Procedural Grammar for Truss Microstructure Design

Ane Zuñiga and Jorge Cañada
Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology

Abstract

Truss structures play a pivotal role across diverse domains like civil engineering, space exploration, and healthcare, owing to their adaptability and structural efficacy. Despite the longstanding success of conventional design approaches in crafting large-scale truss configurations, the realm of microscale truss design remains largely untapped. In this project, we leverage procedural grammars, computational tools for design space exploration, to systematically generate cube-shaped truss structures. By defining a procedural grammar tailored for truss design, we aim to expand the design space beyond typical structures and explore novel configurations. We provide open-source code implementing the proposed grammar and generate a dataset of cube-shaped truss structures. Through this work, we contribute to the advancement of truss design methodologies.

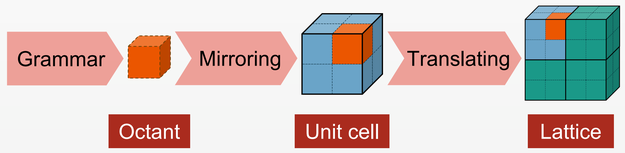
Background

Truss structures:
- They are assemblies of elements such as beams that form a rigid structure.
- They are widely used for construction, ranging from megastructures to nanostructures.
- They comprise an infinite design space hard to explore through traditional inspection methods.

Procedural grammars:
- They are computational tools for the systematic exploration of design spaces.
- They generate examples through the sequential application of pre-defined rules.
- The rules can include awareness of local and global conditions within the designs (context-sensitive grammars).

Assumptions and Feasibility Constraints

Assumptions:
- Target structures are cubic-shaped and symmetrical.
- Design focuses on a single octant of a unit cell.
- Octant is mirrored across three axes to construct the complete unit cell.
- Lattice structure is formed by translating the unit cell.



Feasibility Constraints:
We define feasibility as ensuring both the absence of floating vertices or disconnected structures and continuity preservation during mirroring and translation operations.
- To guarantee continuity of the full structure, we impose:
  - All vertices in the octant are interconnected (there is at least one path that runs through all the vertices).
  - There is at least one vertex in each face of the octant.
- To avoid the formation of non structurally sound structures, we impose:
  - Any given vertex located on a face of the octant must be connected to at least:
    + Two vertices on the same face OR
    + One vertex outside of the face.
  - Any given vertex located inside the body of the octant must be connected to at least two other vertices.

Grammar Rules

Symbols:
- E: essential vertices.
- B: additional vertices inside the body.
- F: additional vertices on faces.
- h: non-terminal edge.
- I: terminal connection.
- J: terminal disconnection.

Parameters:
- N: total number of vertices.
- M: maximum number of interconnected vertices
- *FulfilledVertex*: Indicates if a vertex fulfills connectivity conditions

Rules:
- P1: Initial Vertex and Non-terminal Edge Generation.
  Generates n_e essential vertices, n_b extra vertices within the body, n_f extra vertices on faces, and all non-terminal edge connections among each pair of vertices.

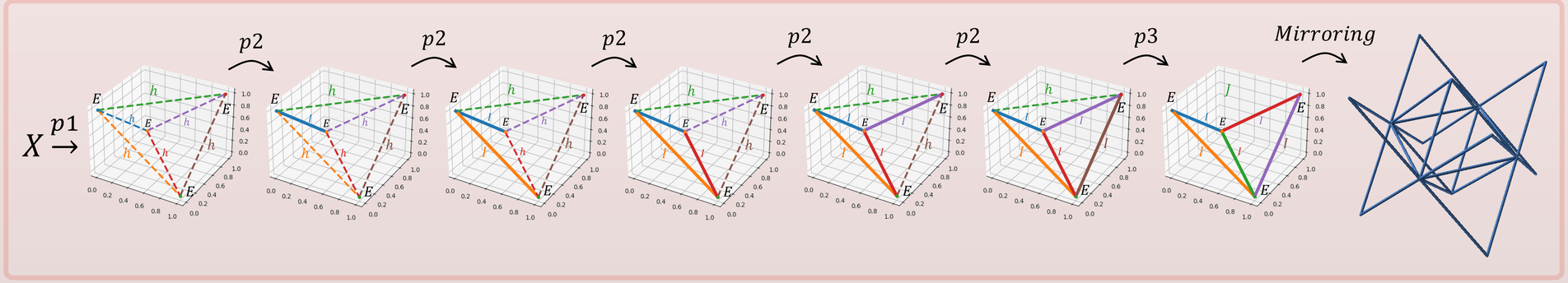
  $$None < X > None : None \rightarrow n_e E, n_b B, n_f F, \frac{N(N-1)}{2} h$$

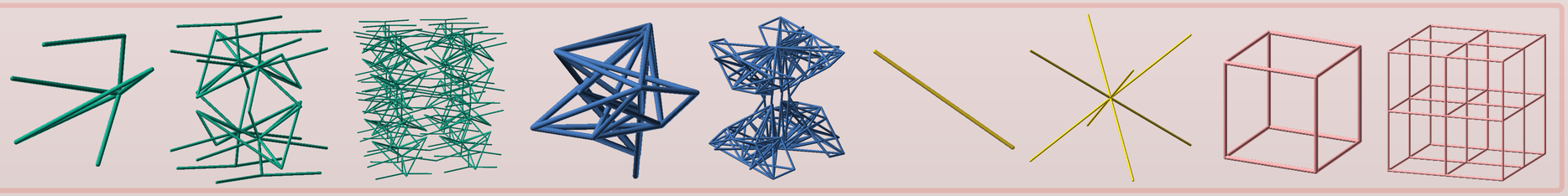
- P2: Vertex Connection.
  Converts a non-terminal edge into a terminal connection, irrespective of the current state.

  $$None < h > None : None \rightarrow I \text{ , } update\ FulfilledVertex\ and\ M$$

- P3: Vertex Disconnection.
  Converts a non-terminal edge into a terminal disconnection. It is context-sensitive and can only be applied when all vertices are interconnected, and both vertices connected by the edge satisfy the vertex connection criteria.

  $$FulfilledVertex < h > FulfilleddVertex : M == N \rightarrow J \text{ , } update\ FulfillVertex\ and\ M$$





Discussion

The proposed grammar exhibits certain limitations:
- The explored design space is restricted to cube-shaped structures generated through octant mirroring. Even though this space likely includes many of the most relevant types of truss, it is limited.
- Volume is not considered, and intersections are not addressed.

Additionally, our grammar relies on computational steps (of parameters) that are not explicitly included in the rules. This allows to keep the grammar concise but makes it not self-contained. A self-contained version would require the addition of several new symbols and rules.

Conclusion

The formulation of a context-sensitive shape grammar for truss microstructure design has been presented. The proposed grammar is capable of generating a diverse set of examples of cube-shaped trusses produced through the mirroring of an octant across three axes. While the grammar is concise, it is capable of covering a wide design space that includes some of the most commonly used truss designs (e.g., octet). This grammar enables the systematic exploration of the truss design space, allowing to discover numerous designs in an otherwise unfathomable design space. In combination with simulation tools, the proposed grammar can allow the discovery of optimal designs for specific applications.

Acknowledgements

We extend our sincere gratitude to Liane Makatura, PhD student at the MIT Computational Design and Fabrication Group, for her input and guidance throughout this project.

Code available at:
https://github.com/anezuniga/grammar_truss_microstructure