Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

Kristiane Snarvold Sletten

Adopting Functional Grammar in Conceptual Design

Design Exploration in a Parametric Environment

Master's thesis in Civil and Environmental Engineering
Supervisor: Nils Erik Anders Rønnquist
Co-supervisor: Sverre Magnus Haakonsen
June 2022

NTNU
Kunnskap for en bedre verden

Kristiane Snarvold Sletten

Adopting Functional Grammar in Conceptual Design

Design Exploration in a Parametric Environment

Master's thesis in Civil and Environmental Engineering
Supervisor: Nils Erik Anders Rønnquist
Co-supervisor: Sverre Magnus Haakonsen
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

NTNU

Kunnskap for en bedre verden

Department of Structural Engineering
Faculty of Engineering
NTNU – Norwegian University of Science and Technology

ACCESSIBILITY

MASTER THESIS 2022

SUBJECT AREA:	DATE:	NO. OF PAGES:
Conceptual Structural Design	07.06.2022	8 + 62 + 15

TITLE:

Adopting Functional Grammar in Conceptual Design

Design Exploration in a Parametric Environment

Anvendelse av Funksjonell Grammatikk i Konseptuell Design

Designutforsking i Parametrisk Miljø

BY:

Kristiane Snarvold Sletten

SUMMARY:

New and advanced digital tools, like parametric design software, have positively influenced the synergy between architects and engineers. Even though the existing computational tools enhance the collaboration, it lacks the features to take advantage of the cooperation's potential in the conceptual phase. Architectural modelling tools design geometry in the absence of structural performance, while engineers' structural analysis tools need a predefined geometry to be analysed. *Functional grammar*, introduced by William J. Mitchell, aims to integrate these methods by combining design exploration and structural analysis. This thesis addresses this topic, and the approach is based on *shape grammar*. That is, shapes get modified and generated when rules are applied to an initial shape. The intention of the digital implementation of functional grammar is to achieve a generative design exploration parallel with analysing the structural performance. It involves developing rules as plug-in components for Grasshopper to perform these operations. In addition to integrating structural analysis software, other aspects can be combined with shape grammar to exploit the approach's power. For instance, this thesis adds price estimation and a prediction of CO_2 equivalents to accomplish another dimension for evaluating structures. After establishing the algorithm, a manual design exploration was executed to demonstrate the power of functional grammar as a design approach. The investigation discovered that different configurations were the most favourable based on which objective was validated the highest. Further, it illustrated the effectiveness and simplicity of the method when it already was constructed. A multi-objective optimisation aimed to minimise displacement, price estimation and CO_2 equivalents with utilisation and buckling load factor as constraints. The optimisation generated less valuable architectural language than the manual design alternatives. It requires short and efficient scripts to achieve quick and accurate computations and feasible structures. The study has demonstrated the flexibility and usefulness of shape grammars in early-stage design.

RESPONSIBLE TEACHER: Nils Erik Anders Rønnquist

SUPERVISOR(S): Sverre Magnus Haakonsen

CARRIED OUT AT: Department of Structural Engineering, NTNU, Trondheim

Preface

This Master thesis was written as a concluding part of my Master of Science degree at the Norwegian University of Science and Technology (NTNU), Department of Structural Engineering. Within five months, from January to June 2022, I have researched and written to finalise it.

During my years at NTNU, I got the opportunity to combine architecture and engineering through a Minor in Architecture. I learned how architecture and engineering have evolved throughout thousands of years and how close cooperation between the two disciplines could positively influence a design in the conceptual phase. My summer internships and courses like AAR4209 and TKT4198 have given me an insight into parametric design and its huge potential. The previous semester's project assignment concerned the conceptual design of a pedestrian bridge. I had both the structural and architectural responsibility of developing and analysing the structural behaviour in the parametric software Rhino Grasshopper. Buildings increase in complexity and are frequently given an unusual shape; hence early involvement of engineers and the usage of parametric design is essential. With this in mind, in addition to my growing interest in the field, I wanted to explore conceptual structural design further.

This master thesis would not have been possible without my advisors' feedback and guidance. PhD-student Sverre Magnus Haakonsen has been an exceptional advisor, available always; thank you! Thanks to Professor Nils Erik Anders Rønnquist for discussing different topics for this thesis and Marcin Lukowski for giving me a helping hand when needed. I would also like to thank Bunji Izumi for the help and everything I have learned from him.

The greatest thanks to my family and friends who have supported and believed in me. Not only have they been there through this last semester, but all my years with studies in Trondheim. I am grateful for all the daily conversations that have kept me calm and committed throughout the process and their contribution to discussions to further develop this thesis and me as an engineer. Thanks to my fellow students who have inspired me with their knowledge and as human beings. It has been a pleasure to learn from them and be a part of this society.

Finally, I would thank myself for these five years and for writing a master's thesis that contributes to my overall knowledge as a structural engineer.

Trondheim, 7th June 2022

Kristiane Snarvold Sletten

Abstract

New and advanced digital tools, like parametric design software, have positively influenced the synergy between architects and engineers. Even though the existing computational tools enhance the collaboration, it lacks the features to take advantage of the cooperation's potential in the conceptual phase. Architectural modelling tools design geometry in the absence of structural performance, while engineers' structural analysis tools need a predefined geometry to be analysed. *Functional grammar*, introduced by William J. Mitchell, aims to integrate these methods by combining design exploration and structural analysis.

This thesis addresses this topic, and the approach is based on *shape grammar*. That is, shapes get modified and generated when rules are applied to an initial shape. The intention of the digital implementation of functional grammar is to achieve a generative design exploration parallel with analysing the structural performance. It involves developing rules as plug-in components for Grasshopper to perform these operations. In addition to integrating structural analysis software, other aspects can be combined with shape grammar to exploit the approach's power. For instance, this thesis adds price estimation and a prediction of CO_2 equivalents to accomplish another dimension for evaluating structures.

After establishing the algorithm, a manual design exploration was executed to demonstrate the power of functional grammar as a design approach. The investigation discovered that different configurations were the most favourable based on which objective was validated the highest. Further, it illustrated the effectiveness and simplicity of the method when it already was constructed. A multi-objective optimisation aimed to minimise displacement, price estimation and CO_2 equivalents with utilisation and buckling load factor as constraints. The optimisation generated less valuable architectural language than the manual design alternatives. It requires short and efficient scripts to achieve quick and accurate computations and feasible structures. The study has demonstrated the flexibility and usefulness of shape grammars in early-stage design.

Sammendrag

Nye og avanserte digitale verktøy, som for eksempel programvarer innenfor parametrisk design, har hatt en positiv effekt på samarbeidet mellom arkitekter og ingeniører. Selv om de eksisterende beregningsverktøyene forbedrer samspillet, mangler de funksjoner for å utnytte det fulle potensialet ved et samarbeid i den konseptuelle fasen. Det betyr at arkitekter med sine modelleringsverktøy designer geometrien uten å ta hensyn til konstruksjonens bærevne, mens ingeniørenes analyseverktøy har behov for en forhåndsdefinert geometri for å utføre beregningene. *Funksjonell grammatikk*, introdusert av William J. Mitchell, har som mål å integrere metodene ved å kombinere designutforskning og konstruksjonsanalyse.

Denne masteroppgaven fordyper seg i funksjonell grammatikk som baserer seg på *formgrammatikk*, det vil si at design blir generert basert på regler som endrer på den initielle formen. Den digitale implementering av funksjonell grammatikk for å oppnå en generativ designutforskning parallelt med konstruksjonsanalyse innebærer å utvikle en programvareutividelse i Grasshopper som inneholder et sett av regler for å endre på geometrien. I tillegg til å integrere programvare for analyse, kan ytterligere aspekter legges til for å utnytte potensialet ved metoden. For eksempel, tar denne oppgaven for seg prisestimering og en prediksjon av CO_2-ekvivalenter som medfører flere dimensjoner å vurdere konstruksjonsystemer ut i fra.

Etter å ha etablert algoritmen, ble det en rekke designalternativer manuelt utarbeidet for å demonstrere potensialet til funksjonell grammatikk ved designutforsking. Studiet oppdaget at det varierte hvilke konfigurasjoner som var den mest gunstige basert på hvilket aspekt som ble validert høyest. I tillegg illustrerte den hvor effektiv og lettvint bruken av metoden er når den allerede har blitt etablert. Den multi-objektive optimaliseringen ble utført med mål om å minimere forskyvning, prisestimering og CO_2 ekvivalenter med utnyttelse og knekklastfaktor som begrensninger. Det viste seg at optimaliseringen genererte mindre verdifulle arkitektoniske uttrykk enn de designalternativer som var valgt manuelt. Det kreves korte og effektive koder for å oppnå raske og nøyaktige beregninger og gjennomførbare konstruksjoner. Dette studiet har demonstrert fleksibiliteten og nytten av formgrammatikk i tidlig designfase.

Table of Contents

1 Introduction

Advanced computer technology creates endless possibilities within the design, analysis, fabrication and shape of structures. Previously, the shape dominated the structural performance, whereas today, due to technology, complex and unusual forms are built even though they are materially inefficient. Adopting various design tools facilitates a parallel evaluation between structural performance and design generation. Bridging the gap between these approaches enables greater structural input integration during the conceptual phase.

Functional grammar aims to bridge this gap with rule application that incorporates engineering and fabrication knowledge. It produces shapes that satisfy the functional requirements, as well as realisable materials and fabrication processes. The approach is based on *shape grammar* which modifies a shape derived from rule application. Functional grammar as a design strategy involves generative design exploration (shape grammar) combined with analysing structural behaviour.

With today's numerous requirements for structures and the development of a computational architecture which embraces parametric design and form-finding, multi-objective optimisation (MOO) is relevant for solving complex design problems. An optimisation is highly relevant for a conceptual study to detect the best solution. The outcome depends on the design goals, for instance, structural performance, price and CO_2 equivalents. Parameters such as cross-section and material could be adjusted to improve the computed results. Methods like form-finding and optimisation may generate designs beyond a designer's mind.

The presented work aims to integrate and evaluate the use of shape grammars in a parametric environment. By defining example grammars with and without functional requirements, the resulting structures are analysed and compared to demonstrate the flexibility and usefulness of shape grammars in early-stage design.

This thesis is based on ongoing research within shape grammar at Conceptual Structural Design Group (CSGD), Department of Structural Engineering at NTNU. Section 4 focuses on the digital implementation of functional grammar proposed by William J.Mitchell. That is, to transform the shape rules into plug-in components. These plug-in components are used as a design approach, creating high flexibility in design exploration. Integrating Karamba to perform an FE-analysis allows the structural performance to be continuously interpreted. To further explore the opportunities within the functional grammar, scripts to calculate volume, price, and CO2 equivalents have been developed. By adding an architectural model of the geometric shapes, the thesis demonstrates a broad aspect of functional grammar. Lastly, a MOO targeting minimised displacement, price estimation and CO_2 equivalents for steel and timber structures is completed.

2 Background

2.1 Architects and Engineers

Up to about 1450, there was neither something called architects nor engineers. They went by the profession *architekton*; an ancient Greek word that can be translated into "master builder", responsible for both the artistic and technical design. The desire to build bigger and better than their predecessors have constantly required innovations (Addis, 2015, pp. 6–8). Innovations have led to the fundamental knowledge about structures that newer designs are based on and influenced by.

(a) Dome of Pantheon, Rome, Italy. Image by Heracles Kritiko/Shutterstock.

(b) As Duomo di Milano shows, Gothic cathedrals introduced spires and pointed arches. Image by Simone Simone.

(c) Felix Candela's roof design in eight hyperbolic paraboloid thin concrete shells. Los Manatials, Mexico. Image by Felipe Gabaldón

(d) Implementation of iron enable long spans for the St. Pancras station in London. Source: ('10 Things to Do at London's St Pancras', n.d.)

Figure 2.1: Impressive and innovative buildings through the years and decades.

The Greek and Roman engineers accomplished impressive masonry vaults, like the Pantheon in Rome (see Figure 2.1a). They based themselves on qualitative science, which entails assumptions from former experience (Addis, 2015, p. 60). Ambitions to build higher, more beautiful, and span further let the architecture unfold. The domes were replaced with spires and the semicircular arches by pointed arches as the Gothic era arose (see Figure 2.1b) (Popovic Larsen, 2016, pp. 41–43). The implementation of iron enabled tension members and trusses, initiating an expression of lightness due to slender and lightweight structures, known as skeleton frames. Roofs at railway stations spanned further, achieving spaces free from supports (see Figure 2.1d) (Popovic Larsen, 2016, pp. 59–67). Furthermore, the invention of reinforced concrete generated solid, three-

dimensional forms and had the ability to cantilever out beyond their supports (see Figure 2.1c) (Addis, 2015, p. 629). Unlike the traditional materials, there was no empirical knowledge related to new materials.

Aligned with the development of science, mathematics and new materials, the specialisation of the two disciplines created a clear distinction between their contribution to a project. The architect became the designer and solved the architectural issues, whereas the engineer was responsible for the technical challenges (Popovic Larsen, 2016, p. 1).

Even though there is still a definite difference between the roles and their contribution to a project, we experience a positive tendency of engineers' impact and participation in the design process. With increased focus on the benefits of cooperating in the conceptual phase, engineers are increasingly involved in early *form finding*. The parallel process of geometric design and structural analysis is possible due to the available computational tools (Popovic Larsen, 2016, p. 101). New technology, like *parametric design* software, enables complex geometric shapes and high execution, easily adaptable to adjustments. It reduces the time and costs without compromising the architectural expression or structural performance. Hence, it provokes innovative and advanced structures, resulting in a high level of design and constantly surpassing previous barriers.

Despite the roles, there are recently developed projects designed by engineers. The state-of-the-art building, Burj Khalifa in Dubai in Figure 2.2, was designed by the two structural engineers Owings and Merrill ('The Future of Structural Engineering Will Survive the High-Tech Revolution', 2016). They utilised simulation and modelling tools to rapidly analyse and iterate forms that optimised material quantities and minimised wind loads.

Figure 2.2: Burj Khalifa in Dubai, United Arab Emirates ('The Future of Structural Engineering Will Survive the High-Tech Revolution', 2016).

2.2 Conceptual Design

A building project involves several stakeholders who have various interests and influence in the different phases. With the highest uncertainty in the initial stage, identifying the risk factors like multiple design changes increases the likelihood of success. As Figure 2.3 indicates, a project is more sensitive to modification when reaching the schematic design, affecting the project costs negatively. As figure Figure 2.4 demonstrates, structural engineers gradually replace the architects' role in a project. When reaching the phase "construction documents", the architects are almost no longer involved in the project. In this traditional approach, engineers have little impact on the geometric outcome, generating less efficient design. This structural efficiency can be increased by integrating structural principles in the conceptual phase. However, it requires high commitment and anchorage to achieve significant global design decisions.

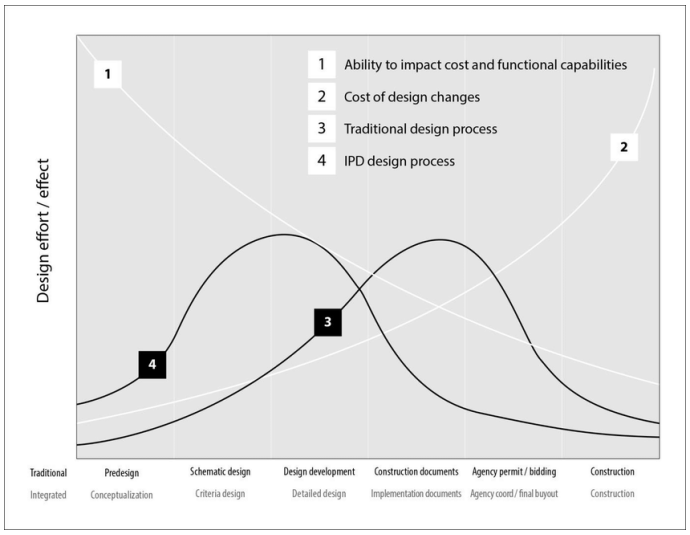


Figure 2.3: The MacLeamy Curve. With time the project becomes more sensitive of changes. Image by Saeed Talebi alternated from the American Institute of Architects (Talebi, 2014).

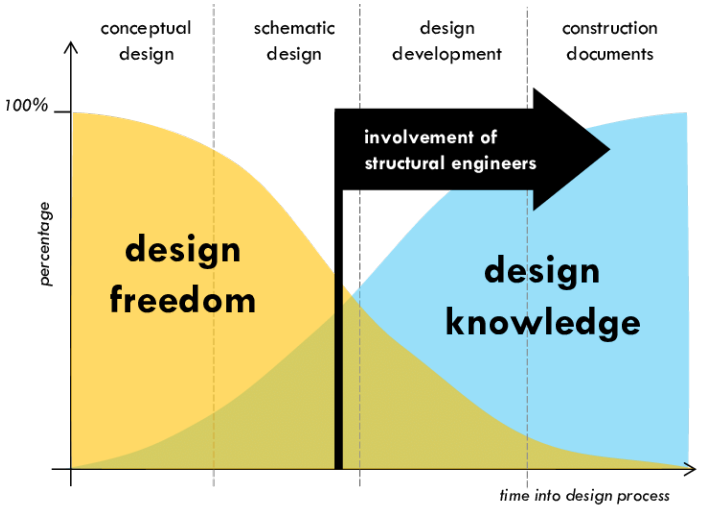


Figure 2.4: Traditionally the geometric shape are fixed in the conceptual design phase, while the structural consideration increase outwards in the process. Image by Caitlin T. Mueller.

As Mueller states, based on history, theory and nature, the form has a higher impact on the structural performance than material, member size and internal topology. It is a reasonable statement since the form distributes the forces' flow and magnitude. By solving architectural and structural problems simultaneously, there is accomplished unique and original design. In addition, it contributes to an integrated structural design that utilises material properties and natural law, rather than irrational execution (Mueller, 2014, p. 22). For instance, stave churches in Norway have an enduring structural form (see Figure 2.6). A more recent example in Figure 2.5 is the Dulles Airport Terminal by architect Eero Saarinen, where the structural form reveals the flow of forces.

Figure 2.5: Dulles Airport Terminal by Eero Saarinen. Image by MWAA.

Figure 2.6: Heddal stave church in Norway. Image by John Erlandsen. License: CC BY SA 3.0

The advanced software has positively influenced the conceptual work. Continuous generating and analysing concepts facilitate effective design exploration and *optimisation*. An enhanced and efficient structural form during the conceptual design enables the possibility to reduce material quantities, hence costs and resources (Mueller, 2014, p. 21). Thus, the conceptual phase has the potential to reduce the environmental impact of the building industry - one of the most significant challenges ahead.

2.3 Computational Design

The methods involving computational design have developed alongside its adoption. The different approaches enable complex design, and as Figure 2.7 illustrates, the terms overlap each other, which has led to confusion and inconsistent use of the terms. In the article *Computational design in architecture: Defining parametric, generative and algorithmic design*, literature from various authors is reviewed to analyse, compare and discuss the different computational design terms. It suggests an improved taxonomy on which the definitions in the following subsections are based.

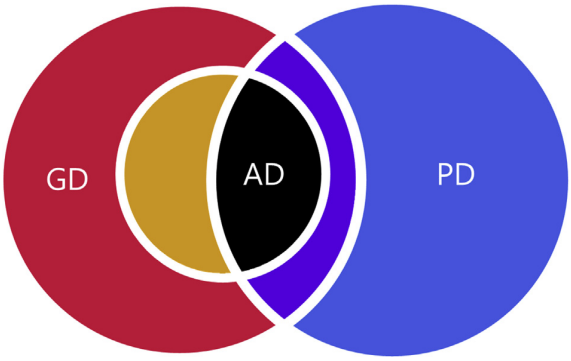


Figure 2.7: Representing the overlapping of the terms regarding computational design. Image by Inês Caetano. License: CC BY-NC-N.

2.3.1 Parametric Design

Parametric designs is defined as "...a design approach based on the use of parameters to describe sets of designs". Accordingly, it is a design that depends on its parameters to modify and generate solutions. These parameters representing geometric properties are either fixed (constraints) or variables (parameters). Hence, a restricted design domain occurs for the model to fluctuate within.

2.3.2 Generative Design

Generative design is defined as "... a design approach that uses algorithms to generate designs". Moreover, the approach is based on algorithmic or ruled-based processes that generate numerous realistic, complex design proposals (Caetano et al., 2020). Using rule-based systems, or *shape grammars*, permits design generation beyond parametric limitations (Mueller, 2014, p. 83).

2.3.3 Algorithm Design

Litterateur has an inconsistent definition of algorithm design due to overlapping parametric and generative design. However, the article defines it as "...a generative design approach characterised by an identifiable correlation between the algorithm and its outcome".

2.4 Optimisation

Design space in computational design contains all possible solutions to a problem system. *Optimisation* is a numerical method based on an algorithm to identify the most favourable solution within a given design space. However, the optimised solutions are constrained due to the problem formulation.

There are different types of structural optimisation: size, shape, and topology. Size optimisation decides the cross-section based on predetermined geometry and element configuration, whereas shape optimisation determines the overall structural form (Mueller, 2014, pp. 35–38). The last

class, typology optimisation, aims to maximise the structural performance by generating the optimal connective arrangement of elements in a structure. This type can also be combined with size and shape optimisation. These different optimisation types are illustrated in Figure 2.8

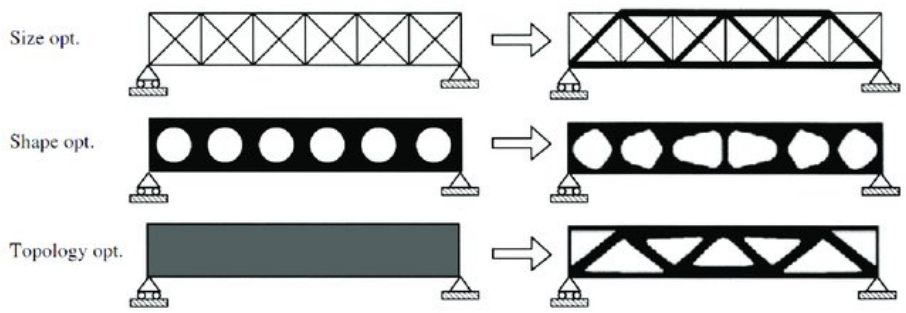


Figure 2.8: Comparative illustration of size, shape and topology optimisation (Gebisa & Lemu, 2017). License: CC BY 3.0.

As a practical approach, physical models have been used to explore structural forms. Antoni Gaudi was one of the first to implement the method in conceptual structural design when designing the Santa Coloma church (Popovic Larsen, 2016, pp. 84–88). He achieved the most efficient structural form for the particular load case using graphic statics and inverting funicular models. His architecture reflects his inspiration by materials, geometry and configuration of forces, resulting in several impressive buildings with a high complexity of form; the private residence churches, Park Quell, and the unfinished Sagrada Familia. Today, these forms can be achieved through a digital form-finding process. Even though several methods exist, the process' essence is the same: to find the best shape related to the design goals.

Work performed before the computational era pushed the limits of materials. Currently, other parameters influence the complexity of generating environmental and sustainable design. As a result of higher demands for newer constructions than previously, further requirements emerge to satisfy aspects such as climatic conditions and material quantities. Respectively it is necessary to perform a multi-objective optimisation (MOO) to connect multiple parameters to obtain the best possible solution (Popovic Larsen, 2016, p. 101). In conceptual studies, parametric design and MOO are often utilised to conduct the form-finding process and structural optimisation.

A recent construction which has reduced material usage by performing structural analysis optimisation is the innovative pedestrian bridge Striatus. The 3D-printed pedestrian bridge was designed by the architecture firm Zaha Hadid (Figure 2.9). The bridge is made from hollow concrete wedges (Figure 2.10), where members are held together by gravity and compression ('Striatus 3D concrete printed masonry bridge', n.d.). By combining ancient building knowledge with computational design, both reinforcement and mortar have been eliminated.

Figure 2.9: Striatus bridge assembled and in use. Image by naaro

Figure 2.10: The hollow wedge shows the reduction of concrete. Image by Alessandro Dell'Endice.

3 Shape Grammar

This section first introduces shape grammar as an approach to generate design and its historical evolution. Thereby, *Functional Grammar: An Introduction* published by William J. Mitchell (J. Mitchell, n.d.) and labelled shape are defined. In addition, the software tools used for this thesis to demonstrate the advantage of shape grammar are presented.

3.1 Defining Shape Grammar

As mentioned earlier, the architect typically defines a design space, and then the engineer has a restricted domain to work within. In comparison, shape grammar involves geometric operations and transformations to generate design beyond this topology set by the architect. These geometry modifications are based on a set of rules applied recursively, replacing subshapes with new shapes (see Figure 3.1). Hence, it contributes to a diverse and wide range of design options. It is both a descriptive and generative approach ('Introduction', n.d.). The rules themselves describe potential forms to generate, and the rules generate the designs. This design generation involves operations such as addition and subtraction and spatial transformations - shifting, mirroring and rotating - to the shape. Shapes are defined by points, lines, planes or volumes.

Shape Grammar is a spatial algorithm that treats shapes as nonatomic entities ('Introduction', n.d.). That means that the designer has the liberty to decompose and recompose them as wanted. Another aspect of shape grammar is that it is nondeterministic. The user may have several rules to choose from, how to apply them, and when they are applied in the computation process affects the output.

3.2 The Evolution of Shape Grammar

A world-class structural engineer is capable of brainstorming a range of creative concepts within a typology or system and then, by intuition, estimating their performance (Mueller, 2014, p. 80). However, it is not a process that can guarantee the omission of bias and human errors, and this vulnerability makes computational operations valuable to explore beyond topological boundaries. Additionally, computational operations enable a comparison of designs across typologies.

After Stiny first proposed *shape grammar* as a strategy to generate unlimited and unexpected design ideas, Koning and Eizenberg presented a grammatical study of Frank Lloyd Wright's prairie house. A study that emphasising the significance of a rule-based approach (Mueller, 2014, pp. 83–84). Figure 3.1 shows the potential of how simple rules can generate increasingly diverse and complicated forms.

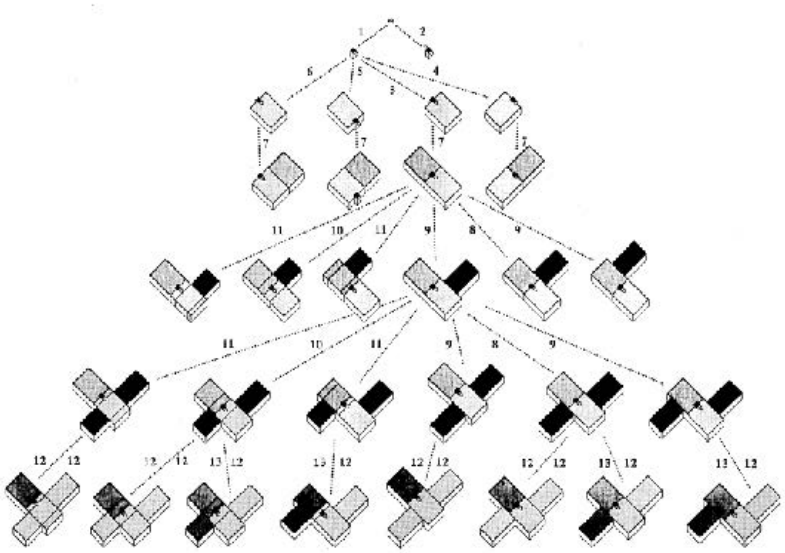


Figure 3.1: A selection of the possible rules applied in the grammatical study of Frank Lloyd Wright prairie house (Koning & Eizenberg, 1981).

Furthermore, William J. Mitchell introduced *functional grammar*, see Figure 3.2. The approach includes rules which consider engineering and fabrication knowledge. He meant that these requirements were necessary conditions for a solution to a design problem. Together with Cagan, he combined grammar with performance goals. Resulting in the computational technique, shape annealing; a stochastic simulated annealing optimisation process to generate approximate optimum rule-based shapes (Mueller, 2014, p. 86).

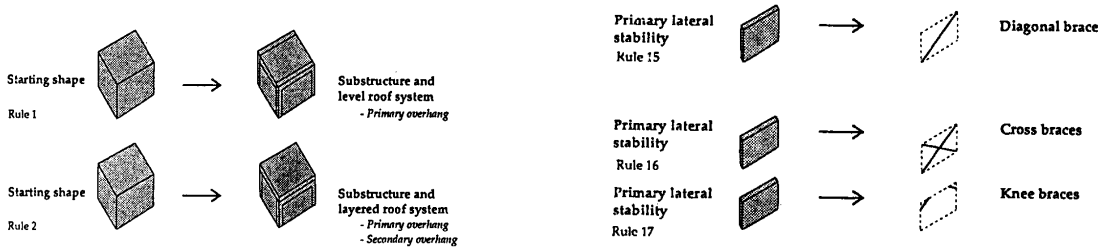


Figure 3.2: Mitchell was the first to use rules application. The images shows a small section of the rules proposed (J. Mitchell, n.d.).

Cagan then teamed up with Shea to further developed the shape annealing approach (Mueller, 2014, p. 86). They applied the method to truss structures and achieved a wide range of high-performing designs within a limited problem domain. It demonstrated that shape annealing is most suitable after the conceptual design, once the structural typology is set.

Geyer's recent progress within functional grammar involved combining it with multidisciplinary optimisation, see Figure 3.3. His study of a planar gravity and lateral frame for a large hall resulted in trans-typological designs. Even though the approach does not take full advantage of grammars due to slightly flexible rules, the optimisation makes the approach valuable for comparing predetermined designs after the conceptual phase.

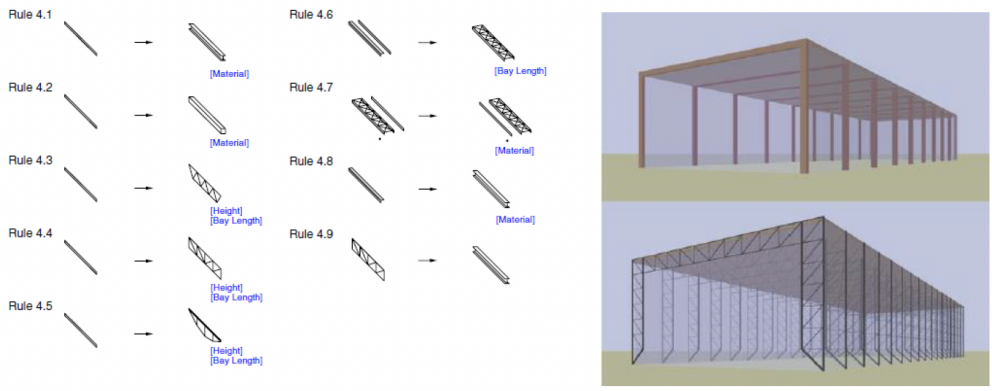


Figure 3.3: Sample of rules and a structure due to multidisciplinary optimisation of functional grammar (Geyer, 2008).

In a later study, Caitlin Mueller introduces the term *trans-typology grammar*. Her approach involves three types of computational classes: shapes, grammars and analysis engines (Mueller, 2014, p. 88). A shape class contains data or properties that include geometric information, as well as internal organisation and hierarchy. While a grammar class contains a list of rules that can be applied to and modifies particular geometric objects. The analysis class can provide a performance score based on a particular geometric object's properties.

3.3 Functional Grammar

In 1991 William J. Mitchell published a study about Functional Grammar. The previous section gave a short introduction to his concept. However, a more in-depth embroider is necessary to understand the essence of this thesis. Therefore the following sections concerns the rules and his terms *terminals* and *markers*.

3.3.1 Terminals and Markers

Mitchell operates with the terms *terminal vocabulary Vt* and *marker vocabulary Vm* (J. Mitchell, n.d.). He points out that they could be differentiated through a labelling scheme. However, that labels not gives a clear distinction between the two. He defines markers as following:

> *"Markers have interface requirements and functions. The interface requirements are represented diagrammatically by boundaries and connection points, details of the element's shape and internal organisation are left unspecified. (Thus a supporting element marker might be described by the end points of its central axis, a room marker by its outer boundaries, and so on.) The functions are presented by input and output parameters, plus partial specifications of the mappings from inputs to outputs. Typically, such as partial specification is a mathematical expression containing terms with unknown values."*

While the definition of terminals is:

> *"Terminals has known geometry and behaviours. (That is, it can be drawn in complete detail, and the expression mapping from input to outputs do not contain terms with unknown values.) The functional description of a terminal represents empirical knowledge of how this type of element actually behaves. This knowledge may be complete or incomplete, approximate or accurate, correct or incorrect. It can be substantiated or disproved in the usual ways - by building and testing a prototype, for example. "*

3.3.2 Rules

Functional grammar is a rule-based approach, either specified as a top-down refinement, bottom-up assembly or a combination of them both (J. Mitchell, n.d.). Although the determination of the design is controlled differently, both rules convey the same knowledge. For instance, in Figure 3.4 the beam converts a point load into two point loads at each beam end and then to the columns, which transfers these loads to the ground. Thus, a bottom-up design begins with the known (terminals) and determines a subsystem's assembly with the desired behaviour. Instead of beginning the process with terminals, top-down rules start with a combination of markers. Hence, a general goal including constraint of geometry, interfaces, inputs and outputs of the initial shape in Figure 3.5 results in an assembly of columns and a beam. For top-down refinement, there is a special case, the termination rule. Those rules replace markers with terminals or combinations of terminals.

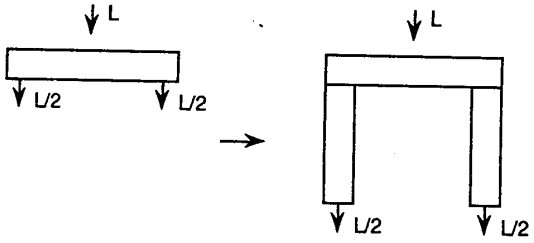


Figure 3.4: A bottom-up assembly rule. Image by William J. Mitchell.

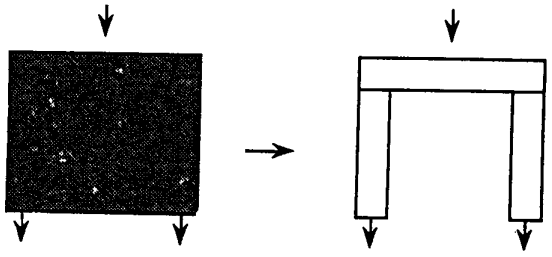


Figure 3.5: A top-down refinement rule. Image by William J. Mitchell.

As rules are applied, the initial structural shape is modified due to geometric operations and transformations, and structural properties. Recursive rule application leads to an infinite number of sequences to conclude unique designs.

3.4 Labelled Shapes

Even though Mitchell omits the use of labelled shapes, both Stiny and Mueller adopt it to define the language of shapes directly in terms of labelled and parameterised labelled shapes. A state label enables control of the order of the various rules that can be applied to a structural shape (Mueller, 2014, pp. 90–91). As Mueller specifies in the trans-typology structural grammar approach, a structural shape always has a particular state. When a rule can be applied depends on the state of the current structural shape, and when it is applied, it changes the state of the structural shape, which affects what rules can be subsequently applied. However, it can also remain in its current state. In short, a rule is restricted to either one or more specific states that dictate the following shape, rules and state. A state can be related to several rules and vice versa. Further, a shape can return to a previous state.

Stiny has the following precise interpretation of labelled shapes when he defines shape grammar (Stiny, 1980):

- R is a finite set of *shape rules* of the form $\alpha \rightarrow \beta$, where α is a labelled shape in (S, L)$^{+}$ and β is a labelled shape in (S, L)*
- I is a labelled shape in (S, L)$^{+}$ called the *initial shape*

Likewise, this thesis combines labelled shapes with Mitchell's theory of functional grammar. The states are specified in Figure 3.7 and Table 4.3. Establishing these state labels is valuable since it creates a general order for rules to be applied. Although, the implementation of state labels is simplified compared to Mueller's complexity of relations and actions. That is, the order depends on the states for the given rules. Each rule is related to a specific state and is restricted to a particular consecutive rule.

3.5 Software Tools

In this thesis, the following software has been adopted for geometric design generation, in addition to structural analysis and optimisation operations.

Rhinoceros 7

Rhinoceros 3D, also known as Rhino, is developed by Robert McNeel Associates (Associates, givenun=0, n.d.). It is a Computer-Aided-Design (CAD) software developed to produce complex geometry. The geometric design in Rhino is done manually, which makes it less adaptable to changes.

Grasshopper

Grasshopper is a visual programming language connected to Rhinoceros as its visual studio (Network, n.d.), see Figure 3.6. It is a parametric software that allows easy real-time geometry manipulation through changing parameters. The language is based on dragging components into the canvas and connecting them through wires. A component receives input from the previous component, performing functional operations and then distributing outputs. Then the algorithm will automatically transform into a digital geometric model in Rhino. In addition, the software allows downloading plug-ins or packages for a more sufficient and compatible design. Further, the scripting of additional plug-ins enables complex projects.

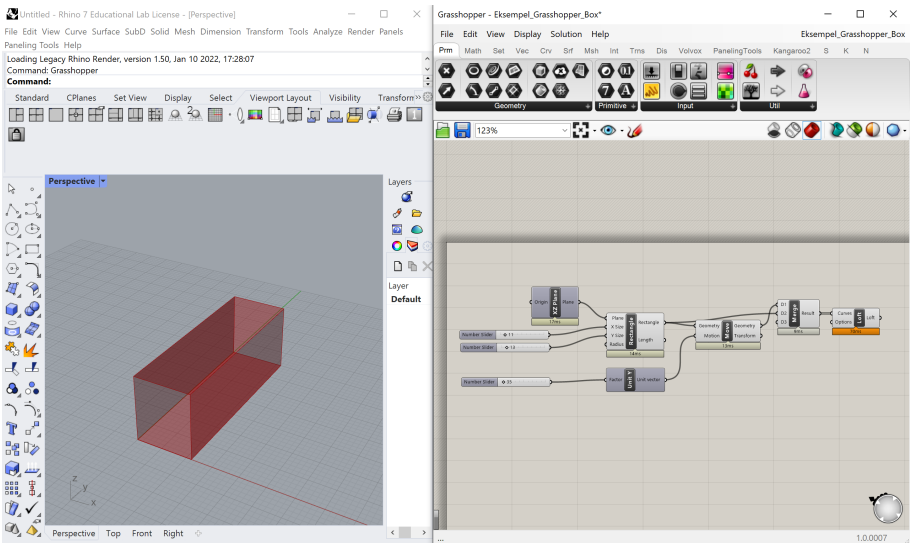


Figure 3.6: A box created in Grasshopper (right) and automatically viewed in Rhino (left).

Karamba 3D

Karamba 3D is a parametric structural engineering tool that can be downloaded and integrated into Grasshopper ('Karamba3D – parametric engineering', n.d.). The plug-in easily combines parametric design with finite element analysis (FEA) and optimisation. It provides accurate calculations of spatial trusses, frames and shells. For this thesis, Karamba will be adopted to evaluate the structural performance of the structure.

Octopus

Octopus is a MOO tool that allows searching for many goals at once and introduces the Pareto principle for multiple goals ('Octopus', 2012). The Octopus component can connect to multiple parameters or *genes*, which can either be NumberSliders or GenePool components. Octopus will then explore possible solutions by varying the genes within the individual range settings. These genes sets the design space. The *phenotype* is the solution mesh from the parametric modelling. Octopus expect both a number parameter and a text parameter. The number parameter contains the numeric objective values, that is, the fitness values of the solution. It requires a minimum of

two, and the maximum is theoretically unlimited. The text parameter includes textual objectives descriptions, that is, short names describing the objective values. Constraints can be ensured by implementing a boolean parameter. Octopus expects a "true" value for valid solutions.

Visual Studios and C#

Visual Studio is an integrated development environment (IDE) from Microsoft, a code editor for C Sharp (C#) development ('Visual Studio', n.d.). C# is an object-oriented programming language, and the software will be used to develop Grasshopper components to enable structural grammar operations.

Simple Grammar

Simple Grammar is a plug-in developed by Sverre Magnus Haakonsen for an ongoing project within shape grammars at Conceptual Structural Design Group (CSDG), Department of Structural Engineering, NTNU. The corresponding Visual Studio project contains both classes and components related to shape grammar (Haakonsen & Izumi, 2022). The UML diagram in Figure 3.7 presents the required classes and their relationships.

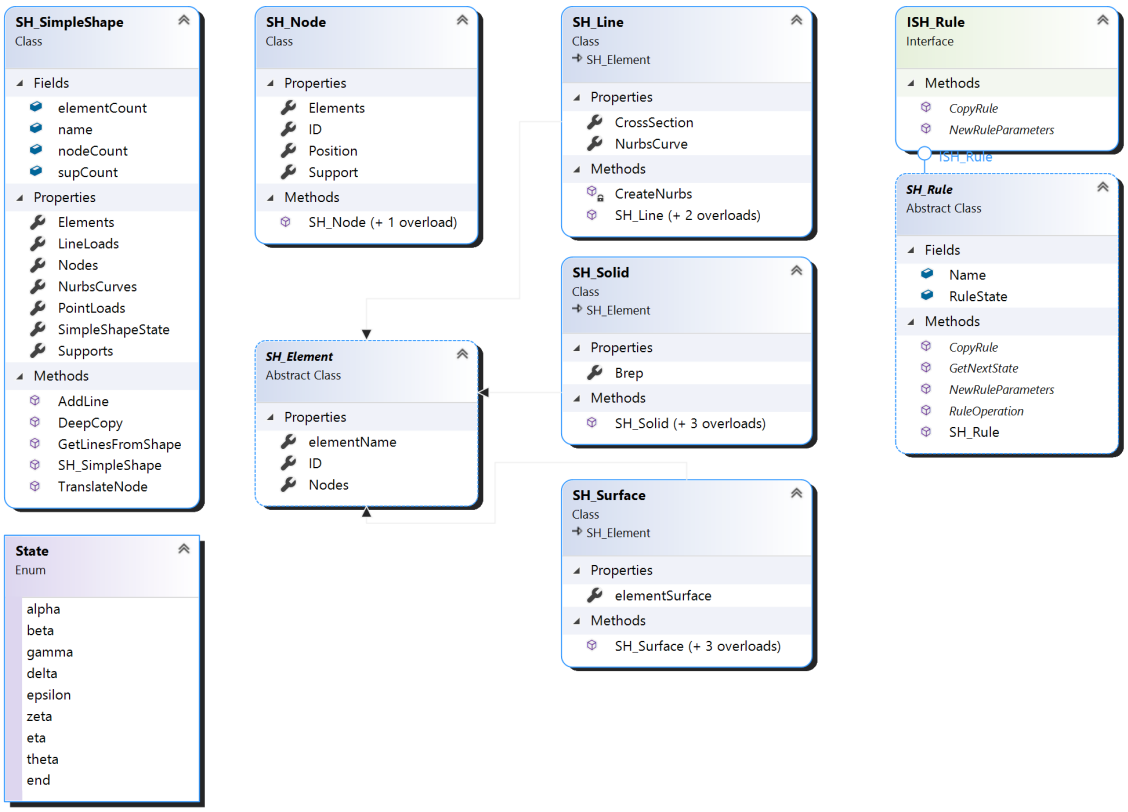


Figure 3.7: UML diagram for SimpleGrammar.

The following existing component is adapted in this thesis.

GrammarInterpreter

The rule application depends on the *GrammarInterpreter* component. As figure 3.8 shows, the component has the *Simple Shape* (initial geometry) and *Rules* as input, and outputs the *Modified Shape*. This output is an external class, thus not Rhino geometry. Hence, assembling the model requires an additional code before it is displayed in Rhino.

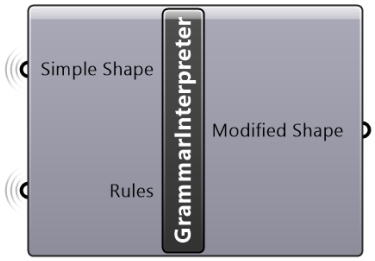


Figure 3.8: This component modify the Simple Shape due to applied rules.

4 Digital Implementation of Functional Grammar

The first part of this section undertakes the functional grammar proposed by William J. Mitchell by transforming his theory into Grasshopper components. The study examines the top-down refinement rule, which was defined in Section 3.3.2. Further, the concept is illustrated by showing the step by step process behind one structural configuration, including a volumetric model. The volumetric model is further described in the last sub-section.

4.1 Mitchell Rules

This section will focus on the different rules developed to transform the initial geometry into its final and modified form. The subsections will describe the function of each Mitchell Rule component which inherits the properties from abstract class *SH_Rule* in *SimpleGrammar*. This relationship is shown in Figure 4.1, including details behind each class. Other classes and their properties and methods that have had an impact on the codes are displayed in Figure 3.7.

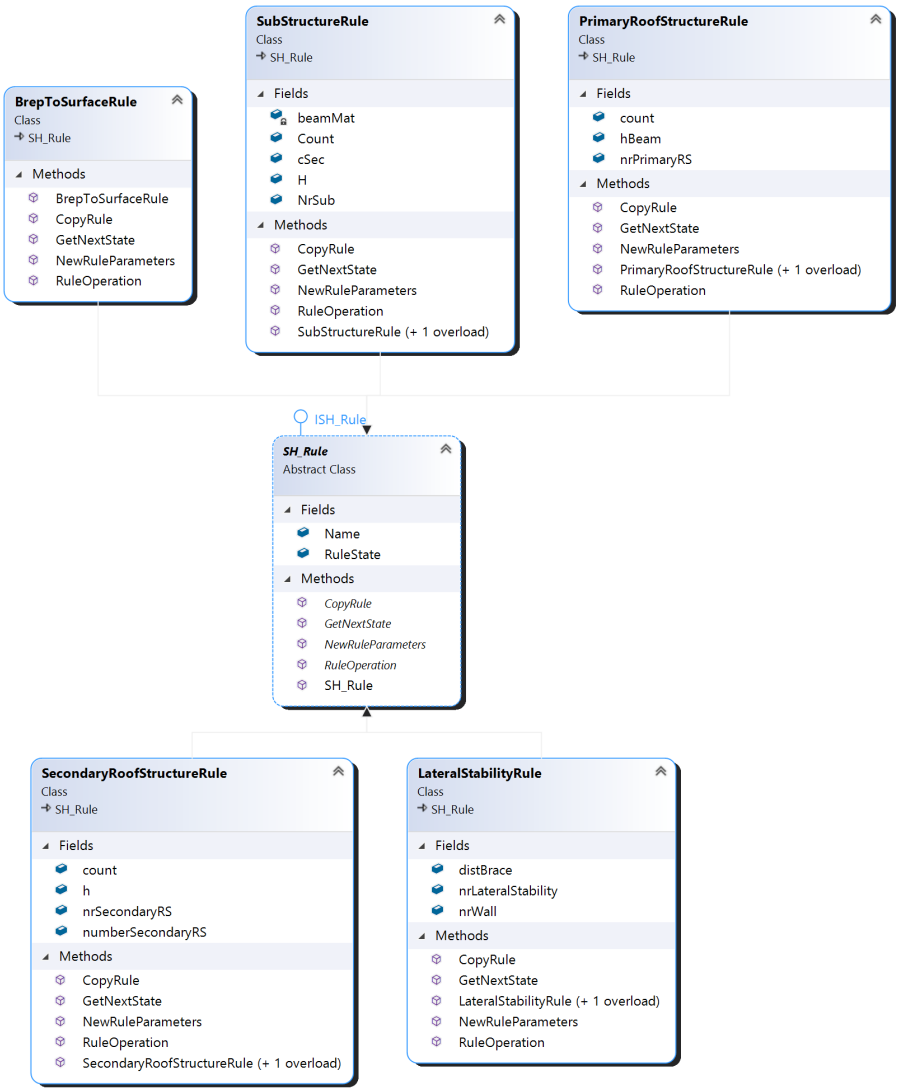


Figure 4.1: UML diagram for Mitchell Rules.

To differentiate between the two framework's terminologies, will the developed rule components be referred to as "Mitchell Rule #", while the rules from functional grammar by J. Mitchell with only "Rule #". Table 4.1 provides an overview of this relation, as well as a short description of the plug-in components. Since the Rules are merged into fewer Mitchell Rules, there is necessary with a numerical value to represent the different structures. The structural elements and their input value to select and generate the design is presented in Table 4.2. In some cases, the name includes a number representing the input value to generate the corresponding structure. Those cases without a number have only one corresponding structure.

Table 4.1: The table illustrates the relationship between the developed components and the rules proposed by J. Mitchell.

Functional Grammar by William J. Mitchell	**Component name**	**Description of the component**
	Mitchell Rule 1	Deconstruct volume into surfaces.
Rule 1 Rule 2 Rule 3 Rule 3	Mitchell Rule 2	Generates substructure.
Rule 26 Rule 27 Rule 28 Rule 29	Mitchell Rule 3	Generates primary roof structure.
Rule 9 Rule 10 Rule 11 Rule 12 Rule 13 Rule 14 Rule 15	Mitchell Rule 4	Generates secondary roof structure.
Rule 15 and 20 Rule 16 and 21 Rule 17 and 22 Rule 18 and 23 Rule 19 and 24	Mitchell Rule 5	Generates lateral stability.

Gathering the Rules that belong to the same stage of the process into one component, for instance, deciding the primary roof structure or the bracing, reduces the amount of Mitchell Rules. Having one component that includes several structural elements makes the model more adaptable when the designer wants to change the structure. Rather than replacing components, the designer only

adjusts the parameters, hence the design. It contributes to a more user-friendly interface and is less time-consuming.

Instead of operating with volumes as William J. Mitchell, this study is based on mainly nodes and lines. Processing these geometric properties simplifies the geometry, making it adaptable for both a visual and analytical model. This is preferable when working with functional grammar. This inequality is also noticeable in Table 4.1 where Rule five to eight is not listed since they are volume operations that get simplified through the use of lines. Likewise, for Rule twentynine and thirty, which is incorporated in Mitchell Rule 5.

Table 4.2: The table presents the different structural elements proposed by J. Mitchell. The elements' names in the plug-in are listed in column two.

Functional Grammar by J. Mitchell	**Corresponding name and input value**
Substructure and level roof system	Substructure 0
Substructure and layered roof system	Substructure 1
Substructure and pitched roof system	Substructure 2
Substructure and bowed roof system	Substructure 3
Truss	Primary Roof Structure 0
Beam	Primary Roof Structure 1
Pitched Truss	Primary Roof Structure
Bowed Trusses	Primary Roof Structure
Pitched Trusses	Secondary Roof Structure 0
Bowed Truss	Secondary Roof Structure 1
Beams	Secondary Roof Structure 2
Flat Trusses	Secondary Roof Structure 3
Joists	Secondary Roof Structure
Joists	Secondary Roof Structure
Diagonal brace	Lateral Stability 0
Cross braces	Lateral Stability 1
Knee brace	Lateral Stability 2
Shear wall	Lateral Stability 3
None	Lateral Stability 4

As mentioned in Section 3.4, labelled shapes are adapted in this thesis. The order for rules to be applied coincides with the order in Table 4.1. Furthermore, the state labels for the Mitchell Rules are specified in Table 4.3.

Mitchell Rule #	State Label
1	α
2	β
3	γ
4	δ
5	ε

Table 4.3: State labels for Mitchell Rules.

Geometry

The initial volume is constructed in Rhino and referenced to a brep container in Grasshopper. The dimensions of the box illustrated in Figure 4.2 set the width, length and minimum height. When rules are applied to *GrammarInterpreter*, the geometry gets modified based on the instructions from the rules. However, for the initial shape to be responsive to these modifications, it needs to be converted into a *SimpleShape* class before it acts as an input. This was accomplished through the code in Listing 1, which was written in Grasshopper's C# component.

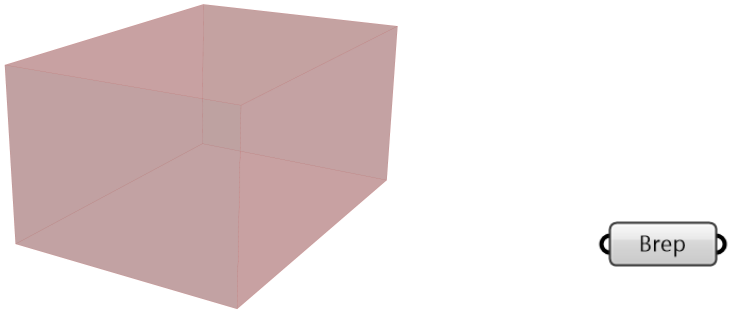


Figure 4.2: The initial geometry before rule application.

```
Brep b = RhinoGeometry;

    SH_SimpleShape ss = new SH_SimpleShape();

    ss.Elements["Solid"] = new List<SH_Element>();
    SH_Solid s = new SH_Solid();
    s.Brep = b;
    s.elementName = "Brep";
    ss.Elements["Solid"].Add(s);

    SimpleShape = ss;
```

Listing 1: The code converts Rhino geometry into Simple Shape geometry.

Mitchell Rule 1

The component displayed in Figure 4.3 requires no additional input since it only processes the *SimpleShape* through the *GrammarInterpreter*. The initial box gets deconstructed into six surfaces as this rule (MRule1) is applied. Each surface is thereby assigned an element name and stored in *SimpleShape*. *SimpleShape* in Figure 3.7 shows that *SH_SimpleShape* store a property named *Elements*. That is a dictionary which stores a list of SH_Elements as values, with the key "Surface". A *SH_Element* has the element name and geometry as properties. By evaluating the Z component of the normal vector for each surface, their element name is either set as top, longest wall, shortest wall or bottom. The shortest walls are equivalent to the transversal walls of the box, and the longest walls are the longitudinal walls.

Even though the component does not correspond to a specific Rule in functional grammar (J. Mitchell, n.d.), it produces necessary elements for further rules to be based on. The element names contribute to the correct selection of surfaces to modify in Mitchell Rule 2.

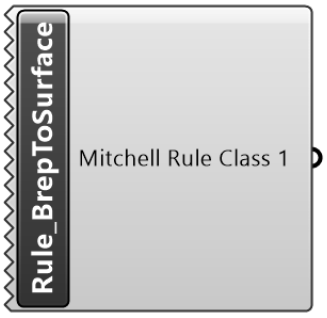


Figure 4.3: Mitchell Rule 1 (MRule1) deconstruct the volume into surfaces.

Mitchell Rule 2

The Mitchell Rule 2 (MRule2) component in Figure 4.4 addresses Rule one to eight. However, Mitchell originally converts the surfaces into volumes rather than lines and nodes that this developed component will generate. The rationale for this is to simplify further modifications and later structural analysis.

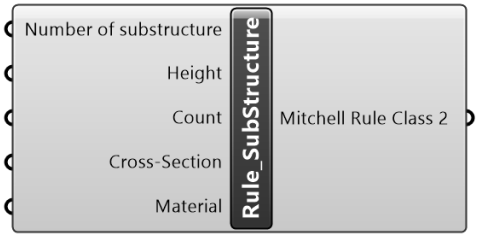


Figure 4.4: This component generate lines and nodes for each possible substructure.

The component has an input named *Number of Substructure*, which requires a number between zero and three. Based on this number, the GrammarInterpreter generate the corresponding substructure (shown in Figure 4.5 and Table 4.2). It produces a set of lines and nodes that further rules will depend on. Through the input *Height* the incline of the pitched roof of substructure 2 is decided, likewise for substructure 3 where it determines the height of the bowed roof. In addition, this substructure requires a discretization of the arch. Thus, the input *Count* is added to control the number of segments the arch is divided into. If the number is odd, the code converts it to an even number by adding one. This applies to all developed components that include "Count".

Default values are set as zero for the substructure, one for the height and six for the count. Substructure 0 and 1 have similar outputs, except for the element name, which is linked to the substructure number. Subsequent rules will generate different structural outcomes for these two.

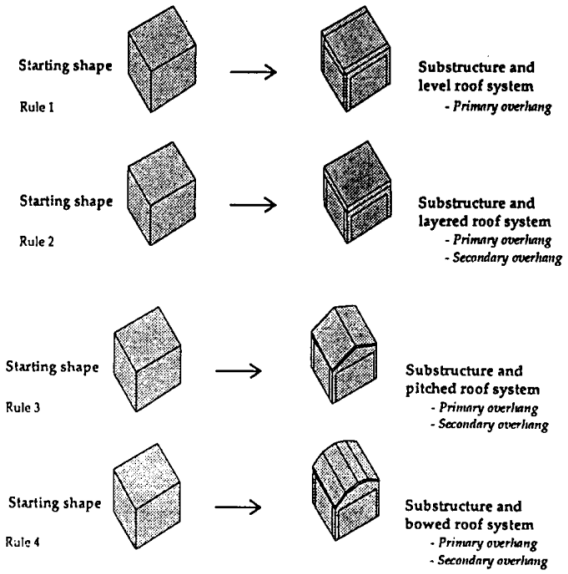


Figure 4.5: The substructures identified by J. Mitchell. From the top, substructure 0 is followed by substructure 1, substructure 2 and at the bottom is substructure 3. Image by William J. Mitchell.

The different objects are stored in the *Elements* dictionary, but with the key now set as "Line". The lines correspond to columns, transversal and longitudinal beams implied by the element name. In addition, each object has an id and two nodes for line construction. For substructures 0 and 1, there will be eight nodes, while substructure 2 has additional two nodes (top nodes for the pitched roof). The additionally amount of nodes for substructure 3 depends on the value for *Count*.

Further, at this stage, the name of the material and cross-section are selected, hence the inputs *Material* and *Cross-Section*. Their default values are respectively "timber_C20" and "200x200" [mm x mm]. These properties are added to all subsequent generated elements.

Mitchell Rule 3

Figure 4.6 shows Mitchell Rule 3 (MRule3) determines the primary roof structure, that is, Rule twenty-five to twenty-eight. However, the configurations have some limitations since the substructure from MRule2 omits some combinations. For instance, the bowed roof, equivalent to substructure number 3, dictates that the primary roof structure must be a bowed truss. Even though Mitchell only proposed four structures, there is a huge variety of possible truss systems in reality.

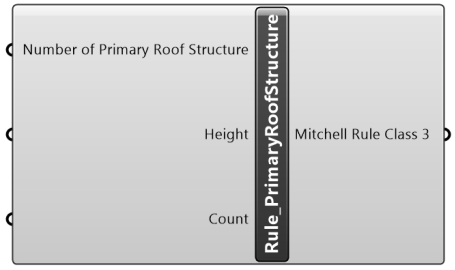


Figure 4.6: Primary Roof Structure are constructed when this component is applied to Grammar-Interpreter.

The component has three inputs; *Number of Primary Roof Structure*, *Beam Height* and *Count*. The election of the primary roof structure is relevant for substructures 0 and 1, likewise for Beam Height. The last input determines the number of segments for the truss system for all four substructures. The default value is set as zero for the primary roof structure, while the default value for height and count is equivalent to one and four.

Mitchell Rule 4

The component Mitchell Rule 4 (MRule4) in Figure 4.7 addresses the election of the secondary roof structure. Initially, there are six secondary roof structures stated as Rule nine to fourteen, but the feasible selection depends on the substructure and primary roof structure. Like previous, this process is applied through input with the name *Number of Secondary Roof Structure*. Additional inputs are secondary truss *Height* for pitched and bowed structures and *Count* which decides the

number of secondary structures. There is also an input called *Amount of Secondary Roof Structure* which represents the number of joists and beams as the secondary roof structure. Zero, one, six and two are respectively the default value for the secondary roof structure, height, count and amount of secondary roof structure.

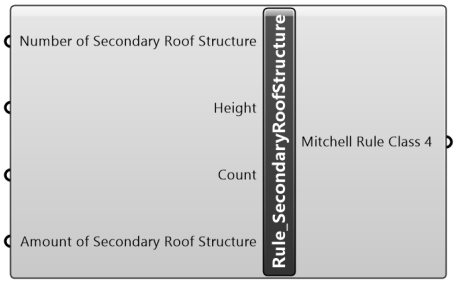


Figure 4.7: Mitchell Rule 4 modify the current shape by adding secondary roof structure.

Mitchell Rule 5

Rules fifthteen to twentyfour manage primary and secondary lateral stability and are merged into one component, Mitchell Rule 5 (MRule5) showed in Figure 4.8. *Number of Lateral Stability* selects sort of bracing. In addition, the component includes an input *Number of Wall* for which wall to brace. Both default values are set to zero. Wall zero and one correspond to the transversal walls, while two and three are the longitudinal walls. To be able for multiple walls to be braced, the inputs take in a list of values. The last input *Distance Corner* is only applicable for knee braces (Lateral Stability 2), and it decides the distance from the top corner that the knee brace will intersect with the top beam and column.

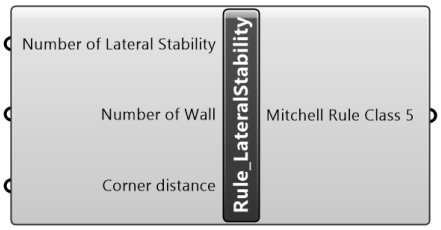


Figure 4.8: The component adds lateral stability to the structure. All four walls can be braced.

The shear wall is modelled as a surface to match the necessary input in Karamba. Different solutions for modelling the knee braces have been evaluated. Since shape grammar involves design generation, the bracing has been modelled as a part of the component. That implies that the column and beam intersecting with the bracing element get cut into two elements.

The selection of lateral stability affects the feasible material options for the columns. For instance, only masonry columns are sufficient if no lateral stability is chosen, i.e. Lateral Stability = 4. Hence, this component also restricts the bracing options based on the columns' material.

4.2 Additional Plug-ins

There are developed two components to extract the necessary elements to be viewed in Rhino. One generates the geometry for the structural analysis, while the other converts the geometry into a volumetric model.

CreateGeometry

As mentioned previous, the modified shape from *GrammarInterpreter* is an external class called SimpleShape. Therefore the geometry will not be displayed in the Rhino viewport when generated. The *CreateGeometry* component in Figure 4.9 solves this by converting the SimpleShape geometry into Rhino geometry - Lines, Surfaces, and Points that are equivalent to the output "Support". These three geometric properties are necessary for structural analysis in Karamba.

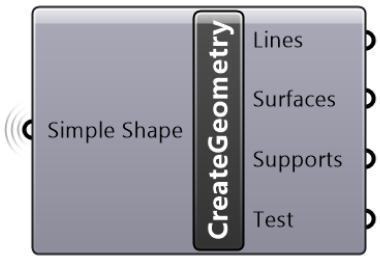


Figure 4.9: *CreateGeometry* convert and supply with necessary geometry.

CreateVolume

To generate a volumetric model, the *CreateVolume* in Figure 4.10 had been developed. The single input is the Simple Shape from the *GrammarInterpreter* which contains information about the cross-section and material, as well as the geometric elements. Based on these properties, the component generates volumes, which are breps displayed in Rhino.

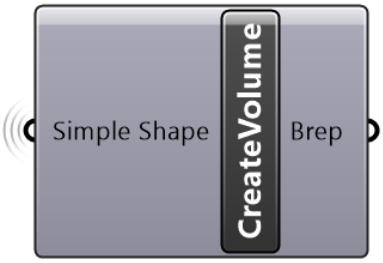


Figure 4.10: *CreateVolume* constructs the model into a volume based on the stored element data.

4.3 Structural Configurations

The flow chart in Figure 4.11 illustrates a structural configuration produced by applying rules to the initial geometry. The initial geometry gets deconstructed into six surfaces when applying MRule1. Further, the pitched roof, Substructure = 2, and its height equal to 1.5 meters are applied by MRule2. This choice of substructure will narrow down selection further. MRule3 request an input value for Count to decide the number of segments for the truss system. This integer is set to six and serves as the lower limit for the number of joists. When applied, the quantity can be regulated in MRule4, which adds a secondary roof structure. For this particular structure, the number of the secondary roof structure is three, which means that each segment from the primary roof structure is divided into a division of three. Lastly, lateral stability is generated twice and chosen both times are the cross-bracing - Lateral Stability = 1 and longitudinal wall.

Additional example configurations are to be found in Appendix A.

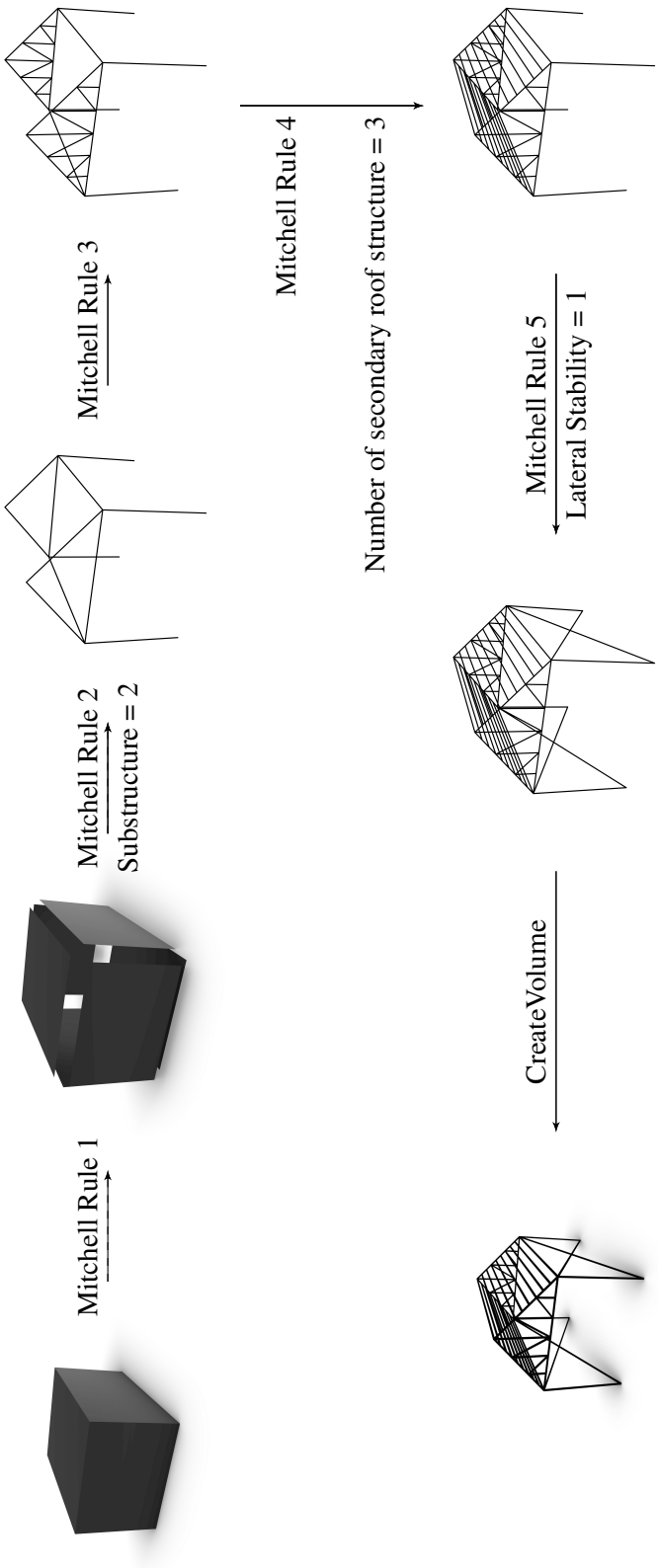


Figure 4.11: Illustration of the digital modification process that generates the same example configuration as Mitchell referred to.

4.4 Volumetric Geometry

Since functional grammar should fulfil architectural purposes in addition to structural ones, each SH_Line was assigned a cross-section and material when constructed. The volumes generated are based on this data.

Substructure 0 and 1 have the same structural model due to simplification. However, the visual model has some distinct differences in beam-column connections. Substructure 0 has a levelled roof system with a primary overhang, while substructure 1 has a layered roof system with primary and secondary overhang. This difference is illustrated in Figure 4.12

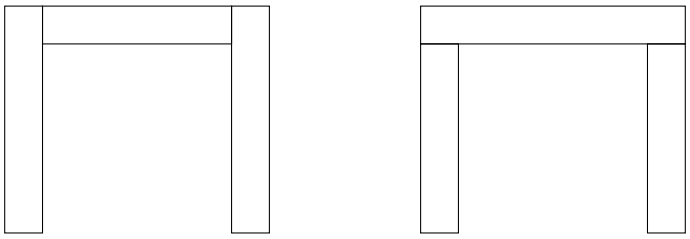
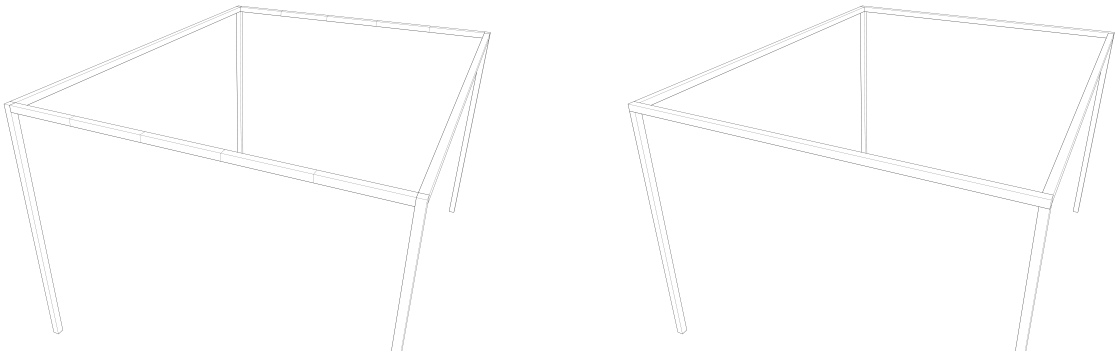


Figure 4.12: The figure illustrates the volumetric differences between substructure 0 and 1. The figure to the right shows the levelled roof system for substructure 0, while to the left is the layered roof system for substructure 1.

Figure 4.13 show the entire structure as a volume after Mitchell Rule 2 is applied.



(a) Substructure 1 **(b)** Substructure 1

Figure 4.13: The volumes for substructure 0 and 1 after applying Mitchell Rule 2. The beam-column connection differs from the two substructures.

5 Structural Analysis

By connecting shape grammar to a structural analysis software like Karamba, the potential of functional grammar as a design strategy is exploited in the conceptual phase. Hence, this chapter will focus on establishing an analytical model. First, the loads acting on the structure are introduced, and these loads are relevant when the following sub-section establishes the Karamba model used to investigate the structural behaviour.

5.1 Loads

Self Weight

Karamba 3D automatically calculates the self-weight for the structure, and it will vary with the type of material and construction. The self-weight from the joints is neglected.

Live Load

The designed model contains a roof structure, bracing and columns without additional floors. Hence, the live load only includes maintenance work on the roof, which will be neglected at this design stage.

Snow Load

The snow load can be calculated according to NS-EN 1991-1-3, where factors for the roof design and location affect the value. Table NA.4.1(901) presents the characteristic value for snow load at terrain for all counties in Norway. For instance, Trondheim could be used as a reference, which entails a snow load equal to 3.5 kN/m^2.

Wind Load

According to NS-EN 1991-1-4, the wind load depends on several factors. For example, the design of the roof, location, et cetera. However, the structure in this thesis has no intended place. Therefore it is conservative to assume the wind load to be 0.7 kN/m^2, without no further advanced calculations.

5.2 Establishing Analytical Model

Karamba is adopted to investigate the structural performance of the geometric outcome. This section focus on establishing the model to perform the FE-analysis. It is necessary to define the different structural elements like shells and beams, material, cross-section, loads and supports. These properties are assembled into a model, and a second-order analysis is performed.

5.2.1 Structural Elements

The shape gets generated by the *GrammarInterpreter* and outputs the structural properties as SimpleShape Class. The component *C# Sort structural elements* separates the parts into beams, bars, columns, bracing and shear wall. Hence, the material and cross-section of the elements may differ from each other. Furthermore, as both Figure 5.1 and Appendix B.1 show, their element names are collected into ID-lists to serve as identifiers for the *Line to Beam (Karamba 3D)*-and *Mesh to Shell (Karamba 3D)*-components. These components convert the geometry into Karamba-geometry and include a drop-down menu *Options* to adjust their properties. The input-option "Bending" from the drop-down menu allows to switch off the bending stiffness of members. This behaviour is desired for members that only transfer axial forces, like vertical and diagonal members in a truss system and bracing. Hence, the value is set to "false" to achieve this property. Another solution to achieve it would be to model pinned joints.

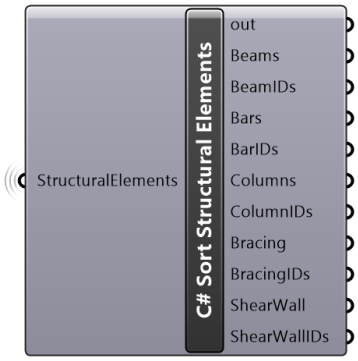


Figure 5.1: *C# Sort structural elements* sorts and delivers the structural members and their ID.

5.2.2 Material and Cross-section

Each structural element classification have its material and cross-section component assigned with the components *Material Selection (Karamba3D)* and *Cross-Section Selection (Karamba3D)*. The material component are connected to the cross-section, and the cross-section to one of the structural element components presented in Figure 5.2.

The material is chosen with the *Material Selection (Karamba3D)*-component, which have a drop-down menu with give quick access to materials. Steel, concrete, glulam timber and aluminium are some of the options within *Family*. The *Name* depends on the chosen family, for example, "S355" is an alternative for steel.

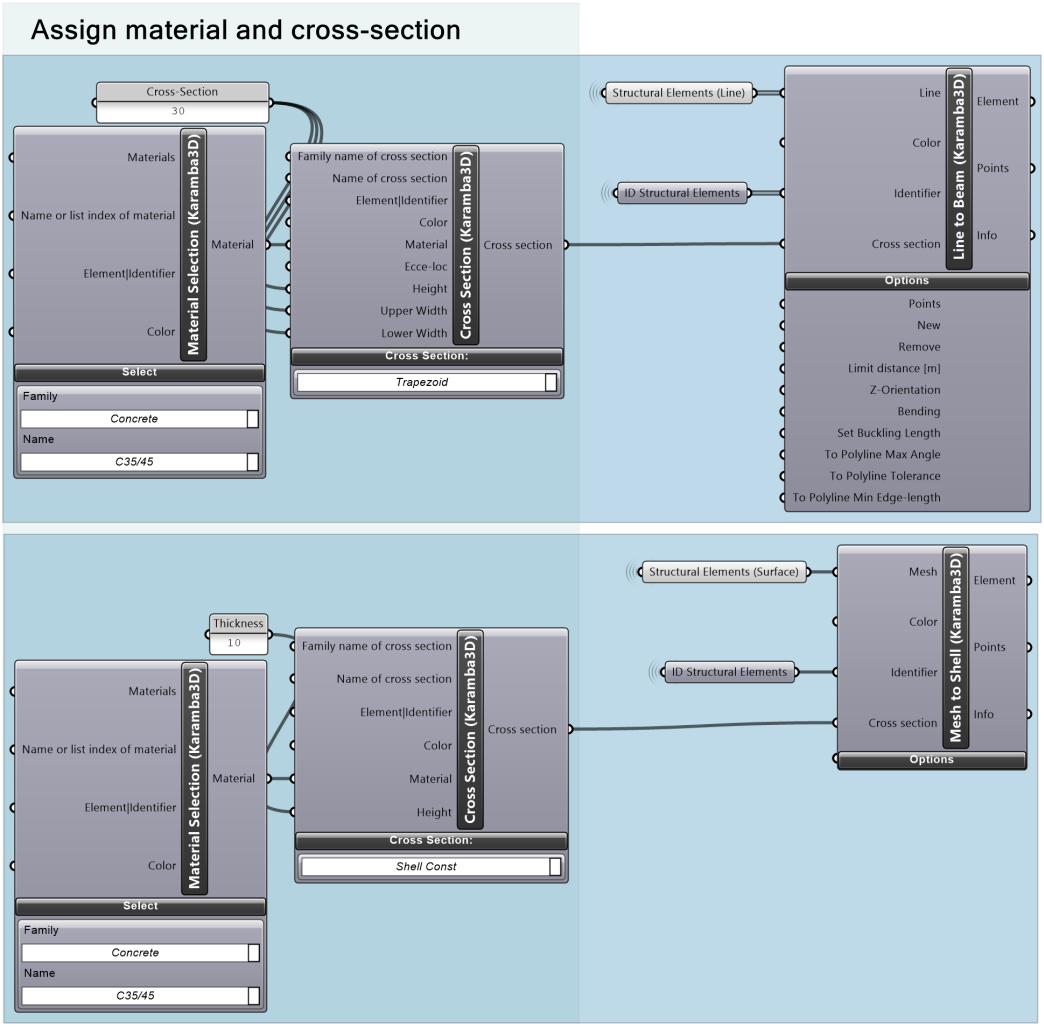


Figure 5.2: Defining structural elements with *Line to Beam (Karamba 3D)*-component and *Mesh to Shell (Karamba 3D)*-component. Further, they are connected to *Material Selection (Karamba3D)* and *Cross Section (Karamba3D)* that assign material and cross-section to the elements.

The selected materials that have been adopted from the Karamba component are listed in Table 5.1 below. Glulam (glued laminated timber) is chosen since it is stronger and more rigid than solid timber due to the lamella effect. The greatest tensile forces occur at the top and bottom of the beam, and consequently, these lamellas have a higher strength class.

Material Name	Material Family
Steel	S355
Concrete	C35/45
GlulamTimber	GL32c

Table 5.1: Material from Karamba is used for the analysis. The material properties are assigned by Karamba.

There have been operated with different cross-sections for different materials. Other than default values, the standard dimensions [mm] of glulam are presented below in Table 5.2.

Width:	90, 115, 140
Height:	90, 115, 135, 180, 225, 270, 315, 360, 405, 450, 495, 540, 585, 630

Table 5.2: Standard dimensions for glulam ('Standard limtre trykkimpregnert', n.d.).

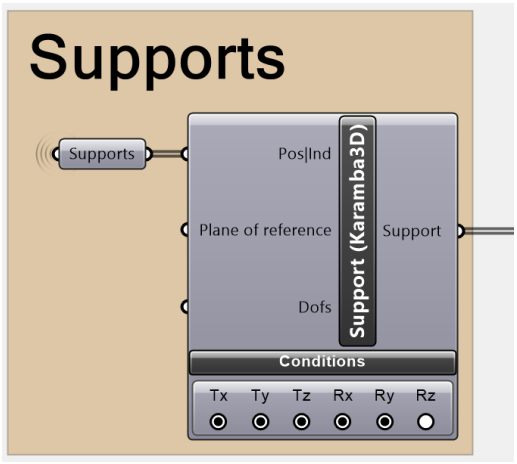
The relevant steel profiles is based on the standard dimensions in *Stålkonstruksjoner - Profiler og formler* (Larsen et al., 1997). The profiles for all materials correspond to the alternatives listed in the cross-section menu included in *Cross-Section (Karamba)*-component. Table 5.3 presents a summary of profiles and dimensions applicable for the materials in addition to the default values.

	Cross-Section					
	Steel		Glulam		Concrete	
Member	Profile	Dim. [cm]	Profile	Dim. [cmXcm]	Profile	Dim. [cmXcm]
Beams	IPE	Default	Trapezoid	11.5x11.5	Trapezoid	30x30
Bars	HFRHS	5	Trapezoid	9x9	Trapezoid	-
Columns	HFRHS	10	Trapezoid	11.5x11.5	Trapezoid	20x20
Bracing	HFRHS	5	Trapezoid	Default	Trapezoid	-
Shear Wall	-	-	-	-	Shell Const	10 cm

Table 5.3: Cross-section applied to members. Profile corresponds to the included cross-section menu in Karamba.

5.2.3 Supports

The supports have been collected from the *CreateGeometry* by extracting the points that are located in World XY-plane with Z-coordinate equal to zero. These support points correspond to the endpoints of the columns that are in contact with the ground. The support conditions are defined through the *Support (Karamba)*-component, which has six degrees of freedom, three translational (Tx, Ty, Tz) and three rotational (Rx, Ry, Rz). As Figure 5.3 indicates are all translational dofs assumed fixed, in addition to Rx and Ry, to prevent torsion and moment.



(a) Support conditions are set by the *Support (Karamba)*-component.

(b) Support condition displayed in Rhino

Figure 5.3: Support condition for the structures.

5.2.4 Loads

The loading for this conceptual study is self-weight, snow load and wind load. How the loads are constructed is shown in Figure 5.4. With the *Load (Kamarba3D)*-component, the type of load is chosen as *MeshLoad Constant* and is globally projected for the snow load case while orientation is set as globally for the wind load. Wind load is assigned as load case zero, and snow load is load case one. *Vec* inputs the surface-load vector [kN/m^2] and the *ElemIds* adds the identifiers to which elements the loads are applied to. Further, the component receives a mesh that the surface load acts on and converts it into line loads.

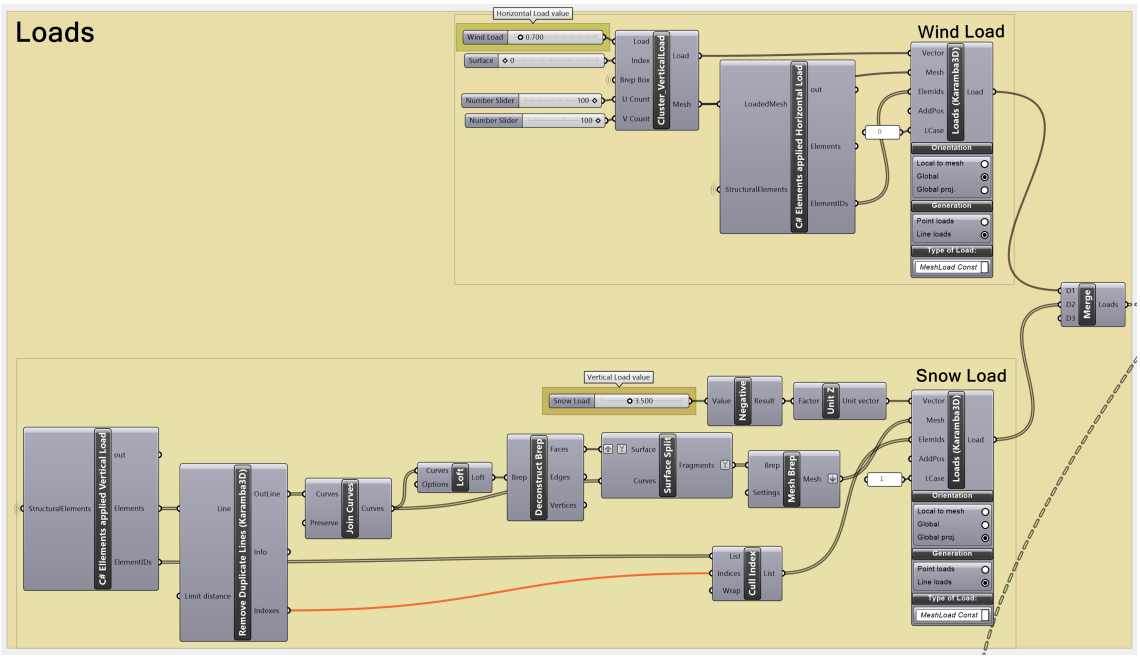


Figure 5.4: The algorithm in Grasshopper to add loads to the analytical model.

Snow Load

In addition to self-weight, snow load is added as vertical loading. The C# script *Elements applied Vertical Load* is developed to collect the correct structural elements. Those elements coincide with the top part of the secondary structure, that is, elements in the longitudinal axis. Based on their element names assigned during the application of Mitchell rules, the necessary element IDs are collected. No duplicate lines are ensured by the *Remove Duplicate Lines (Karamba 3D)*. If there occurs any duplicate line, the index of the deleted item is used to eliminate the corresponding element ID. After that the data is fed into the *ElementIds*. The mesh is constructed based on the *Elements* extracted by the C# code. There should be one mesh between each adjacent line to achieve accurate loading values independent of the roof shape and elements. Therefore these lines have been lofted, thereby deconstructed. Then, the faces are split into surfaces based on the lines before the final meshes are constructed and fed into input *Mesh*.

Wind Load

For the wind load, a cluster has been constructed. The cluster generates the mesh and load vector based on which surface from the initial box to apply the loading. The surface is controlled by the *index* input, while the *U Count* and *V Count* determine the quads of the mesh. In addition, the load value is also taken in as an input. The entire algorithm for the cluster is presented in Figure 5.5. Since loads act on a surface from the initial box, the wind is not accounted for when the structure reaches above the original height.

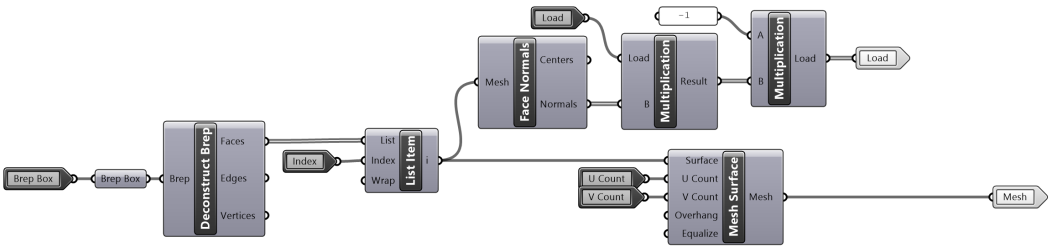


Figure 5.5: The cluster constructed for the wind load. The output *Load* is the load vector, which is determined by the load value - input *Load* set by the user - multiplied by negative one to achieve the correct vector direction. In addition, it is determined by the surface selected with *index* that decides which surface of *Brep Box* to apply the loading.

The C# script *Elements applied Horizontal Load* is developed to access the correct structural elements to load. The inputs are *StructuralElements* that obtain geometry from the *GrammarInterpreter* component, and *LoadedMesh*, that is the mesh the load are applied to in the *Load (Karamba3D)*-component. The structural elements that are located on the mesh are then collected. In this load situation, the loading act on one of the longitudinal walls.

Figure 5.6 shows in which directions the two load cases are applied; positive x-direction for wind load and negative z-direction for snow load.

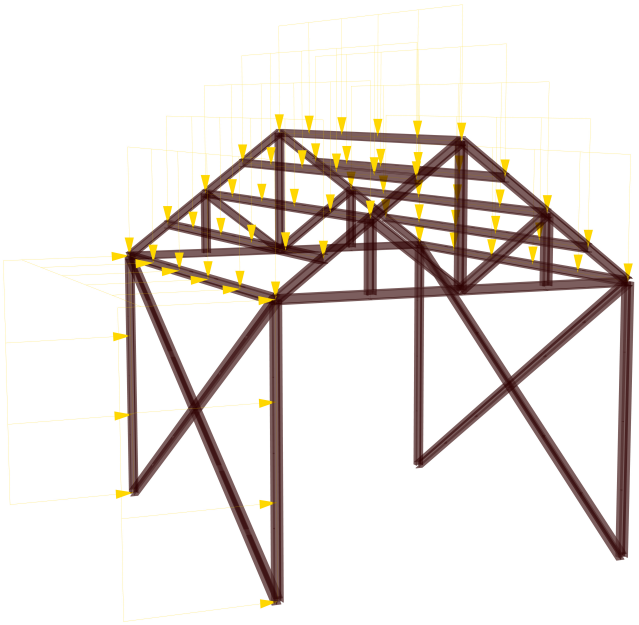


Figure 5.6: Load direction. Vertical load on the roof and horizontal load acting on one of the longitudinal walls (positive x-direction).

5.2.5 Assemble Model And Structural Analysis

The *AssembleModel*-component collects necessary information - structural elements, supports, load, cross-section and material - to run an FE-analysis with Karamba. A second-order analysis is completed with the *AnalyzeThll (Karamba3D)*-component and connected to the *Model View (Karamba3D)*-component (see Figure 5.7). Further, the component is joined with the components *Beam View (Karamba3D)* and *Shell View (Karamba3D)*, which controls the display options. This concerns rendering of such as cross-section, displacement and utilisation. Other results are given by the *Utilization of Elements (Karamba3D)*-component, which returns the utilisation for members, and the *Buckling Modes (Karamba3D)*-component, which calculates the buckling modes and returns the buckling factors.

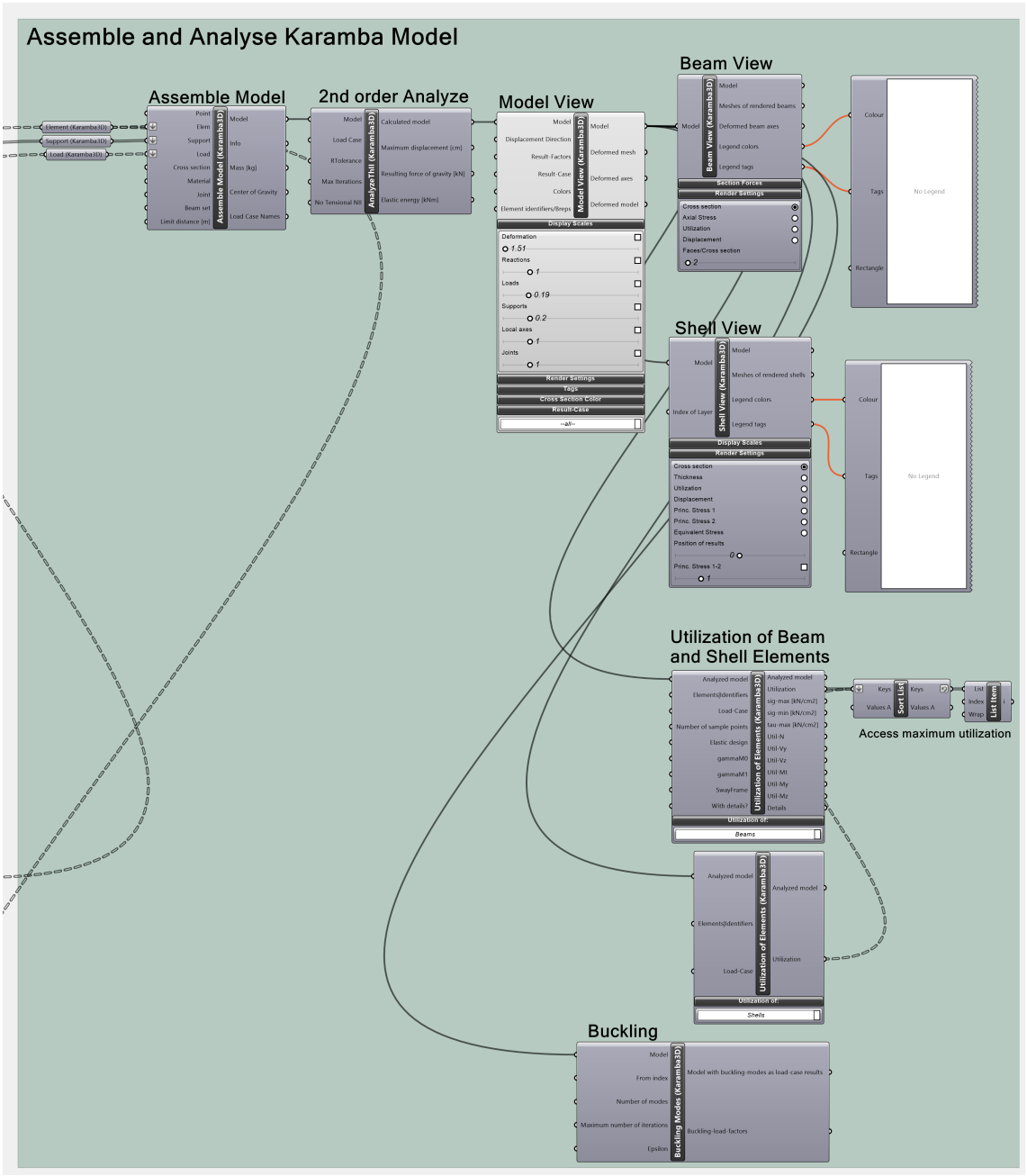


Figure 5.7: Illustration of how the Karamba model is assembled and analysed.

6 Exploration with Shape Grammar

The Karamba model established in the previous section is now used to investigate the structural performances of particular configurations generated from the developed Mitchell Rules. Besides that, a rough price estimation, CO_2 emission and an architectural model are established and will impact the total evaluation of the different shapes. This adds an extra dimension to shape grammar. Further, these shapes are evaluated and compared based on the given results.

The advantage of shape grammars is to rapidly explore and evaluate configurations by changing the parameters. Section 6.3 demonstrates the potential through a manual exploration of structures that also gets analysed. The subsequent section concerns a MOO based on minimising displacement, price estimation and CO_2 equivalents for both glulam and steel structures.

6.1 Price Estimation

A rough estimation of the price has been calculated to get an overview of the price alternatives. The assumed price is not fixed since it will fluctuate within countries, seasons, et cetera. However, it is used to establish an estimation to indicate how the price can affect the choice of structure and the broad potential of shape grammar.

The costs correlate with the volume - the bigger the volume, the higher the costs. Although, as Figure 6.1 indicates, the volume and price are calculated in separate codes since later calculations are also based on the volume. It takes in three inputs; *Elements*, *Material* and *CrossSections*. *Elements* include all structural geometry - lines and surfaces, while their corresponding material properties are obtained from the second input. Whereas the last input value represents the area of cross-sections for lines, it corresponds to the thickness of the surfaces. In addition to the total volume, a dictionary named *Volumes* gets generated. It stores the volumes of each material separately together with their gamma value. This dictionary provides the necessary information to input in later components.

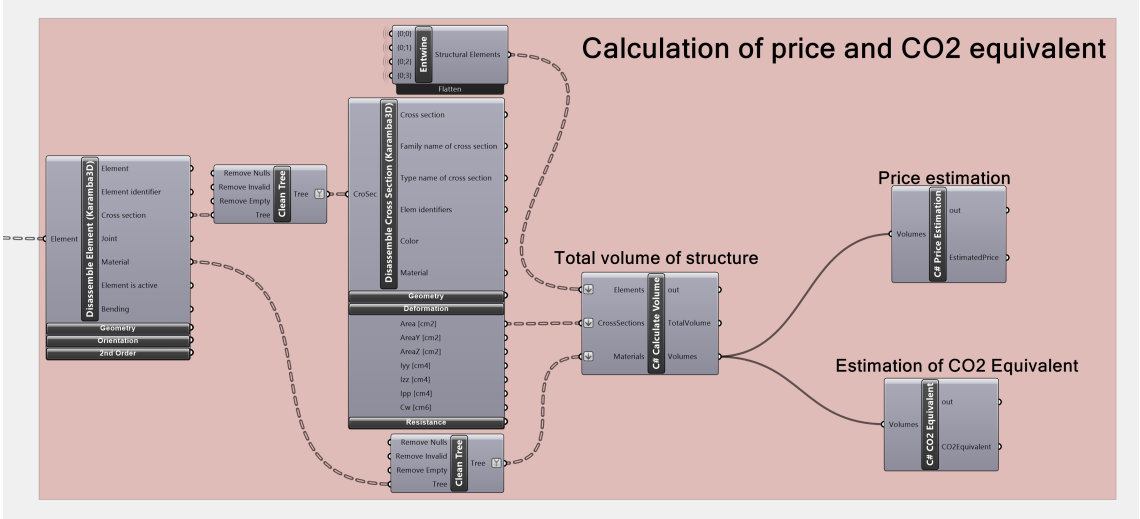


Figure 6.1: The computed total volume [m^3] is used to calculate a price estimation [NOK] and a prediction for CO_2 equivalents [kgCO2eq].

The total price for a particular structure is calculated based on the information stored in the dictionary *Volumes* from *C# Calculated Volume* and assumed prices. The price of concrete is assumed 1400 NOK per m^3, while the timber and steel prices is based on the mass. Hence, both the steel and timber price are assumed 40 NOK per kg. The output is delivered in Norwegian kroners [NOK].

6.2 CO_2 Emission

Another aspect of shape grammar is how an estimation of the CO_2 equivalents can be integrated and evaluated. A highly relevant operation and value for future design approach and shape generation aware of the environmental challenges ahead. Likewise as price estimation, it is based on the information fed by *Volumes*. Table 6.1 presents the utilised life cycle analysis (LCA) coefficients ('ecoinveent', n.d.). Coefficient for A1-A3 are related to production of material.

Material	**A1-A3**	**Units**
		kgCO2eq
Concrete (ready-mix)	238.2	/m^3
Steel (profile)	1	/kg
Timber (glulam)	44	/m^3

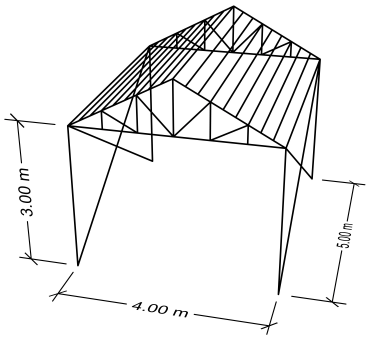
Table 6.1: Life cycle analysis (LCA) coefficients used to calculate a prediction for CO_2 emission.

6.3 Manual Exploration of Shape Grammar

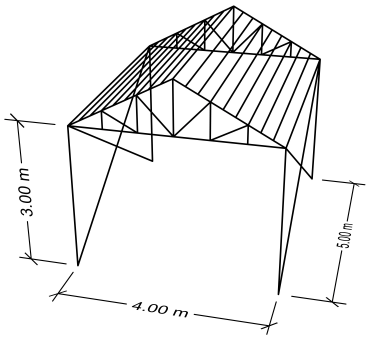
A manual selection of various structures that are analysed are illustrated in Figure 6.2 with corresponding description in Table 6.2. There are combinations where the structural performance are both checked for steel and timber, while others are composite structures of steel and concrete. The initial dimensions of the building are: a column height equal to three meters, longitudinal side is five meters and transversal side has length of four meters. These dimensions are displayed in Figure 6.2a and Figure 6.2b are applicable to all configurations. The cross-sections, profiles and material utilised are presented in Table 5.3 and Table 5.1.

Configuration	Material	Roof Structure	Bracing
1	Steel	Pitched truss roof (Substructure = 2)	Cross-bracing (Lateral stability = 1)
2	Glulam	Pitched truss roof (Substructure = 2)	Cross-bracing (Lateral stability = 1)
3	Steel	Bowed truss roof (Substructure = 3)	Knee braces (Lateral stability = 2)
4	Glulam	Bowed truss roof (Substructure = 3)	Knee braces (Lateral stability = 2)
5	Steel	Flat trusses as primary and secondary roof structure (Primary roof structure = 0, Secondary roof structure = 3)	Cross-bracing (Lateral stability = 1)
6	Glulam	Flat trusses as primary and secondary roof structure (Primary roof structure = 0, roof structure = 3)	Cross-bracing (Lateral stability = 1)
7	Steel columns and bracing + concrete beams	Beams as primary and secondary roof structure (Primary roof structure = 1, Secondary roof structure = 2)	Diagonal bracing (Lateral stability = 0)
8	Concrete	Beams as primary and secondary roof structure (Primary roof structure = 1, Secondary roof structure = 2)	Shear walls (Lateral stability = 3)

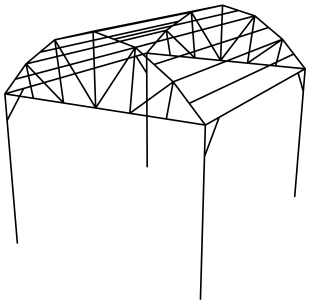
Table 6.2: Description of the configurations manually chosen.



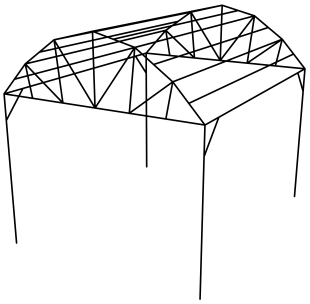
(a) Configuration # 1 - steel framework.



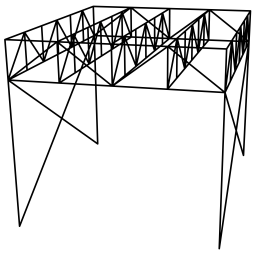
(b) Configuration # 2 - timber framework.



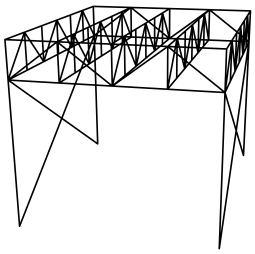
(c) Configuration # 3 - steel framework.



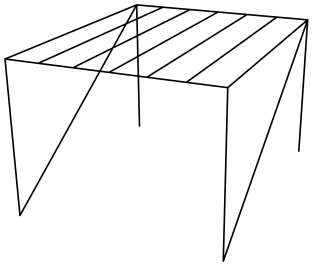
(d) Configuration # 4 - timber framework.



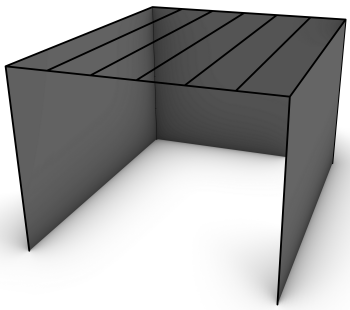
(e) Configuration # 5 - steel framework.



(f) Configuration # 6 - timber framework.



(g) Configuration # 7 - steel columns and bracing & concrete beams.



(h) Configuration # 8 - Concrete

Figure 6.2: The selection of structural configuration that are analysed.

6.3.1 Results

Below in Table 6.3 are the results for the structures in Figure 6.2 presented. These values are extracted from the established Karamba models and C# scripts, and are gathered under the header *Results* in Grasshopper.

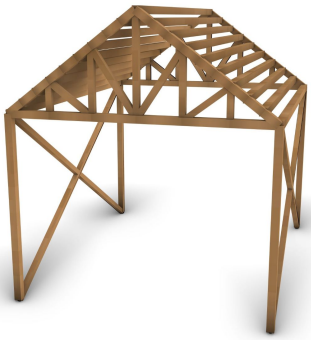
Configuration	Displacement Load Case 0	Displacement Load Case 1	Buckling	Maximum Utilisation	Total Volume	Price Estimation	CO_2 Equivalents
	[cm]	[cm]	[kN/m^3]		[m^3]	[NOK]	[kgCO2eq]
# 1	1.726	0.952	28.166	0.382	0.231	20087	2008.736
# 2	3.774	2.668	11.335	0.376	1.781	9075	78.344
# 3	1.447	1.627	13.148	0.518	0.196	15679	1567.857
# 4	3.248	4.665	4.821	0.346	1.346	6862	59.234
# 5	2.063	0.122	17.398	0.459	0.157	12565	1256.544
# 6	4.454	0.222	10.119	0.136	1.159	5907	50.995
# 7	6.826	0.312	4.903	1.7452	5	4549	992.265
# 8	0.020	0.148	509.680	1.195	4.770	4770	1136.214

Table 6.3: Results for the configurations from the FE-analyse with Karamba, including calculation of total volume, price estimation and CO_2 equivalents.

6.3.2 Architectural Expression of Configurations

In addition to the structural performance, price estimation and prediction of CO_2 equivalents, the architectural aspect affect the choice of structure. Thus, a visual representation has been constructed to evaluate the importance of expression. The models are presented in Figure 6.3 and have been added material colour and cross-section to visualise the architectural language.

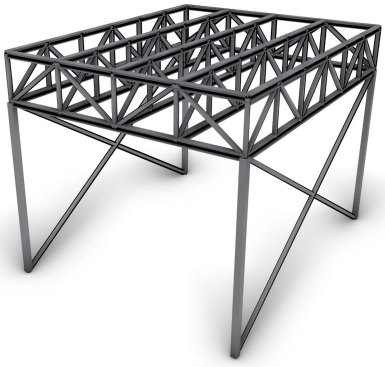
(a) Configuration # 1 - steel framework.



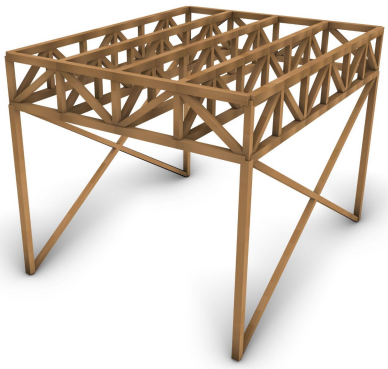
(b) Configuration # 2 - timber framework.

(c) Configuration # 3 - steel framework.

(d) Configuration # 4 - timber framework.

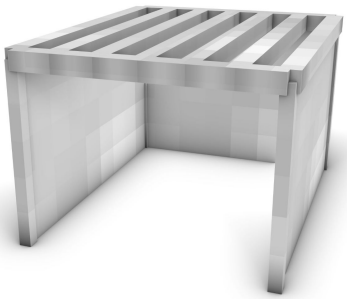


(e) Configuration # 5 - steel framework.



(f) Configuration # 6 - timber framework.

(g) Configuration # 7 - steel columns & concrete beams.



(h) Configuration # 8 - concrete.

Figure 6.3: 3D architectural model of the configurations with material and cross-section added.

6.3.3 Interpretation of Results

The manual exploration results were presented in Section 6.3.1 Section 6.3.2 and are in this section evaluated, compared and discussed.

- Steel structures (configuration 1, 3 and 5) displacement results for both load cases are small. The same configurations with glulam as material remain with approximately two times higher displacements. Except for the structures involving bowed roofs, which achieved greater horizontal than vertical displacement. This is expected knowing the advantages of an arch and how it transfers forces.

  Configuration 8 accomplished minimal displacements for both load cases. Due to three shear walls, the structure achieves sufficient stiffness to resist actions. The two cross-braced walls for configuration 7 are located on the longitudinal side. Since they are perpendicular to the load direction, the structure does not achieve equally good horizontal displacement results. Anyway, as a rule of thumb, ten centimetres displacement is acceptable for a conceptual study. This criterion is fulfilled for all configurations.

- The buckling loading factor is greater than one for all configurations. Thus, no buckling is predicted, and the applied loads are less than the estimated critical loads.

- In the conceptual phase, the aim is to achieve utilisation near one, to indicate if it is realisable or not. The value is well within this requirement for steel and timber structures, implying that the materials' properties can be better utilised by changing the configurations. Maximum utilisation for steel and timber structures has reached less than approximately half of the members' capacity. Even though configuration 8 has a max utilisation equal to 1.195, it achieved good displacement results, and the configuration is assumed achievable. For configuration 7 the utilisation is quite high and has high horizontal displacement.

The intention of including volume, price and CO_2 calculations is to present the broad potential of shape grammars. The result is not realistic without an exact and advanced calculation process, but it provides enough information for discussion. The interpretation of the results is done below but has a greater impact when discussing the results based on user preferences.

- Common for timber and concrete structures is a volume higher than steel structures. As the models imply, a more slender design with steel is possible due to its material properties and fabrication method. This is best shown through the truss system.

- The timber and steel price was both assumed to be 40 NOK per kg. With identical configurations analysed and knowing that timber is a light-weighted material, the results are obvious: much higher price for steel structures. In comparison, the designs consisting of concrete accomplish a relatively low cost.

- Compared to glulam systems, steel systems have much higher CO2 equivalents, whereas structures involving concrete are approximate to steel emissions.

- During a manual exploration, the user generates architecturally desirable shapes that need to be controlled structurally and with the consumers' additional important aspects. Hence, the models in Figure 6.3 are aesthetically pleasant. Although, configuration 7 could be enhanced.

For the structures analysed, timber scored high on price and CO2. In other words, glulam achieves a low environmental footprint and low costs, while the steel framework has potential for improvement. They have approximately equal utilisation, but the load cases induce lower displacements for steel structures. Therefore, reducing either the number of elements, adjusting the configuration, or the cross-sections will increase the utilisation of steel members. These changes will most likely positively affect the CO2 equivalent and price, and the deviation between steel and timber decreases. Alternatively, unlike steel structures, configurations of timber need to be advanced. Thus, the displacement gets reduced, and the price and CO_2 equivalents increases. With improved configurations, structures would reach higher utilisation and affect the architectural language. Taking advantage of the materials' properties would likely lead to more slender steel structures.

The architectural aspect is not as straightforward to evaluate since aesthetics is an individual preference. Which configuration and the importance of architecture also depend on how the consumer validates and weights the different elements. For instance, a person with a strict budget would prioritise a sufficient shape with the lowest price. In contrast, a person with much engagement towards the environment and no economical restriction would most likely choose another structure. This customisation to fit each person's desires is one advantage of functional grammar, and it is practical for design exploration in the early stage.

The configurations impact the result; different materials are exploited through different compositions. The various configurations are feasible for different loading conditions and scales of structure. Steel achieves great structural performance for this initial shape, the given load cases, and the chosen configurations. However, a MOO for both materials is performed in the next chapter to get a better comparison.

6.4 Optimisation of the Design

This section aims to optimise the shape using Octopus with respect to multiple objectives. The loads accounted for in the analysis are presented in Section 5.1, that is, self-weight, wind load and snow load. Both the material properties and the cross-section are fixed during the optimisation (see Table 5.1 and Table 5.3). Two optimisations will be performed: one with timber and one with steel. Lastly, the results from the MOO are then retrieved and compared with the configurations in Section 6.3.

6.4.1 Optimisation Approach with Octopus

The goal is to optimise in terms of structural factors, price and CO2 equivalents. The objective values and the constraints are as follows::

Displacements	Aim to minimise the displacement, which entails a decrease in moments and normal forces.
Price estimation	It is favourable to design a sufficient construction that is also competitive. Hence the structure is optimised to minimise the price.
CO2 equivalents	A suitable objective to calculate and minimise when knowing the environmental challenges this planet faces.
Buckling factor	It is necessary to achieve a buckling load factor > 1, then buckling is not predicted. Through the *C# Buckling Constrain*, corresponding script in Listing 3, this requirement is satisfied when the optimisation is performed.
Utilisation	The requirement is utilisation < 1. This is taken into account through constructing *C# Utilization Constrain* for the Octopus-solver. The related script is displayed in Listing 2.

Maximum displacement includes two values - one for each load case. Therefore the optimisation is based on the maximum of these two to ensure a valid optimisation. It is not optimised with respect to material volume since it correlates with the price. Additionally, it also affects the CO_2 emission. Therefore, by evaluating those two objectives, material volume is considered indirectly.

```
// Access input
    List<double> utilization = utilization;

// Solve
    double maxutilization = utilization.Max();
    bool criteraFeedback = new bool();

    if ( maxutilization <= 1)
    {
      criteraFeedback = true;
    }
    else
    {
      criteraFeedback = false;
    }

// Output
    Requirement = criteraFeedback;
```

Listing 2: This script apply to component *C# Utilization Constrain* and checks if the requirement for utilisation is fulfilled, if not it returns false.

```
// Access Input
    double bucklingFactor = BucklingLoadFactors;

// Solve
    bool criteraFeedback = new bool();
    if ( BucklingLoadFactors >= 1)
    {
      criteraFeedback = true;
    }
    else
    {
      criteraFeedback = false;
    }

 // Output
    Requirement = criteraFeedback;
```

Listing 3: This script apply to component *C# Buckling Constrain* and checks if the requirement for buckling is fulfilled, if not it returns false.

To initialise Octopus it is necessary with a valid configuration, this is set with the parameters in Figure 6.4 and the validation given is from Karamba. There are fifthteen parameters/genomes presented, which are connected to the various Mitchell Rules and contribute to the geometric modifications. Similarly to parameters used in the manually exploration. However, items such as material, cross-section and wall number are omitted. As seen in Figure 6.4, a pink wire attached to the variables connects them to the Octopus component. Octopus will generate different configuration within the individual domain of these variables.

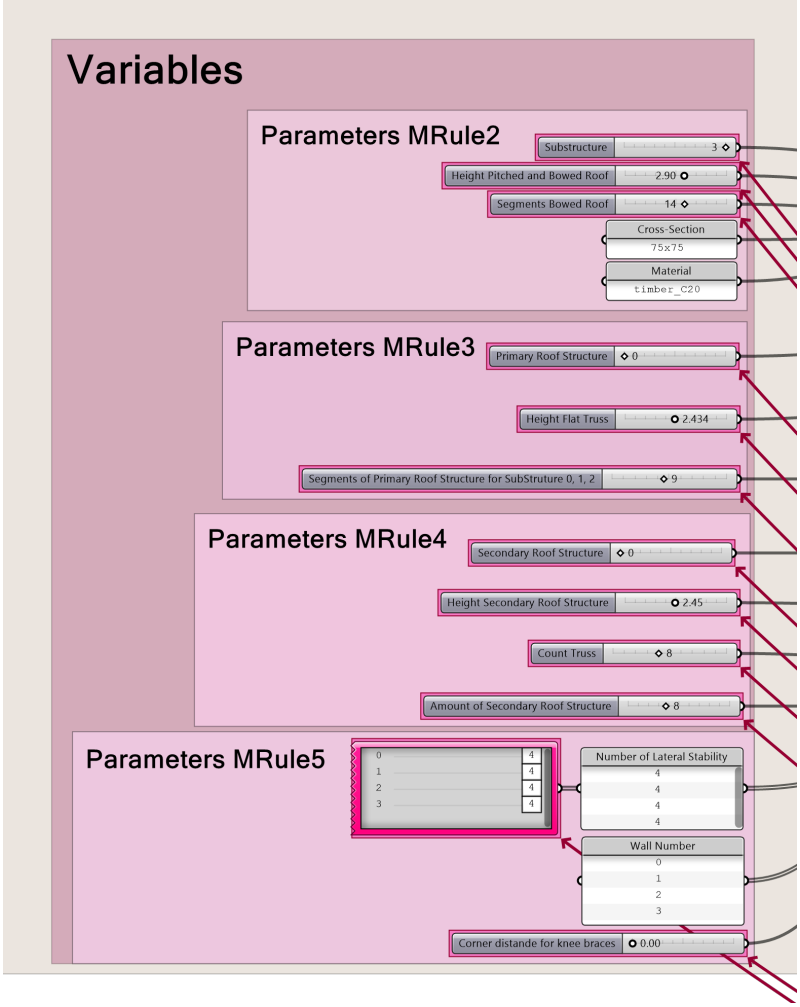


Figure 6.4: The parameters marked with dark pink are the genomes used for MOO with Octopus.

The optimisation process generates a set of solutions within the design space and arranges them in a 3D population field. One axis in the 3D population field represents one of the objectives, hence the three axes names "Displacement", "Price Estimation", and "CO2 Emission". Due to high values for price and CO_2 equivalents, their axis went from highest to lowest, making the results difficult to interpret. Hence the values from the Karamba analysis are multiplied with *0.0001* (steel) or *0.001* (timber) before being added as objective values. Achieving values of the same size as a displacement for those two objectives made the population field more readable.

The optimisation is operated with the default settings. Accurate results are accomplished by running the optimisation for several generations. Therefore, both optimisations were stopped at six generations. Figure C.1 and Figure C.7 display the set-up values in Appendix C.

Figure 6.5 illustrates the optimisation process to find the most feasible shape with the given load case. Based on the parameters and the initial shape, a structure is generated and fed into the Karamba model. Karamba calculates the displacement, buckling and utilisation and outputs the solution mesh. The requirements are checked, and the solution is thrown away if they are not satisfied. In addition, the optimisation seeks minimum displacement, price and CO_2 equivalents.

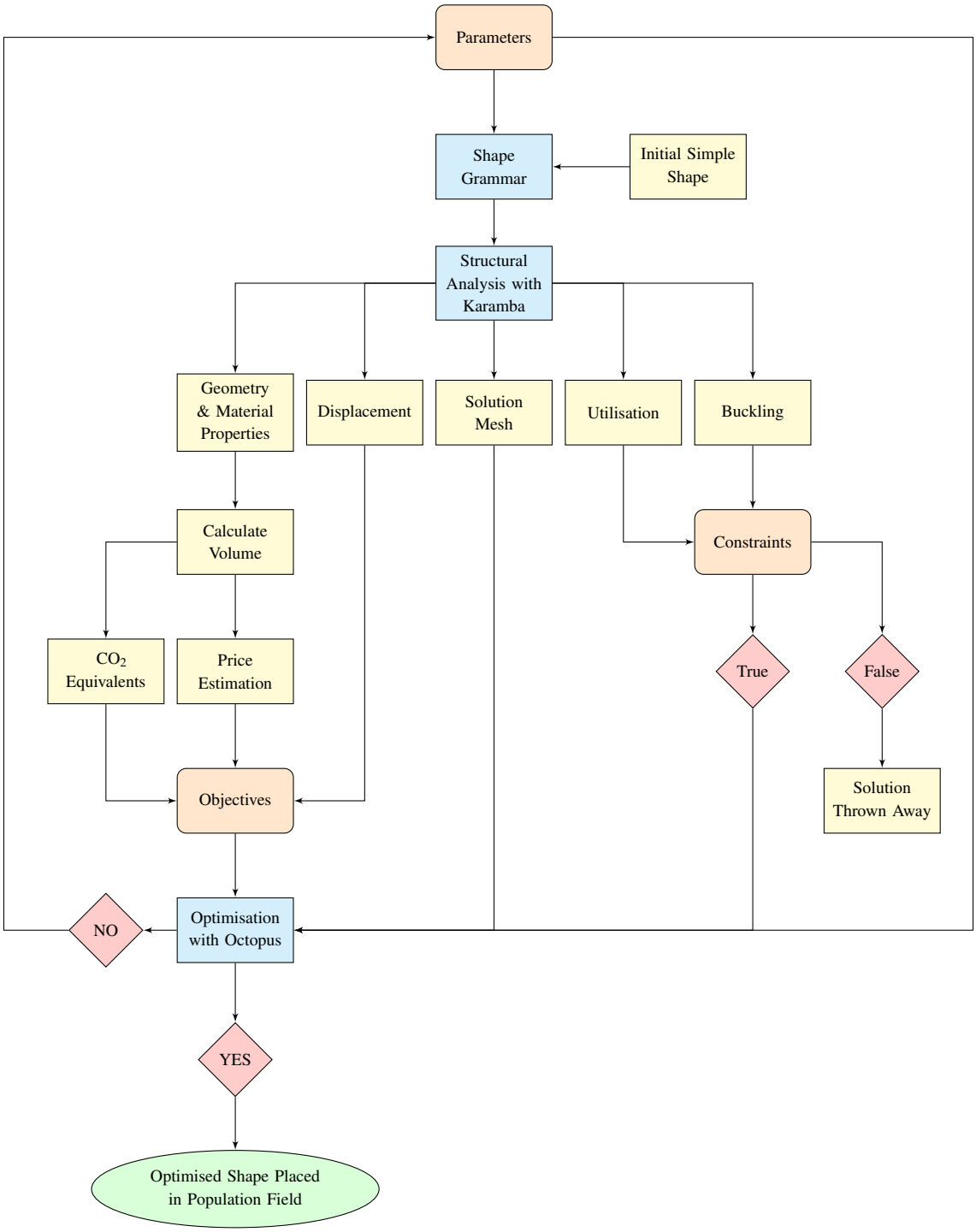


Figure 6.5: Flow chart of the optimisation process. Octopus receives solution mesh, parameters, validation from constraints, and the objectives CO_2 equivalents, price estimation and displacement.

See Appendix D for the Grasshopper-file, and Appendix E for the video demonstrating the concept, including the optimisation process.

6.4.2 Results

The population field in Figure 6.6 and Figure 6.7 shows the shapes at the Pareto-front compared to each other for respectively glulam and steel. All solutions at the Pareto-front are equally good. That is, no solution is better in all criteria. For instance, a solution with better price estimation has worse displacement and CO_2 results and is not located at the Pareto front. That solution would be dominating since the Pareto optimal solutions are non-dominating. The configurations at the Pareto front are exported from Octopus, and the files containing the objective and parameter results are merged in Excel, where they are further evaluated. See Appendix C.2 and Appendix C.5 for file explanation and results.

A detailed result orientation is presented in Appendix C. Retrieved from that data, Table 6.4 presents three shapes per material that represent the lowest value of each objective for both materials. Further, the table includes a configuration for each material representing an example of the designer's preference. These configurations are marked with yellow in Figure 6.6 and Figure 6.7.

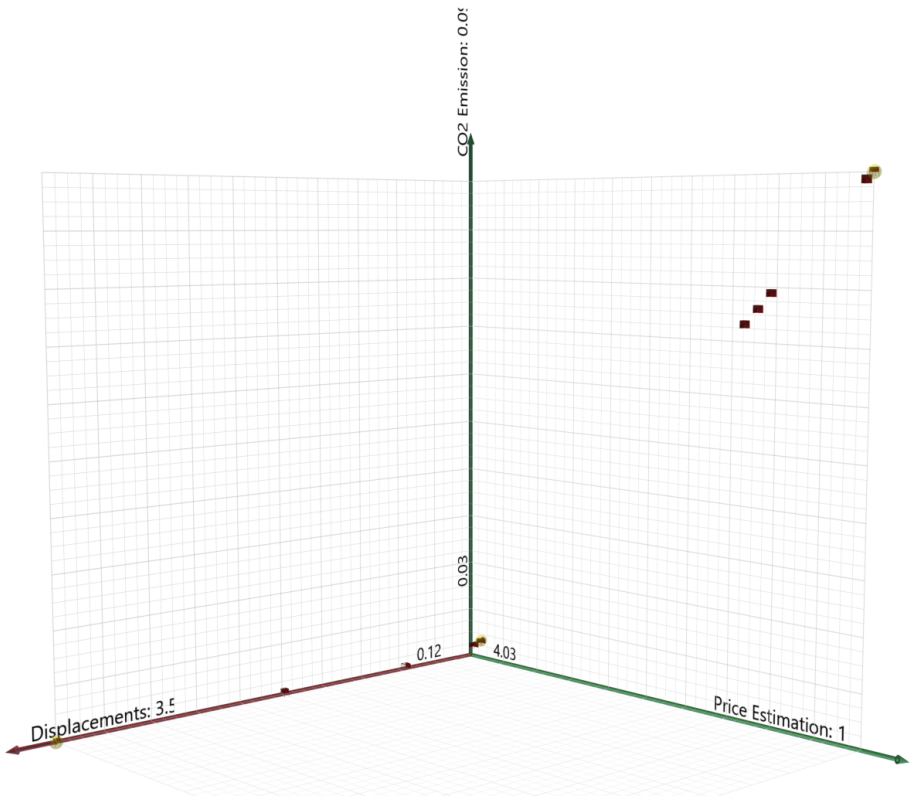


Figure 6.6: The 3D population field result from the MOO with glulam. The yellow-marked structures correspond to the results in Table 6.4.

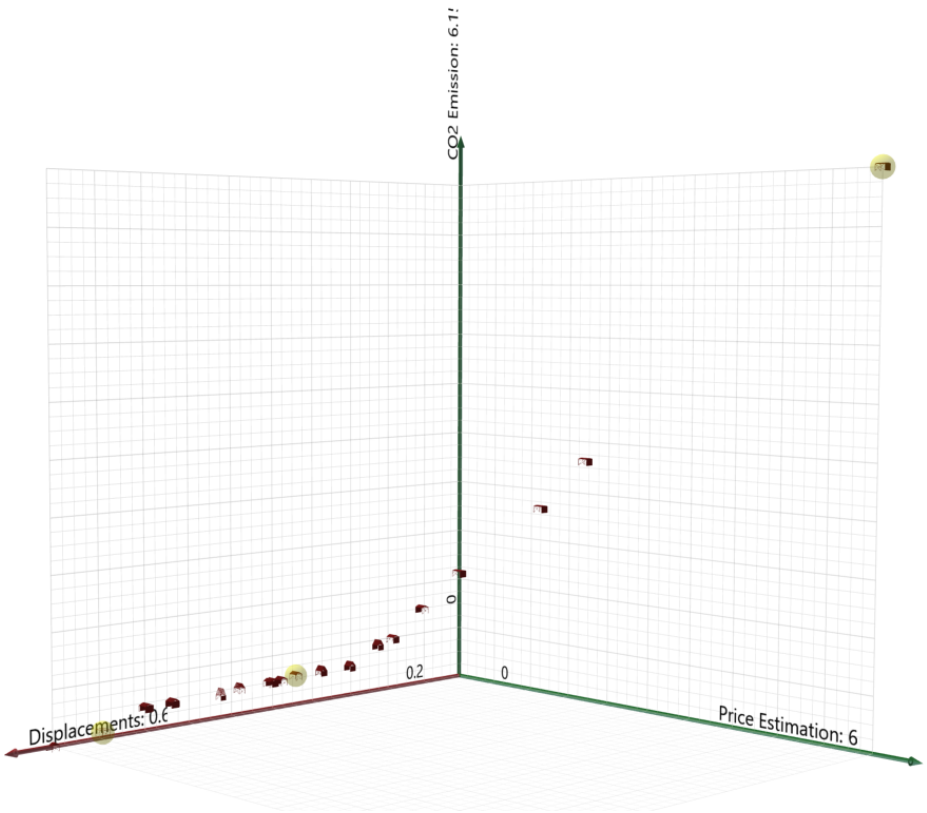


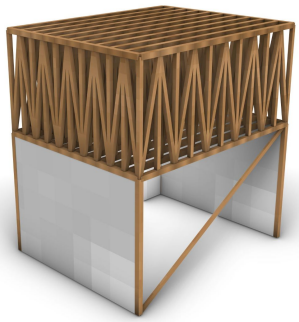
Figure 6.7: The 3D population field result from the MOO with Steel. The yellow-marked structures correspond to the results in Table 6.4.

Configuration	Displacement Load Case 0	Displacement Load Case 1	Buckling	Utilisation	Total Volume	Price Estimation	CO_2 Equivalents
	[cm]	[cm]	[kN/m^3]		[m^3]	[NOK]	[kgCO2eq]
Glulam I	0.123	0.026	87.921	0.271	2.127	10839	93.574
Glulam II	3.545	0.757	6.618	0.927	0.790	4025	34.749
Glulam III	3.545	0.757	6.618	0.927	0.790	4025	34.749
Glulam IV	0.142	0.098	5.328	0.774	0.840	4282	36.965
Steel I	0.186	0.195	10.519	0.260	7.688	615173	61517.271
Steel II	0.606	-	566.386	0.255	0.095	7622	762.205
Steel III	0.606	-	566.386	0.255	0.095	76225	762.205
Steel IV	0.425	0.407	27.802	0.177	0.559	44741	4474.075

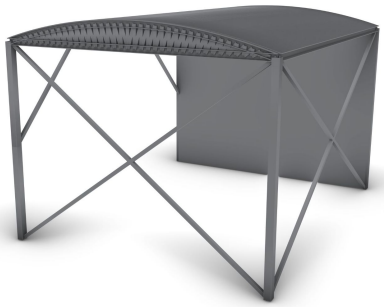
Table 6.4: Results from MOO for both steel and glulam. The shapes listed represent the minimum value for one of the objectives. The name including I represents the configuration with the lowest displacement, II lowest price, while III is the configuration with the lowest CO_2 prediction. Configurations IV is chosen from a designer's perspective.

6.4.3 Architectural Expression of Configurations

Figure 6.8 shows the models corresponding to the stated results in Table 6.4. Material colour is added to obtain an architectural expression.

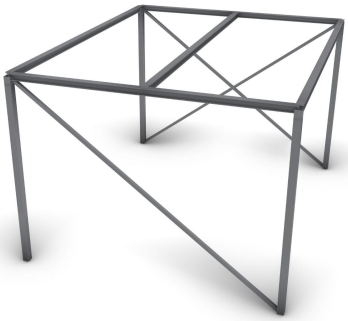


(a) Glulam I



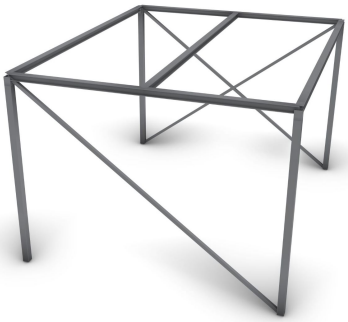
(b) Steel I

(c) Glulam II

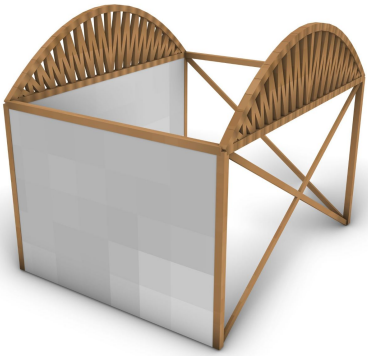


(d) Steel II

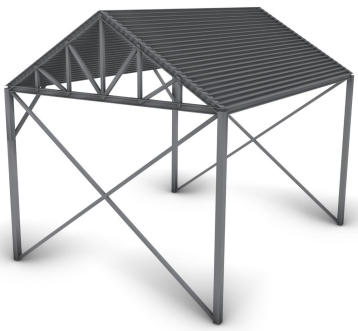
(e) Glulam III



(f) Steel III



(g) Glulam IV - Designers preference



(h) Steel IV - Designers preference

Figure 6.8: Shapes from the MOO for both glulam and steel.

6.4.4 Interpretation of Results

Some observations regarding the results from the MOO presented in Section 6.4.2 are in this section mentioned and discussed.

- Since the buckling and utilisation requirements were constraints during the optimisation process, they are both automatically fulfilled for all configurations. That means that all structures have sufficient capacity, and no buckling is predicted.
- Almost all steel configurations have three or four braced walls, which are logical considering structural stability.
- One solution is located further from the centre in the population field for steel than Steel II/III. However, an error was detected when this configuration was reinstated in Grasshopper. The component calculating the volume failed to run; hence the solution was omitted. Even though the solution was invalid for the evaluation, the actual shape is feasible. This conclusion is supported by the fact that the error message that appeared occurred during the volume calculation and therefore is independent of the structural performance.
- The solutions indicate a linear correlation between the price and CO_2 equivalents, which results from both depending on material volume and has simple estimations.
- Common for the majority of the generated proposals is a high density of the secondary structure. This entails that each point load transferred to the primary structure gets reduced. However, it contributes to higher material usage, which results in a higher price and self-weight.
- Neither of the shapes constructed with glulam in Figure 6.8 has no particularly positive architectural expression. The roof structure got quite an extensional roof height, and since the wind load is not affected by this height, it does not contribute to larger internal forces. If the roof height was taken into account when loads were applied, higher roof height would imply larger forces and would only be favourable to increase to a certain extent.
- Steel II/III achieve a slender structure, likewise for Steel I, despite a compact bowed roof and secondary roof structure.
- Instead of minimising the displacements, it could be implemented into Octopus as a constraint. For instance, a requirement of maximum of ten centimetres displacement could entail additional desirable solutions. The same effect could also appear with a constraint on price or CO_2 equivalents.

The shapes generated from the optimisation fail to achieve a great architectural language. Even though the elements are structurally arranged, they accomplish a disorder when it comes to the expression. Despite the moderate architecture, the process produced unique solutions.

The configuration named designers' preference would vary with the designers and their opinion. The designers' preference for glulam consists of beams as primary roof structure and bowed roof as secondary roof structure, and is chosen due to an improved roof height compared to Glulam

I and II/III. All walls are braced and include several types, which entail a chaotic expression, whereas Glulam II/III achieve a greater braced facade with one cross-braced wall. Glulam IV is preferred rather than II/II due to lower price and CO_2 equivalents and a greater roof system. That indicates that the designer validates these aspects higher than a better bracing system. The same applies to Glulam I; a better bracing system and less displacement are downgraded.

The MOO involving steel generated a solution close to one of the manual configurations. The designers' preference has a resemblance to Configuration 1 in Section 6.3; pitched roof and its design and two cross-braced walls. However, it has an extra wall braced with knee braces. Instead of cross-braced longitudinal walls, Steel IV has the shorter sides braced to resist movement due to wind load - load case 0. Another inequality is the amount of secondary roof structure that is much higher for the optimised shape. The optimised shape has better structural performance, but the price and CO_2 equivalent are more than double what is calculated for Configuration 1.

As assumed, steel realised slimmer structures than timber. Unlike glulam, where the generated configuration in most cases includes concrete shear walls, steel structures manage to achieve sufficient lateral stability on their own.

6.5 Discussion and Remarks

There have been some modifications to simplify the model and calculations and limit the scope of this thesis. However, it entails inaccuracy in the results. The following sources of errors are detected:

- When the Karamba model was established, the members were sorted into beams, bars (vertical and diagonal truss members), columns, and shear walls. Hence, the members within the same classification have the same properties. Arranging them into more specific groups would generate higher accuracy in the analysis. For example, particular members in a truss contribute differently and require different cross-sections, which is common practice.
- The knee braces are modelled to transfer only axial forces. This was done since all bracing types were assembled into one group with the same properties, which was necessary for the MOO. However, in reality, they would create a stiff corner - fixed connections and transfer moment. This could be solved when an independent Karamba model is established, which is further elaborated in Section 7.
- In general, all loads are simplified. There have not been performed calculations to detect the most unfavourable load combination since the focus has been to demonstrate the usefulness of shape grammars. When wind load is applied, the additional height from the roof is not included, and hence the wind load is lower than what has been computed. This has already been discussed when interpreting the optimisation results; the additional roof height should indicate greater loading to achieve rational results.

  Furthermore, the roof form has not affected the snow or wind load, which means the characteristic load had not been multiplied with factors to consider the shape of the roof, its characteristics, et cetera.

- The *Utilization of Elements* component calculates members' utilisation according to NS-EN 1993-1-1. That is Eurocode 3 for steel structures; thus, glulam and concrete results have the wrong material factors.

In addition to the following improvements can be made to accomplish enhanced results:

- In Section 4.4 the differences between substructure 0 and 1 were stated. Even though they have been analysed with the exact same structural system, the load would have an eccentricity for Substructure 0 in reality.
- As *Steel II* and *Steel III* proves, it is necessary with limitations to achieve feasible structures. That is limitations regarding minimum distance for knee braces and truss members. Additionally, a condition considering the angle of diagonal members should be added.
- The MOO showed that it is necessary with several constraints linked to each rule and action to manage realistic configurations. For instance, the amount of secondary roof structure should be limited, and this limitation should depend on which substructure and primary roof structure are chosen.
- Even though the MOO minimised the displacements, it would be beneficial to minimise displacements with respect to mass since higher self-weight cause higher displacement. That is, minimising mass multiplied with displacement.
- The verticals in the bowed truss system are not perpendicular to the bottom beam. This could be improved, or another system could be created.
- For the shape grammars to be adaptable to all proportions of a box, an algorithm that takes column placements and quantity into account should be developed.
- The price estimation and CO_2 prediction were highly simplified. A detailed analysis would incorporate the price of connections, transport, labour, et cetera.

7 Further Work

As this thesis illustrates, there is a wide range of aspects to evaluate within functional grammar. Further research to expand this study will involve a separate Karamba model/plug-ins directly connected to the Mitchell Rules and their inputs. Increasing the complexity and flexibility of the algorithm and process can ensure higher accuracy and a time-efficient evaluation. As for now, the data for material and cross-section is manually attached to Karamba-component to analyse the structure. However, an independent analysis engine with input data from the Mitchell Rules would accelerate the result orientation. MRule2 illustrates this potential, where the given name and cross-section are applied to subsequently generated elements. An advanced approach would distinguish between the different elements' structural properties and analyse based on this given input. Whereas now, the cross-section and material input in shape grammar are insignificant for the structural analysis with Karamba.

Moreover, a score could be added to rate each configuration concerning the users' preferences. That would enhance the user interface - for instance, multiplying each aspect with an individual coefficient to account for the consumers' prioritisation. The fabrication process could also be interesting to check for the different structures. Any other aspects that are favourable to consider during a design process could be integrated with shape grammar to compare different configurations.

Concerning shape grammar in general, it is necessary and desirable to develop rules independent of structures. By making the rules flexible and reusable for various initial shapes, the usefulness of the concept increases. The design exploration could go beyond the typology when an algorithm with flexible rules is created. However, it would require further development of software and computational design to achieve advanced models.

8 Conclusion

This exploration of shape grammar with parametric approaches has captured the essence and potential of early involvement of structural performance. By implementing William J. Mitchell's theory about functional grammar into a digital approach, a design generation was constructed. The idea behind rule application is to investigate different shapes effortlessly and rapidly based on a few parameters. This design approach integrates structural performance with design exploration, bridging the gap between architects and engineers and their separate software tools. It lets the user interact and control the design through computational software.

A manual design generation was completed to prove functional grammar's power. An advantage of the approach is immediate feedback for desired shapes. It was efficient when structures were evaluated and compared. Unlike the MOO, manual generation accomplished great architectural language since the designer determined the shape. The price estimation and the CO_2 equivalent prediction represent an excerpt of possible aspects to integrate with shape grammar. The significance of feature preference when selecting a structure is illustrated by implementing those aspects. Further, the design of a feasible configuration depends on load cases, material and cross-section.

This thesis has researched a relatively new and unexplored topic within the conceptual design. Therefore, collecting relevant data and theories about the concept has been challenging. There are limited studies published surrounding shape and functional grammar. However, with increased attention to its potential, and focus on developing the design approach in the coming years, rule application and functional grammar will increasingly impact future conceptual design. It will be interesting to follow the development of the field in the future.

Bibliography

10 things to do at london's st pancras [TripActions]. (n.d.). Retrieved 27th May 2022, from https://tripactions.com/blog/10-things-to-do-at-st-pancras

Addis, W. (2015). *Building: 3000 years of design engineering and construction*. Phaidon Press, Inc.

Associates, R. M., givenun=0. (n.d.). *Rhinoceros 3d* [Www.rhino3d.com]. Retrieved 6th June 2022, from https://www.rhino3d.com/

Caetano, I., Santos, L. & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research*, *9*(2), 287–300. https://doi.org/10.1016/j.foar.2019.12.008

Ecoinveent [Ecoinvent]. (n.d.). Retrieved 27th May 2022, from https://ecoinvent.org/

The future of structural engineering will survive the high-tech revolution [Redshift EN]. (2016, September 14). Retrieved 20th January 2022, from https://redshift.autodesk.com/future-of-structural-engineering/

Gebisa, A. & Lemu, H. (2017). A case study on topology optimized design for additive manufacturing. *IOP Conference Series: Materials Science and Engineering*, *276*, 012026. https://doi.org/10.1088/1757-899X/276/1/012026

Geyer, P. (2008). Multidisciplinary grammars supporting design optimization of buildings. *Research in Engineering Design*, *18*, 197–216. https://doi.org/10.1007/s00163-007-0038-6

Haakonsen, S. & Izumi, B. (2022). *SimpleShapeGrammar* (Version 1.0.1). https://doi.org/10.5281/zenodo.1234

Introduction. (n.d.). Retrieved 22nd April 2022, from http://www.mit.edu/~tknight/IJDC/page_introduction.htm

J. Mitchell, W. (n.d.). Functional grammars: An introduction.

Karamba3d – parametric engineering. (n.d.). Retrieved 4th June 2022, from https://www.karamba3d.com/

Koning, H. & Eizenberg, J. (1981). The language of the prairie: Frank lloyd wright's prairie houses. *Environment and Planning B: Planning and Design*, *8*, 295–323.

Larsen, P. K., Clausen, A. H. & Aalberg, A. (1997). *Stålkonstruksjoner: Profiler og formler* (2nd ed.). Tapir Akademisk Forlag.

Mueller, C. T. (2014, May 2). Computational exploration of the structural design space.

Network, S. D. c. t. N. (n.d.). *Grasshopper*. Retrieved 6th June 2022, from https://www.grasshopper3d.com/

Octopus [Food4rhino]. (2012, December 6). Retrieved 5th June 2022, from https://www.food4rhino.com/en/app/octopus

Popovic Larsen, O. (2016). *Conceptual structural design: Bridging the gap between architects and engineers, second edition.* [OCLC: 1096811847]. ICE Publishing.

Standard limtre trykkimpregnert [Moelven]. (n.d.). Retrieved 21st May 2022, from https://www.moelven.com/no/no/limtre/standar-limtre-trykkimpregnert/

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343–351. https://doi.org/10.1068/b070343

Striatus 3d concrete printed masonry bridge. (n.d.). Retrieved 15th November 2021, from https://www.striatusbridge.com/

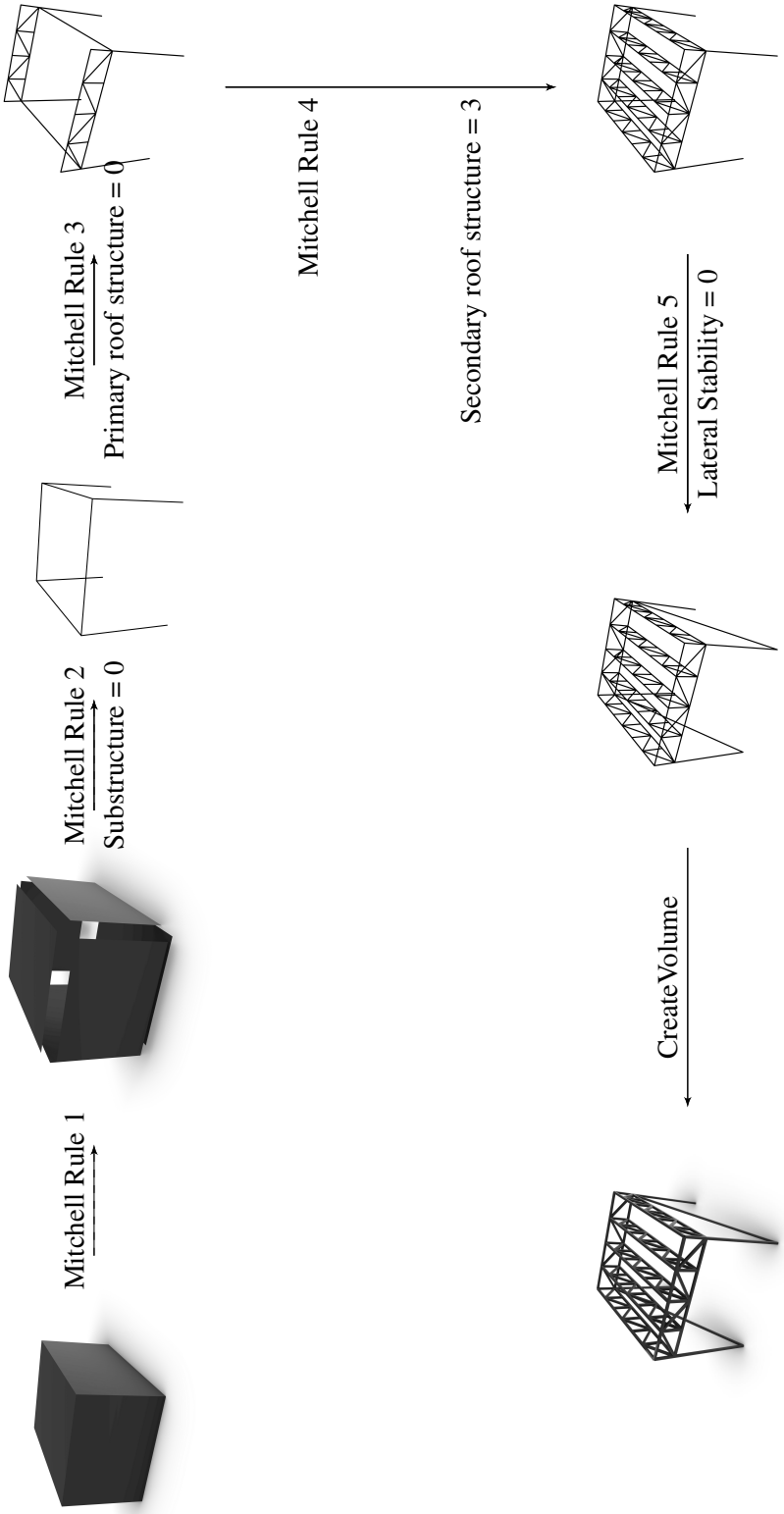
Talebi, S. (2014). Exploring advantages and challenges of adaptation and implementation of bim in project life cycle.

Visual studio: IDE and code editor for software developers and teams [Visual studio]. (n.d.). Retrieved 6th June 2022, from https://visualstudio.microsoft.com

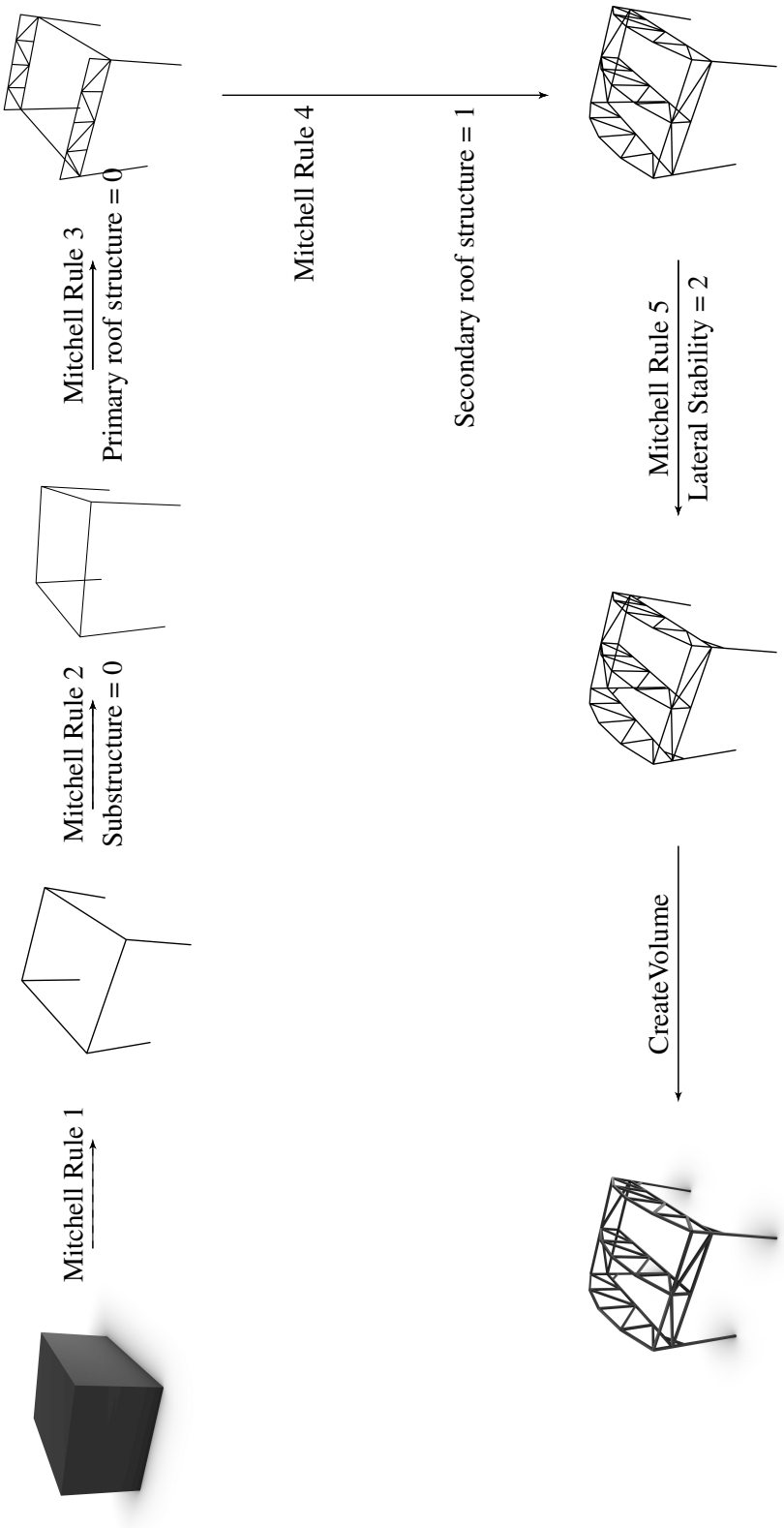
Appendix

A Configurations

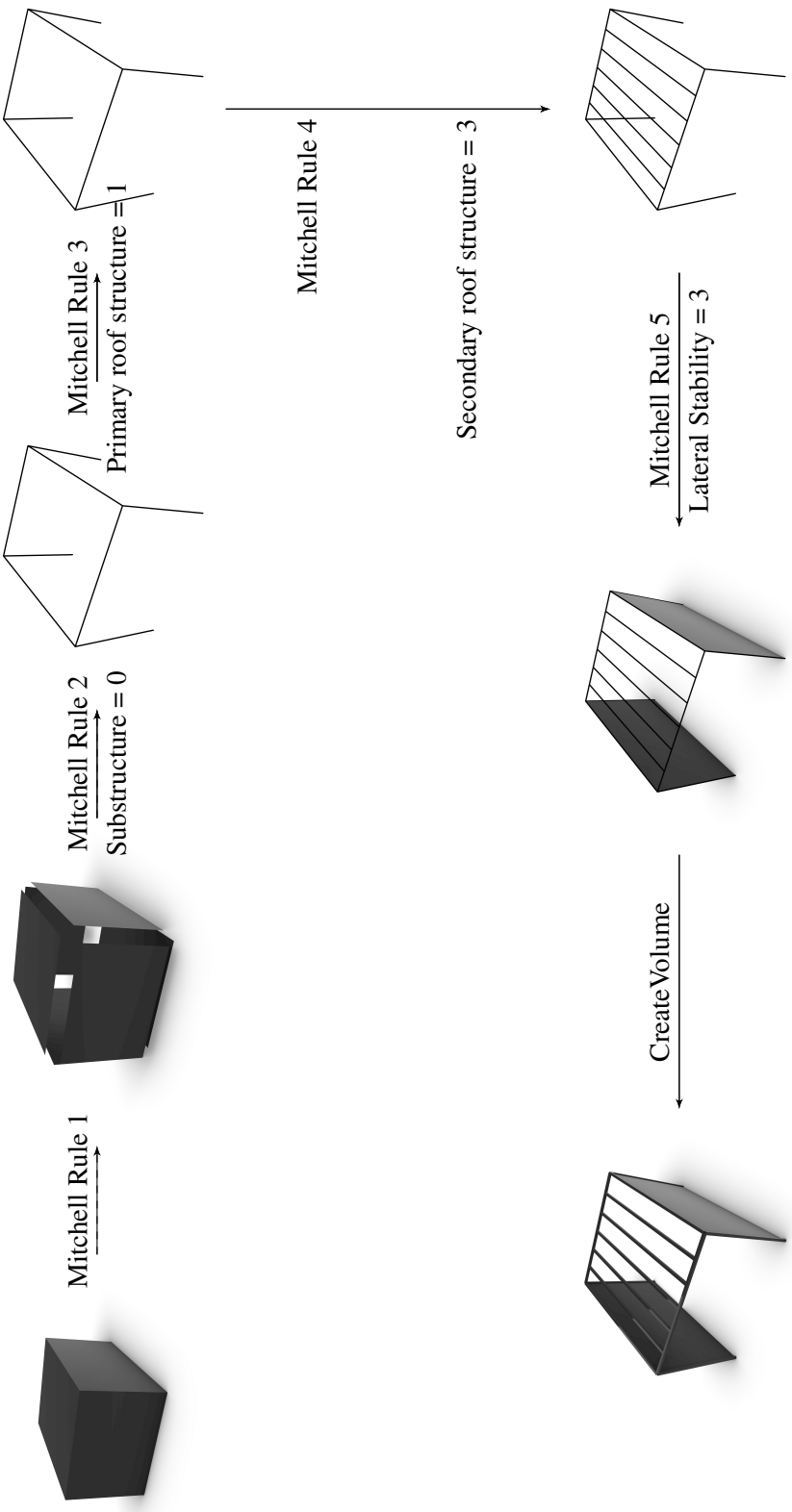
A.1 Configuration I



A.2 Configuration II



A.3 Configuration III



B C# Scripts

The script in Appendix B.1 shows the methodology, other script is found in Appendix D.

B.1 C# Sort structural elements

```
// Input
var _ss = (SH_SimpleShape) StructuralElements;

//Solve
//Access columns
var col = from c in _ss.Elements["Line"]
    where c.elementName.Contains("Column")
    select c;

List <Line> columns = new List<Line>();
List <String> columnIDs = new List<String>();
foreach ( SH_Line c in col)
{
  Curve crv = c.NurbsCurve;
  Line cLine = new Line(crv.PointAtStart, crv.PointAtEnd);
  columns.Add(cLine); // Convert to Rhino geometry
  columnIDs.Add(c.elementName);
}

// Access shear wall (= surface, therefore single output)
var shearW = from sWall in _ss.Elements["Surface"]
    where sWall.elementName.Contains("ShearWall")
    select sWall;

List <Surface> shearWall = new List<Surface>();
List <String> shearWallIDs = new List<String>();
foreach ( SH_Surface s in shearW)
{
  shearWall.Add(s.elementSurface); // Convert to Rhino geometry
  shearWallIDs.Add(s.elementName);
}

// Access bracing
var brace = from b in _ss.Elements["Line"]
    where b.elementName.Contains("Brace")
    select b;

List <Line> bracing = new List<Line>();
List <String> bracingIDs = new List<String>();
foreach ( SH_Line b in brace)
{
  Curve bCrv = b.NurbsCurve;
  Line bLine = new Line(bCrv.PointAtStart, bCrv.PointAtEnd);
  bracing.Add(bLine); // Convert to Rhino geometry
  bracingIDs.Add(b.elementName);
}

//Access bars
List <Line> bars = new List<Line>();
List <String> barIDs = new List<String>();
var keys = new HashSet<string>(){"dTruss","vTruss"};
foreach (string key in keys)
```

```
    {
      var bar = from b in _ss.Elements["Line"]
          where b.elementName.Contains(key)
          select b;

      foreach ( SH_Line b in bar)
      {
        Curve bCrv = b.NurbsCurve;
        Line bLine = new Line(bCrv.PointAtStart, bCrv.PointAtEnd);
        bars.Add(bLine); // Convert to Rhino geometry
        barIDs.Add(b.elementName);
      }
    }

    //Access all beams
    var beamElement = from b in _ss.Elements["Line"]
        where b.elementName.Contains("Mitchell")
        select b;

    List <Line> beamElements = new List<Line>();
    List <String> beamIDs = new List<String>();
    foreach ( SH_Line b in beamElement)
    {
      Curve bCrv = b.NurbsCurve;
      Line bLine = new Line(bCrv.PointAtStart, bCrv.PointAtEnd);
      beamElements.Add(bLine); // Convert to Rhino geometry
      beamIDs.Add(b.elementName);
    }

    // Beams
    List<Line> beams = beamElements.Except(bars).ToList();
    foreach (string id in barIDs)
    {
      if (beamIDs.Contains(id) == true)
      {
        beamIDs.Remove(id);
      }
    }

    // Output
    Beams = beams;
    BeamIDs = beamIDs;
    Bars = bars;
    BarIDs = barIDs;
    Columns = columns;
    ColumnIDs = columnIDs;
    Bracing = bracing;
    BracingIDs = bracingIDs;
    ShearWall = shearWall;
    ShearWallIDs = shearWallIDs;
```

C Optimisation Results

C.1 Optimisation Set-up and Population Field Results for Steel

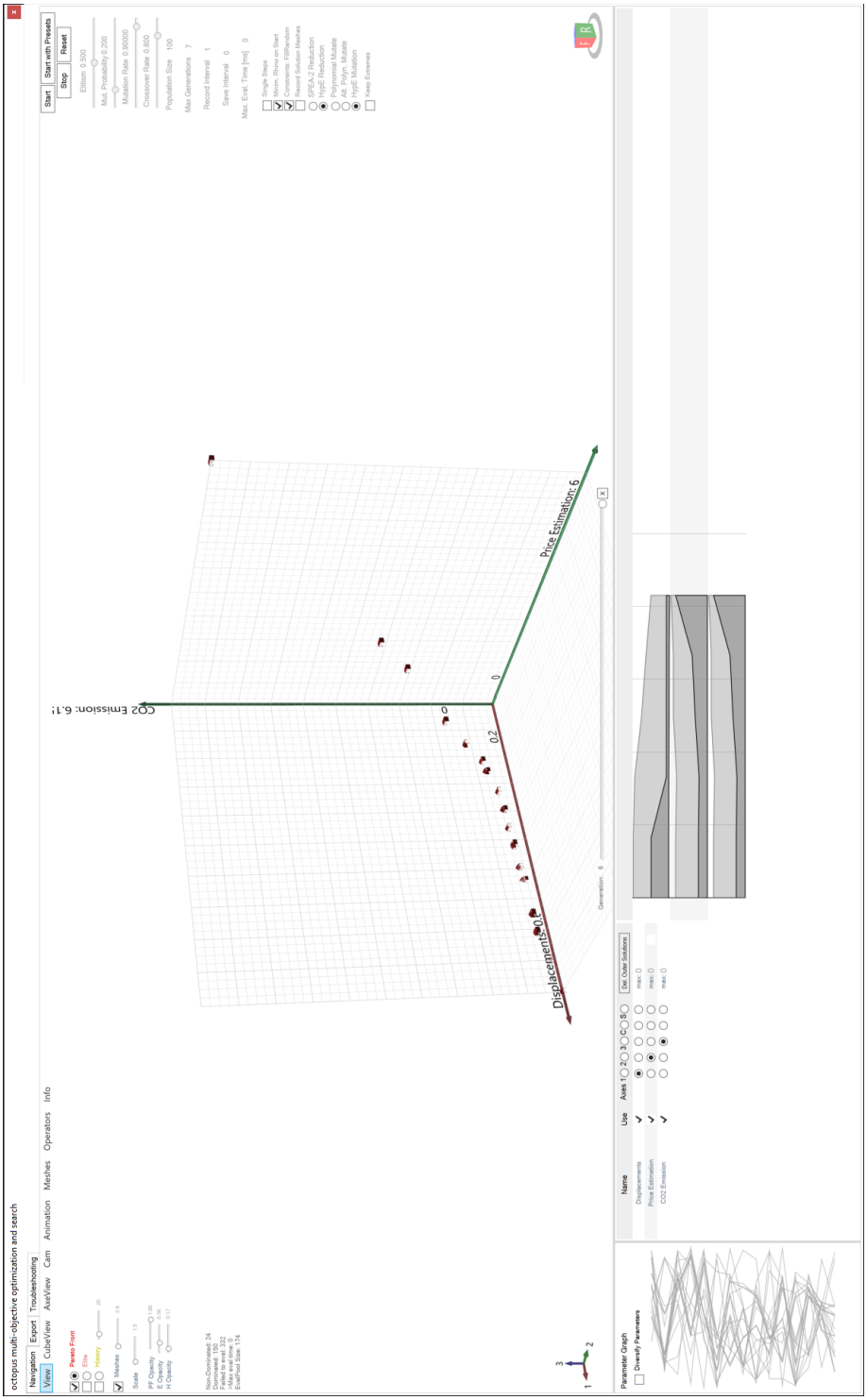


Figure C.1: Displays the population field and the optimisation set-up.

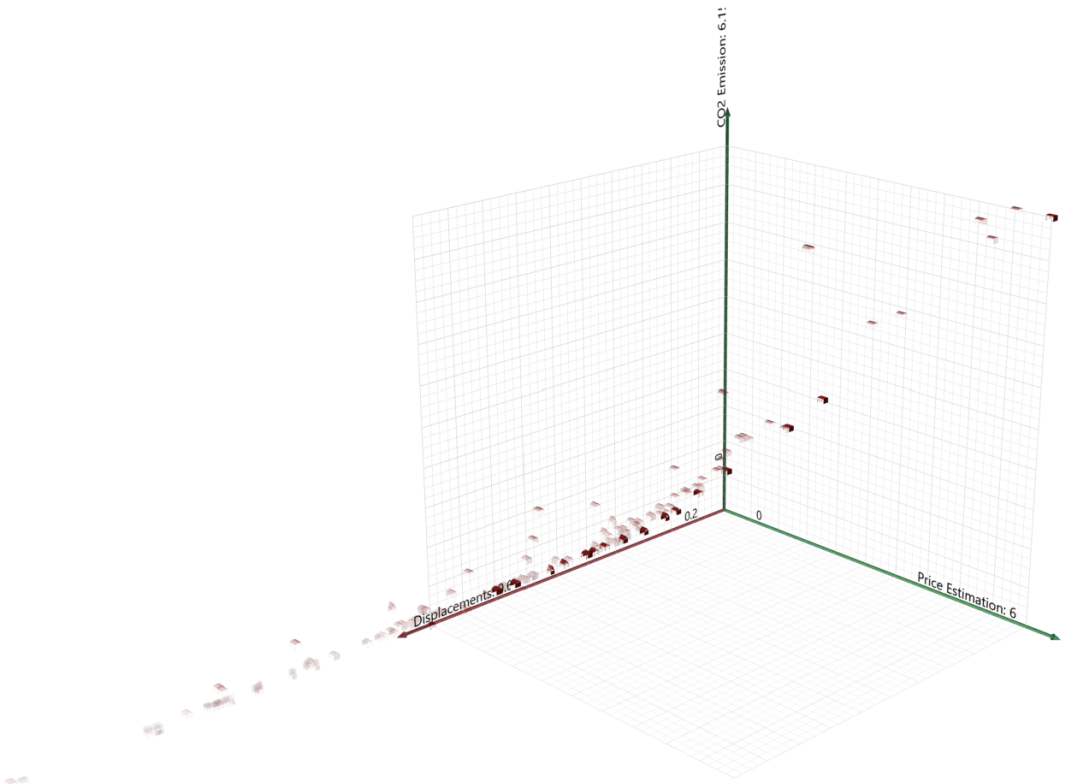


Figure C.2: The transparent shapes represent Elite configuration, whereas the red is the Pareto-front solutions.

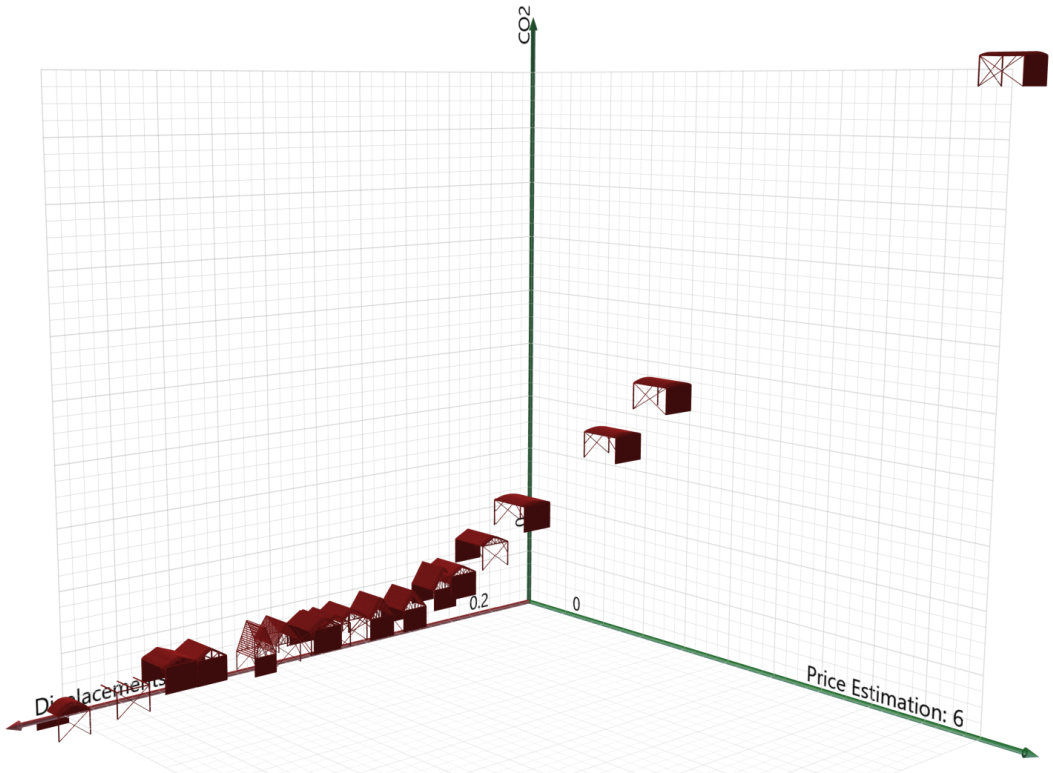


Figure C.3: Close up of population field to give an indication of configurations generated.

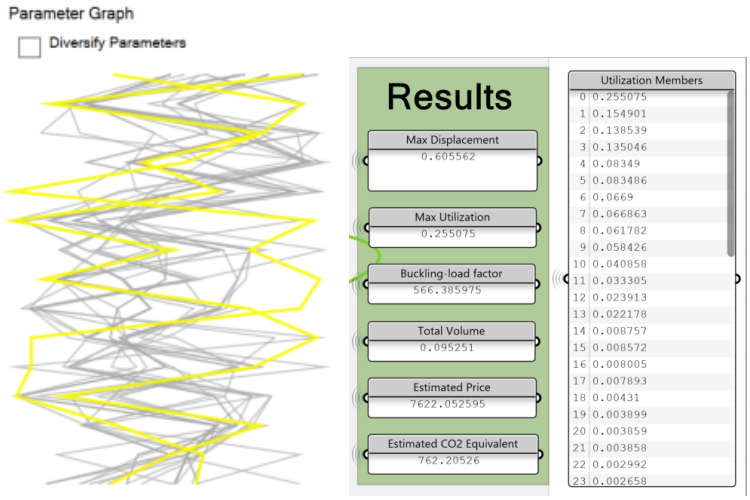
C.2 Objective and Parameter Results from MOO Steel

The MOO results were exported as txt-files. However, they were imported into Excel and converted to make them understandable. The results are attached as ZIP-file, file named C.2. *SteelOptimisationResults*.

Column one stores the objectives. Row one represents the lowest displacement, row two the price estimation and row three the CO_2 equivalents. Price and CO_2 values need to be multiplied by 1000 to get the actual value. Then they coincides with the results in Appendix C.3. A collection of numbers represent one configuration. The parameters in column two are a percentage of the individuals' parameter domain.

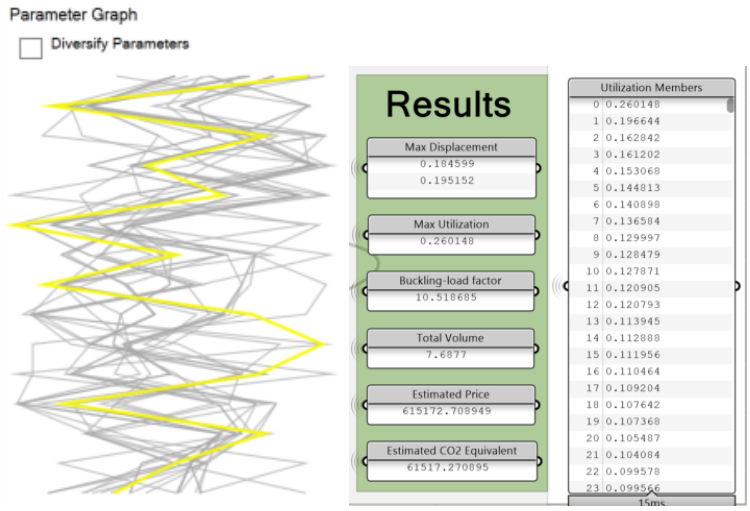
The order of the results depends on when they were generated. Thus, they are not sorted and do not match the locations in the population field. The red marked values belong to the invalid configuration, while those marked with green correspond to the evaluated configurations. The yellow-marked values belong to the designer's preferred shape.

C.3 Results for Steel Configuration



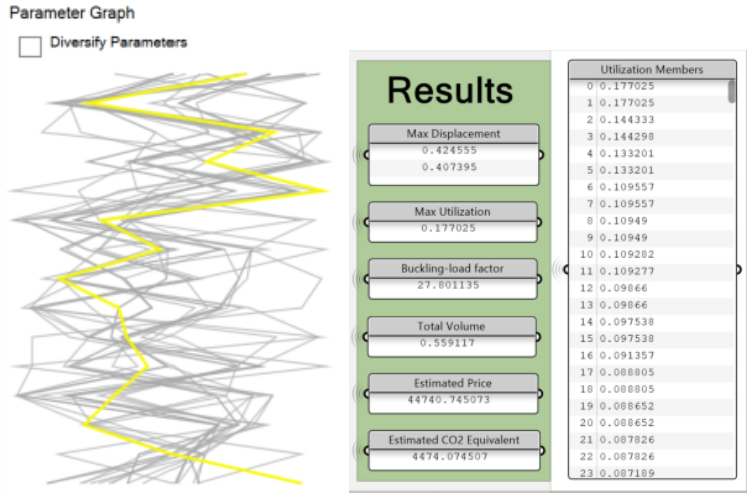
(a) Parameter graph **(b)** Results.

Figure C.4: Results for the configuration with the lowest value for price and CO_2 equivalents.



(a) Parameter graph **(b)** Results

Figure C.5: Results for the configuration with the lowest displacement.



(a) Parameter graph **(b)** Results

Figure C.6: Results for the configuration chosen from the designer.

C.4 Optimization Set-up and Population Field Results for Glulam

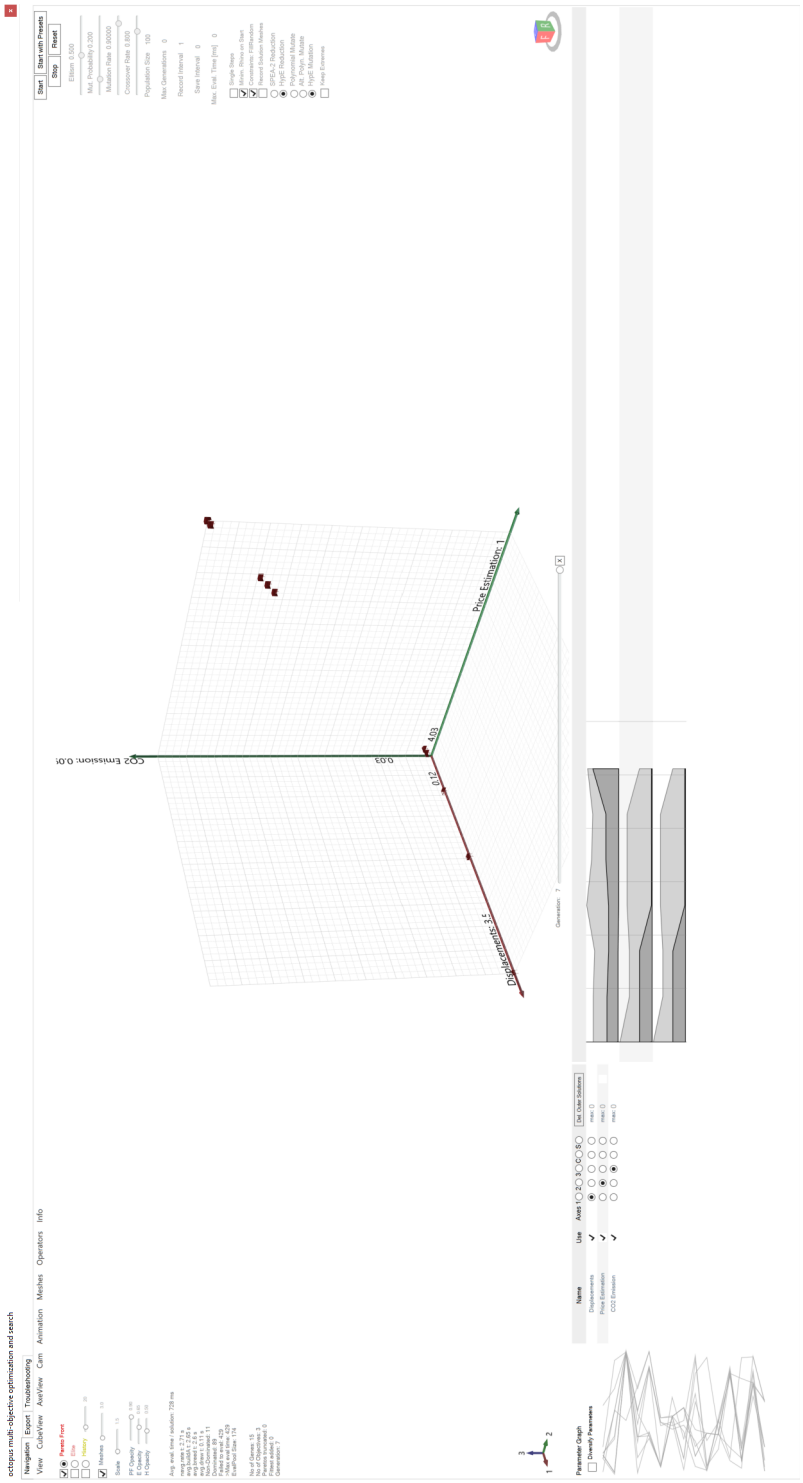


Figure C.7: Displays the population field and the optimisation set-up.

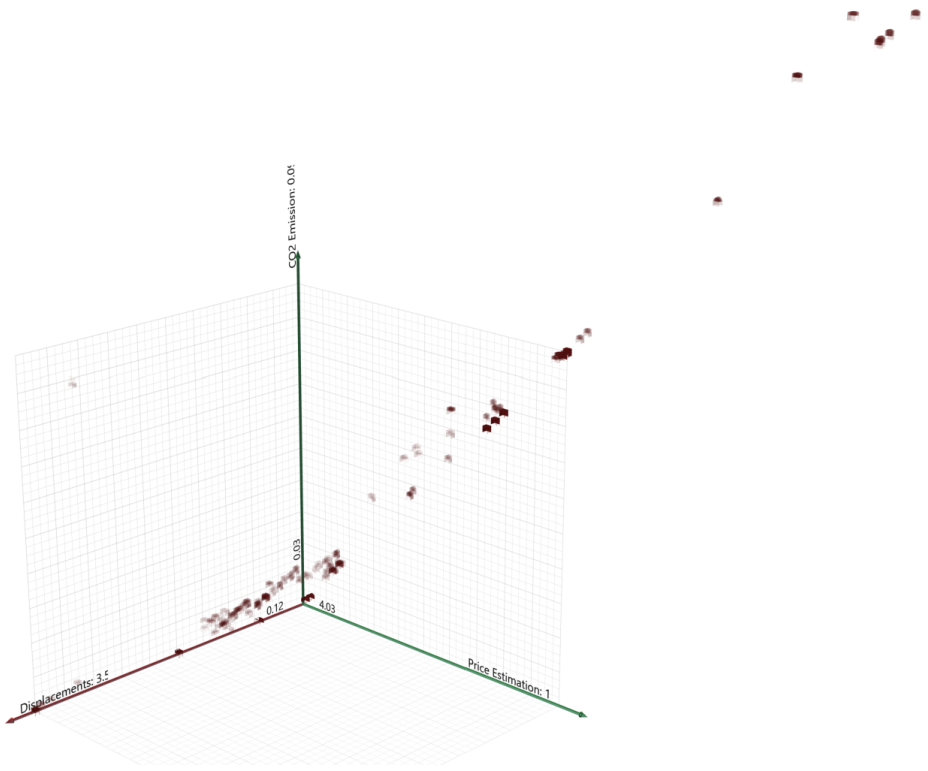


Figure C.8: The transparent shapes represent Elite configuration, whereas the red is the Pareto-front solutions.

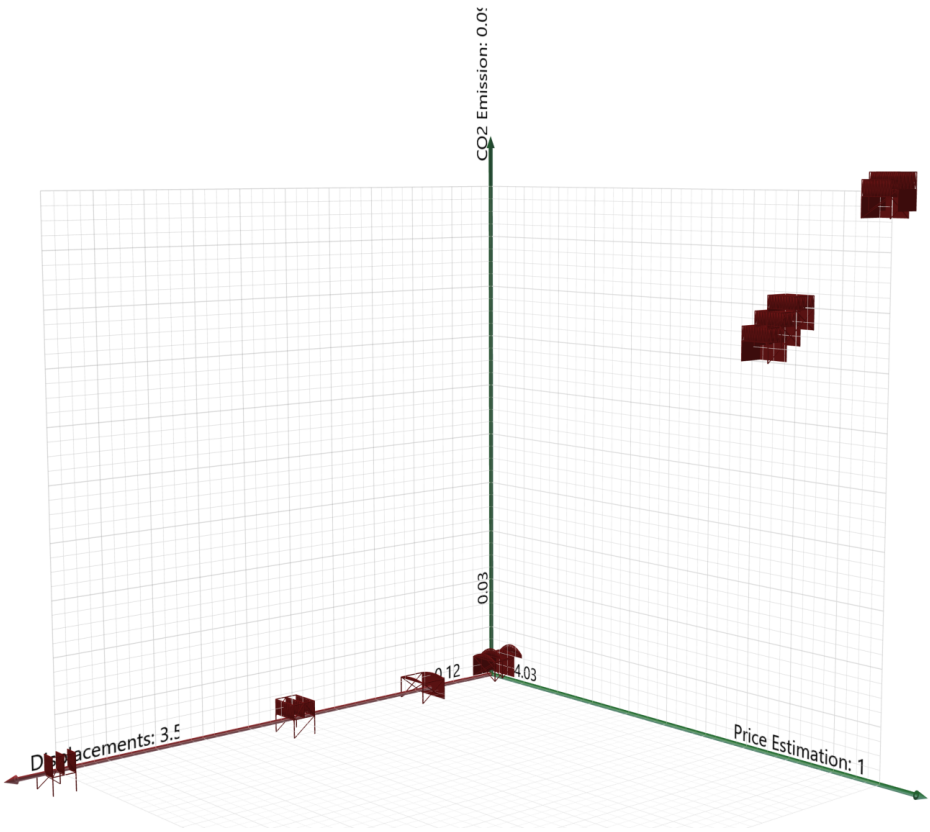


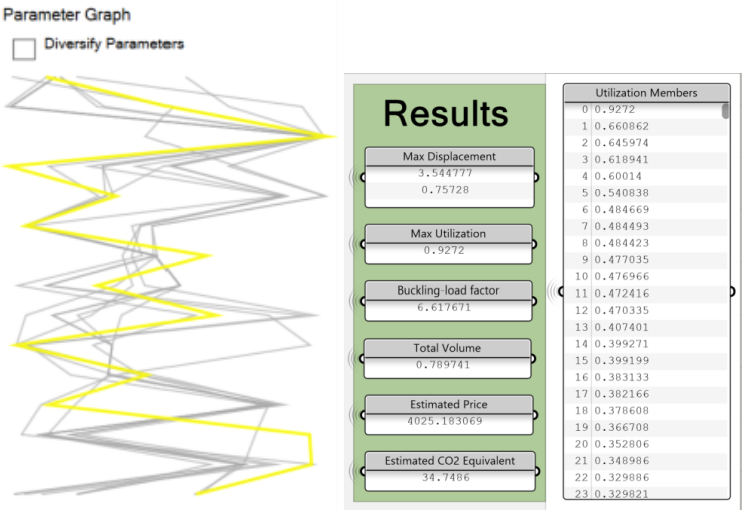
Figure C.9: Close up of population field to give an indication of configurations generated.

C.5 Objective And Parameter Results from MOO Glulam

The document-explanation in Appendix C.2 applies for glulam results as well. Although, instead of multiplying the price and CO_2 values with 1000, it needs to be multiplied by 100 to coincide with the results in Appendix C.6.

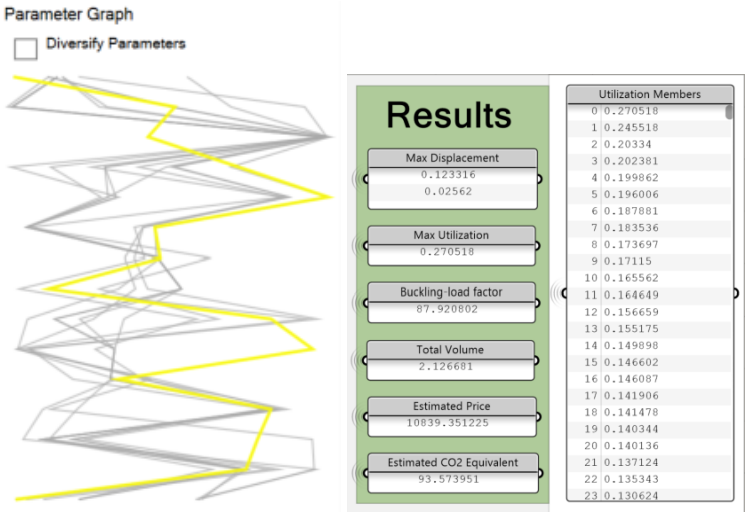
The results are attached as ZIP-file, file named C.5. *GlulamOptimisationResults*.

C.6 Results for Glulam Configurations



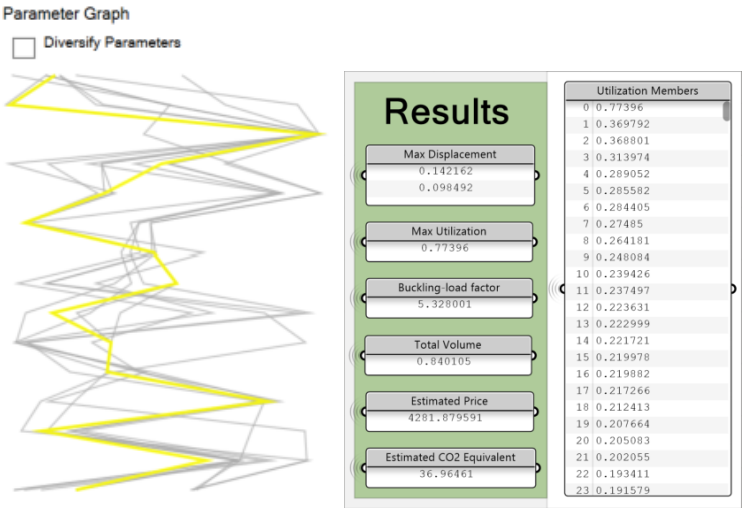
(a) Parameter graph **(b)** Results.

Figure C.10: Results for the configuration with the lowest value for price and CO_2 equivalents.



(a) Parameter graph **(b)** Results

Figure C.11: Results for the configuration with the lowest displacement.

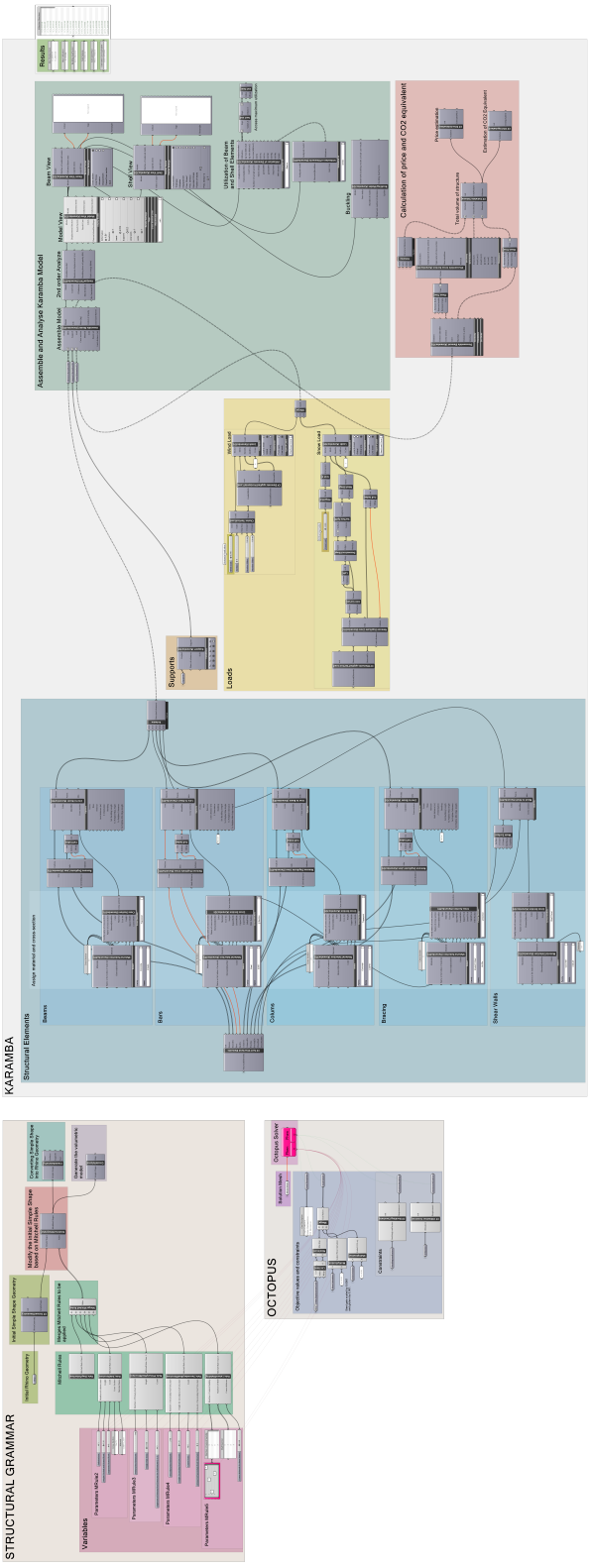


(a) Parameter graph **(b)** Results

Figure C.12: Results for the configuration chosen from the designer.

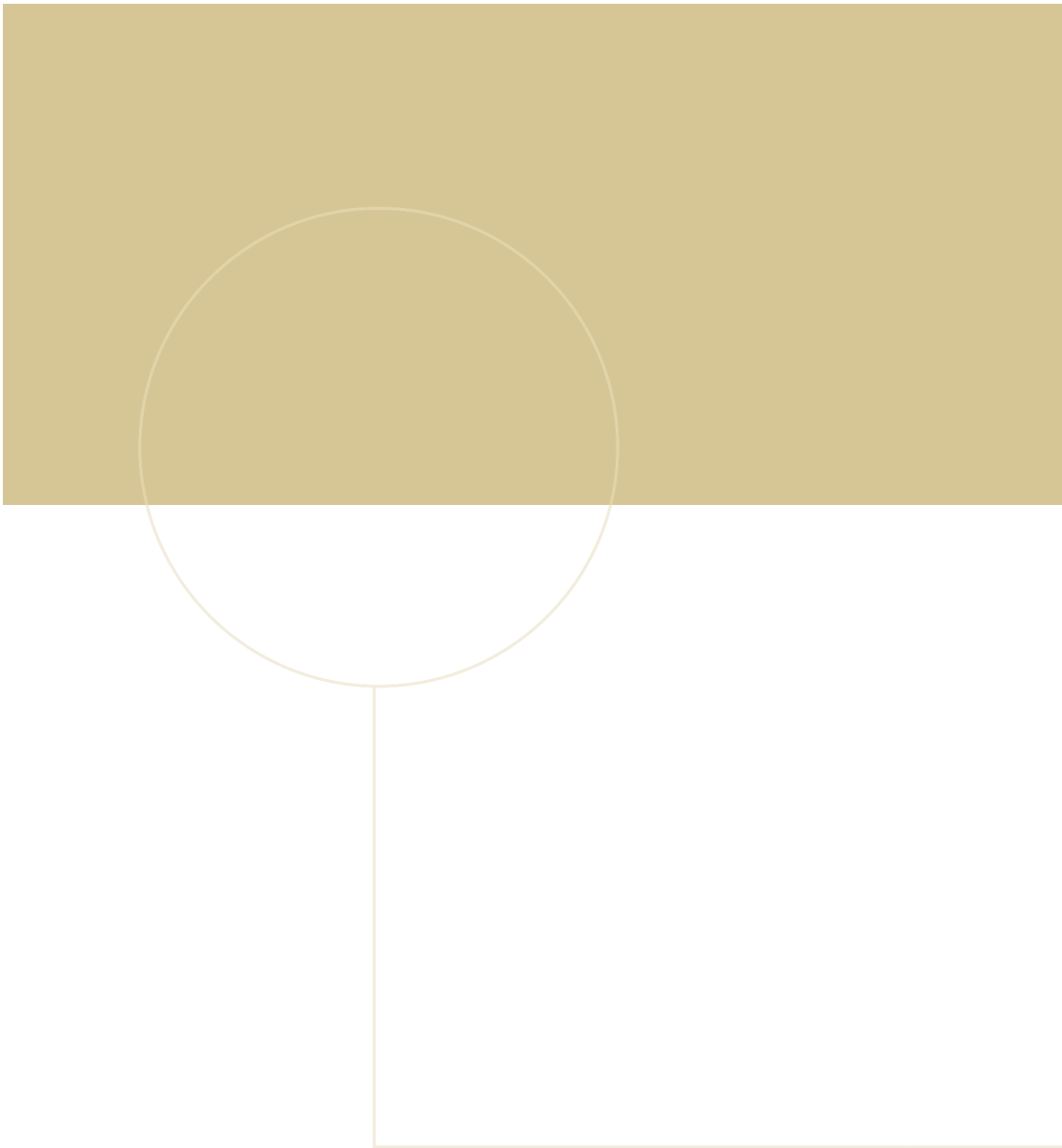
D Grasshopper

Grasshopper file is attached as a ZIP-file.



E Video

Video demonstrating functional grammar is attached as a ZIP-file.



NTNU

Kunnskap for en bedre verden