Generation of Celtic Key Patterns with Tree-based Collage Grammars

Renate Klempien-Hinrichs and Caroline von Totth

Abstract. Tree-based collage grammars are a syntactic device for modelling visual languages. Celtic art provides such languages, which follow precise rules of construction, for instance key patterns and knotwork. In this paper, we study the syntactic generation of Celtic key patterns using tree-based collage grammars. Moreover, we compare the regulation mechanisms employed here in order to ensure that only consistent key patterns are generated with those that were used in a previous study for knotwork.

1 Modelling with collage grammars

Collage grammars are a device to generate sets of pictures in a syntactic manner [HK91,HKT93,DK99]. They were developed by equipping hyperedge replacement grammars with spatial information: nonterminals come with a location and an extension in some Euclidean space, and the role of terminals is played by collages, where a collage is a union of parts and a part is a set of points. Equivalently, collage grammars may be seen as a combination of a tree grammar and an algebra interpreting the generated trees as collages [Dre00,Dre06], so-called tree-based collage grammars.

The initially proposed collage grammars were context-free. More powerful grammar types include the table-driven (or T0L) grammars, where nonterminals are replaced in parallel with rules from one of finitely many tables [KRS97] (for T0L collage grammars see [DKK03]). The behaviour of a table-driven tree grammar may be emulated by a regular tree grammar generating unary trees as input for a top-down tree transducer; such a unary tree basically states the sequence of the applied tables in the T0L grammar [Dre06, Lemma 2.5.7].

For any picture-generating mechanism, it is of particular interest to find out whether a certain class of pictures can be generated. For table-driven collage grammars, the case study in [DK00] (see also [Dre06, Sect. 3.5]) shows a way to produce Celtic knotwork. The idea is to identify a small pattern that is repeated in a tiling-like manner to form the whole design (see [Bai90,Slo95] for distinct ways to divide a design).

In addition to knotwork, there are three other major design types in Celtic art: key patterns, spirals, and interlacings with animal or human forms (see [All93,Bai51] for overviews). Historically, the designs were drawn in medieval manuscripts, carved on standing stones, or forged in precious metalwork.

In this paper, we present a case study modelling Celtic key patterns, sometimes also called maze patterns, by means of collage grammars. Where a Celtic knot consists of continuous cords that interweave, key patterns have continuous

F. Drewes, A. Habel, B. Hoffmann, D. Plump (Eds.): Manipulation of Graphs, Algebras and Pictures. Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday, pp. 205–222, 2009.

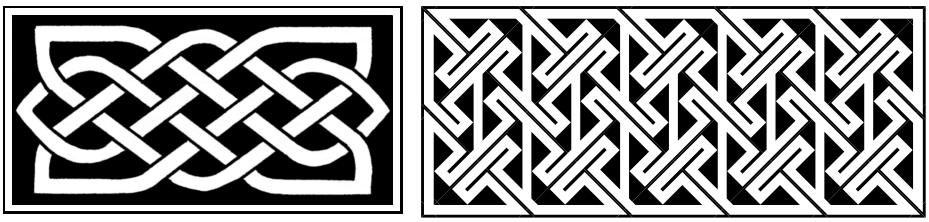

Fig. 1. Celtic designs: a knot (left) and a key pattern (right)

paths that follow straight lines, usually at 45° angles, and meet at crossings (see Figure 1 for examples).

Methods for the construction of key patterns by hand, that may have been used by the original Celtic artists, are described in [Bai90,Bai93,Mee02]; these methods are mainly oriented at which straight long lines may be drawn. On the other hand, small patterns or *tiles* may be identified that are repeated throughout the whole design. For this, Sloss [Slo97] overlays a key pattern with a grid whose axes run parallel to the pattern border, obtaining rectangles of various sizes. In contrast (and more in line with [Bai93]), we propose to use triangles and squares with a 45° slope as tiles. We then use tree-based collage grammars to show how these tiles can be structurally combined to yield well-formed key patterns. Moreover, we discuss the structural differences between knotwork [DK00] and key patterns. All collage grammars for key patterns that we developed were implemented using Frank Drewes' system Treebag [Dre06, Sect. 8].

The paper is organised as follows: In Section 2, basic notions of tree-based collage grammars are recalled. Section 3 contains a structural analysis of basic key patterns and a description of the collage grammar that we used to generate them. In Section 4, variations of the first model are proposed. The paper concludes with a brief summary and ideas for various lines of future research.

2 Tree-based collage grammars

The basic notions for tree-based collage grammars collected in this section follow presentations given in [DKL03,DEKK03,Dre06]. Please consult these, [Dre06] in particular, for more detail, including examples.

Let $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ denote the set of natural numbers, and $[n] = \{1, \dots, n\}$ for $n \in \mathbb{N}$. Moreover, $\mathbb{R}$ denotes the set of real numbers, and $\mathbb{R}^2$ the Euclidean plane, which contains points $(x, y) \in \mathbb{R}^2$.

We will use expressions to represent collages and call such an expression a *term* (or *tree*) over a certain signature. In general, a *signature* is a finite set Σ of symbols, each symbol $f \in \Sigma$ being assigned a unique *rank* $\mathrm{rank}_\Sigma(f) \in \mathbb{N}$. The set T_Σ of *terms over* Σ is the smallest set such that for $n \in \mathbb{N}$, $f[t_1, \dots, t_n] \in T_\Sigma$ for all $f \in \Sigma$ with $\mathrm{rank}_\Sigma(f) = n$ and all $t_1, \dots, t_n \in T_\Sigma$. For $n = 0$, we may omit the parentheses in $f[\,]$, writing just f instead.

It is now possible to define collages and the relationship between collages and terms in T_Σ formally. A *part* $p \subseteq \mathbb{R}^2$ is a bounded subset of the Euclidean plane; intuitively, p is a set of black points. A collage C is a finite set of parts, and the set of all collages is denoted by $\mathcal{C}$. Transformations $f\colon \mathbb{R}^2 \to \mathbb{R}^2$ on $\mathbb{R}^2$ are canonically extended from points to parts and collages, i.e., $f(p) = \{f(x,y) \mid (x,y) \in p\}$ for a part p, and $f(C) = \{f(p) \mid p \in C\}$ for a collage C. For affine transformations $f_1, \ldots, f_n$ on $\mathbb{R}^2$, $\langle\!\langle f_1 \cdots f_n \rangle\!\rangle$ denotes the operation $f\colon \mathcal{C}^n \to \mathcal{C}$ given by $f(C_1, \ldots, C_n) = \bigcup_{i \in [n]} f_i(C_i)$ for all $C_1, \ldots, C_n \in \mathcal{C}$. A *collage operation* is either an operation of the form $\langle\!\langle f_1 \cdots f_n \rangle\!\rangle$ for affine transformations $f_1, \ldots, f_n$ ($n \geq 1$) or a constant collage, viewed as an operation of arity 0. A *collage signature* is a finite signature Σ consisting of collage operations, where ranks coincide with arities. For a term $t \in T_\Sigma$, its *value* is $\mathrm{val}(t)$, i.e., the collage $\mathrm{val}(t) = f(\mathrm{val}(t_1), \ldots, \mathrm{val}(t_n))$ if $t = f[t_1, \ldots, t_n]$.

By such an interpretation of ranked symbols as collage operations, appropriate grammatical devices for generating sets of terms allow us to generate sets of collages. Among such devices, we consider the regular tree grammar and the top-down tree transducer, and we call the combination of a tree generator and a collage algebra a *tree-based collage grammar*. Since we deal exclusively with tree-based collage grammars in this paper, in the following we will refer to them simply as collage grammars.

A regular tree grammar works analogously to a regular string grammar, but by replacing nonterminals in terms and having nonterminals occur only without subterms. Formally, a *regular tree grammar* is a system $G = (N, \Sigma, P, S)$ consisting of

- a finite set N of *nonterminals*, which are considered to be symbols of rank 0,
- a signature Σ disjoint with N,
- a finite set $P \subseteq N \times T_{\Sigma \cup N}$ of *productions*, and
- an *initial nonterminal* $S \in N$.

A term $t \in T_{\Sigma \cup N}$ *directly derives* a term $t' \in T_{\Sigma \cup N}$, denoted by $t \longrightarrow_P t'$, if there is a production $A ::= s$ in P such that t' is obtained from t by replacing an occurrence of A in t with s. The *language generated by* G is $L(G) = \{t \in T_\Sigma \mid S \longrightarrow_P^* t\}$, where $\longrightarrow_P^*$ denotes the reflexive and transitive closure of $\longrightarrow_P$; such a language is called a *regular tree language*.

A tree transducer transforms some input tree into an output tree by traversing the input tree symbol by symbol, possibly storing some information in one of finitely many states. Formally, a *top-down tree transducer* is a system $td = (\Sigma, \Sigma', \Gamma, R, \gamma_0)$ consisting of

- finite *input* and *output signatures* Σ and Σ',
- a finite signature Γ of *states* of rank 1, where $\Gamma \cap (\Sigma \cup \Sigma') = \emptyset$,
- a finite set R of *rules*, and
- an *initial state* $\gamma_0 \in \Gamma$.

Each rule in R has the form

$$\gamma[f[x_1, \ldots, x_n]] \to t[\![\gamma_1[x_{i_1}], \ldots, \gamma_m[x_{i_m}]]\!],$$

where $\gamma, \gamma_1, \ldots, \gamma_m \in \Gamma$ are states, n is the rank of symbol $f \in \Sigma$ and $x_1, \ldots, x_n$ are pairwise distinct variables (of rank 0 and not occurring in $\Sigma \cup \Sigma' \cup \Gamma$); and t is a term with symbols from Σ' and $m \in \mathbb{N}$ subterms of the form $\gamma_j[x_{i_j}]$ with $i_j \in [n]$, i.e., $x_{i_j} \in \{x_1, \ldots, x_n\}$.

There is a *computation step* of td from a term s to a term s', denoted by $s \Rightarrow_R s'$, if R contains a rule $\gamma[f[x_1, \ldots, x_n]] \to t[\![\gamma_1[x_{i_1}], \ldots, \gamma_m[x_{i_m}]]\!]$, s has a subterm of the form $\gamma[f[t_1, \ldots, t_n]]$, and replacing this subterm in s with $t[\![\gamma_1[t_{i_1}], \ldots, \gamma_m[t_{i_m}]]\!]$ (which is formed as in the rule, but with corresponding subtrees $t_{i_1}, \ldots, t_{i_m}$ from s instead of variables $x_{i_1}, \ldots, x_{i_m}$) yields s'. The *tree transformation computed by* td is given by $td(s) = \{s' \in T_{\Sigma'} \mid \gamma_0[s] \Rightarrow_R^* s'\}$ for every tree $s \in T_\Sigma$, where $\Rightarrow_R^*$ denotes the reflexive and transitive closure of $\Rightarrow_R$.

3 The structure of Celtic key patterns

In this section, we show a way to structure Celtic key patterns so that they can be easily generated by collage grammars. We first identify a small set of tiles – the constant collages – that allows us to put together a whole pattern, and then describe the syntactic synthesis.

Following [Bai93], a typical key pattern will often be a coherent composition of isosceles right-angled triangles to which a border line needs to be added, see Figure 2 left.

Moreover, note that the smooth borders must be produced by triangle hypotenuses. In particular, the top and bottom border triangles have horizontal hypotenuses, whereas the hypotenuses of all other triangles are vertical. This

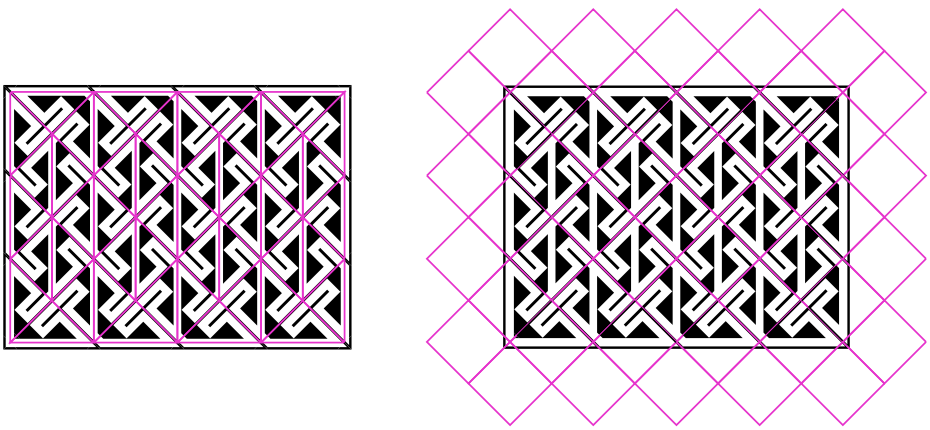

Fig. 2. Division of a key pattern into triangles (left) and squares (right)

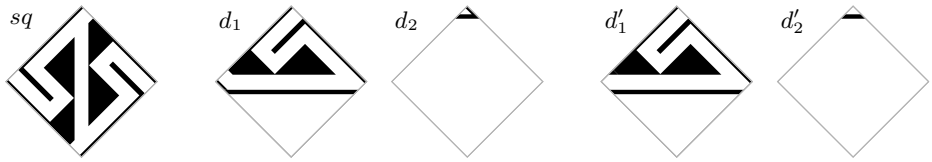

Fig. 3. The set of basic tiles

suggests a division into square tiles as indicated in Figure 2 right, which also takes care of drawing the border lines. The resulting set of basic tiles (modulo reflections and 90° rotations) is shown in Figure 3. At first glance, the last two of the five tiles appear to be identical to the preceding two. There are, however, subtle differences, for two reasons. First, key patterns have two different types of corners since a corner is obtained by so-called mitring, which means reflecting a basic triangle at one of its shorter sides; compare, e.g., the two right corners of a pattern in Figure 2. Secondly, a line always comes with a thickness, and where a hand-drawn line lies on the connecting sides of two tiles, then each of the graphic tiles gets a line of half the thickness. Thus, the two middle tiles d_1 and d_2 in Figure 3 have a short line at a 135° angle that is not present in the last two tiles, and in tile d'_1 the left corner of its left black triangle is pointed.

Now let us come to the syntactic arrangement of these tiles in a rectangular key pattern. The task is to overlay the pattern with a tree whose interpretation through collage operations yields the pattern. In order to find the tree, we reuse the idea from [DK00] for rectangular knotwork, namely to develop the pattern from its centre. One can immediately observe from Figure 2 that the left half of the pattern may be obtained from the right half by a rotation of 180° about the centre. The right half in turn may be seen as having a horizontal backbone through the centre, where each vertical line of square tiles above the backbone may be obtained from the same line below the backbone by a rotation of 180° about the crossing point of the vertical line with the backbone. (The border, however, will need special treatment, which we will discuss at the end of the section.) Thus, we can concentrate on the lower right section as sketched in Figure 4 to get an idea of the required collage signature.

As proposed earlier, the figure displays a portion of the tree for the key pattern in Figure 2 overlaying the pattern structure. The constant symbols sq, d_1 and d_2 refer to the tiles given in Figure 3; their placement results from the operations above them in the tree and reflects the layout of the tiles in the pattern. Constants r_1 and r_2 result from mitring d_1 and d_2, respectively, as described above. Constant sq_m results also from a reflection of sq, but this time in the vertical (or, equivalently, horizontal) axis through the middle of the tile. This reflection is necessary because two adjoining tiles need to agree in the path linking them (this is unlike the knotwork tiles considered in [DK00], where the cord always leaves through the centre of a tile side).

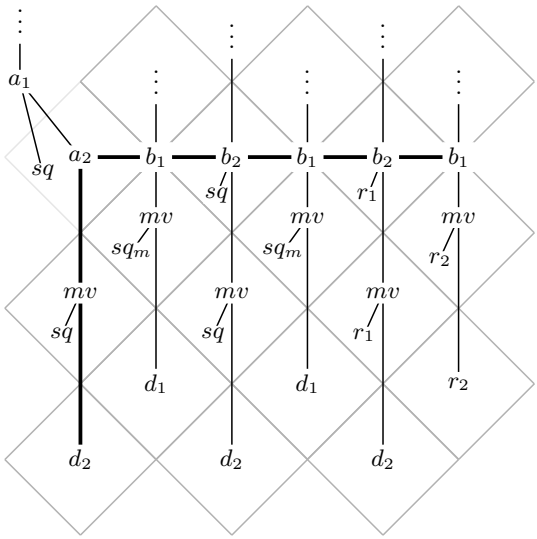

Fig. 4. A sketch of a tree over collage operations that generates the lower right section of the key pattern in Figure 2

The operations work as follows. For all collages C_1, C_2, C_3, C_4:

- $mv(C_1, C_2)$ puts C_1 at the current position (meaning the affine transformation applied to C_1 is the identity), and translates C_2 downward for the diagonal length of a tile;
- $b_1(C_2, C_3, C_4)$ moves C_2 half this length downward, C_3 half the length to the right, and C_4 half the length downward followed by a rotation of 180° about its current position;
- $b_2(C_1, C_2, C_3, C_4)$ behaves as b_1, but has in addition collage C_1 which is put at the current position;
- $a_2(C_1, C_2)$ moves C_1 one diagonal length downward and C_2 half the length to the right; and
- $a_1(C_1, C_2, C_3)$ puts C_1 and C_2 both at the current position and rotates C_3 by 180° about that position.

Now these operations have to be organised in order to yield well-formed key patterns. Let us first take a look at a successful generation process, like the one shown in Figure 5, where pictorial representations of terms are given for easier understanding. Starting from some initial item, the idea is to specify the width of the pattern before the height.

In order to get uniform growth both horizontally and vertically, some regulation mechanism has to be employed. For this, we use a regular tree grammar producing unary terms over the symbols w_1, w_2, h_1 of rank 1 and h_2, wh_2 of rank 0 such that the terms without parentheses belong to the string language of the regular expression $w_1^*(w_2h_1h_1^*h_2|wh_2)$. Symbols w_i specify the width and symbols h_j the height of the pattern, symbols x_1 (i.e., symbols with the index 1) the

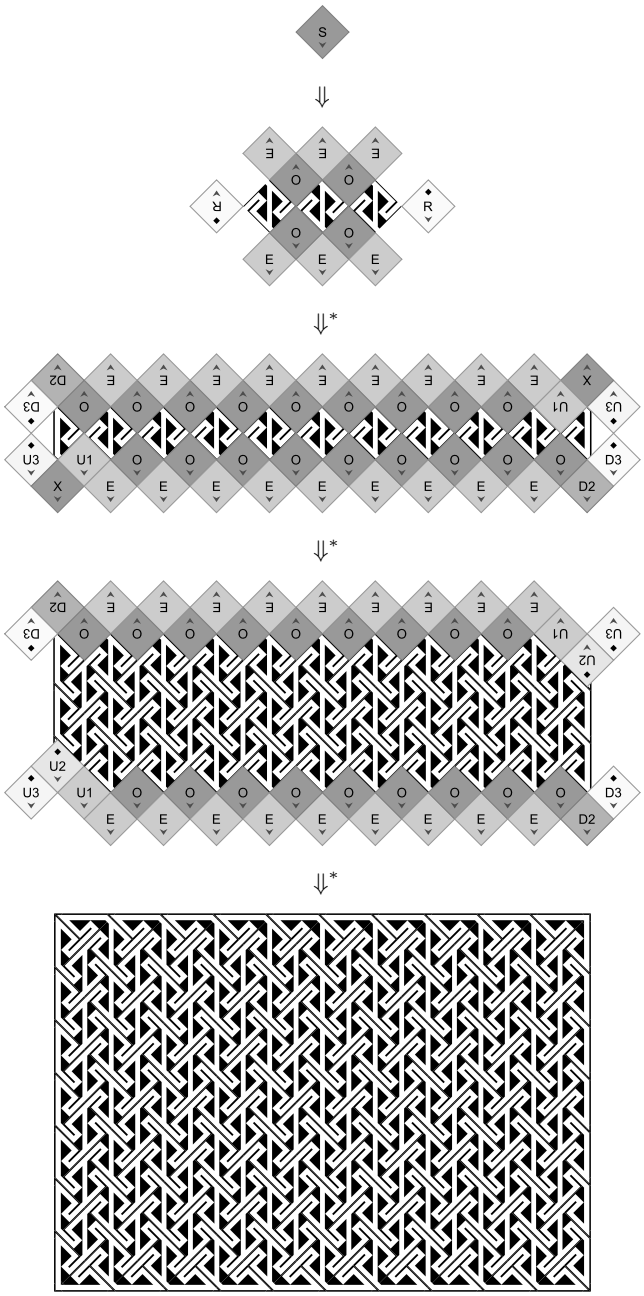

Fig. 5. Typical derivation sequence for a key pattern

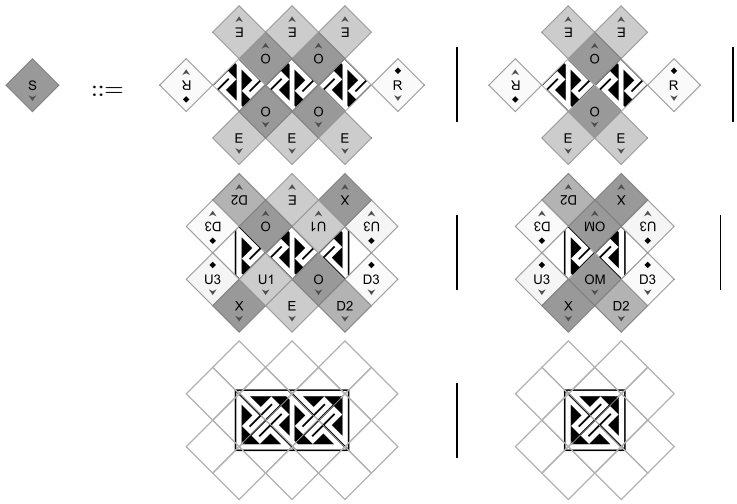

Fig. 6. Transducer rules to start a computation from the initial state

internal growth and symbols y_2 the border construction, with symbol wh_2 for the special case that no vertical growth is to take place. These control terms are then used as input for a top-down tree transducer that consumes in each step the first symbol of the term and copies the rest of the term identically to all newly introduced states. The rules of the transducer are shown in Figures 6–10, in the same pictorial representation as above, and with grey squares representing the named transducer states.

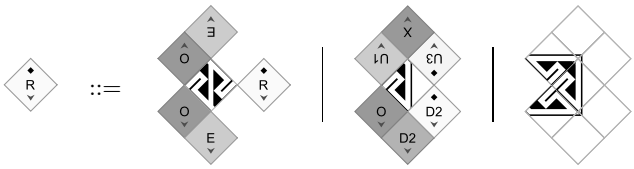

Fig. 7. Transducer rules to grow a design horizontally

– State S is the initial state and may encounter symbols w_1, w_2, wh_2 (Figure 6). As the centre of a key pattern may either lie inside a tile sq or inbetween two such tiles, there are two options for processing each of the three symbols, making the transducer nondeterministic. The right-hand-sides in the first row of Figure 6 are created when w_1 is consumed by S. The next two right-hand-sides result when state S encounters immediately symbol w_2, i.e., when no horizontal growth takes place. In this case, state OM is introduced to work analogously to state O by producing tiles sq_m, but producing tile d'_1 instead

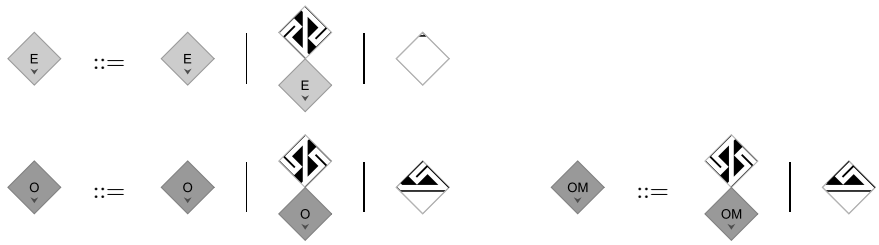

Fig. 8. Transducer rules to grow a design vertically

of d_1 for the top and bottom border (Figure 8 on the right). We leave it to the interested reader to find out why nevertheless the key pattern shown in Figure 1 cannot be generated by the resulting set of transducer rules, and how to remedy the problem.

- State R executes horizontal growth and may encounter symbols w_1, w_2, wh_2 (Figure 7). The first right-hand-side is the result of processing w_1, the second results from consuming w_2, and the last from consuming wh_2, respectively.
- State E ignores symbols w_i (i.e., it consumes them and proceeds to their subtree without any further action) and produces the even-numbered vertical lines using tile sq; analogously, state O produces the odd-numbered vertical lines using tile sq_m. Finally, state OM behaves nearly the same as O, with the difference that it places tile d_1' instead of d_1 for the border (Figure 8).
- States D2 and D3 encounter only symbols h_i, from which they grow the lower half of the right border (and the upper half of the left border).

Fig. 9. Transducer rules to grow the vertical borders

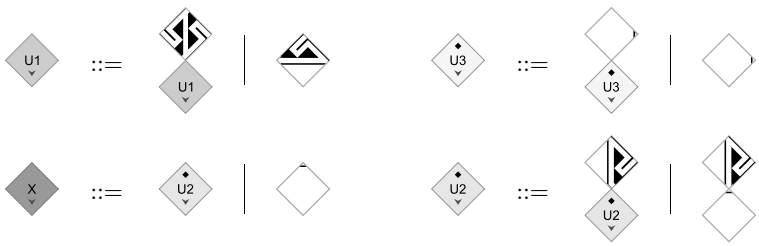

Fig. 10. Transducer rules to produce the second type of corner

– The upper half of the right border (and the lower half of the left border) is grown through states U1, U2 and U3 from symbols h_i (Figure 10). These states produce the alternative corner with tiles d'_1 and d'_2 as follows: As may be seen in Figure 5, state U2 is one step behind the other two states, with the retardation provided by state X on the first occurrence of symbol h_1. This is because U2 has to produce (a rotation of) tile d_1 while reading symbol h_1, but on encountering h_2, i.e., for the corner, it has to be tile d'_1 instead, which is suitably combined with tile d'_2.

Reconsidering the complete model, one may ask why one should start growing key patterns from their centre, and not rather from the centre of one of their sides, or indeed a corner. These are, in fact, viable and sometimes even preferable alternatives. The question will be discussed with the variations of key patterns studied in the next section.

4 Variations of Celtic key patterns

In Celtic key patterns, both the basic tiles and their structural arrangements offer many possiblities for distinctive designs. A small collection of alternatives is presented in this section.

The considerations of the preceding section started with identifying a basic triangle tile in a complete key pattern. In the original Celtic artwork, many variations of this first, very typical, triangle may be found. A small collection of triangles is shown in the first row of Figure 11. Note that in all triangles, the

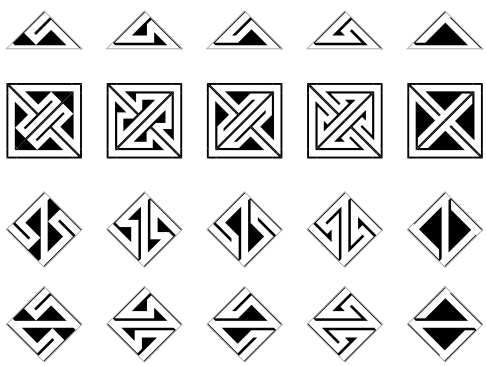

Fig. 11. Various basic triangles, smallest corresponding key patterns, and derived square tiles for larger key patterns

path enters at the same position close to the right angle, continues along the hypotenuse with half the original width, and leaves again at the acute angle opposite of the entrance. Any triangle complying with these geometric constraints can be used instead of the basic triangle of the previous section to produce a

well-formed pattern with continuous paths. The smallest such patterns, consisting entirely of border triangles, are shown in the second row of the figure. In order to grow larger patterns with the method of the preceding section, one needs to compose a triangle with its 180° rotation about the centre of the hypotenuse to form a square tile. The derived square tiles are shown in the last two rows of Figure 11: The third row contains the squares with vertical centre path (as used in the preceding section), and the fourth row the squares with horizontal centre path, obtained from the upper squares by a 90° rotation. A small selection of key patterns using these squares is shown in Figure 12.

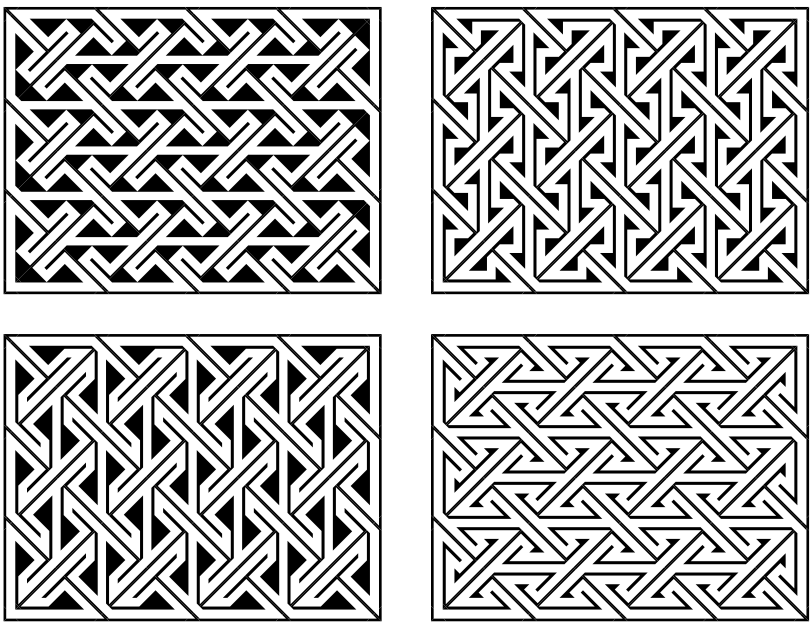

Fig. 12. Key patterns based uniformly on a triangle

The geometric constraints for triangles imply similar constraints for squares: A path enters at a fixed position on each side of a square, and the paths within a square are connected in some way. In Celtic art, many such squares have been used (see Figure 13 for a small sample). Since these squares cannot be divided satisfyingly into triangles, they have to be combined with border triangles so that full key patterns may be generated. Some samples are shown in Figure 14.

One can also combine more than one square type in a key pattern. To avoid disorderly arrangements, suitable syntactic rules may be employed. One such rule is to distinguish between constants sq and sq_m (rather than having sq_m as a reflection of sq) and thus admit interpretations by different squares; the key patterns in Figure 15 are of this kind. This rule may be refined by requiring

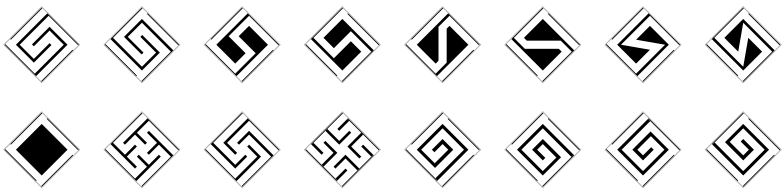

Fig. 13. Sample collection of square tiles

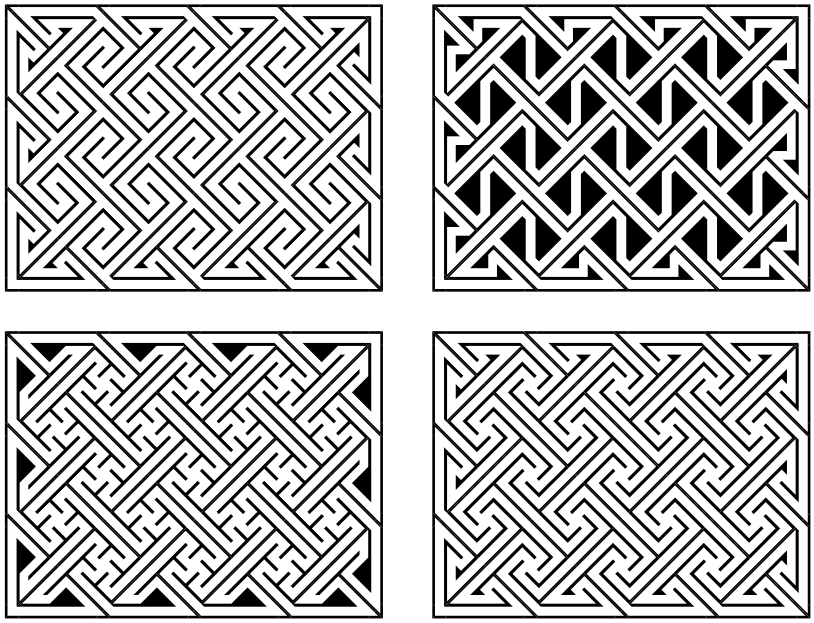

Fig. 14. Key patterns based on a square with a triangle border

that two distinct squares be used alternately to interpret sq, which leads to key patterns as shown in Figure 16.

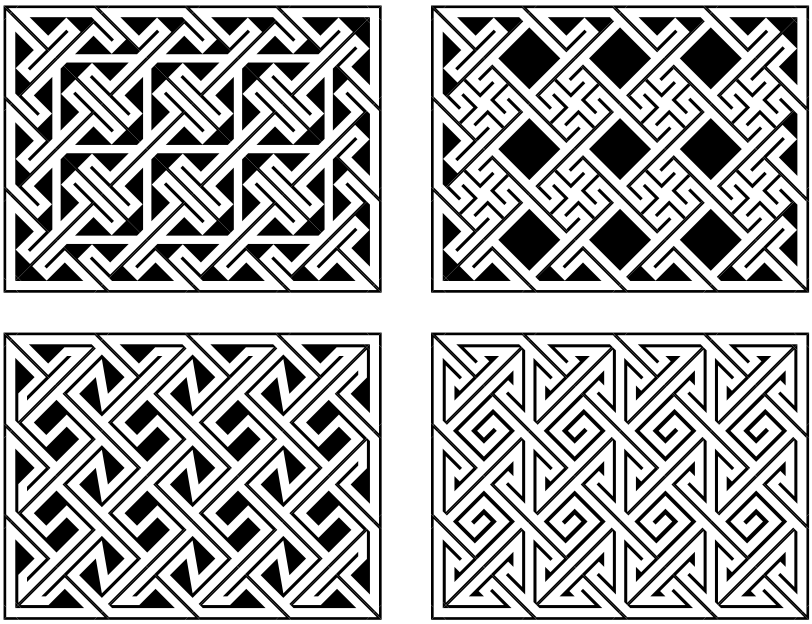

Fig. 15. Key patterns where sq and sq_m are interpreted differently

Further distribution rules for different squares in a key pattern include:

- Using different squares for the lower half and the upper half; this may be implemented in our model by doubling the states of the transducer so that the two halves may be treated independently, but to the same vertical extension as given by the input term.
- Using different squares for each row of squares, but having the choice for the upper half reflected in the lower half; this may be implemented in the input term of the transducer by replacing the symbols h_1 with references to the squares that shall be used.
- Finally, one may use branching synchronization to achieve a more general symmetric distribution of synchronized squares. The method used to generate breaklines within the rectangular Celtic knot designs in [Dre06] is well-suited for this. A nesting depth of 2 for the resulting branching collage grammar is sufficient to ensure that the overall design maintains horizontal and vertical symmetry even though blocks of different squares may alternate in a nondeterministic fashion.

Coming back to the question at the end of Section 3, the two rules above are good reasons to start the growth of a key pattern from the centre at least of the

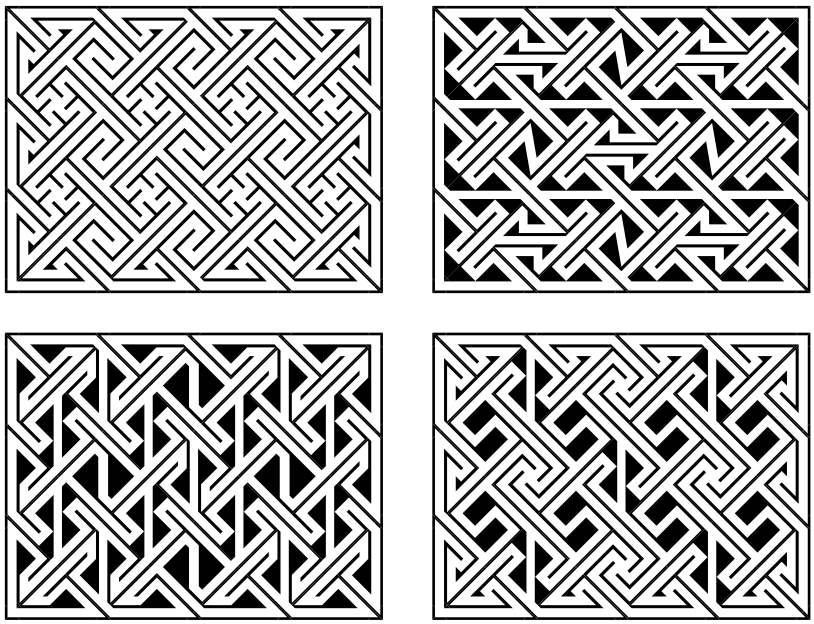

Fig. 16. Key patterns where *sq* is interpreted alternately by two distinct squares

vertical direction. A further reason lies in the last four tiles shown in Figure 13, i.e., the dead-end spiral, the only type of tile given in that figure that does not have (at least) 180° rotational symmetry. Our model admits rotation of this tile between the lower and the upper half of a key pattern; see, e.g., the last pattern in Figure 15. If, however, it is desired that all occurrences of this tile have the same orientation, then it is preferable to start vertical growth from the top (or the bottom) of the pattern.

5 Conclusion

In this paper we have shown how rectangular key patterns with interior and border variations can be modelled using tree-based collage grammars to interpret the generated terms. All considered key pattern variations are covered by a class of tree generators that combine a regular tree grammar for unary terms with a top-down tree transducer.

The model for rectangular knotwork as proposed in [DK00,Dre06] is also based on square tiles, but uses branching tree grammars to encode a horizontally and vertically symmetric distribution of so-called breaklines over a knot. Consequently, the syntactic structure of rectangular knotwork with breaklines may be assumed to be one level more complex than the structure of rectangular key patterns with regular distribution of tiles.

For both knotwork and key patterns, it is interesting to study how evenly placed holes of varying sizes and shapes can be added to the base pattern. These holes can either be left plain, or they can be filled with any type of Celtic tiling. The designs so produced are called carpet-page designs. We note that for an authentic look, the holes will have to be distributed in a symmetric fashion across the expanse of the base design.

While the basic shape of such a hole is rectangular, in Celtic art holes also come in L, cross or crosslet shapes. The boundary of the hole itself needs to be sealed off with specific border tiles. Of course, holes need not be restricted to the interior of the key pattern boundary, but may also lie directly on it, breaking up the rectangular border. If these border cutouts are regularly distributed as well, the resulting key pattern designs display a multitude of interesting shapes, of which the cross is the most basic.

In [DK00,Dre06], a way to include such holes is proposed for square knotwork panels that are grown diagonally from the centre to the corners and thus come with a natural vertical/horizontal synchronisation. It would be nice to have a generalised method that works for *rectangular* patterns (and therefore needs additional synchronisation), in any kind of tiling with defined border tiles.

In Celtic art, it is often customary to fill such holes with some contrasting decorative pattern. This presents a modelling problem, since whatever pattern is created may not grow beyond the boundaries of the hole, and collage grammars do not offer context-sensitive queries. No matter whether the new pattern is created by subdivision or growth, information about the shape and size of the hole is required in order to create a correct pattern with a border that seamlessly joins the boundary of its parent hole. Then, a formalism is required that can deal with the multitude of possible shapes and sizes and create matching patterns. Additionally, some synchronisation between the scale of the tiles in the base panel – which inform the dimensions of possible holes – and the scale of hole-filling tiles must take place.

The creation of round key pattern (or knotwork) panels with a circular tesselation pattern calls for another type of construction method altogether. Collage grammars as they are used here rely on local replacement, which cannot be used to recompute the scaling and placement operations necessary to evenly place tiles along a growing circle. A suitable construction method for circular tesselation patterns might also shed some light on how to generate Celtic spiral patterns. It may be interesting to note that in [Dre06], a method is suggested for generating a tiling of concentric rings of triangles, and from there spiral tilings, including the Frazer spiral. This which might work for circular Celtic patterns, as the basic idea is growing concentric rings of tiles outward from an inner circle which is subdivided in a fan-like arrangement of triangles. This process, however, cannot generate a truly circular outer border by subdivision.

In the illuminated pages of Celtic manuscripts, key patterns come with colour. Often, this just entails giving a different colour to the paths than to the background, which can be achieved just like having white paths on a black background. Just as often, however, a sophisticated colouring scheme is used based

on colour blocks in rectangular or lozenge shapes. How to add colour to key patterns in such a way is an open problem, though colour operations as presented in [Dre06] might allow simulating at least a simplified version of the original Celtic colour schemes.

Finally, there are two classes of Celtic key patterns for which we have not yet devised a collage grammar-based generation technique. The first class uses squares tiles that are obtained from basic triangles by reflection at the hypotenuse. Consequently, these tiles do not have rotational symmetry with respect to path entry points at their sides, so that they have to be arranged differently to form entire patterns. The second class uses hook-like squares such as the four last squares in Figure 13, but some of the path entries may be sealed off. The reason for this can be seen in Figure 17: There are many long straight black lines that do not finish off by properly meeting with other black lines at each end. Of course, lengthening these lines at their ends requires the two neighbouring tiles which form the line end to agree. Moreover, the orientation of the tiles in hook patterns is not so uniform as in the key patterns considered in this paper. Next steps for future work may include writing collage grammars for these classes of key patterns, too.

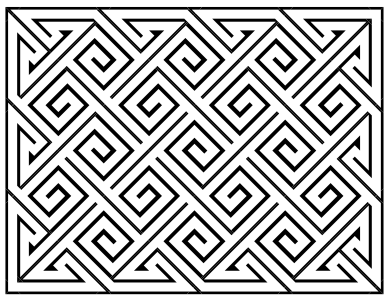

Fig. 17. Pseudo hook pattern

References

[All93] J. Romilly Allen. *Celtic Art in Pagan and Christian Times*. Bracken Books, London, 1993.

[Bai51] George Bain. *Celtic Art. The Methods of Construction*. Constable, London, 1951.

[Bai90] Iain Bain. *Celtic Knotwork*. Constable, London, 1990.

[Bai93] Iain Bain. *Celtic Key Patterns*. Constable, London, 1993.

[DEKK03] Frank Drewes, Sigrid Ewert, Renate Klempien-Hinrichs, and Hans-Jörg Kreowski. Computing raster images from grid picture grammars. *Journal of Automata, Languages and Combinatorics*, 8(3):499–519, 2003.

[DK99] Frank Drewes and Hans-Jörg Kreowski. Picture generation by collage grammars. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages, and Tools*, chapter 11, pages 397–457. World Scientific, 1999.

[DK00] Frank Drewes and Renate Klempien-Hinrichs. Picking knots from trees – the syntactic structure of celtic knotwork. In *Proc. 1st Intl. Conference on Theory and Application of Diagrams 2000*, volume 1889 of *Lecture Notes in Artificial Intelligence*, pages 89–104. Springer, 2000.

[DKK03] Frank Drewes, Renate Klempien-Hinrichs, and Hans-Jörg Kreowski. Table-driven and context-sensitive collage languages. *Journal of Automata, Languages and Combinatorics*, 8(1):5–24, 2003.

[DKL03] Frank Drewes, Hans-Jörg Kreowski, and Denis Lapoire. Criteria to disprove context freeness of collage languages. *Theoretical Computer Science*, 290:1445–1458, 2003.

[Dre00] Frank Drewes. Tree-based picture generation. *Theoretical Computer Science*, 246:1–51, 2000.

[Dre06] Frank Drewes. *Grammatical Picture Generation – A Tree-Based Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[HK91] Annegret Habel and Hans-Jörg Kreowski. Collage grammars. In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proc. 4th Intl. Workshop on Graph Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 411–429, 1991.

[HKT93] Annegret Habel, Hans-Jörg Kreowski, and Stefan Taubenberger. Collages and patterns generated by hyperedge replacement. *Languages of Design*, 1:125–145, 1993.

[KRS97] Lila Kari, Grzegorz Rozenberg, and Arto Salomaa. L systems. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages. Vol. I: Word, Language, Grammar*, chapter 5, pages 253–328. Springer, 1997.

[Mee02] Aidan Meehan. *Maze Patterns.* Thames & Hudson, London, 2002.

[Slo95] Andy Sloss. *How to Draw Celtic Knotwork: A Practical Handbook.* Blandford Press, 1995.

[Slo97] Andy Sloss. *How to Draw Celtic Key Patterns: A Practical Handbook.* Blandford Press, 1997.

Dr. Renate Klempien-Hinrichs

Fachbereich 3 – Informatik
Universität Bremen
D-28334 Bremen (Germany)
rena@informatik.uni-bremen.de
http://www.informatik.uni-bremen.de/~rena

Renate Klempien-Hinrichs was a member of Hans-Jörg Kreowski's team from 1993 to 2009. In 2000, she received her doctoral degree from the University of Bremen under Hans-Jörg's supervision.

Caroline von Totth

Fachbereich 3 – Informatik
Universität Bremen
D-28334 Bremen (Germany)
caro@informatik.uni-bremen.de
http://www.informatik.uni-bremen.de/~caro

Hans-Jörg Kreowski supervised Caroline von Totth's diploma thesis. Since 2004, she is a member of his group, working on her doctoral thesis, again under his supervision.