing.

Concepts such as selecting nodes and edges or undo and redo can be implemented in a straightforward manner. But the concept of copy and paste will require further calculations and decisions to make it work for the crochet domain. A selection of a pattern might need to be adapted for insertion at a different position in the graph. In crochet patterns, especially when using the increase or decrease methods, a whole layer cannot be simply duplicated, like you would duplicate a line in text. Here, a program would need to find domain specific solutions and handle various functions separately, such as layer duplication and exact selection copies.

5.2 Partial Patterns for Components, New Stitches, and Copy-And-Paste

Several advanced editing features may be added to tools such as the reuse of patterns as components in other patterns, the definition of custom stitches, and non-linear editing including copy-and-paste. All of these features require support for incomplete graphs. For example, when copying a sub-graph of a pattern containing two nodes that both insert into the same other node, we have to preserve this structure, while extracting the sub-graph. Further, designers may want to design custom stitches to represent techniques beyond the basic stitches, such as the shell stitch resulting from working five double crochet stitches into a single stitch [6].

On the graph level, all these use cases are an insertion of a partial sub-graph into an existing graph. When such partial graphs are inserted into a pattern, the resulting pattern should still be crochetable. So far, the way crochet pattern graphs are constructed in the editor prototype, all patterns are crochetable by construction. We expect to be able to support this through introducing *placeholder* nodes in the graph (see Figure 14). When creating a partial graph, designers can use placeholder nodes to denote the connections between

the partial graph and a pattern graph later on. For example, when creating the shell stitch (see Figure 14), designers have to place one placeholder not to get an initial previous edge. They also need one placeholder edge to denote that all five double crochet stitches are inserted into the same, at this point virtual, node. Finally, the shell stitch can lead to another node, thus the designers need a last placeholder node for the outgoing previous edge. By using placeholder nodes, the shell stitch partial graph is now reusable and can be applied repeatedly in different patterns.

To insert a subpattern expressed with placeholders, users have to replace the placeholder nodes with appropriate nodes in the graph. For example, in the case of the shell stitch, users first have to select a suitable node that can have an outgoing previous edge for the initial placeholder node (the left-most placeholder node in Figure 14). Then they have to select a node in which they want to insert the five double crochet stitches (the bottom placeholder node in Figure 14). Finally, they may choose to replace the final placeholder node but may choose not to in case they want to continue to manually add stitches afterward. By simply selecting two nodes in an existing pattern, users were able to re-use the whole shell stitch subgraph quickly. The editor can check for the crochetableity of the resulting overall pattern before actually adding the subgraph, and thereby we can again ensure that the requirements of the stitches are met by construction.

Placeholder nodes enable designers to create reusable sub-patterns by denoting the required nodes of the subpattern when inserting it into a larger pattern. As a result, designers can create custom stitches or modular patterns that consist of individual partial graphs and an overall pattern that describes how these partial graphs are integrated into a crochetable pattern.

5.3 Advancing Crochet-Specific Tools

An environment which uses our graph structure for representing crochet patterns can now support the designer in crochet-specific ways.

As described earlier, typical formats for pattern instructions are charts and textual descriptions. Charts can be generated from our graph as we have shown with the 2D and 3D visualizations. Also, we can already derive textual instructions, even though they are still more repetitive than necessary. A more detailed analysis of repetitions in the graph could be used to create succinct instructions. A realistic rendering of the resulting object, similar to the rendering of knitted garments, would also be beneficial, but would require a more detailed modeling of the physical yarn behavior [34].

Based on the graph structure, several automatic analysis features can be supported. A pattern can be automatically characterized based on its difficulty, the estimated duration of reproducing it, and estimated yarn usage. Further, by incorporating information about stitch heights and the physical

yarn behavior, a predication may be made whether a pattern results in a 2D or a 3D shape.

Beyond mere analysis, tools may now also support transforming patterns. For example, based on an estimate of the difficulty, a pattern may be simplified, or patterns for garments may be scaled for different sizes. Both use cases would, however, again require modeling yarn behavior in more detail.

6 Conclusion

Crochet designers do not yet benefit from domain-specific tools for digital pattern design and have to resort to generic text or graphics editors. We propose a visual, domain-specific language for representing crochet patterns as a graph. This language covers all concepts required to unambiguously express instructions for crochet patterns. Based on the language, we illustrate the potential for tool building in a prototype of an editor providing 2D and 3D perspectives. Our evaluation shows that designers can express 2D, as well as 3D patterns in the language, and that they also regard resulting tool support as beneficial.

Thereby, our language may lay the foundations for more advanced crochet tools that will provide professional designers with new capabilities, reduce the effort for pattern creation, and may ultimately also make pattern design more accessible for more crocheters.

Acknowledgments

We also gratefully acknowledge the financial support of HPI's Research School² and the Hasso Plattner Design Thinking Research Program³.

References

- [1] Adriprints. 2020. *Stitchin*. <https://www.fonts.com/font/adriprints/stitchin>
- [2] Vasco Asturiano. 2020. *3D Force-Directed Graph*. <https://github.com/vasturiano/3d-force-graph>
- [3] Mike Bostock. 2020. *D3.js*. <https://d3js.org/>
- [4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D³ data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.
- [5] Merrow Sewing Machine Company. 2020. *Merrow 18-E Crochet Stitch Industrial Sewing Machine*. <https://www.merrow.com/crochet-sewing-machines/18e>
- [6] Craft Yarn Council's. 2020. *www.YarnStandards.com*. <https://www.craftyarncouncil.com/standards/crochet-chart-symbols>
- [7] Jared B. Counts. 2018. *Knitting with Directed Graphs*. Bachelor's Thesis. Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science. <https://web.archive.org/web/20200914160119/https://dspace.mit.edu/bitstream/handle/1721.1/119547/1076272960-MIT.pdf?sequence=1>
- [8] Sander de Bruijne. 2020. *Stitch Fiddle*. <https://www.stitchfiddle.com/>
- [9] Andrea Ehrmann, Johannes Fiedler, and Nils Grimmelsmann. 2018. Crochet Machine. DE Patent WO002018114025A1.
- [10] Mikhaila Friske, Jordan Wirfs-Brock, and Laura Devendorf. 2020. Entangling the Roles of Maker and Interpreter in Interpersonal Data Narratives: Explorations in Yarn and Sound. In *DIS '20: Designing Interactive Systems Conference 2020, Eindhoven, The Netherlands, July 6-10, 2020*, Ron Wakkary, Kristina Andersen, Will Odom, Audrey Desjardins, and Marianne Graves Petersen (Eds.). ACM, 297–310. <https://doi.org/10.1145/3357236.3395442>
- [11] Richard Gangi. 1988. Warp Knitting/Crochet Warp Knitting Machine. US Patent 4,761,973.
- [12] Nils Grimmelsmann, Christoph Döpke, Svea Wehlage, and Andrea Ehrmann. 2019. The Largest Crocheting Machine in the World. *Mel-liand International* 25 (06 2019), 99–100.
- [13] Runbo Guo, Jenny Lin, Vidya Narayanan, and James McCann. 2020. Representing Crochet with Stitch Meshes. In *SCF '20: Symposium on Computational Fabrication, Virtual Event, USA, November 5-6, 2020*, Emily Whiting, John Hart, Cynthia Sung, Nadya Peek, Masoud Akbarzadeh, Daniel Aukes, Adriana Schulz, Hayden Taylor, and Jeeun Kim (Eds.). ACM, 4:1–4:8. <https://doi.org/10.1145/3424630.3425409>
- [14] David Henderson and Daina Taimina. 2001. Crocheting the Hyperbolic Plane. *The Mathematical Intelligencer* 23 (06 2001), 17–28. <https://doi.org/10.1007/BF03026623>
- [15] Megan Hofmann, Lea Albaugh, Ticha Sethapakadi, Jessica Hodgins, Scott E. Hudson, James McCann, and Jennifer Mankoff. 2019. KnitPick-ing Textures: Programming and Modifying Complex Knitted Textures for Machine and Hand Knitting. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (New Orleans, LA, USA) (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 5–16. <https://doi.org/10.1145/3332165.3347886>
- [16] HookinCrochet. 2020. *HookinCrochet Crochet Graph Designer Software*. <http://www.hookincrochet.com/>
- [17] HookinCrochet. 2020. *Symbols Basics Font Software*. <http://www.hookincrochet.com/>
- [18] Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. 2008. Knitting a 3D Model. *Comput. Graph. Forum* 27, 7 (2008), 1737–1743. <https://doi.org/10.1111/j.1467-8659.2008.01318.x>
- [19] Morton James. 1933. Warp Knitting Machine. US Patent 1,924,649.
- [20] Alexandre Kaspar, Liane Makatura, and Wojciech Matusik. 2019. Knitting Skeletons: A Computer-Aided Design Tool for Shaping and Patterning of Knitted Garments. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology, UIST 2019, New Orleans, LA, USA, October 20-23, 2019*, François Guimbreti re, Michael Bernstein, and Katharina Reinecke (Eds.). ACM, 53–65. <https://doi.org/10.1145/3332165.3347879>
- [21] Alexandre Kaspar, Tae-Hyun Oh, Liane Makatura, Petr Kellnhofer, and Wojciech Matusik. 2019. Neural Inverse Knitting: From Images to Manufacturing Instructions. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3272–3281.
- [22] Christina Koch. 2017. *The Computer Science of Knitting*. <https://christinalk.github.io/blog/2017/02/06/knitting-programming>
- [23] James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica K. Hodgins. 2016. A compiler for 3D machine knitting. *ACM Trans. Graph.* 35, 4 (2016), 49:1–49:11. <https://doi.org/10.1145/2897824.2925940>
- [24] Vidya Narayanan, Lea Albaugh, Jessica K. Hodgins, Stelian Coros, and James McCann. 2018. Automatic Machine Knitting of 3D Meshes. *ACM Trans. Graph.* 37, 3 (2018), 35:1–35:15. <https://dl.acm.org/citation.cfm?id=3186265>
- [25] Hinke M Osinga and Bernd Krauskopf. 2004. Crocheting the Lorenz Manifold. *Mathematical Intelligencer* 26, 4 (Sept. 2004), 25–37. <https://doi.org/10.1007/BF02985416>

²<https://hpi.de/en/research/research-school.html>

³<https://hpi.de/en/dtrp/>

- [26] Mariana Popescu, Matthias Rippmann, Tom Van Mele, and Philippe Block. 2018. Automated generation of knit patterns for non-developable surfaces. In *Humanizing Digital Reality*. Springer, 271–284.
- [27] Afroditi Psarra, Sadaf Sadri, Esteban Agosin, Grace Barar, Rylie Sweem, Cindy Xu, Ruoxi Song, and Zoe Kaputa. 2021. Sensing Textures: Tactile Resistance. In *ISWC 2021: Proceedings of the 2021 ACM International Symposium on Wearable Computers, Virtual Event, September 21–26, 2021*, Daniel Roggen, Katia Vega, and Hsin-Liu Cindy Kao (Eds.). ACM, 211–215. <https://doi.org/10.1145/3460421.3478833>
- [28] Johanna Riedl and Emanuel Gollob. 2014. *Luftmaschinen Häkelmaschine*. <https://www.youtube.com/watch?v=LFyey3wCnNU>
- [29] Arnim Schachtschabel. 2020. *Berliner Häkelschrift*. <http://www.schachzabel.de/>
- [30] Stitchworks Software. 2020. *Crochet Charts*. <http://stitchworksoftware.com/>
- [31] D.J. Spencer. 2001. *Knitting Technology: A Comprehensive Handbook and Practical Guide*. Elsevier Science.
- [32] Ella Taylor-Smith, Colin F. Smith, and Michael Smyth. 2018. Democratic Participation through Crocheted Memes. In *Proceedings of the 9th International Conference on Social Media and Society, SMSociety 2018, Copenhagen, Denmark, July 18–20, 2018*. ACM, 178–186.
- <https://doi.org/10.1145/3217804.3217910>
- [33] Martijn ten Bhömer, Hai-Ning Liang, and Difeng Yu. 2019. Machine Learning Enhanced User Interfaces for Designing Advanced Knitwear. In *HCI International 2019 - Posters - 21st International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings, Part II (Communications in Computer and Information Science, Vol. 1033)*, Constantine Stephanidis (Ed.). Springer, 212–219. https://doi.org/10.1007/978-3-030-23528-4_30
- [34] Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph.* 31, 4 (2012), 37:1–37:12. <https://doi.org/10.1145/2185520.2185533>
- [35] Elena Zaharieva-Stoyanova and Stefan Bozov. 2017. Application of XML-based Language for Digital Representation of Crochet Symbols. *Digital Presentation and Preservation of Cultural and Scientific Heritage* 7, 1 (2017), 181–189.
- [36] Luigi Omodeo Zorini and Marco Carnevale Miino. 2007. Crochet Galloon Machine. US Patent 7,251,960.

Received 2022-07-12; accepted 2022-10-02