PAUL MERRELL

Supplementary Report: Example-Based Procedural Modeling Using Graph Grammars

Author's address: Paul Merrell, paul@merrells.org.

1 REPORT INTRODUCTION

This is a supplementary report for "Example-Based Procedural Modeling Using Graph Grammars". It shows additional results and a more detailed analysis. We show additional results related to Sections 5.4 and 5.5 of the main paper. We show the graph grammars that were used in Figure 9. We show how our graph grammars deconstruct several complex shapes.

Some of our graph grammars are quite large and have a lot of rules. Note that even large grammars can be computed quickly. Most can be computed in a fraction of a second with the longest one taking 2.4 seconds.

Additional information can be found on our website: https://paulmerrell.org/grammar/

2 REDUCING A GRAPH WITH ITS DESCENDANTS AND STUBS

2.1 Diagonal Example

In Section 5.4 of the main paper, we explain how we can reduce a graph using its descendants. We explain that this method is useful, but not strictly necessary. Usually our algorithm can generate a graph grammar equally capable of producing the same graphs without using any of the tools described in Section 5.4. However, such graph grammars are more complicated because they contain more rules. In the diagonal example in Section 5.4, the graph grammar generated with these tools consists of 6 rules. Otherwise, it requires 21 rules:

Fig. 1. The Example Shape

Fig. 2. The Graph Grammar Without Using the Tools of Section 5.4 (21 Rules)

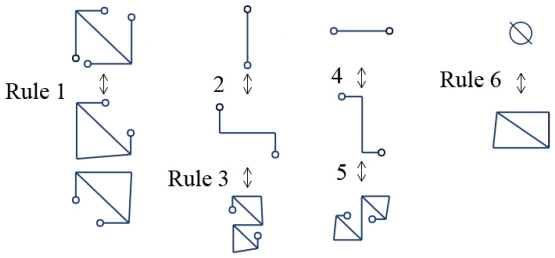
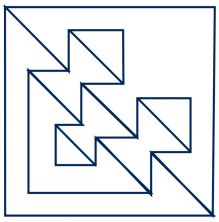
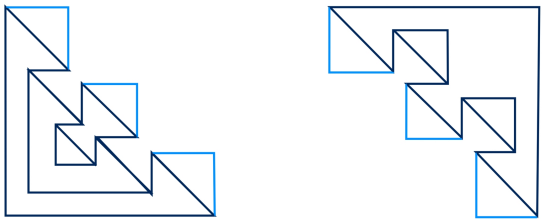


Fig. 3. The Graph Grammar Using the Tools of Section 5.4 (6 Rules)

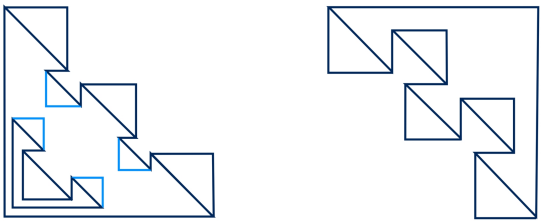
When the graph grammar is simple, we can quickly walk through all the steps necessary to deconstruct a complex shape. Note that constructing the shape is the same process in reverse.



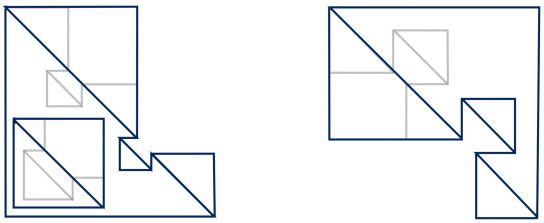
(a) Initial Shape



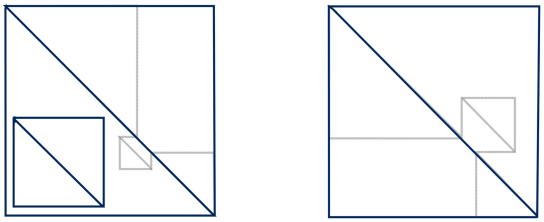
(b) Apply Rule 1



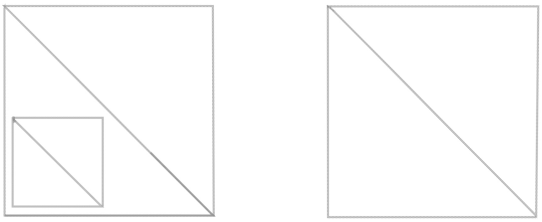
(c) Apply Rule 1



(d) Apply Rule 3



(e) Apply Rule 3



(f) Apply Rule 6

2.2 Stubs

The same thing can be said for the example in Section 5.5. The stubs are very useful, but not strictly necessary. Again, our algorithm can generate an equally powerful graph grammar without using any of the tools mentioned in Sections 5.4 or 5.5, but without them the graph grammar is much larger. Without using these tools, the graph grammar contains 314 rules. (See next page.) By introducing the concept of stubs, we can solve the problem with only 10 rules.

Fig. 5. The Example Shape

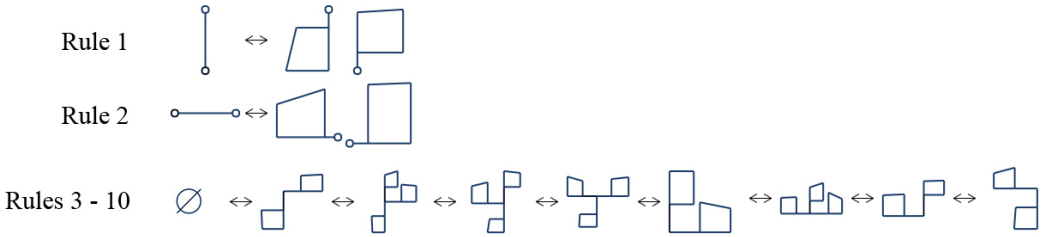
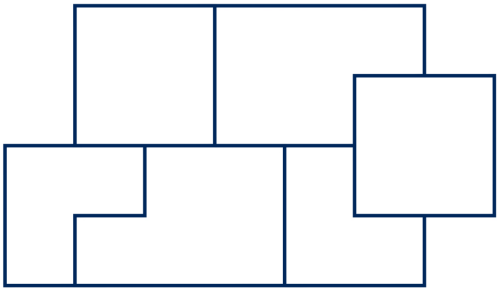
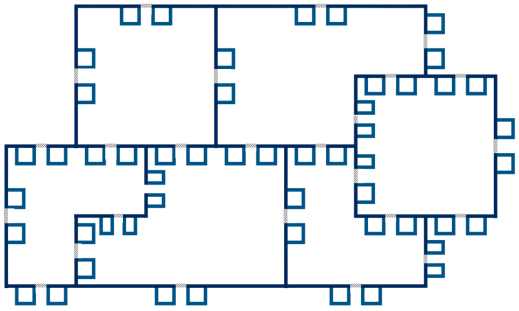


Fig. 6. The Graph Grammar Using the Tools of Section 5.4 - 5.5 (10 Rules)

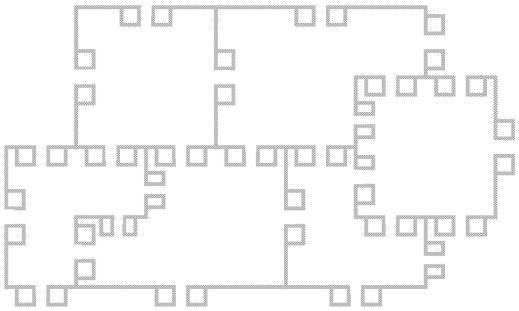
With the simple graph grammar it is not difficult to walk through all the steps of deconstructing a complex shape:



(a) Initial Shape



(b) Apply Rules 1 and 2



(c) Apply Rules 3 to 10

Fig. 8. The Graph Grammar Without Using the Tools of Section 5.4 - 5.5 (314 Rules). Only one page is displayed. The grammar would continue for three more pages.

3 GRAPH GRAMMARS

In this section, we go through several examples in more detail. We show the input shape, the automatically generated graph grammar, and several output shapes.

3.1 Castle

Figures 9 - 11 below show the input, output shape and graph grammar for the Castle Example (Figure 1 in the main paper). We also show how we can use our method to optimize for particular properties. As discussed in Section 6.3, we use the Metropolis-Hastings Algorithm to optimize for various properties. We can guide the output to produce shapes that have more of a certain type of edge or vertex. Figure 11 shows the result when we optimize for producing more bridges, towers, or buildings.

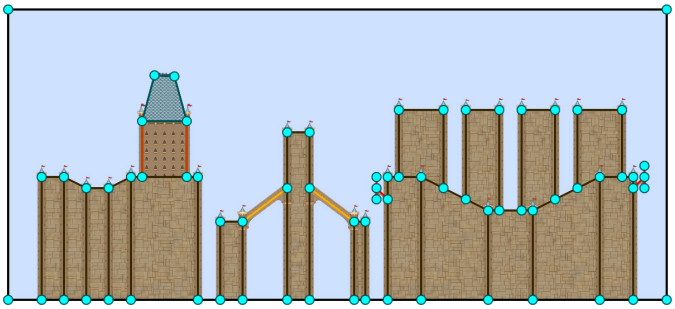


Fig. 9. Castle Example Shape

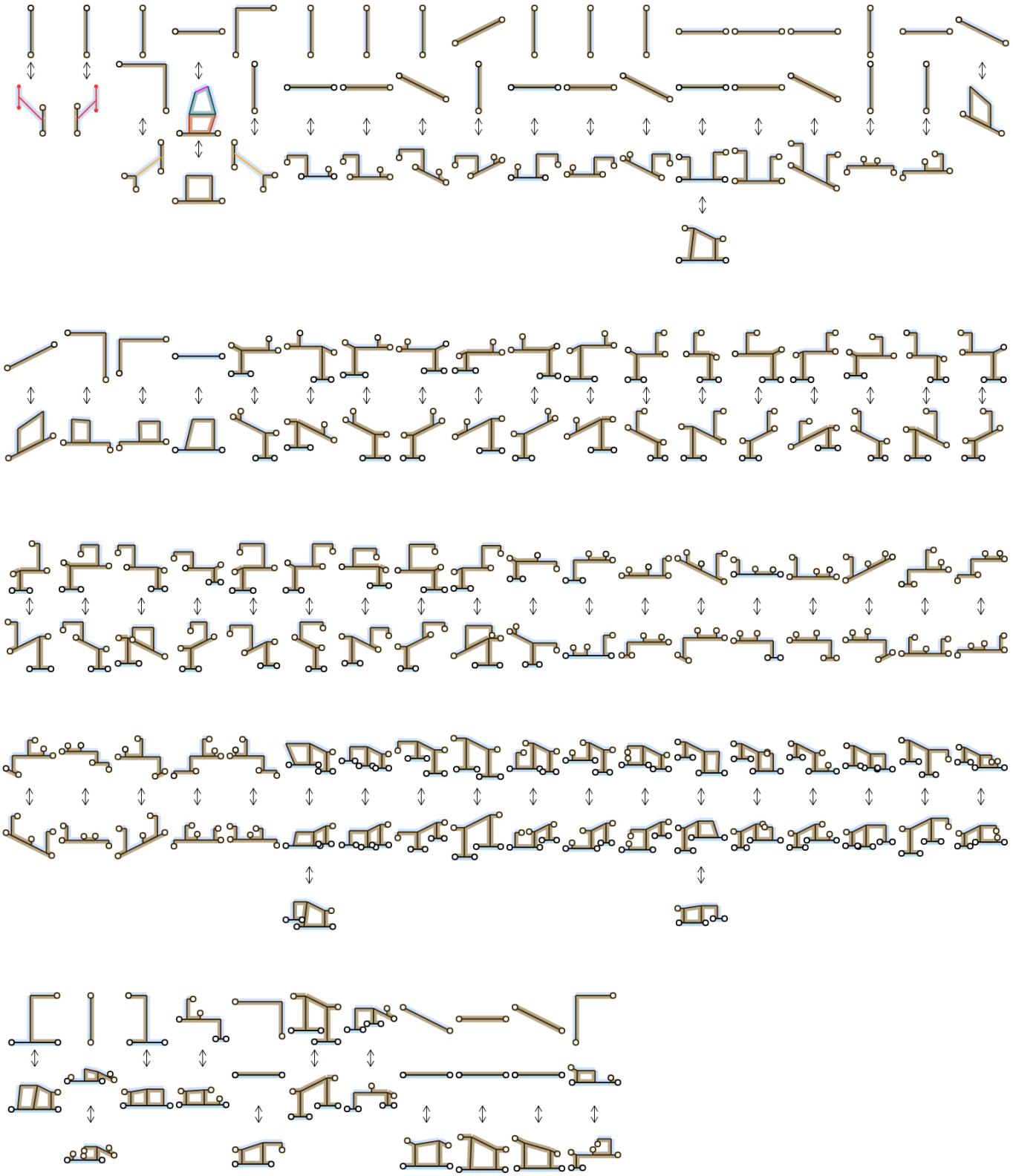


Fig. 10. Castle Graph Grammar (88 Rules)

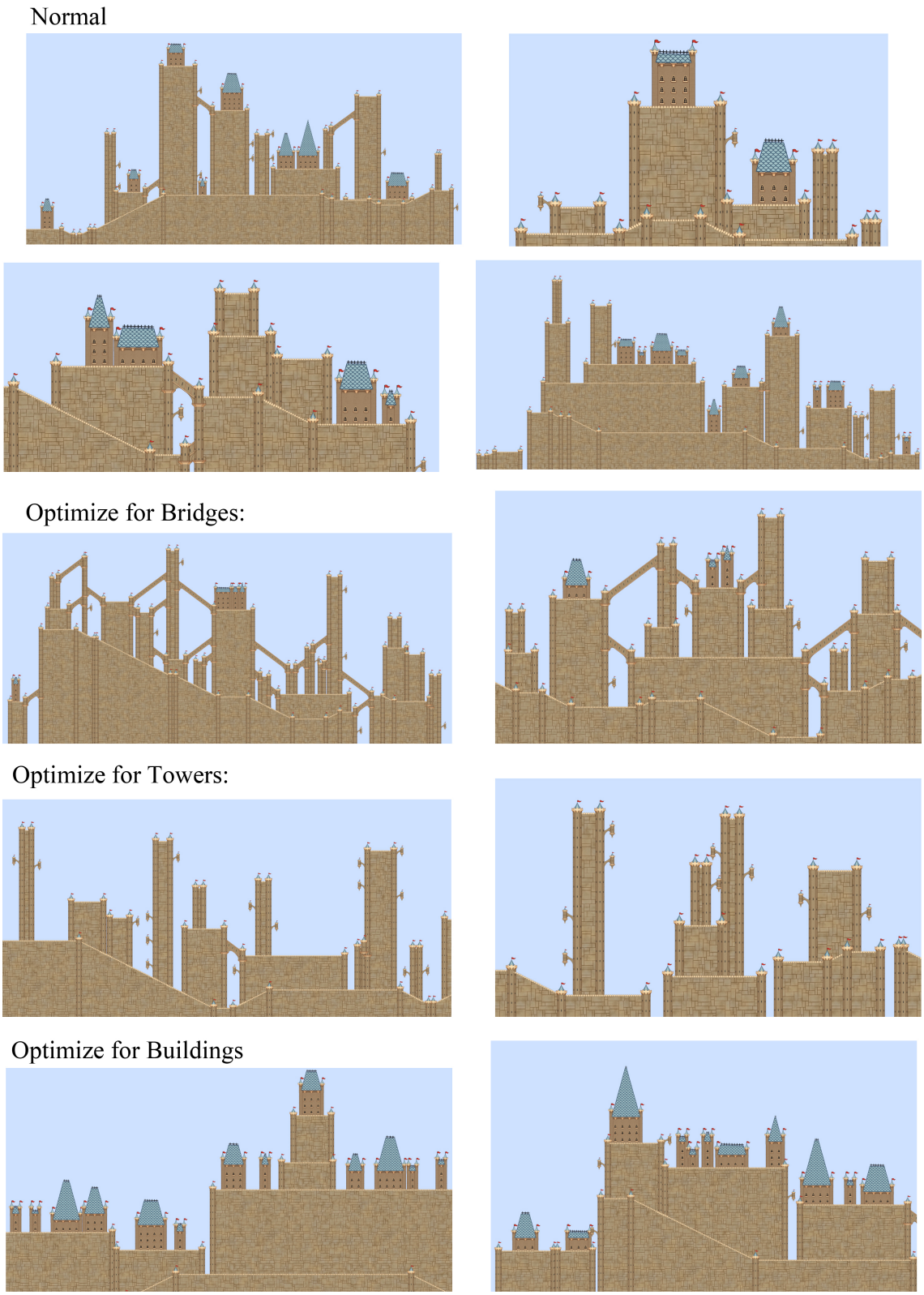


Fig. 11. Castle Output Shapes

3.2 Cave

This corresponds to Figure 9a in the main paper.

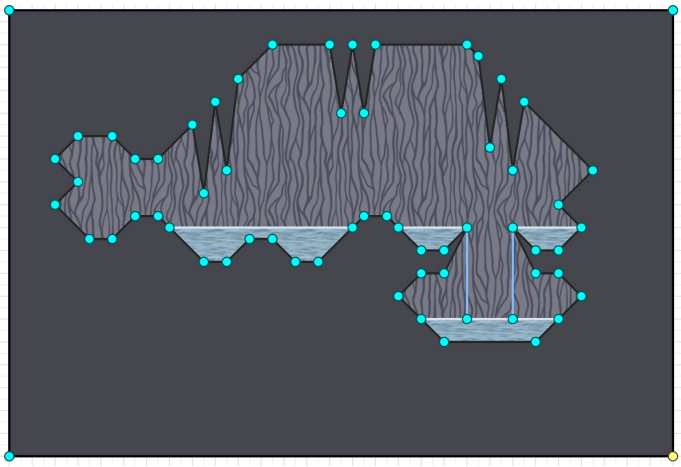


Fig. 12. Cave Example Shape

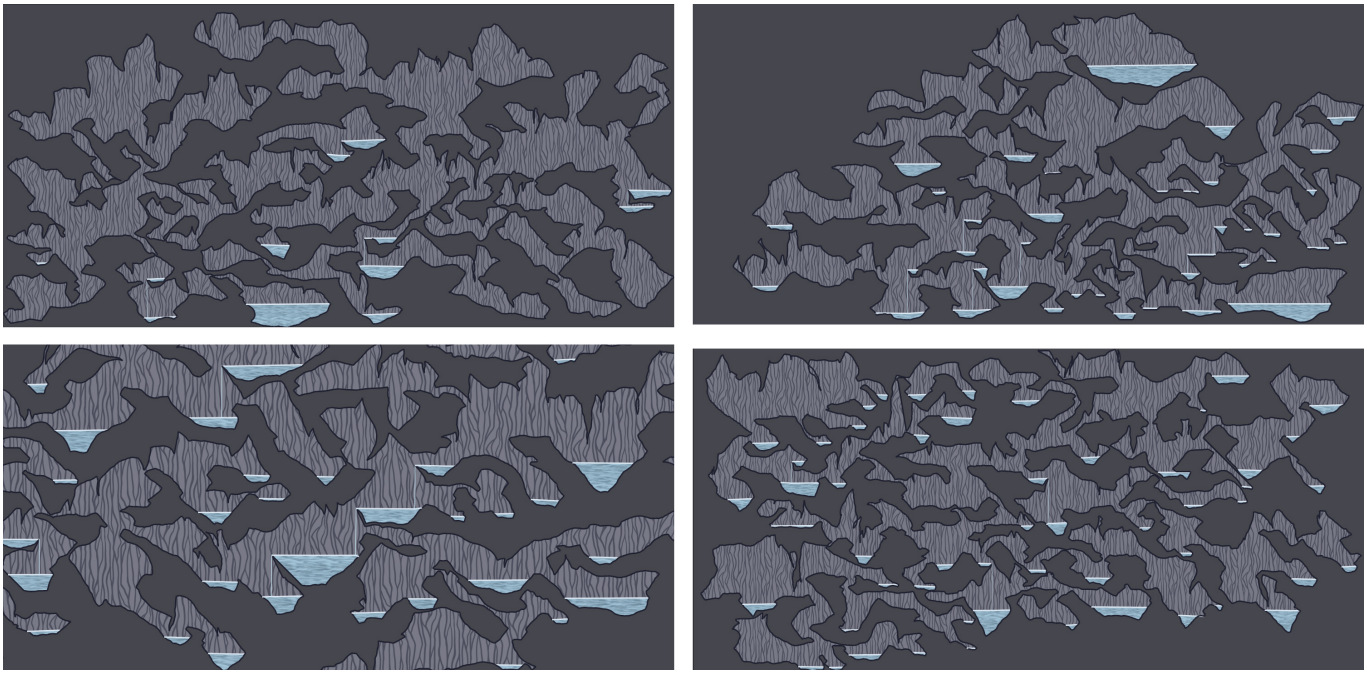


Fig. 13. Cave Output Shapes

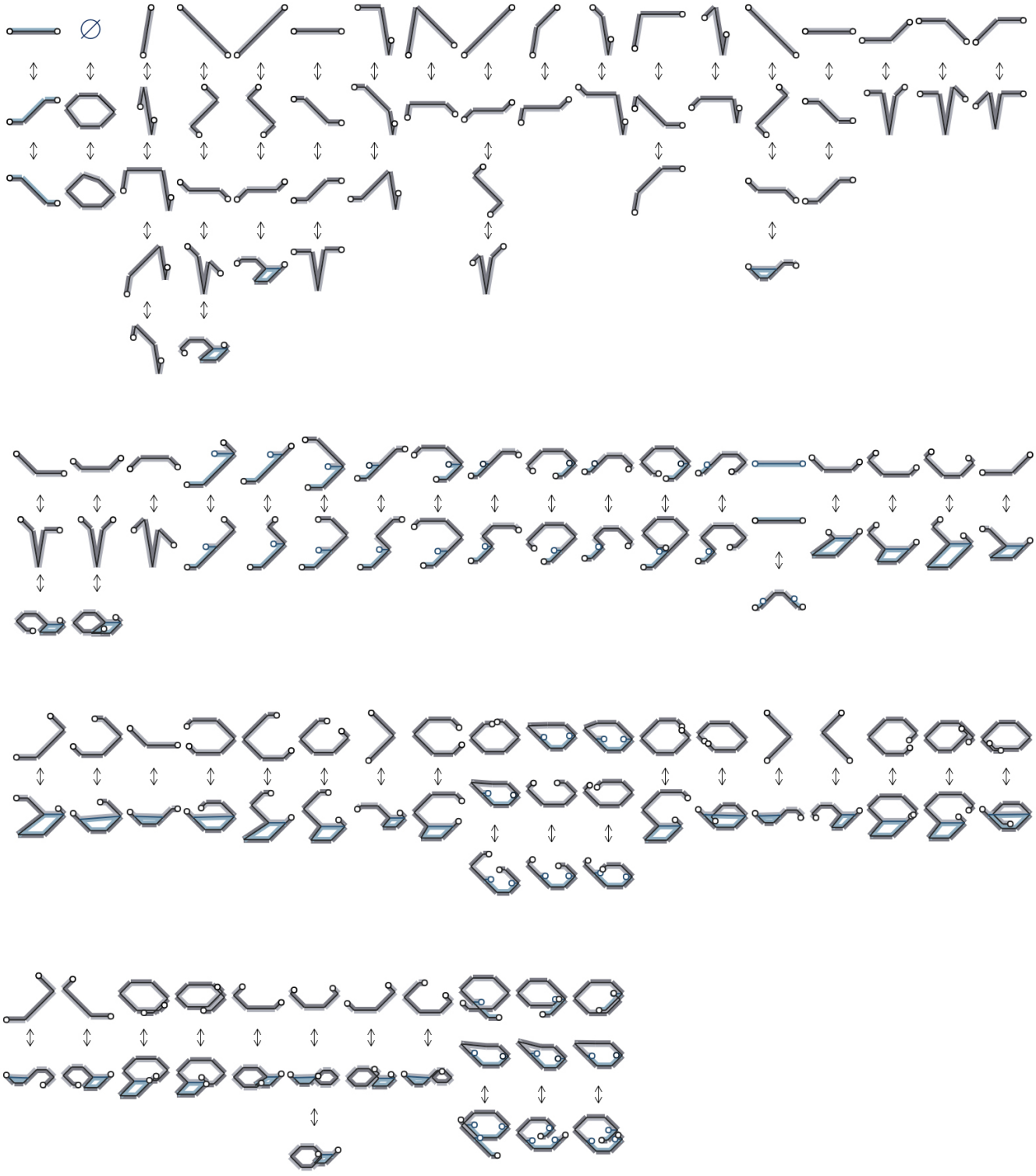


Fig. 14. Cave Graph Grammar (87 Rules)

3.3 Space Station

This corresponds to Figure 9b in the main paper.

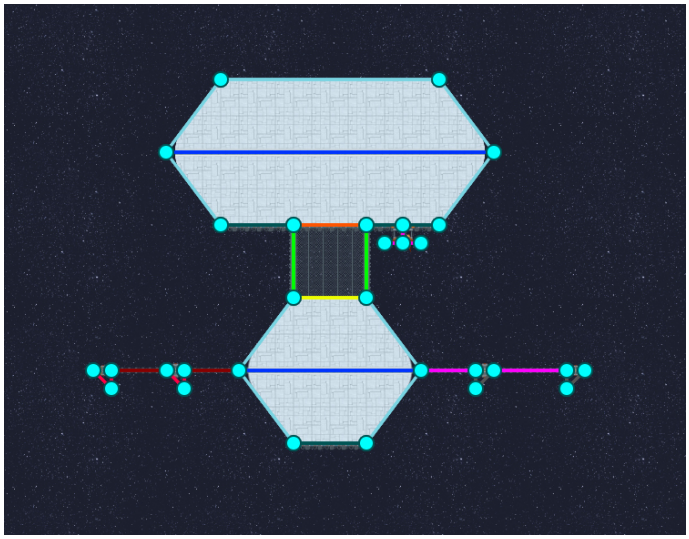


Fig. 15. Space Station Example Shape

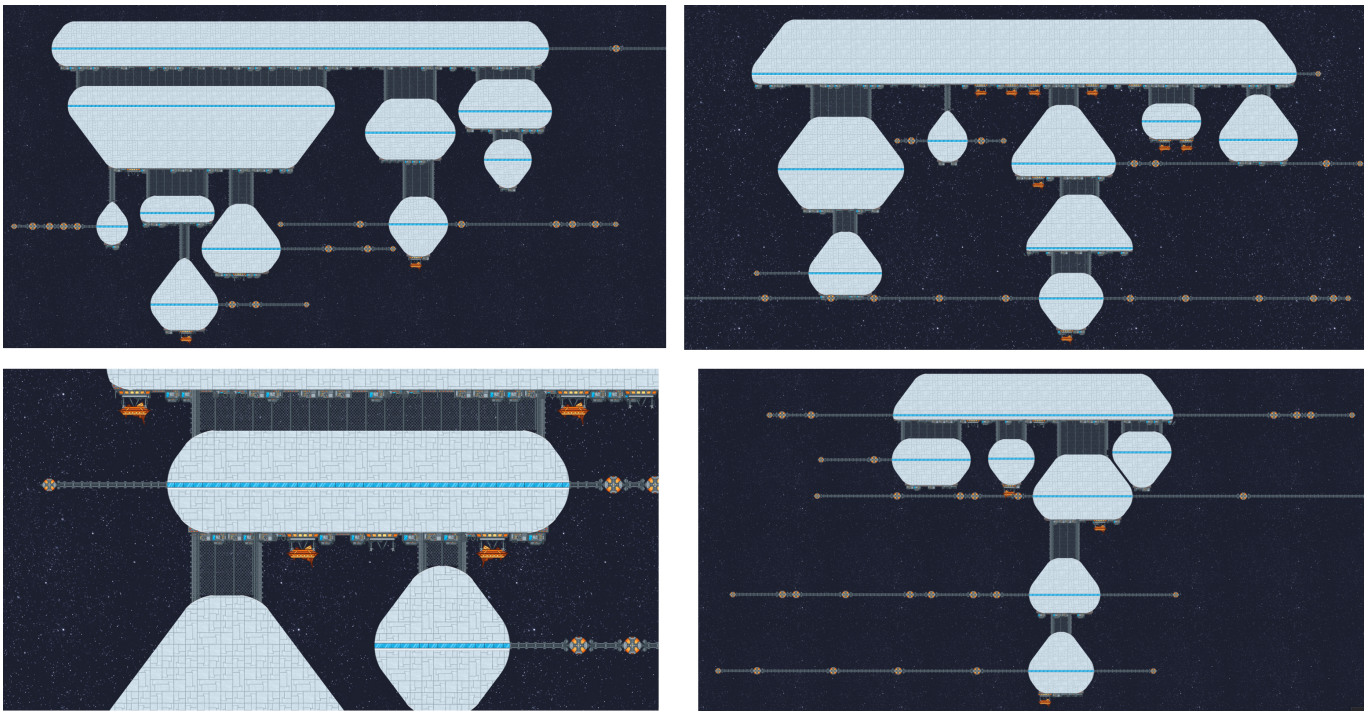


Fig. 16. Space Station Output Shapes

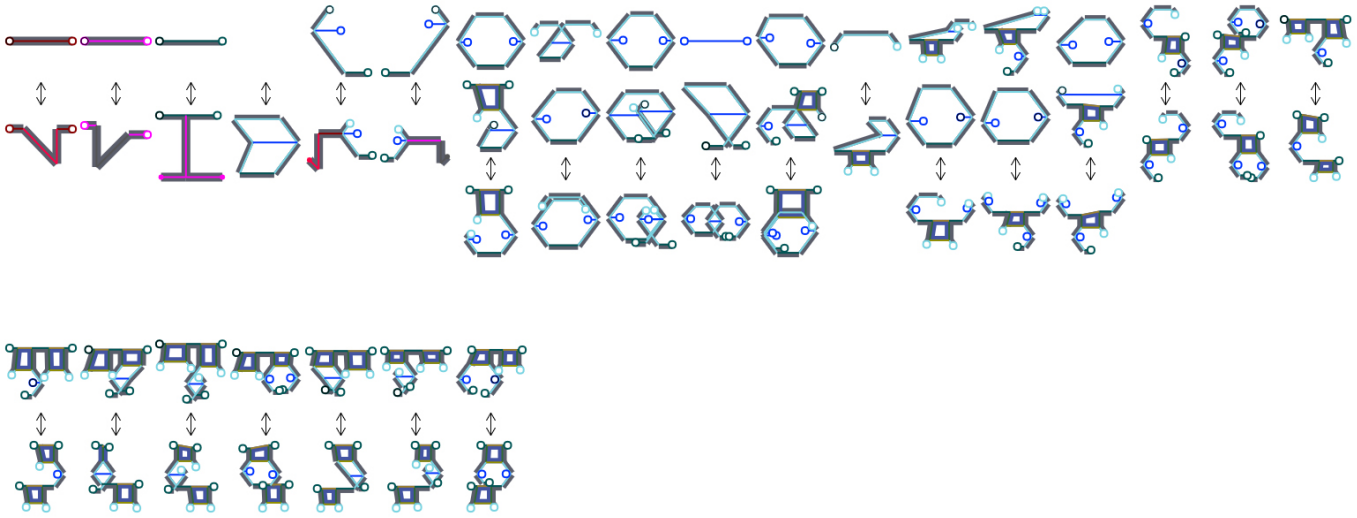


Fig. 17. Space Station Graph Grammar (25 Rules)

3.4 Processing Plant

This corresponds to Figure 9c in the main paper.

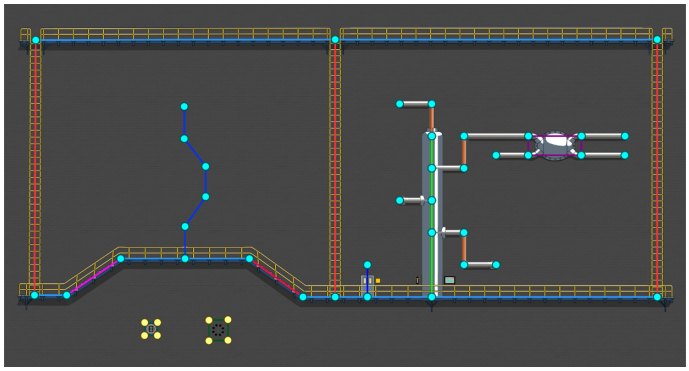


Fig. 18. Processing Plant Example Shape

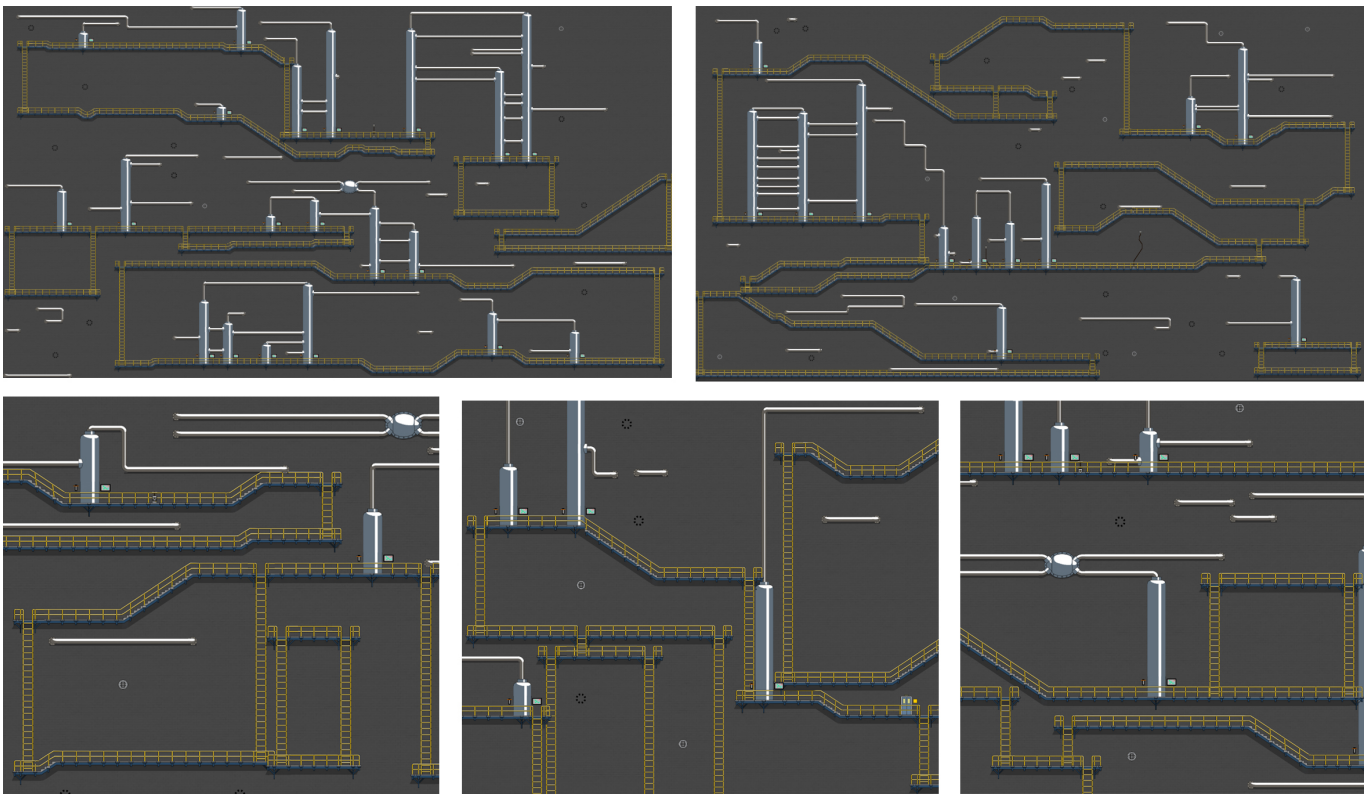


Fig. 19. Processing Plant Output Shapes

Fig. 20. Processing Plant Graph Grammar, Page 1

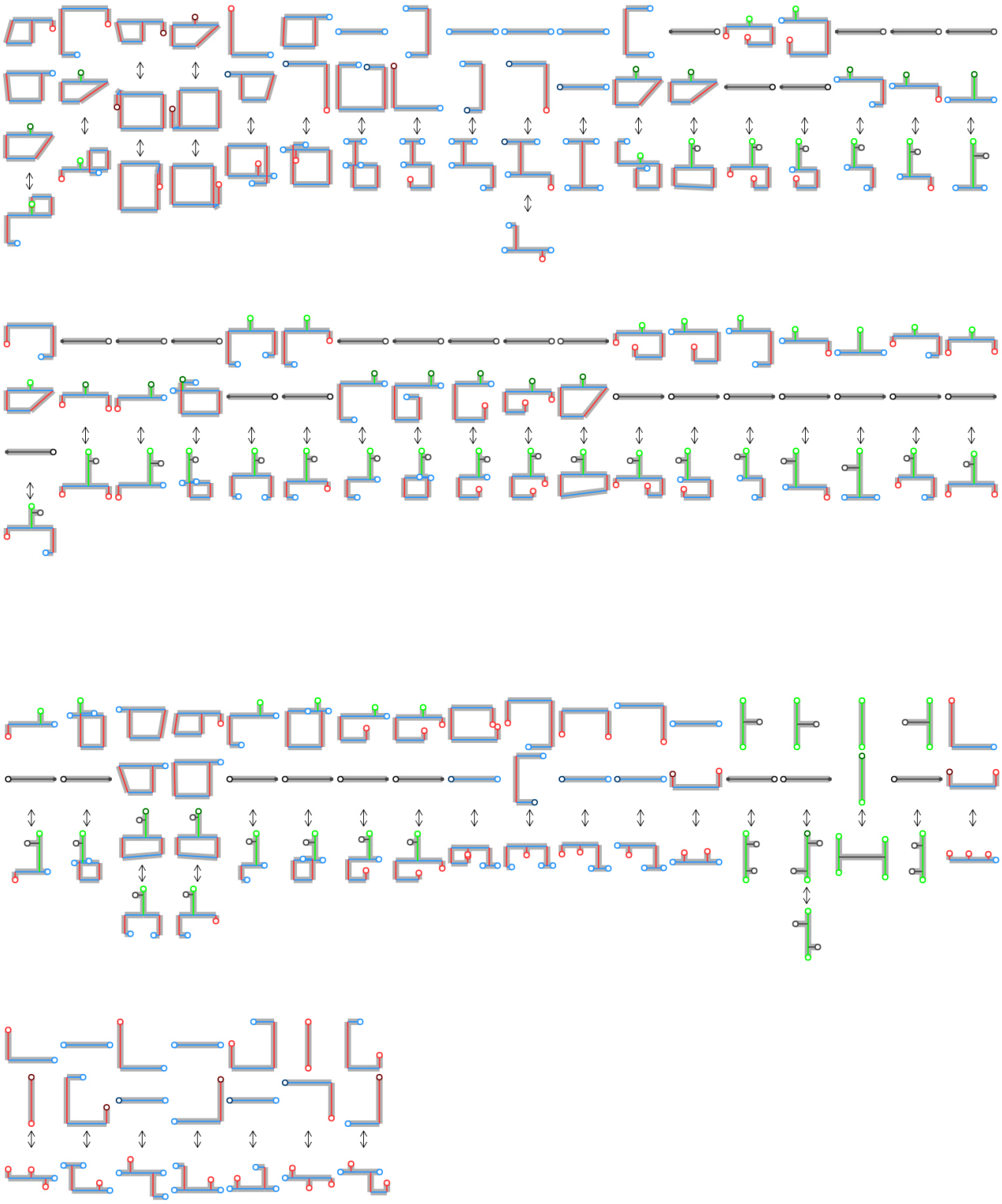


Fig. 21. Processing Plant Graph Grammar, Page 2 (154 Total Rules)

3.5 Islands

This corresponds to Figure 9d in the main paper.

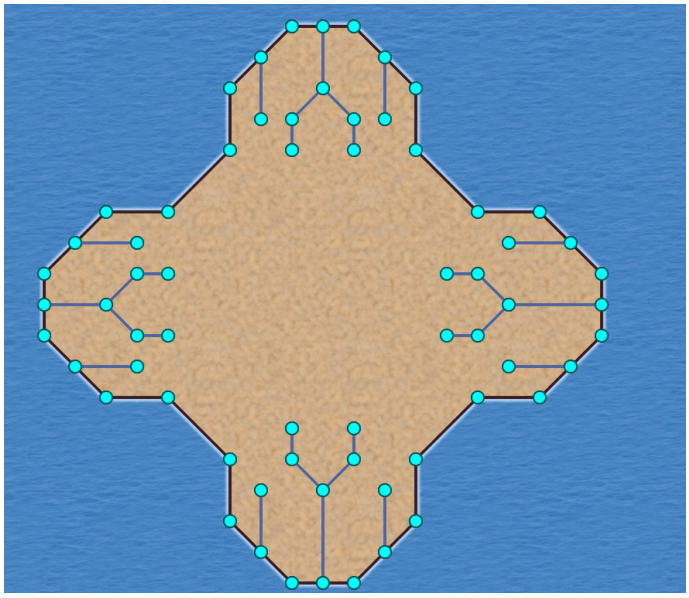


Fig. 22. Islands Example Shape

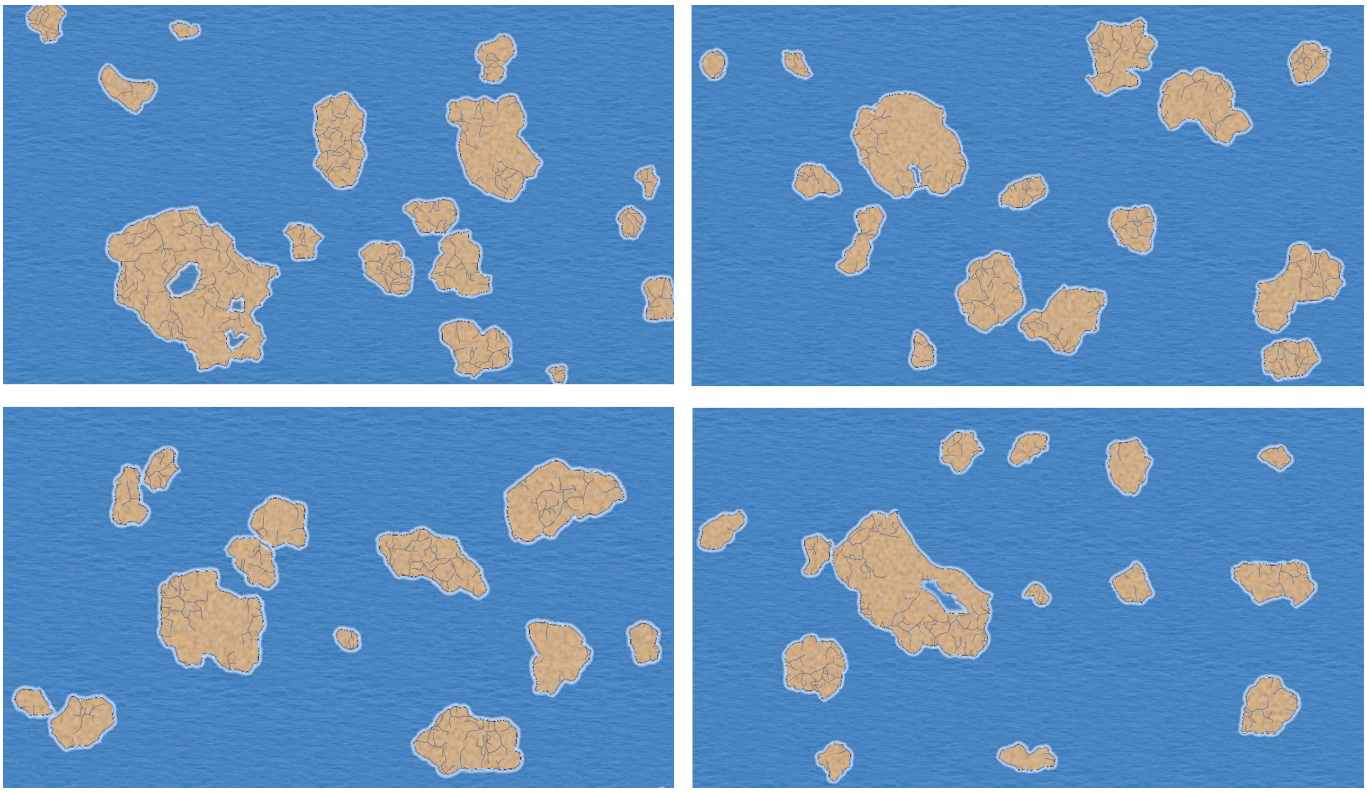


Fig. 23. Islands Output Shapes

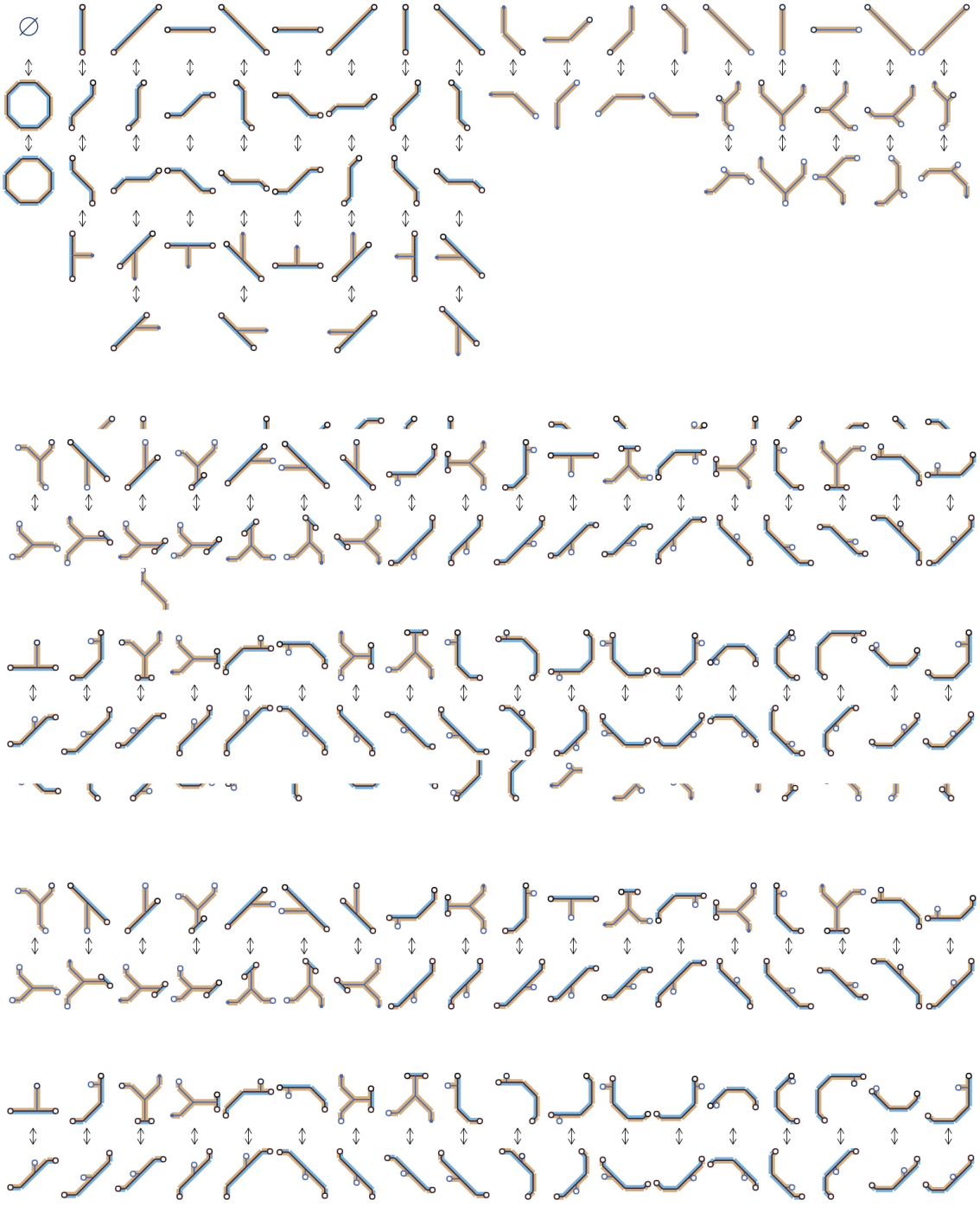


Fig. 24. Islands Graph Grammar (367 Rules). Only one page shown.

The islands graph grammar is very complicated. The full list of rules would extend four pages. The reason it is so complicated is because we are limiting ourselves in the rules we can use. The river edges should not be allowed to form a loop. We explicitly disallow this using the method described in Appendix A in the main paper. This limits the types of rules we can use and consequently we need more of them. If we allowed loops to form the graph grammar could be much simpler, only requiring 50 rules.

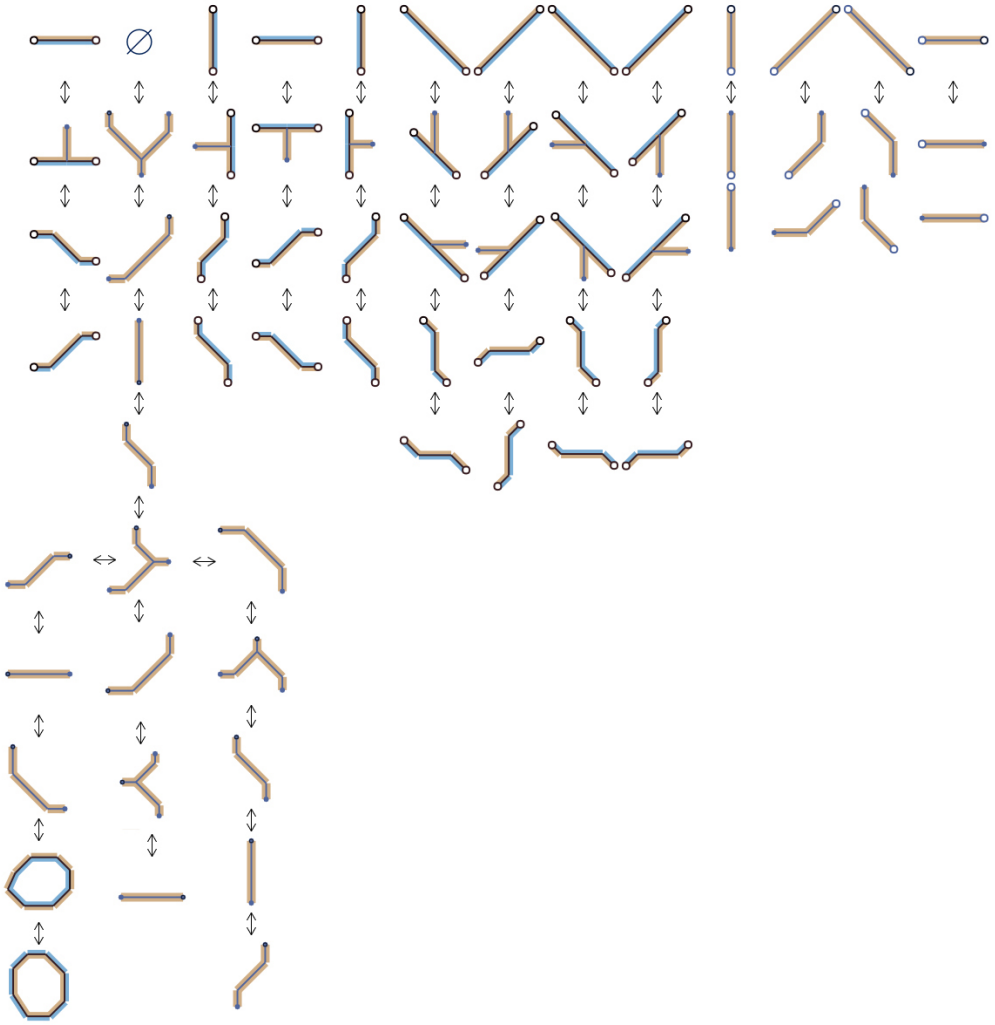


Fig. 25. The Islands Graph Grammar With Loops Allowed (50 Rules)

3.6 Neighborhood

This corresponds to Figure 9e in the main paper.

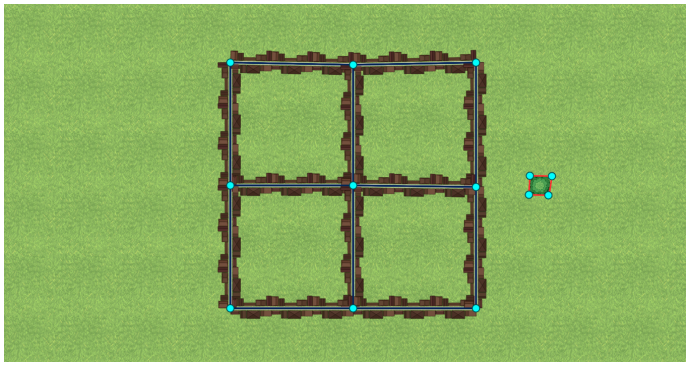


Fig. 26. Neighborhood Example Shape

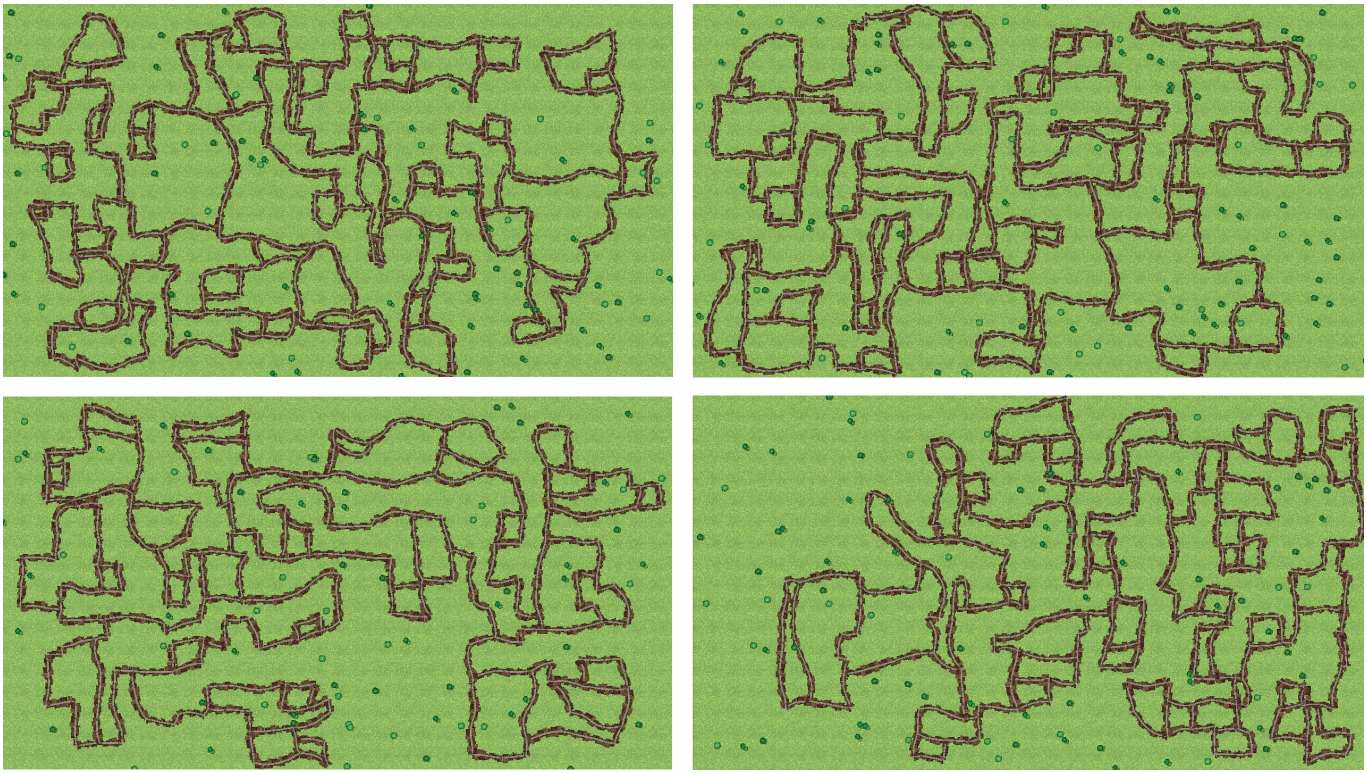


Fig. 27. Neighborhood Output Shapes

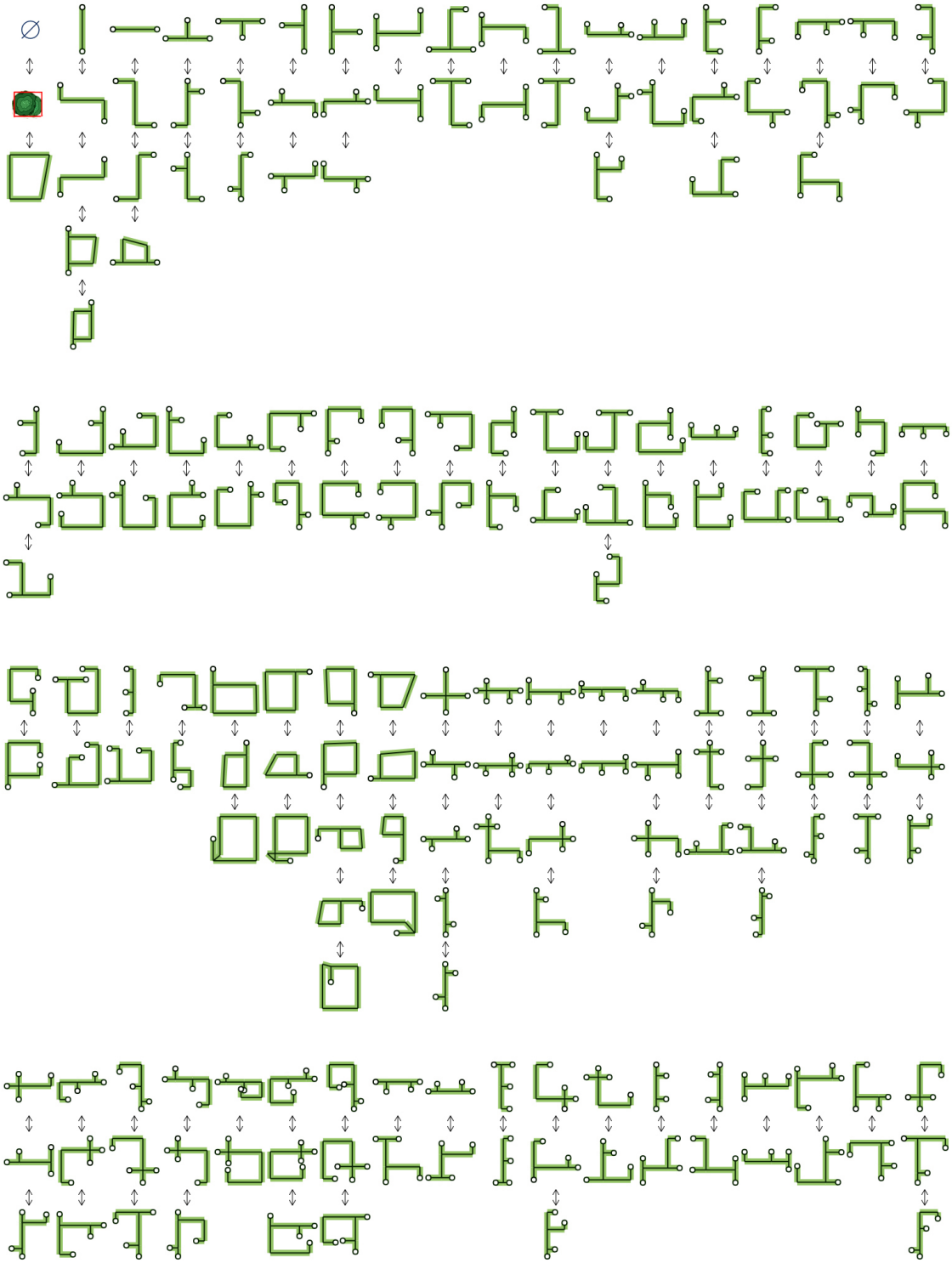


Fig. 28. Neighborhood Graph Grammar (216 Rules). Only one page shown.

The neighborhoods graph grammar is also complicated for reasons similar to the island example (See previous example, Section 3.5). The complete set would take three pages. The reason it is complicated is because we are limiting ourselves because we want the neighborhood streets to be fully connected. We accomplish this by using the method described in Appendix A. This limits the types of rules we can use. We cannot allow a rule that splits an edge into two pieces because this could cause the shape to become fully disconnected. If we were to allow splitting the solution would be simple. We could use stubs to immediately solve the problem with just 12 rules. See Section 5.5 in the main paper.

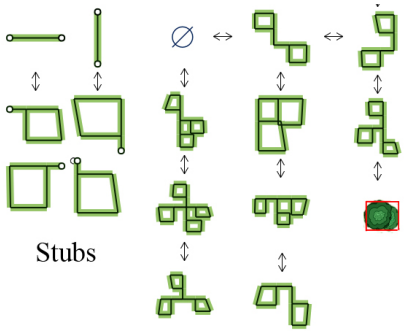


Fig. 29. Neighborhood Graph Grammar (12 Rules)

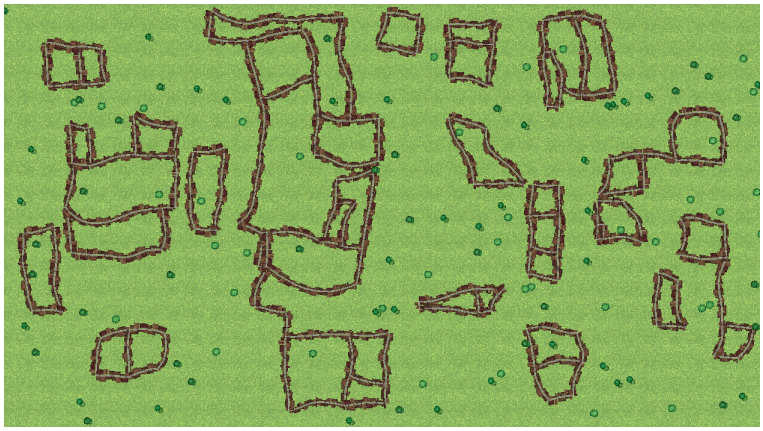


Fig. 30. The Output Shapes When Not Requiring Fully Connectivity. We do not use the simpler graph grammar in the main paper because we do not want the streets to become disconnected like this.

3.7 Flowers

This corresponds to Figure 9f in the main paper.

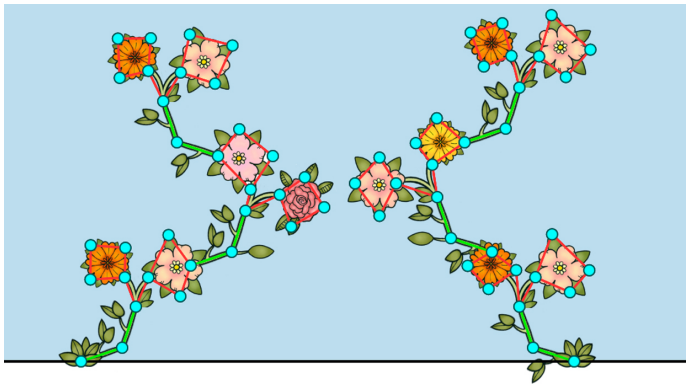


Fig. 31. Flowers Example Shape

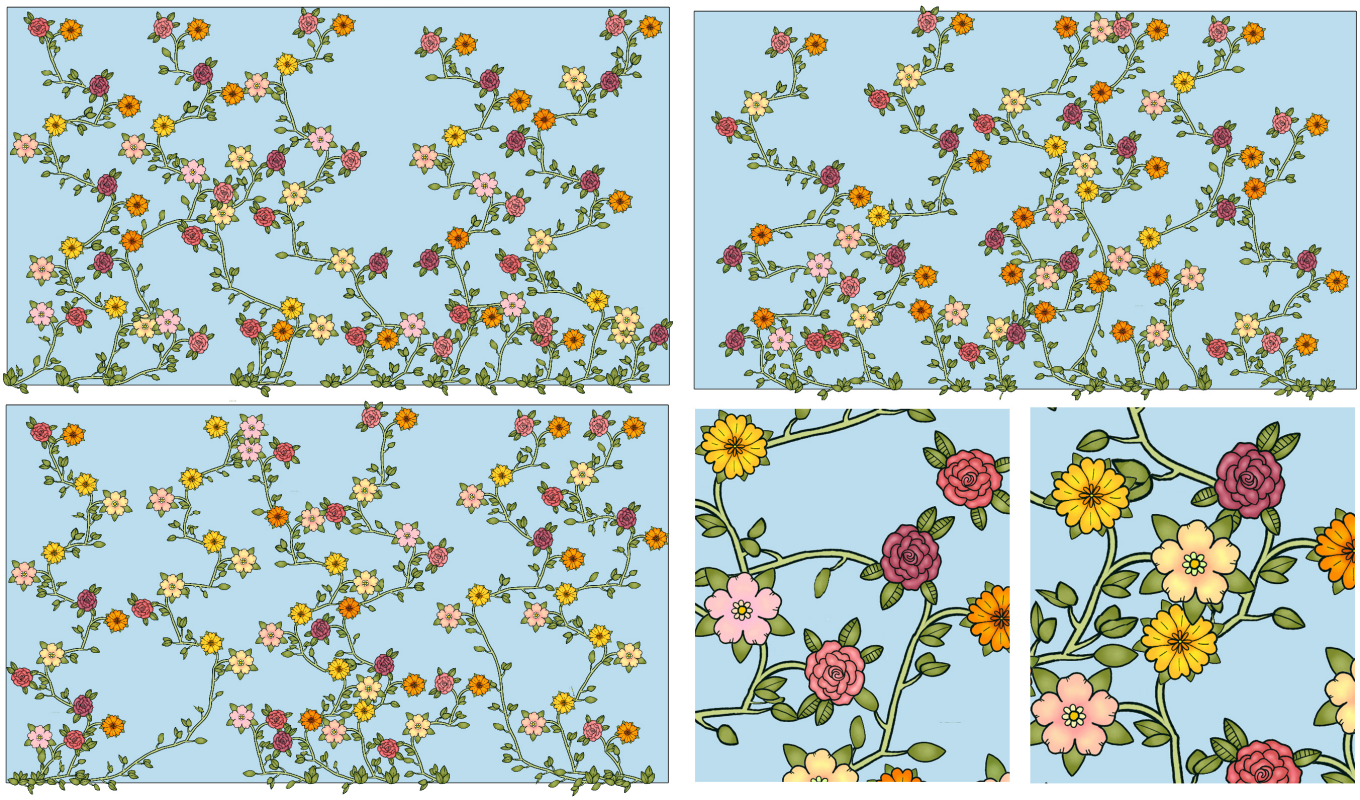


Fig. 32. Flowers Output Shapes

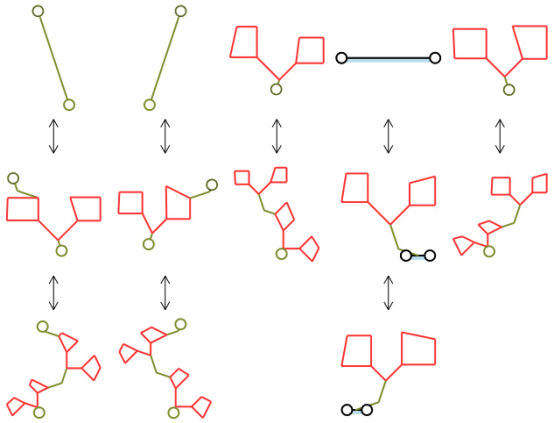


Fig. 33. Flowers Graph Grammar (8 Rules)

3.8 Village

This corresponds to Figure 9g in the main paper.

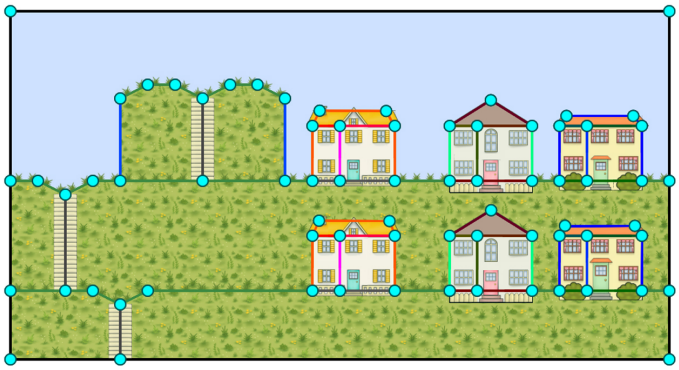


Fig. 34. Village Example Shape

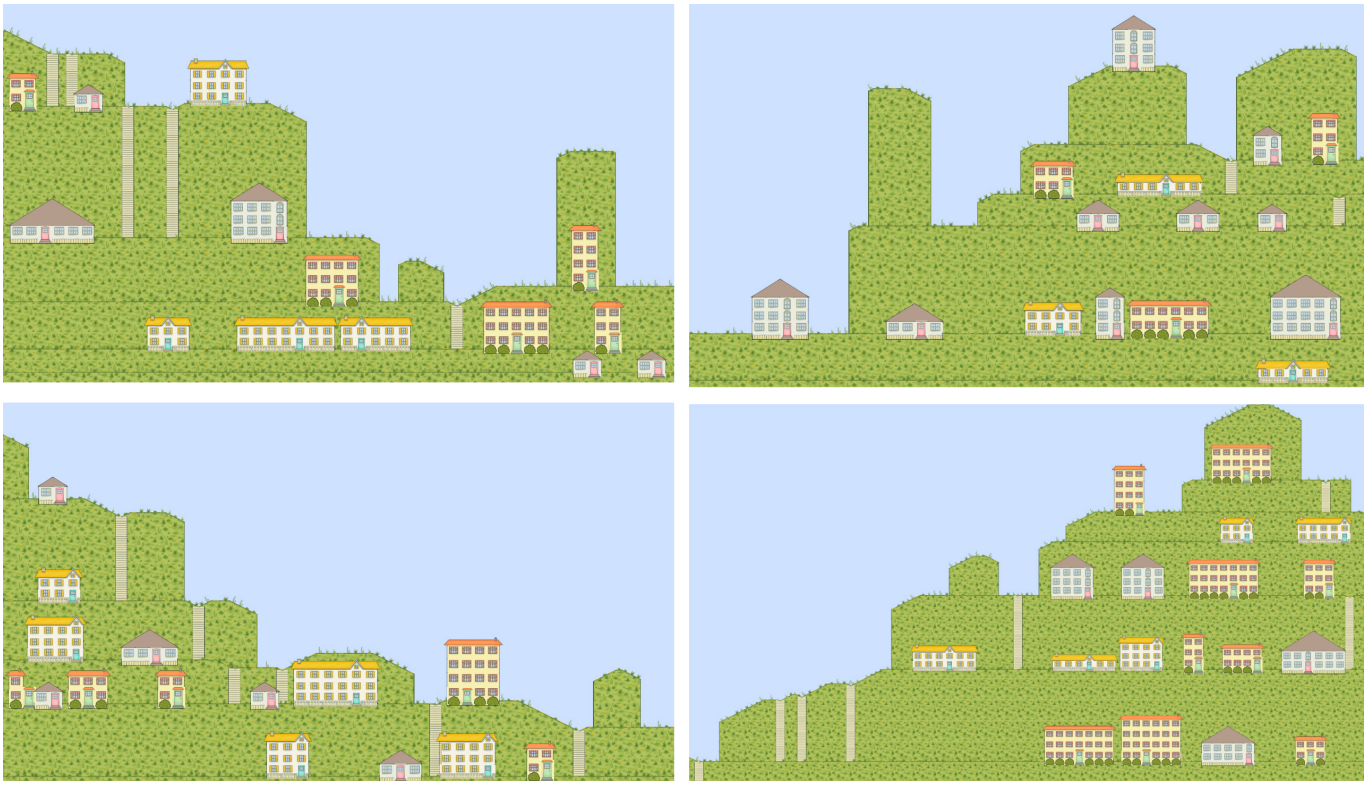


Fig. 35. Village Output Shapes

Fig. 36. Village Graph Grammar (55 Rules)

3.9 Trees

This corresponds to Figure 9h in the main paper.

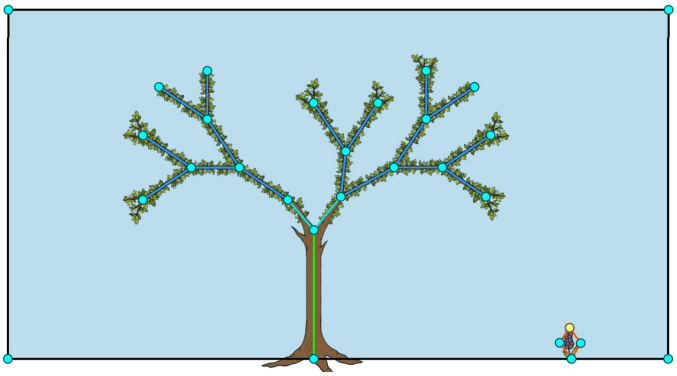


Fig. 37. Trees Example Shape

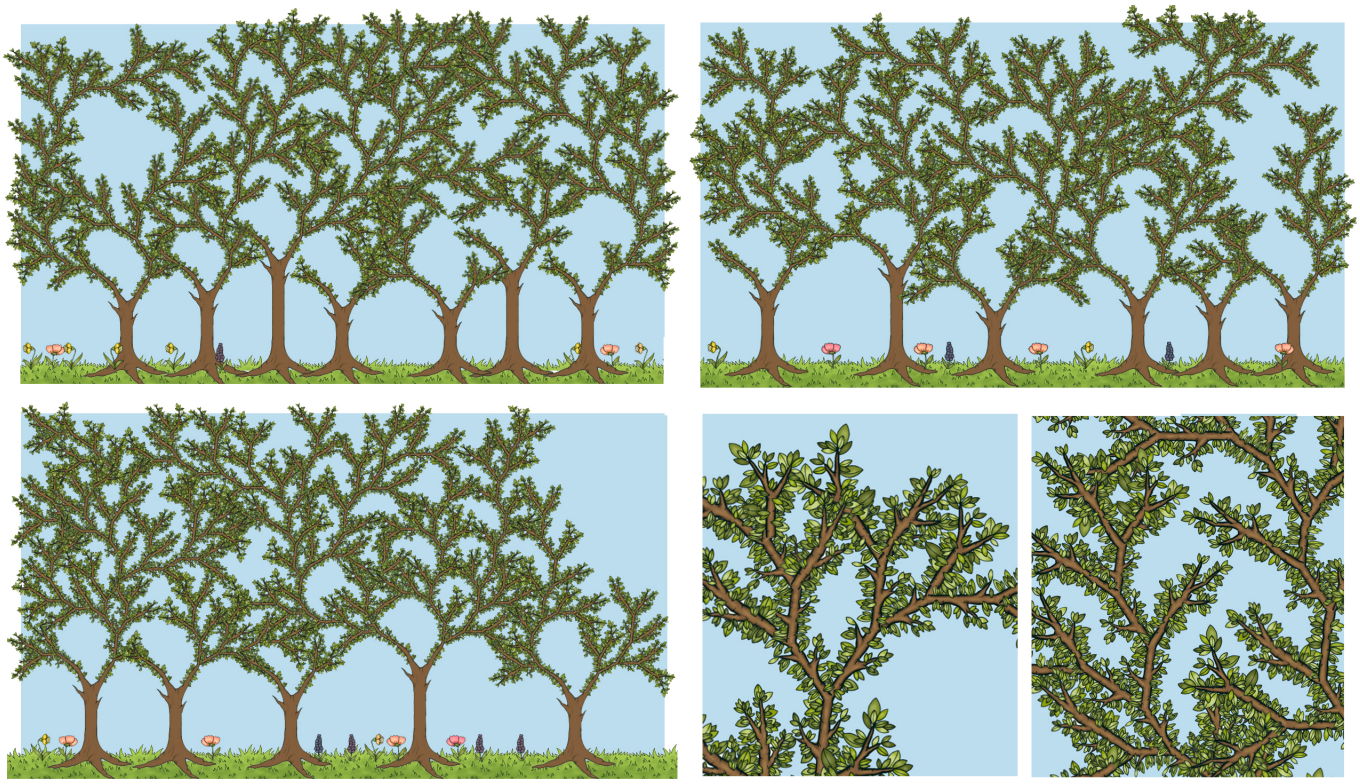


Fig. 38. Trees Output Shapes

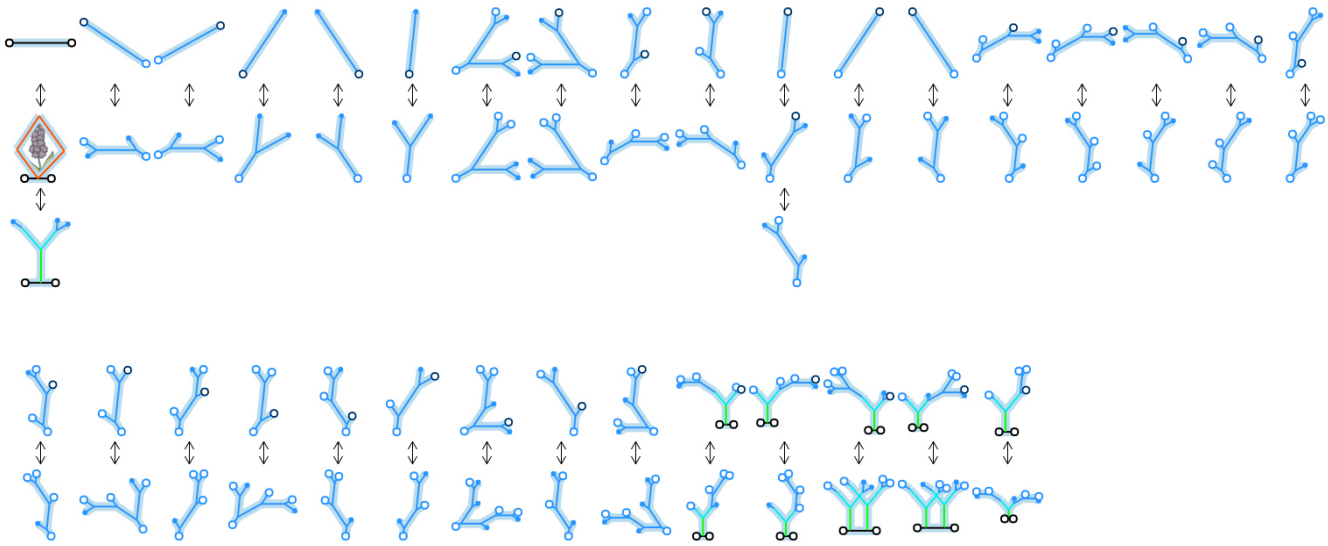


Fig. 39. Trees Graph Grammar (33 Rules)

3.10 Boardwalk

This corresponds to Figure 9i in the main paper.

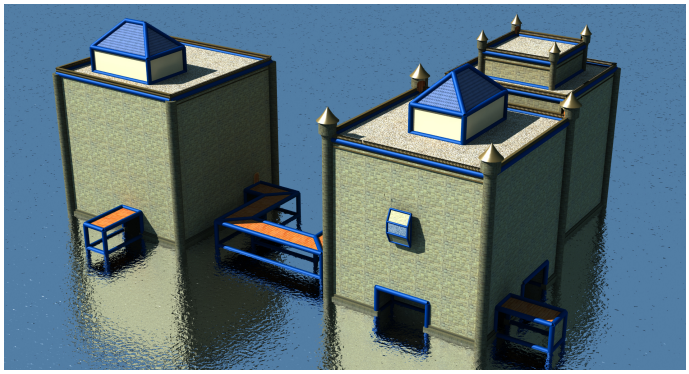


Fig. 40. Boardwalk Example Shape

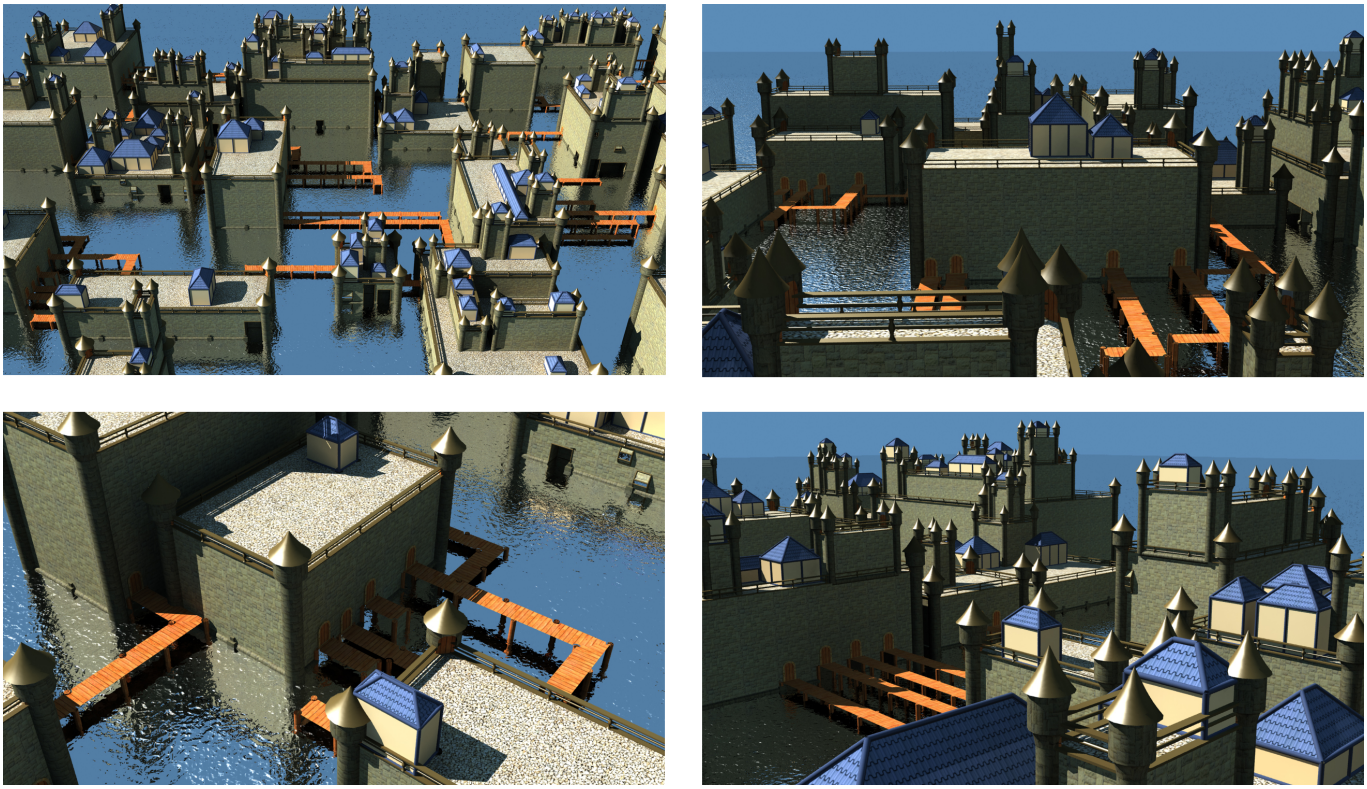


Fig. 41. Boardwalk Output Shapes

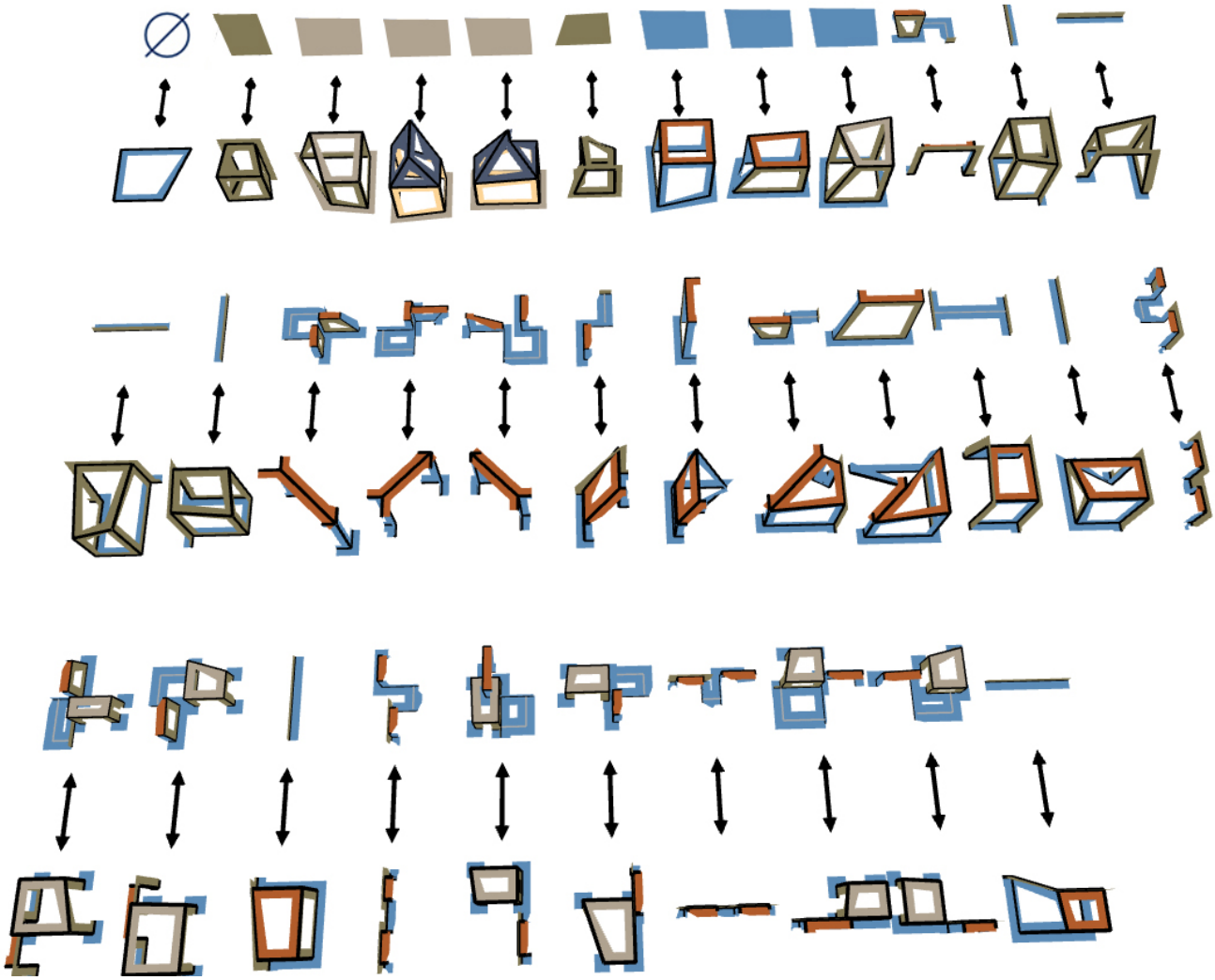


Fig. 42. Boardwalk Graph Grammar (34 Rules)

3.11 Skyscrapers

This corresponds to Figure 9j in the main paper.

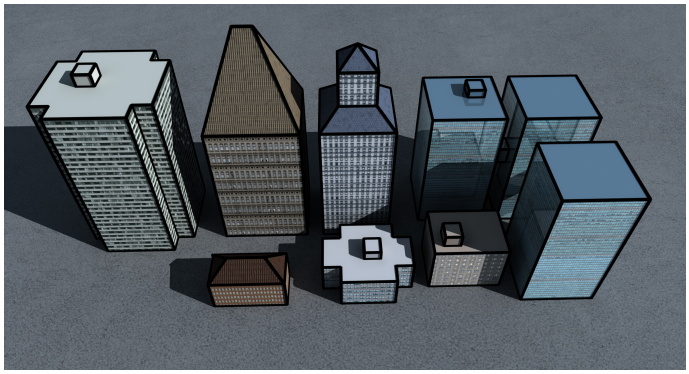


Fig. 43. Skyscrapers Example Shape

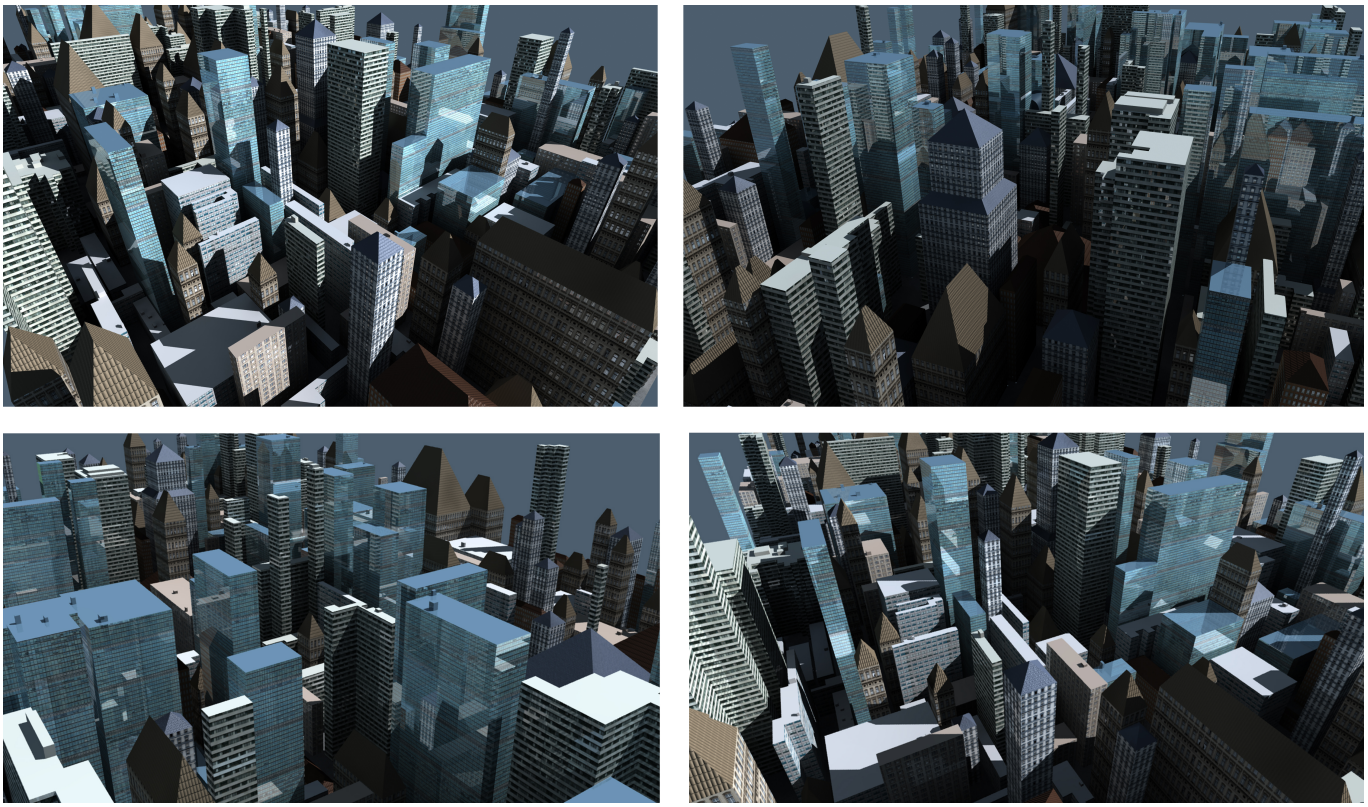


Fig. 44. Skyscrapers Output Shapes

Fig. 45. Skyscrapers Graph Grammar (44 Rules)

3.12 Houses

This corresponds to Figure 9k in the main paper.

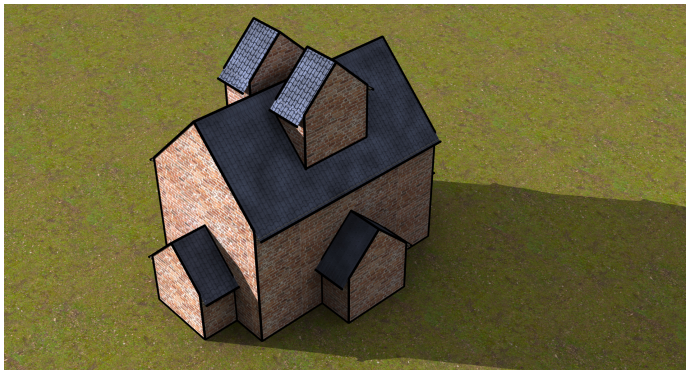


Fig. 46. Houses Example Shape

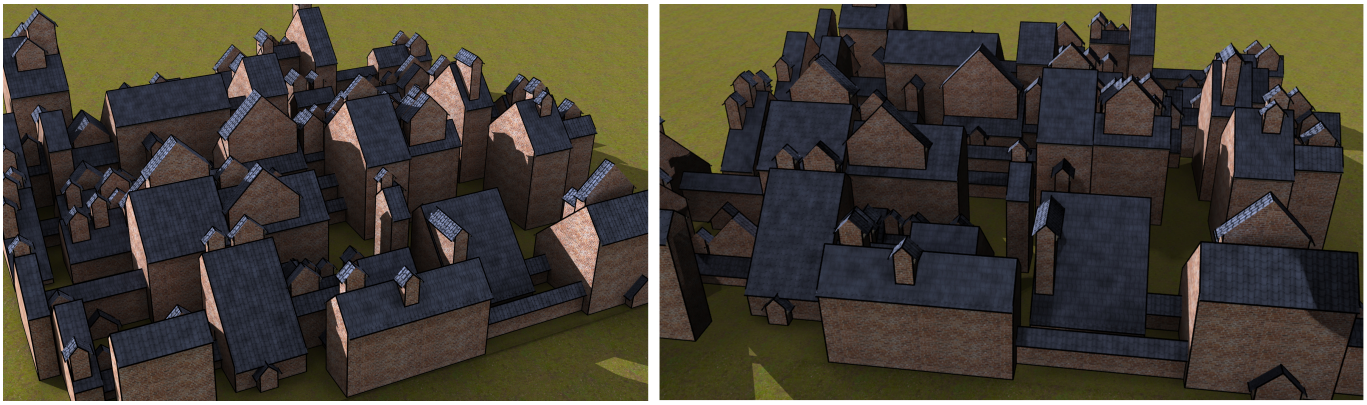


Fig. 47. Houses Output Shapes

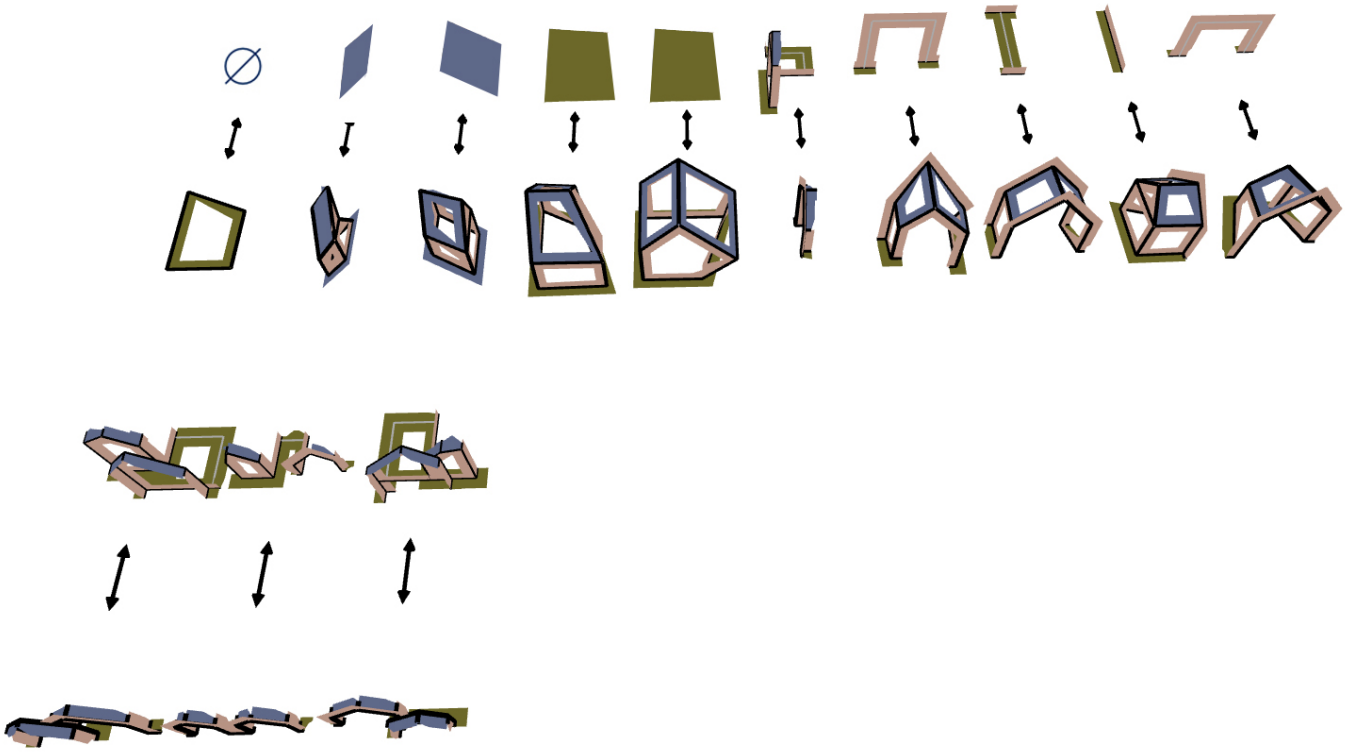


Fig. 48. Houses Graph Grammar (13 Rules)

3.13 Sci-Fi

This corresponds to Figure 9l in the main paper.

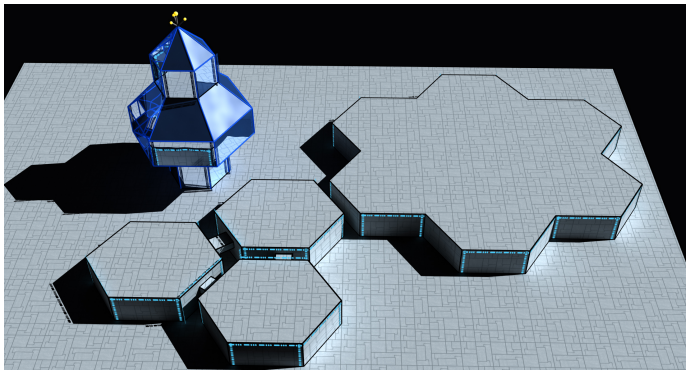


Fig. 49. Sci-Fi Example Shape

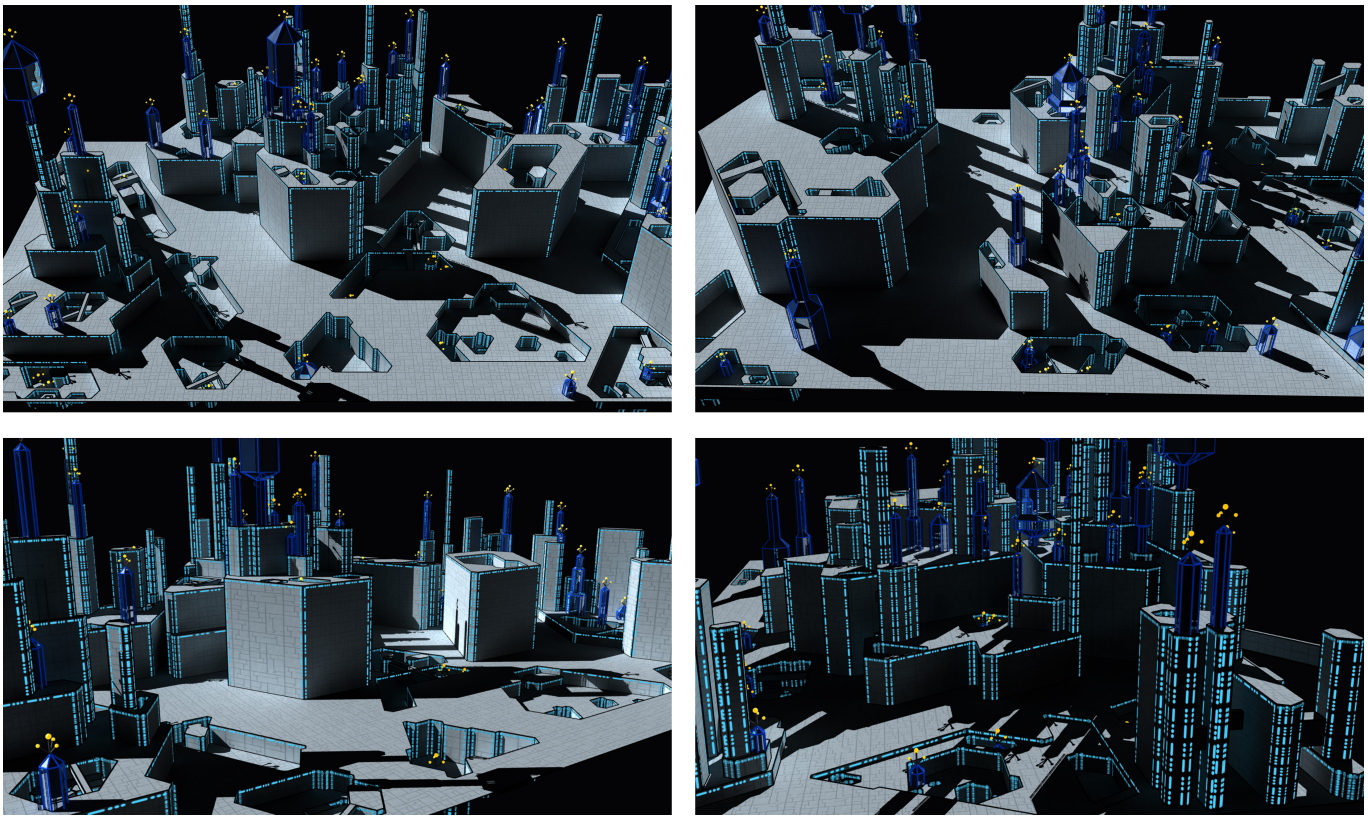


Fig. 50. Sci-Fi Output Shapes

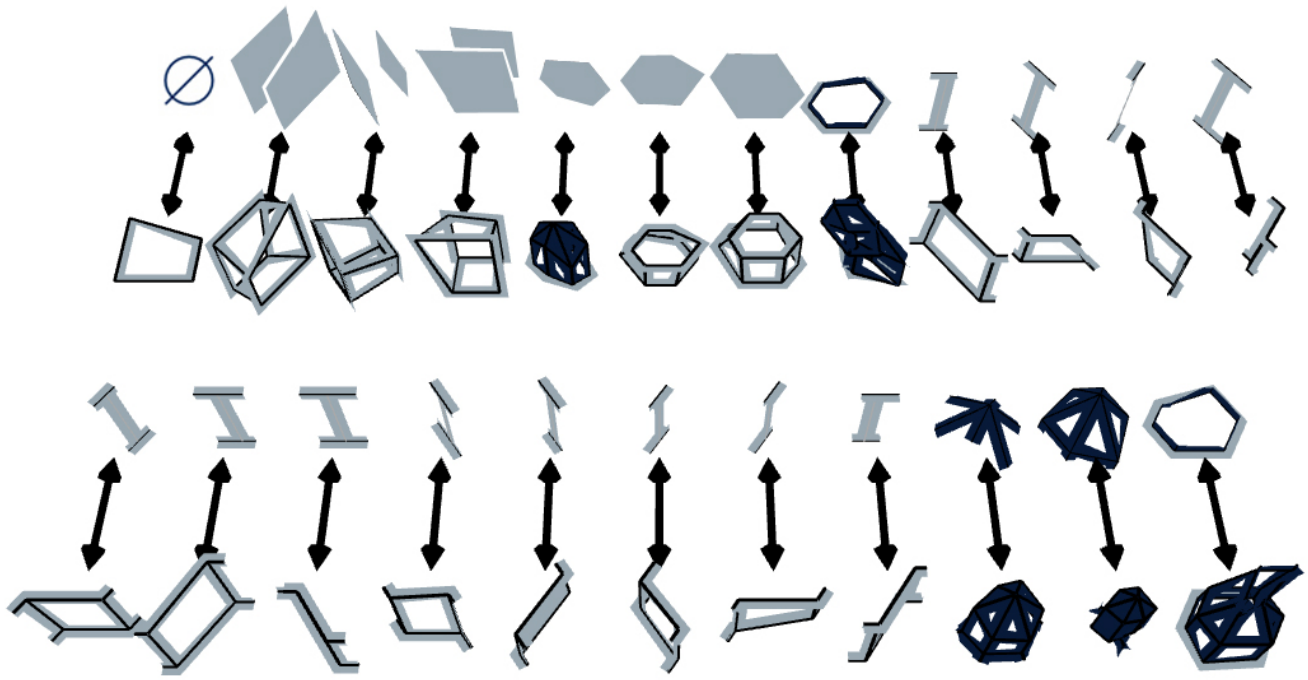


Fig. 51. Sci-Fi Graph Grammar (23 Rules)

4 DECONSTRUCTION STEPS

Here we show how to deconstruct several complex shapes. As explained in Section 5.2 (main paper) deconstruction and construction are the same processes in reverse. If you can deconstruct a shape with a grammar you can also construct it.

4.1 Rectangle

This corresponds to Figure 8a in the main paper.

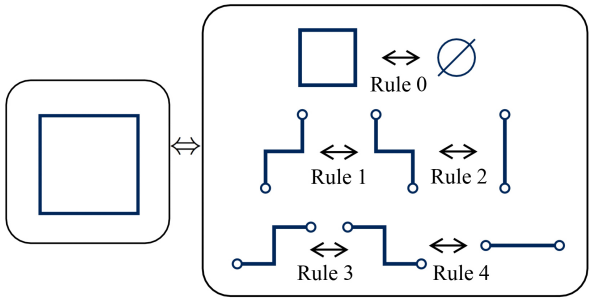
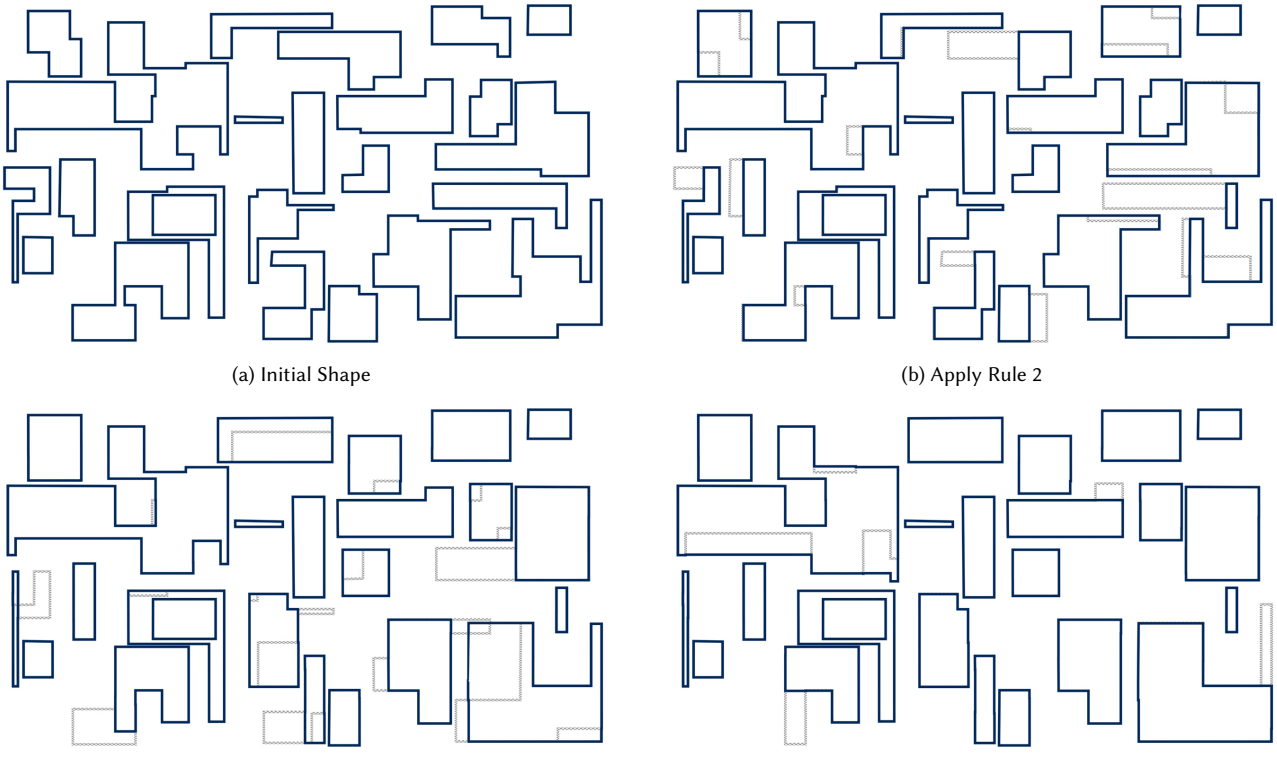


Fig. 52. Input Shape (left). Graph Grammar (right)



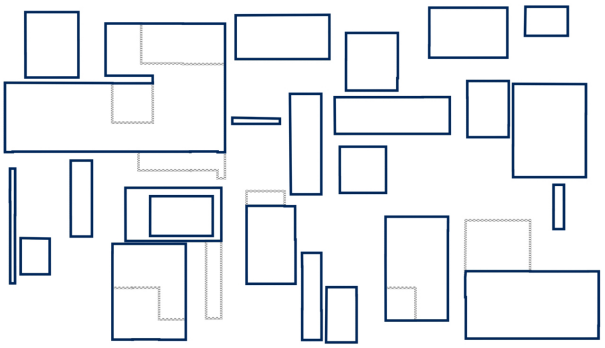
(a) Initial Shape

(b) Apply Rule 2

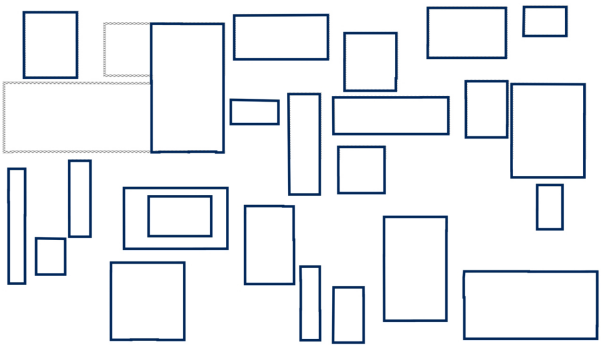
(c) Apply Rule 1

(d) Apply Rule 3

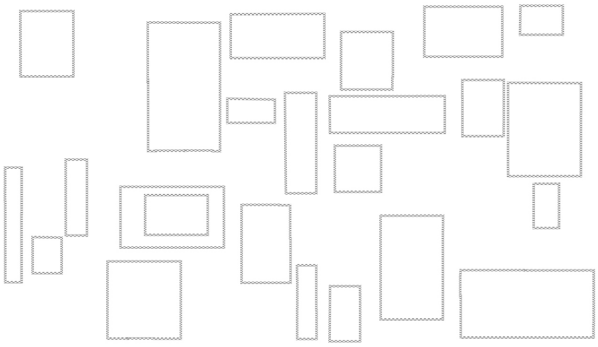
Fig. 53. The steps necessary to deconstruct the initial shape. Continued on next page.



(a) Apply Rule 4



(b) Apply Rules 1 and 2



(c) Apply Rule 0

4.2 Overlapping Rectangles

This corresponds to Figure 8o in the main paper.

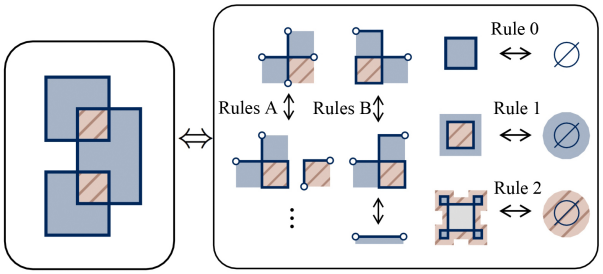
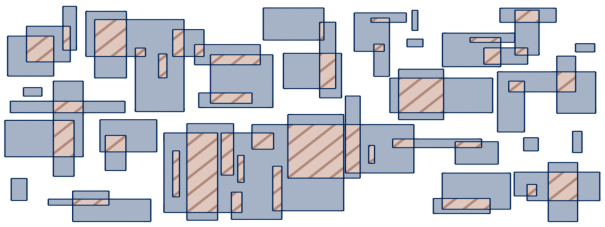
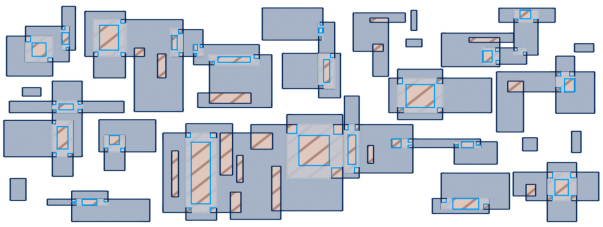


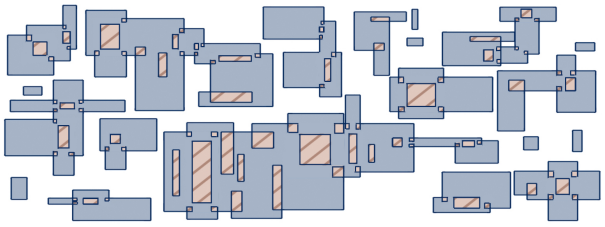
Fig. 55. Example Shape (left) and Generated Graph Grammar (right). The shown rules can be rotated 90°. Rules A and B consist of copies of one rule rotated at 90° angles.



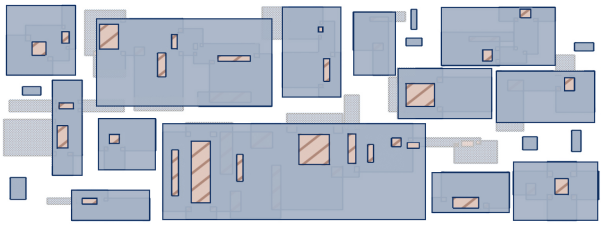
(a) Initial Shape



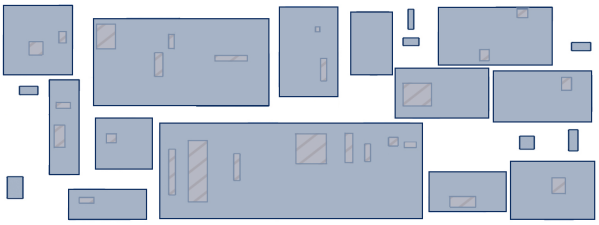
(b) Apply Rules A. The same rule is applied with 90° rotations.



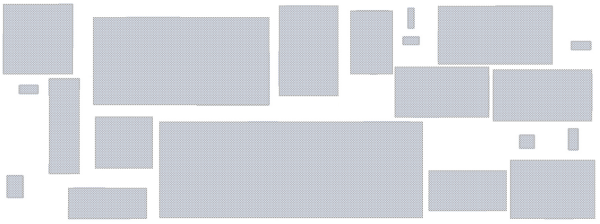
(c) After Rules A Applied



(d) Apply Rules B. The same rule is applied with 90° rotations.



(e) Apply Rule 1



(f) Apply Rule 0

4.3 Stacked Platform

This corresponds to Figure 8n in the main paper.

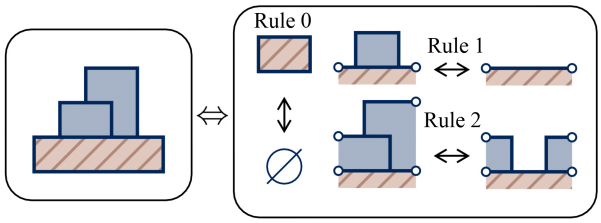
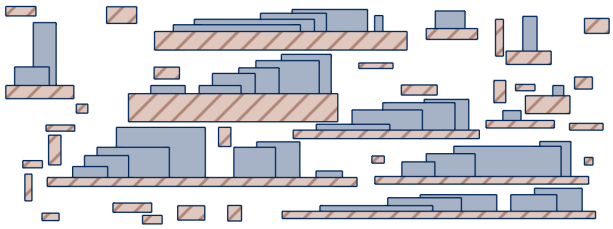
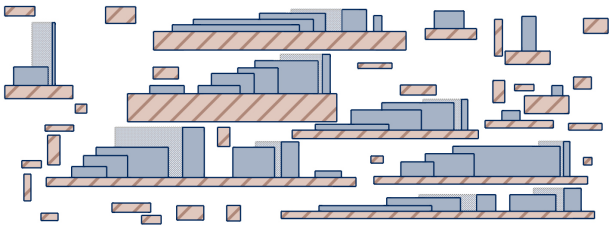


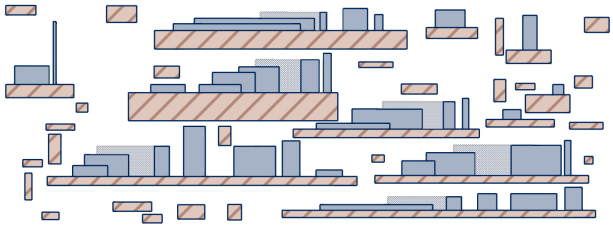
Fig. 57. Initial Shape



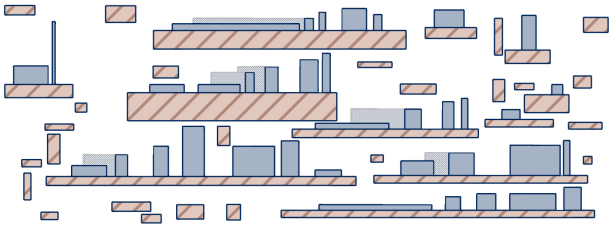
(a) Apply Rule 2



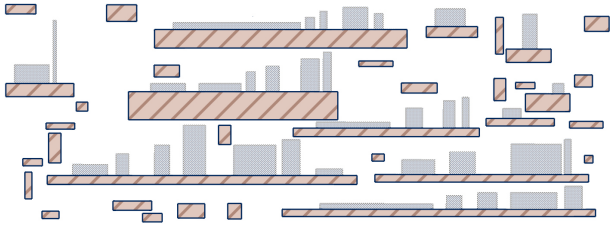
(b) Apply Rule 2



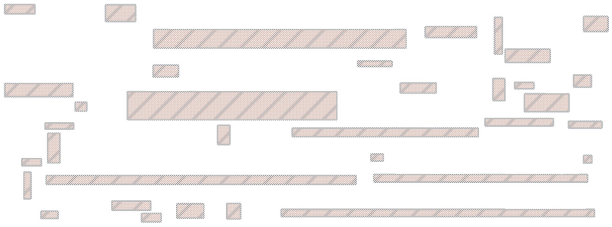
(c) Apply Rule 2



(d) Apply Rule 1



(e) Apply Rule 0



(f) Step 0

4.4 Buildings on Platforms

This corresponds to Figure 8g in the main paper.

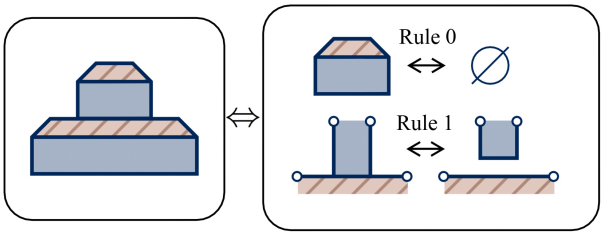
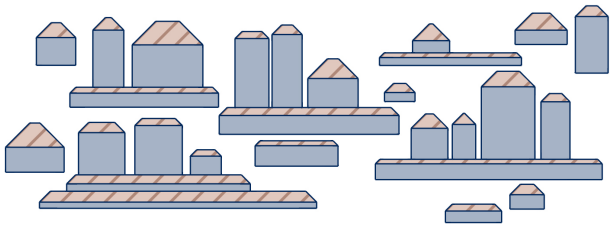
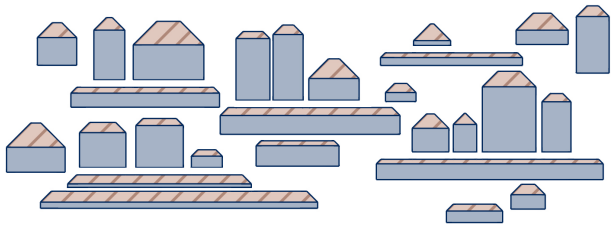


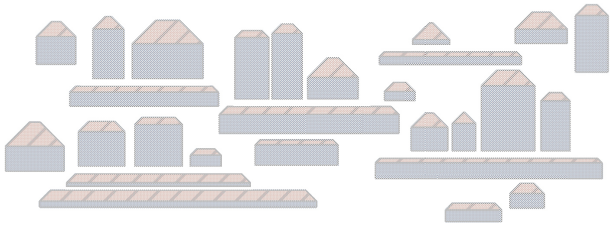
Fig. 59. Example Shape (left) and Generated Graph Grammar (right)



(a) Initial Shape



(b) Apply Rule 1



(c) Apply Rule 0

4.5 Four-way Intersections

This corresponds to Figure 8f in the main paper.

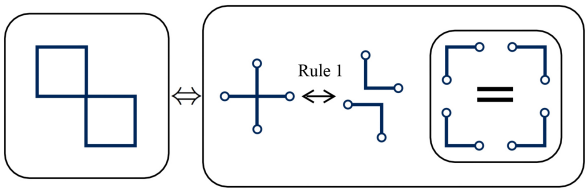
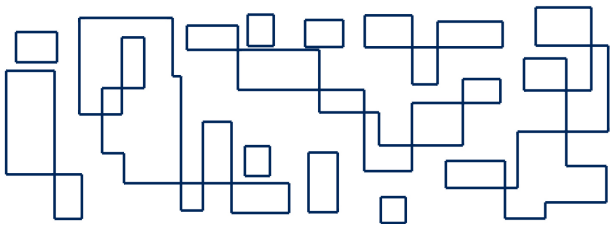
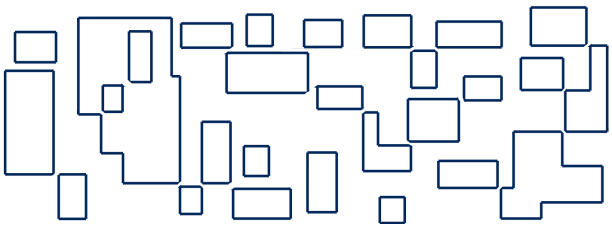


Fig. 61. Example Shape (left) and Generated Graph Grammar (right)



(a) Initial Shape



(b) Apply Rule 1. At this point, the shapes are all locally similar to the rectangle example. The same solution can be applied to deconstruct all the shapes.

4.6 H shape

This corresponds to Figure 8b in the main paper.

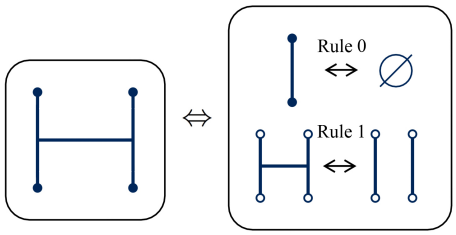
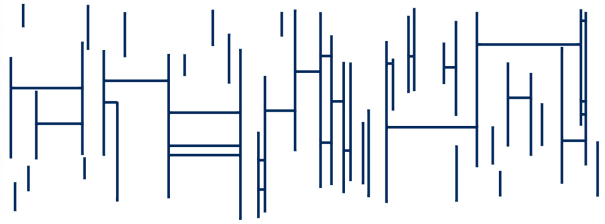
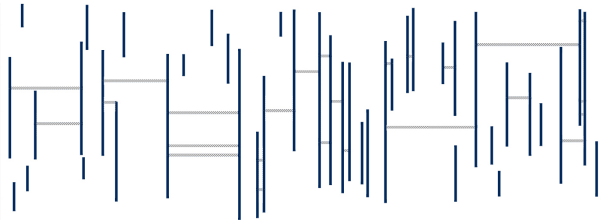


Fig. 63. Example Shape (left) and Generated Graph Grammar (right)



(a) Initial Shape



(b) Apply Rule 1

(c) Apply Rule 0

This is the same as the previous input shape, but we do not allow the grammar to create any loops. This corresponds to Figure 8c in the main paper.

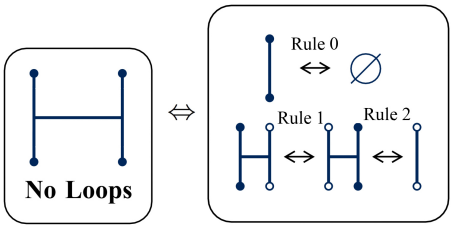
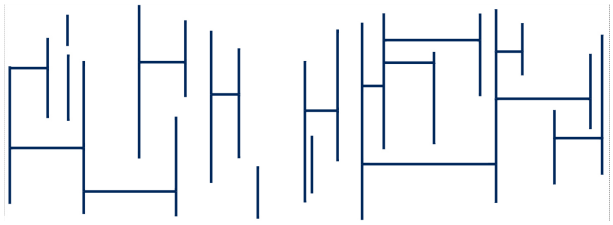
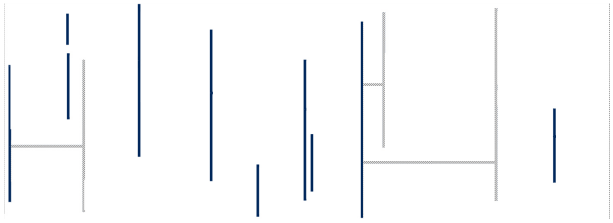


Fig. 65. Example Shape (left) and Generated Graph Grammar (right)

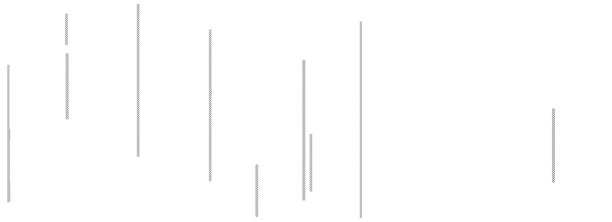


(a) Initial Shape

(b) Apply Rule 2



(c) Apply Rule 2



(d) Apply Rule 0

4.7 Branches

This corresponds to Figure 8d in the main paper.

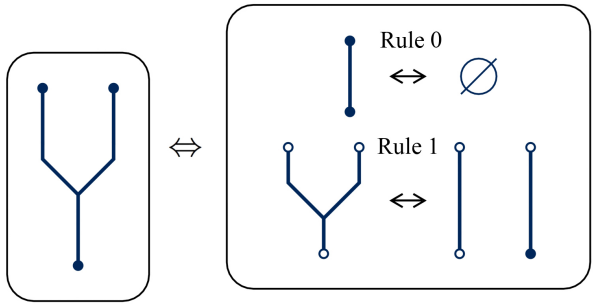
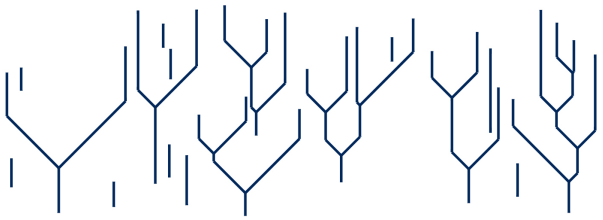
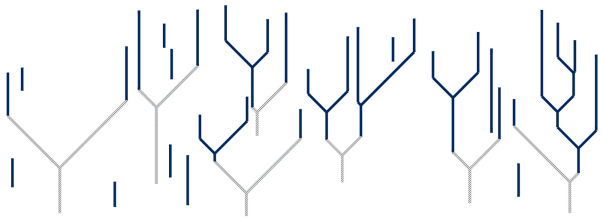


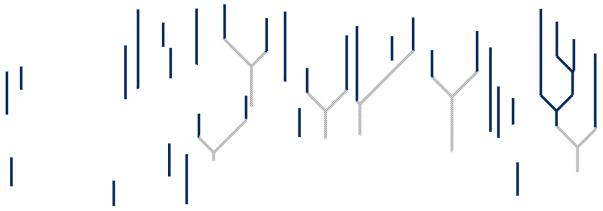
Fig. 67. Example Shape (left) and Generated Graph Grammar (right)



(a) Initial Shape



(b) Apply Rule 1



(c) Apply Rule 1

(d) Apply Rule 1

(e) Apply Rule 0