EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Design Transformations for Rule-based Procedural Modeling — Supplemental Material

Stefan Lienhard[1] Cheryl Lau[2] Pascal Müller[2] Peter Wonka[3] Mark Pauly[1]

[1]EPFL [2]Esri R&D Center Zurich [3]KAUST

Detailed Result Descriptions

In the supplemental material we provide detailed descriptions of the grammars, the semantic matches, their parameters, and the entire co-derivation process for two of our results. We first provide some insights into the transformations between Semper's Sternwarte and the white modern building via two different interpolation paths (Figs. 13 and 14 in the paper). Second, we explain how we transformed the two tree L-systems in the teaser image (Fig. 1 in the paper). The corresponding grammar files are included in the `grammars` folder. We use the notation `@tag` to annotate rules that we consider for semantic matches (Sec. 4.1 in the paper). For each possible pair of two rules (one from each grammar) with the same tag, a semantic match is automatically established.

1. Sternwarte Chain Part 1

Mass Models The first co-derivation step derives the start shape with the axiom rules of both grammars. It passes through some intermediate (untagged) rules before it comes to a halt after all mass model shapes have been instantiated. Their rules are tagged with `@mass_lvl0`, `@mass_lvl1`, or `@mass_lvl2` for shapes on the ground plane and shapes at two higher stacking levels. The tags essentially define a synchronization barrier that allows to apply a match & blend step (Sec. 4.3 in the paper) to the mass models before the co-derivation continues. Fig. 1 shows the result of the shape matching.

Afterwards, in the second co-derivation iteration, the rules split

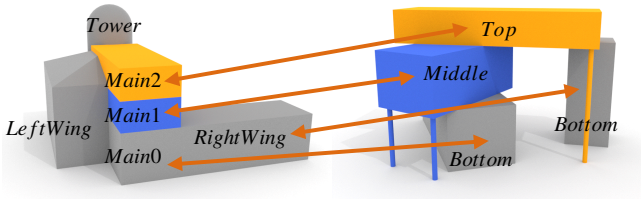

Figure 1: ***Shape Matching*** *The result of the Munkres matching for the first part of the Sternwarte Chain. Only shapes with the same colors can be matched. Gray shapes are tagged as `@mass_lvl0`, blue shapes as `@mass_lvl1`, and orange shapes as `@mass_lvl2`. The unmatched shapes LeftWing and Tower will disappear during the transformation while the other shapes are blended with their matches.*

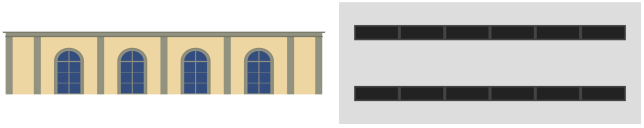

Figure 2: ***Sternwarte and White Modern Building Façades*** *The two façade styles for the upper mass models of both building designs.*

the mass models into their faces for façades and roof tops, and they also generate railings on the Sternwarte roofs and support columns for the white modern building. The component splits of both grammars always produce the same exact shapes. For them, the interpolation parameters have no influence on the outcome of the co-derivation steps since the matched and blended shapes are identical. But the railings and columns are only defined in one of the two grammars, and if these elements are present or not depends on the thresholds of the semantic matches that control the co-derivations ($\langle Main2, Top\rangle$ and $\langle RightWing, Bottom\rangle$). The thresholds are visible in Figs. 3 and 4.

Façade Transformations The next co-derivation steps of interest are the split transformations of the façades. We hierarchically apply the technique presented in Sec. 4.4 of the paper. As example, we describe the transformation of the façades shown in Fig. 2. They are the façades of the long sides of the blue shapes in Fig. 1. The automatically computed edit sequence for that $\langle YellowFacade, Facade\rangle$ co-derivation is:

$\{ Floor_1, Ledge \}$	delete all $Ledge$
$\{ Floor_1 \}$	semantic switch $Floor_1 \rightarrow Floor_2$
$\{ Floor_2 \}$	insert $Floor_2$
$\{ Floor_2, Floor_2 \},$	

where we use the subscripts to distinguish symbols that occur in both grammars. The edit sequence for the next lower level, controlled by the semantic match $\langle Floor, Floor\rangle$, is given in the following. The notation is: P for $Pillar$, Yt for $YellowWinTile$, and Wt for $WindowTile$.

$\{ P, P, Yt, P, Yt, P, Yt, P, Yt P, P\}$	delete all P
$\{ Yt, Yt, Yt, Yt\}$	semantic switch $Yt \rightarrow Wt$
$\{ Wt, Wt, Wt, Wt\}$	insert Wt
$\{ Wt, Wt, Wt, Wt, Wt\}$	insert Wt
$\{ Wt, Wt, Wt, Wt, Wt, Wt\}$	

Window tiles are transformed as vertical splits. Their strings only consist of one single *Window* shape (after removing the *Wall*s) and the edit sequence is a single semantic switch. Even though the edit sequence is very simple, the split transformation still provides a nice effect because the size interpolation smoothly shrinks the window over time.

Parameters In Figs. 3 and 4 we list all semantic matches together with their parameters for both interpolation paths of the transformation between the Sternwarte and the white modern building. For some semantic matches, some of the parameters are unused or ignored and we do not show them in the figures. Possible reasons for that can be:

- If a shape is not the result of a blending operation and if the shape does not initiate a split transformation, t_s and t_e are not used.
- If no discrete decisions need to be made, t_{th} is not needed.
- Split transformations do not need thresholds either.
- Parameters are irrelevant when both grammars generate identical output in a co-derivation step.
- The parameters do not matter if a semantic match is never applied.

2. Tree L-Systems

The second example transforms between the two L-systems by Honda and by Aono and Kunii. We ported the L-systems to our rule-based modeling framework. While it is the example with the lowest number of rules and with only a few semantic matches, understanding it is not trivial due to its recursive nature.

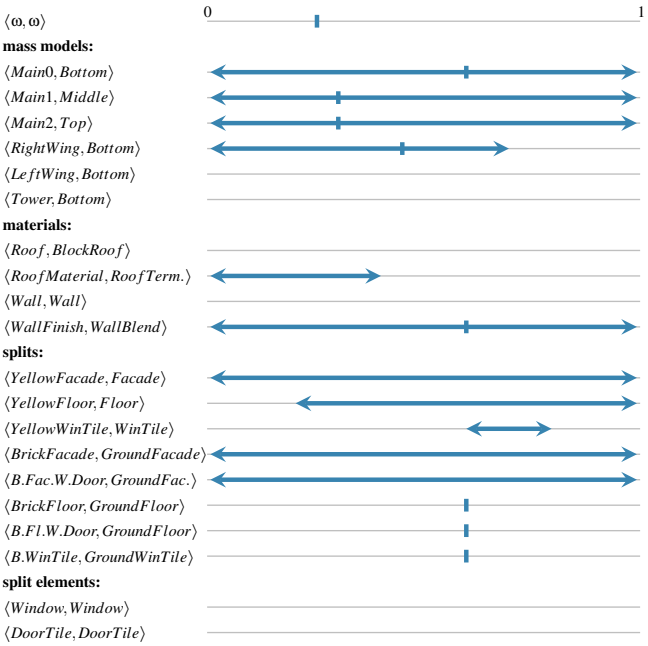

Figure 3: ***Parameters for Sternwarte Chain Part 1*** *Above are the settings that we use to transform the Sternwarte into the white modern building. The threshold of the axiom semantic match decides when the entire LeftWing and the Tower disappear, which happens early on. The last three semantic matches for splits perform their transformations over infinitely small time spans. This effectively results in discrete rule switches, which we represent as thresholds.*

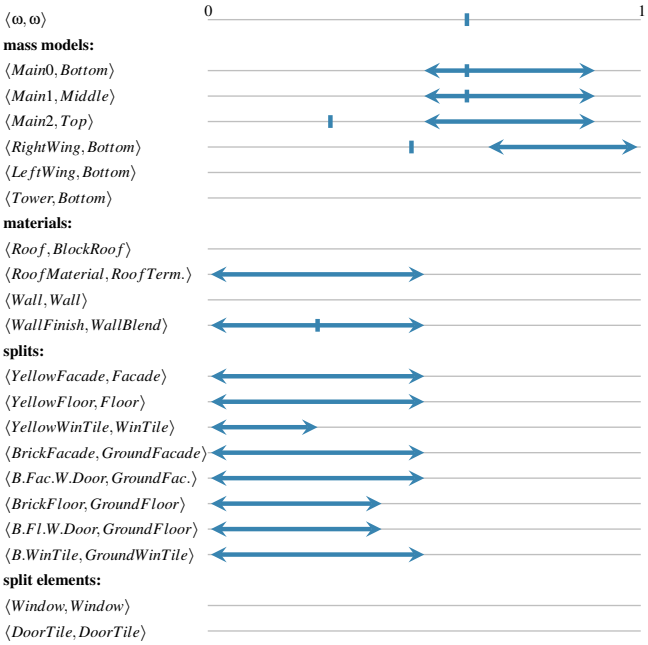

Figure 4: ***Alternative Parameters for Sternwarte Chain Part 1*** *In this version all façade and roof shapes transform before any of the mass models change. This is also evident from the above list of parameters: all split and material transformations are executed over the first half of the sequence, and everything else follows in the second half. At the half-way mark, the resulting design is the Semper Sternwarte completely covered by the façade style of the white modern building. The two thresholds that are not within their corresponding $[t_s, t_e]$ range are responsible removing the railings on top of Main2 and RightWing.*

Grammar Descriptions Executing the first grammar yields a monopodial tree (Fig. 5 top). It has three different branching rules, *A*, *B*, and *C*. They each generate a straight and a lateral segment. Rule *A* is responsible for the tree's vertical growth towards the sky, while rules *B* and *C* generate branches growing to the sides. The only difference between *B* and *C* is that their lateral segments lie on opposite sides of the straight segment. The second grammar grows a sympodial tree (Fig. 5 bottom) that is based on only two branching rules, *A* and *B*. Rule *A* is only applied once at the beginning for the trunk, all remaining branchings are handled by *B*.

Both grammars also use a rule *F* that instantiates a cylinder mesh to represent a segment's geometry (*F* originally stood for *forward*, telling the turtle that executes the L-system to advance). All remaining rules are responsible for gradually diminishing the tree's thickness or they implement the exit conditions for the recursive execution of the L-systems.

We only use two different tags for the symbols: `@branching` for *A*, *B*, *C* in the first grammar and for *A*, *B* in the second grammar, and `@segment` for *F* in both grammars. Putting the `@branching` tag on all branching rules means that we consider them all as

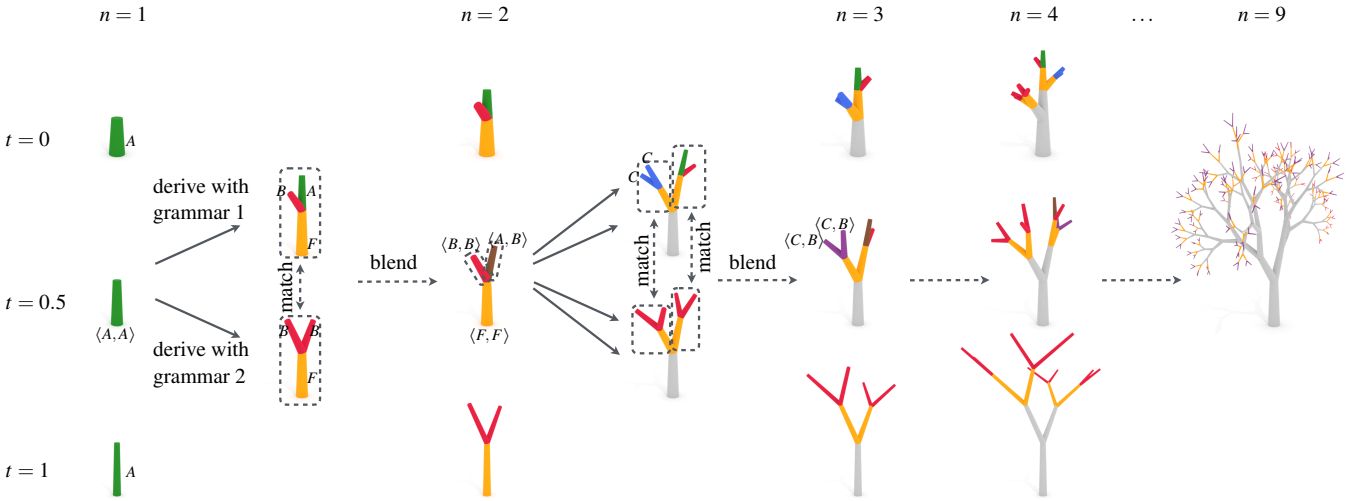

Figure 5: ***Tree L-Systems Transformation*** *The first four co-derivation iterations of both L-systems are shown for times $t = 0$ (top), $t = 0.5$ (middle), and $t = 1.0$ (bottom). The top and bottom rows correspond to the output of the original grammars. We color code the different shapes according to their symbols: As are green, Bs are red, Cs are blue, and Fs are orange. Shapes generated in previous iterations are colored gray. Since blended shapes interpolate their colors we also see brown (for $\langle A,B\rangle$ blends) and purple (for $\langle C,B\rangle$ blends). Between the first and second iterations, we show how the shape matching and blending step works for the semantic match $\langle A,A\rangle$ at $t = 0.5$. The same is shown for semantic matches $\langle A,B\rangle$ and $\langle B,B\rangle$ between iterations two and three.*

interchangeable. The tags results in the six semantic matches listed in Fig. 6.

In the following, two simplified grammars show the overall structures of both L-systems. The ellipses stand for affine transformations that are of no relevance for this explanation. Untagged in-between rules have been removed.

$$\omega \to \ldots A \qquad \omega \to \ldots A$$
$$A \to F \ldots [\ldots B] \ldots A \qquad A \to F \ldots [\ldots B] \ldots B$$
$$B \to F \ldots [\ldots C] \ldots C \qquad B \to F \ldots [\ldots B] \ldots B$$
$$C \to F \ldots [\ldots B] \ldots B$$

Co-Derivation Steps A few iterations of the growth structure encoded by these rules are shown in the top and bottom rows of Fig. 5. The interesting parts of the entire co-derivation happen in the co-derivation steps of our interchangeable branching rules and we explain it by means of the semantic match $\langle A,A\rangle$. The three parts of that co-derivation step are:

1. The given shape is derived with the A rules of both grammars. Both generate three successor shapes each, one F shape at the attachment point that leads to two new branching shapes.
2. The subtree collapsing step can be skipped since the simplified grammars do not generate any shapes without semantic matches.
3. For the shape matching and blending, we choose the Munkres strategy. In this case it works on two sets of three shapes each. The F shape from the first grammar can only be matched with the F shape from the second grammar. The remaining shapes are all compatible for matching. All shapes are matched and therefore all of them are blended, resulting in three shapes labeled $\langle F,F\rangle$, $\langle B,B\rangle$, and $\langle A,B\rangle$. The blending interpolates branching angles, segment lengths and radii, and color. This exact matching and blending is visualized between the first two iterations in the middle row of Fig. 5.

Co-derivation steps for the other three semantic matches for branchings ($\langle A,B\rangle$, $\langle B,B\rangle$, and $\langle C,B\rangle$) are analogous since all branching rules always generate three similar successor shapes. Examples for these are present in columns three and four of the figure.

Parameters Note that the semantic matches $\langle B,A\rangle$ and $\langle C,A\rangle$ do not appear in the user interface (Fig. 6) even though the tags automatically establish these correspondences. This is because such a situation can never occur. Rule A in the second grammar is only used in a co-derivation step once. That one occurrence is controlled by the $\langle A,A\rangle$ semantic match. The user interface further shows that all parameters are set to their default values for this tree L-system transformation.

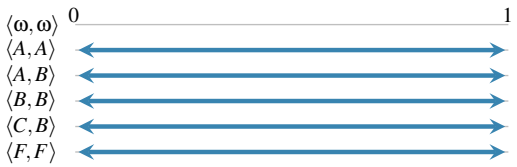

Figure 6: ***Transformation Parameters for Tree L-Systems*** *All the parameters of all semantic matches are set to their default values. Start times are set to 0 and end times to 1. No thresholds are shown since no discrete choices have to be made for this transformation: both designs do not use textures, the branch meshes are identical in both grammars, and the co-derivation steps never end up with any unmatched shapes.*