EXPLORATIVE ANALYSIS AND STRUCTURED SPECULATION: RETHINKING COMPUTATIONAL DESIGN AS AN APPROACH FOR ARCHITECTURAL FORM EXPLORATION

Ferro Yudistira[1]*
[1]) Department of Architecture, Universitas Indonesia, Depok, Indonesia
*e-mail: f.yudistira@ui.ac.id

ABSTRACT

This study examines the potential of computational thinking principles as an approach in architectural form explorations. This study proposes an alternative perspective in the computational design approach, which often emphasizes automation and optimization in problem-solving processes. Current research and discussion in computational design also tend to focus on algorithms as the main idea, overlooking other computational thinking principles. Using House IV—designed by Peter Eisenman—as a case, this study seeks to examine the possibilities of implementing computational thinking principles in a speculative form-making process. House IV was selected due to two main aspects related to its design process: 1) the involvement of rules and procedures, and; 2) the focus on architectural formal exploration instead of program. The processes consist of an analytical and a generative phase. Analytical phases employ three computational thinking principles—decomposition, pattern recognition, and generalization/abstraction—to deconstruct and identify the essential parts and underlying set of rules from House IV. The generative phase utilizes understanding from analysis – in the form of generalized rules – to build an algorithm to reconstruct variations of forms. This approach open possibilities for exploration/speculation of new or unique formal iterations while keeping the character and aesthetic quality of the House IV. Reflection on the processes highlights the importance of explorative analysis and structured speculation in architectural design methods. The findings show the importance of explorative aspect in the analytical phase, along with traceability and explicitness in generative phase of design process. This study proposes a more holistic implementation of computational thinking principles in speculative form-making exploration. This study solely focuses on architectural form exploration, without any consideration for other elements such as space, context and program/function.

Keywords: *Computational Thinking, Computational Design, Explorative Analysis, Traceability, Structured Speculation*

INTRODUCTION

The advancement of digital technology opens a range of possibilities in architectural design that were previously difficult or even impossible to perform through conventional techniques. However, this technology is still often used merely as a tool for emulating conventional practice (Burry, 2011), especially in the production phase, mainly for creating a drawing/drafting and making a model/maquette.

An approach called Computational Design (CD) has been developed to revolutionize the traditional design process based on the idea of computation (Caetano, Santos and Leitao, 2019; Knippers, 2021). As a design approach, CD is closely related to but does not necessarily involve digital tools. Simple use of a certain CAD tool without explicit implementation of a computational concept is not a CD. A computational concept that is often associated with CD is the algorithm. Deterministic nature of algorithms introduced crucial changes in architectural design thinking. Architects/designers no longer design by directly drawing the shape. Instead, they develop an algorithm that can generate various forms based on input given to it (Oxman, 2017). Algorithms usually pair with an objective related to automation or optimization, in which the algorithm is developed to find the most efficient way to solve complex design problems (Çolakoğlu and Yazar, 2007; Koyama and Igarashi, 2018). However, an algorithm contains other elements such as values, rules and conditions. A question then can be asked about the involvement of other computational principles in extracting and/or formulating these elements within the CD approaches. In other words, the current state of CD-based approach in architectural design discussion still does not fully embrace the full and comprehensive principles of Computational Thinking (CT).

This study aims to trace back and reconnect the idea of CD with the principles of CT and then position it as an approach in speculative architectural design process. Based on its etymology, speculation is derived from the Latin word *specere* with further derivative *speculari* and then *speculatio*, which can be translated as 'to observe', or 'meditate and even theorize' (Partridge, 1958). It is arguably a contemplative act performed to search for a meaningful or new answer (hence the 'theorize or theoretical' meaning, since the answer has never been seen in practice before). In this study, speculation related to implementation of CT in the process of exploring or imagining new/unique, alternative and unexpected design results. Therefore, it is different from a probability-based process in which all options are already known beforehand, but with different percentages (such as in the classic case of coin toss or dice rolls). Dunne and Raby (2013) position probable results in the first level (most common) regarding the connection between the present and future, which describes probable as a situation or result that is likely to happen unless some extreme event occurs, while speculation is at the third level in which future and imaginative/alternative scenarios were proposed based on critical reflection of today's situation.

However, in CT-based design approach, this unique and/or unexpected result does not come from random trials and errors; instead, it is achieved through a structured process. It is a process that is akin to exploratory creativity (Du Sautoy, 2019), in which we identify and map what we have in the present (in a particular field

or context), and then exhaustively explore the limits and its potentials while still bound by certain rules or constraints. In other words, CT-based design speculation is seen as a structured and declarative/intentional act instead of something vague that purely relies on intuition (Burry, 2011). Furthermore, speculation in design is not merely about imagining the future, but also how it helps us to critically assess our understanding of the past and present (Dunne and Raby, 2013). It is then why having a declarative/intentional and structured process is crucial; so, we can trace back the (alternative/unexpected) result with the process, to analyse and deeply understand the pattern/logic and rules between the two.

This study aims to seek the possibility to implement four principles of CT—decomposition, pattern recognition, generalization/abstraction, and algorithm—in a speculative and creative process of architectural form exploration. The involvement of other CT principles such as decomposition and pattern recognition will provide an insight into the role of certain parts (geometrical components) along with underlying rules/structures (variable, constant, expression) of the algorithm. The results and discussion of the investigation demonstrate the value of explorative analysis along with traceability and structured exploration. This finding related to the effort in developing more holistic approaches of CT-based architectural design process, which allows the analytical and generative process to be conducted with an explicit logic, rules and structure while still open for a unique formal exploration/speculation.

THEORY / RESEARCH METHODS

Computation in Architectural Design: Production, Automation and Speculation

The involvement of digital technology – especially of computers – in architectural design is not a new thing. However, there is still a tendency to utilize computers to emulate conventional working practice (Burry, 2011). Computers are used merely as a tool that aids (hence the utilization of 'computer-aided') in traditional design approaches, particularly in production-related processes such as drafting and model making (McGill, 2001; De Boissieu, 2022).

The involvement of computers can be seen through the lens of computational capabilities of the hardware, software and/or infrastructure (Gopsill *et al.*, 2023), or as a thinking paradigm (De Boissieu, 2022). Utilization of computers as a digital tool in the design process does not necessarily involve a computational thinking approach. Conversely, computational design does not always involve digital tools and media (McGill, 2001; Caetano, Santos and Leitao, 2019). However, the adoption of computational thinking that takes advantage of the power of computational tools will enhance its potential to expand new boundaries in architectural design (De Boissieu and Watts, 2021). The question then is in which direction the computational approach is positioned in architectural design.

Computational approaches in architectural design can be divided into two directions: automation or speculation (Burry, 2011). Both arguably involve an exploration process that may be difficult to perform through traditional processes. However, the automation perspective tends to focus more on efficiency and

optimization. Computational design moves beyond utilization of digital tools to create representation/visualization, into a method for finding the most efficient solutions to solve complex design problems through powered by computational power, algorithms, and/or artificial intelligence (Çolakoğlu and Yazar, 2007; Koyama and Igarashi, 2018; Sunarya, 2022; Maksoud, 2024; Ploennigs and Berger, 2024). Some parties connect this idea of automation and optimization with fabrication and utilization of material in modern industrialization or manufacture (Kyratsis, 2020; Thomsen, 2022; Haghnazar, 2024; Sun *et al.*, 2024; Önalan *et al.*, 2025).

This study takes an alternative perspective, in which computation power and approach is directed towards creativity and speculation instead of efficiency and optimisation. Previous attempts in this direction lean more towards the possibility of creating new form/shape that arguably difficult to create through conventional/manual method, such as demonstrated by Greg Lynn exploration of forces and time-based technique of 'animate form' (Lynn, 1999) and *splines*-based curves and shapes of Frank Gehry's metal fish statue (Carpo, 2023). However, instead of immediately jumping into computational design (CD), this study revisits the idea of computational thinking (CT) and explores the possibilities to adopt its principles in the architectural design process.

Retrace the Role of Computational Thinking Principles in Computational Design

The current discussion and practice regarding CD-based approach in architecture is mostly focused on the idea of algorithms, which is a set of rules, information and instructions/operations created by architects to generate the form (Ahlquist and Menges, 2011; Oxman, 2017; De Boissieu, 2022). The idea of algorithms 'force' architect to be explicit and deterministic in their intention instead of vague and ambiguous. Some even argue that algorithm – or the ability to think algorithmically – in CD is more important than computer as a tool, which makes utilization of computers not mandatory in CD (McGill, 2001; Çolakoğlu and Yazar, 2007).

Another discourse in CD within the architecture field is related to an effort to clarify its definition by dividing CD into several subsets. Caetano, Santos and Leitão (2019) argue that CD can be divided into three main subfields, which are: Parametric Design (PD), Generative Design (GD), and algorithmic design (AD). (De Boissieu, 2022) proposes a similar argument; she argues that knowledge regarding CD and its subset is important to understand their position in design practice, along with roles and skillsets related to them. It can be argued that both emphasize the importance of understanding CD and its subsets/subfields to fully unlock its potential in design practice.

Instead of branching out the understanding of CD into several design-related subfields, this study focuses on tracing back the connection between CD and computational thinking (CT). This study identifies four main principles of CT, which are decomposition, pattern recognition, abstraction and algorithms, and discusses their possible implementation in architectural design thinking and process. Some authors, for example, Mikaelsson (2022), along with (Kelly and Gero, 2021), have mentioned these concepts in their study. However, the discussion has not fully developed around this concept and still tends to solely focus on algorithms. Algorithm is indeed one of

the computational thinking principles. However, narrowing the focus in algorithms can be questionable in an explorative and creative-based field such as design (including architecture), since we can trap ourselves into repeating the same algorithm (Du Sautoy, 2019).

This study then expands the discussion of computational approaches in architectural design through the inclusion of other CT-based principles. This study does not aim to negate the role of algorithms, but it seeks to introduce a more holistic CT-related approach in the architectural design process. The aim is to explore the possibilities to adopt these CT-based principles as a method in the creative making process instead of efficient problem-solving techniques.

Method

This study adopts the core concepts/principles of CT to deconstruct and reconstruct a certain architectural form. CT is commonly understood as a thinking process and skill to propose solutions for a problem in a form that can be executed by both humans and machines (Wing, 2006; Buitrago-Flórez *et al.*, 2021; Wu *et al.*, 2024). However, CT nowadays is positioned as a fundamental skill for everyone and applicable in various contexts/situations (Mills *et al.*, 2021). Chytas et al. (2022) explore the implementation of CT principles, such as algorithmic thinking and pattern recognition, in helping youth to create more complex and expressive design projects.

Hence, this study argues that CT does not necessarily have to be about problem-solving, or the 'problem' itself can be interpreted in different ways. The problem is a deconstruction and reconstruction process of a certain architectural form. The architectural form used as the case in this study is House IV, designed by Peter Eisenman, which is selected based on two reasons:

1. The involvement of 'rules' and 'step-by-step procedures in the design process (as explicitly stated on the Eiseman Architect official website)
2. Emphasize exploration of architectural form, while shifting the focus away from other traditional notions such as function and meaning. This emphasizes allowing this study to fully focus on how complex architectural form can be constructed (and reconstructed) through traceable and structured processes.

Both analytical (deconstruction) and generative (reconstruction) phases based on the official information – mainly drawings and diagrams – provided by House IV (accessed through the official website of Eisenman Architect). Both phases will be conducted based on a certain set of CT principles. There are several concepts/principles that frequently appear in the CT-related discussion, which are: decomposition, pattern recognition, generalization/abstraction, and algorithm (Beecher, 2017; De Jesus and Martinez, 2020; Buitrago-Flórez *et al.*, 2021; Wu *et al.*, 2024).

Decomposition, pattern recognition and abstraction are arguably related to the analytical phase. Decomposition is a process of breaking down problems or data into smaller parts or components (Riley and Hunt, 2014; Beecher, 2017; De Jesus and Martinez, 2020). These parts can include a process and/or some form of relationship

between the parts. In this study, decomposition aims to break down and acquire an initial understanding of the parts that construct House IV.

After decomposing the parts, pattern recognition is a process to find similarities or patterns between these parts (De Jesus and Martinez, 2020). It may not immediately be visible, but it is a crucial phase for making a more powerful understanding of the problem/objective (Beecher, 2017). In pattern recognition, this study further analyses the parts of House IV to search for similarity and repetition, specifically in how these parts being multiplied and placed or configured in the forming process of House IV. For example, forming process of House IV involves a certain state of transformation called '3x3 matrix', in which different geometrical elements are multiplied and placed and configured within the same matrix.

Abstraction is a way to find and express or formulate important features and characteristics of the pattern, while hiding or deemphasizing less important details (Riley and Hunt, 2014; Beecher, 2017). In other words, abstraction involves the act of generalization; it simplifies the complexity into some form that could work in different situations (with similar pattern/repetition). This phase extracts relevant value/data related to the parts and generalizes the relation/condition between them (based on pattern recognition). The aim is to formulate a set of rules (value, variable, expression, condition) that are relevant to building the algorithm of House IV.

While decomposition, pattern recognition and abstraction/generalization are about analysis, an algorithm is the generative phase of the process (reconstruction). An algorithm is a procedure or set of instructions/operations that are structured to achieve a certain objective (Ahlquist and Menges, 2011; De Jesus and Martinez, 2020). In this study, algorithm development is based on data/information and understanding from analysis and aims to generate architectural forms that resemble House IV. This study, however, does not aim to replicate the same form as House IV, nor is it trying to fully translate the step-by-step procedural transformation – as Eisenman illustrates through a series of diagrams on his website – into a CT-based process. This study positions House IV as a reference; a source to extract a set of elements and transformation rules, which are used to develop a generative algorithm. These set of transformational rules come in the form of expressions that consist of variable (changeable) and constant (static/permanent), in which the variable itself can also have a specific range. These rules allowing the algorithm to be further modified to generate different sets/iterations of architectural forms that still resemble similar characteristics to House IV. The objective then leans more towards exploration/speculation instead of replication.

This study uses Grasshopper (GH) to develop the algorithm. Grasshopper was used due to its capability to 'record' the whole form construction and transformation process. It is also relatively easy to customize the GH component to create a specific expression and/or condition. However, there is a concern related to potential 'bias' of the tool, in which the algorithm development becomes fully rely on or determined by the GH component instead of the pattern and abstraction identified through the analysis. Hence, to help in reducing this potential bias, the reconstruction process will involve a pseudocode.

Pseudocode is a way to describe an algorithm without the use of any specific programming language (Beecher, 2017). There is no single convention or standard

regarding syntax to write pseudocode. However, pseudocode still involves terms/concepts/elements and structure that are commonly/frequently and widely used in programming languages such as variables, assignment, selections/conditions, and repetition/loops. In other words, pseudocode need to involve some general principles/structure related to programming logic – to make it easy to understand by wide audience of reader, but still flexible enough to be adopted and adjusted for specific case/context of the algorithm. Therefore, pseudocode in this study contains two layers; the first layer shows the general structure of the algorithm, while the second layer is related to the contextual reconstruction process of House IV.

As an example, Figure 1 shows a pseudocode that follows general programming structure and logic that consists of: 1) variable declaration; 2) assignment, and; 3) function execution. However, this general structure contains specific value and instruction related to the early phase of House IV reconstruction process, which is: 1) initiation and construction of set of points (related to variable declarations and assignment); 2) translation process (move and copy) of this set of points (related to function execution). Figure 2 shows similar examples with more complex structure for the later phase of the reconstruction process. In terms of the general structure, it consists of: 1) index-based item selection and list creation; 2) function execution; 3) conditional selection (which has its own block of code). Related to the reconstruction process of House IV, it contains: 1) selection and reconfiguration of points; 2) surfaces creation and modification; 3) further form development and modification (depending on the condition). Both examples serve as an illustration of how pseudocode helps in preserving explicitness, clarifying and structuring the reconstruction process of House IV.

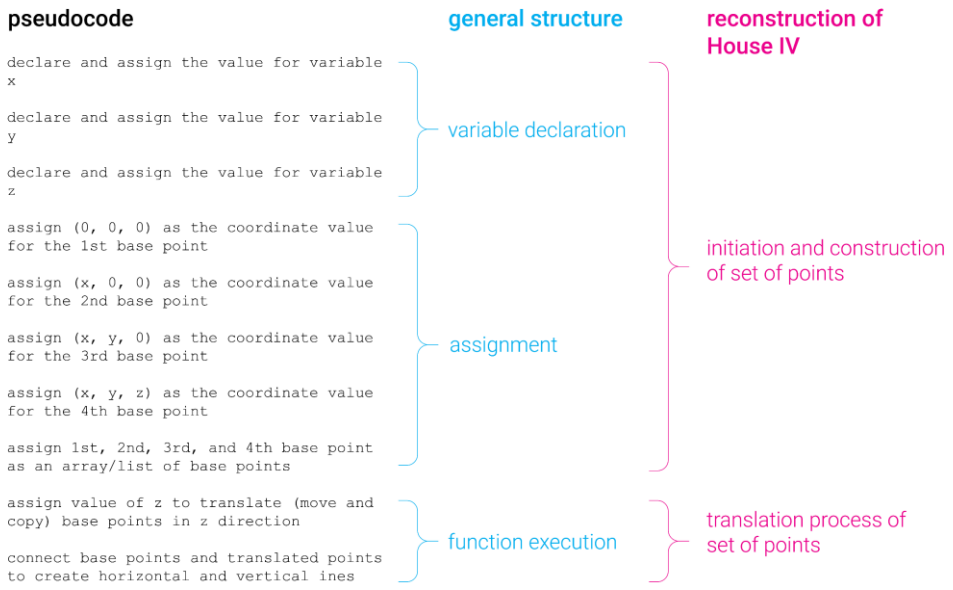


Figure 1. Example of Pseudocode

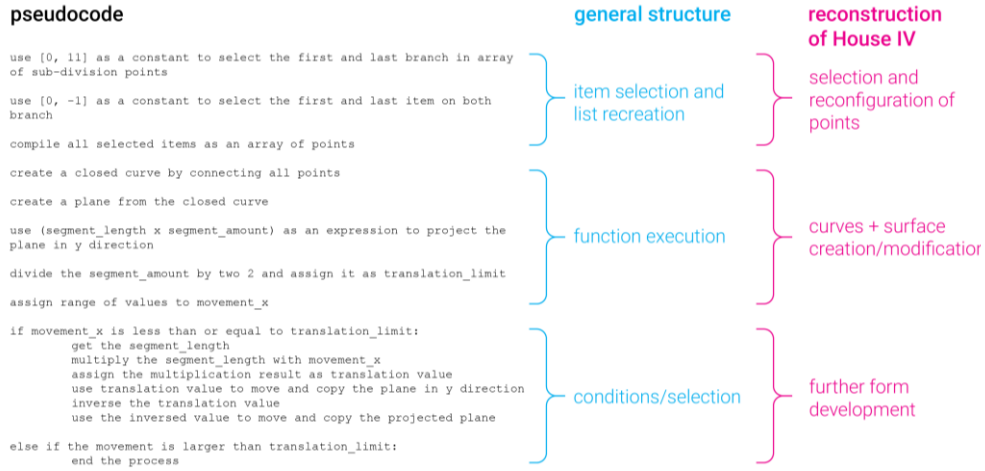


Figure 2. Example of Pseudocode

Similar approach in dismantling the components of existing architecture and extracting its compositional rules and principles can be seen on various study that used Shape Grammar (SG). Mamoli (2020) use SG to analyze the rules and design principles of ancient library, while Wang (2024) implement SG-based approach to uncover the logic and organizational rules of Vernacular house. Even though similar in the overall procedure, SG tend to 'visualize' the logic through 2D-pattern, while this study aims to more generalized rules in the form of textual (pseudocode) and mathematical description (variable, constant and expression). Procedure in SG also quite strict, involving finite sets of shapes, labels, rules, and initial shapes (S, L, R, and I), while the CT-based procedure in this study is more flexible (depending on the objective of the process). Overall theoretical framework of this study can be illustrated through Figure 3 as follows:

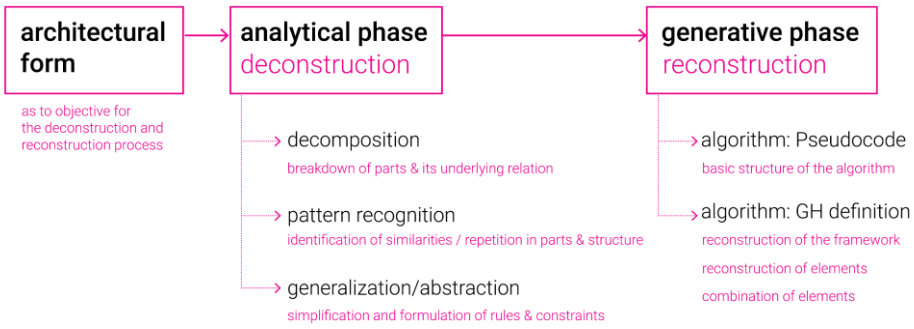


Figure 3. Diagram of Theoretical framework

RESULTS AND DISCUSSION

This section discusses the analytical and generative phase of House IV, along with the reflection from both as a CT-based form-making. Due to the explicit nature of the process, it required very long descriptions to fully explain the whole process (for example, a simple cube required eight points, each of which has its x, y, z coordinates). Hence, this section will pick certain examples or parts that represent the significance of each phase, making the discussion more concise without losing the whole picture

Analytical Phase – Decomposition: Set of Planes, Lines and Volumes

Decomposition aims to break down and gain understanding of both the elements/parts and structure/configuration that construct House IV. Overall, the form of House IV consists of three main elements, which are: 1) a set of planar surfaces or planes; 2) a set of boxes or volumes; 3) and set of columns and beams, that can be simplified as lines. These elements may sound familiar or generic and can be found in any other architectural form. However, the aim of decomposition is not to immediately grab the whole complexity from the start. Instead, it aims to divide the whole into more manageable parts. These parts can then be further analyzed, both individually and in how they connect as a whole.

To analyze the relational complexity between these elements, it is then necessary to identify the underlying 'structure' of the form. This study identifies the presence of a 'construction framework', which is a configuration of points and unseen lines that act as references for constructing elements into the whole. Hence, the next section will analyze the configuration and rules of this construction-framework, and how it governs the placement and transformation of the parts (as the process in constructing the whole).

Analytical Phase – Pattern Recognition: State of Transformation and Construction Framework

The pattern recognition phase aims to analyze and identify the similarity and repetition of House IV elements, specifically related to their initiation, placement and transformation in the framework. This analysis is based on a series of transformations that are illustrated by Eisenman through a set of diagrams. These states of transformation arguably represent both the articulation of the construction framework and the initiation and placement of elements on this framework. The analysis identifies three crucial states, which are based on how the state repeats across all three elements transformation (even though the position of a certain state in the series/sequence of transformational diagram for each element can be different). These crucial states consist of: 1) the initial cube/box; 2) the 3x3 matrices; 3) the extension (Figure 4).

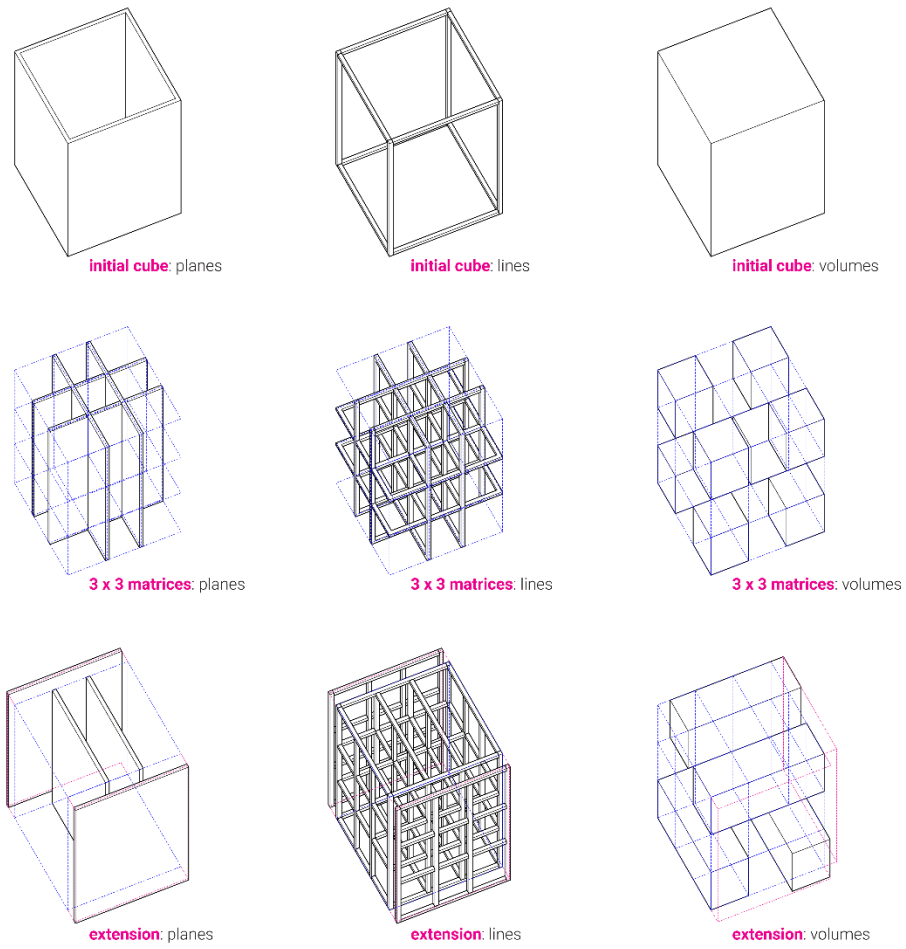


Figure 4. Three Crucial States of Transformations

The initial 'cube / box'

The first crucial state is the 'initial state', which is the starting point for all elements' transformation. In other words, all elements are initiated in a state that represents a simple cube or box. Therefore, this cube/box is the default condition of the framework, in which the next articulation will be based on its properties, along with the elements (planes, volumes, lines) being placed within it.

The '3 x 3 matrices/division'

The next crucial state is the '3 x 3 matrices/division'. In this state, the construction framework has been further articulated and divided into three segments for each horizontal and vertical edge. This articulation results in 3 x 3 matrices of division on all sides/faces. These 3 x 3 matrices introduce various compositional complexities, since each element can be placed and configured differently based on the properties and characteristics of each element (within the same framework).

The 'extension'

The last crucial state is the extension. Here, the placement and configuration of the elements are being extended to the 'outside', but the shape of the elements is still constructed based on the framework. The length of the extension is also arguably still related to the 3 x 3 matrices (will be further explained in the next section). The articulation of the planes and lines elements in this state is similar, in which both are extended to the side of the framework. The articulation of the volumes is a bit different, since the boxes are being extended not only to the side but also to the back.

The previous section explains the similarities of the three states of transformation that repeatedly occur across different elements. These states are then used as a basis to further analyze the series of transformational processes of each element. Specifically, the next section analyzes the similarity and repetition in the transformation that happens between these crucial states. Overall, the change and articulation that occur in this process consist of initial division, translation, and (sub)division.

Initial division

After the first initiation (the initial cube/box form of the construction framework), there is an articulation process that divides the construction framework at its mid-point, resulting in 2 x 2 matrices. A similar process of division happens to planes and lines (Figures 5 and 6). After the division, the planes and lines elements are 're-initiated' based on these matrices. Different division processes happen for the volumes, since they immediately start with 3 x 3 divisions. However, the 2 x 2 and 3 x 3 divisions are arguably closely connected in both forming the construction framework (as later can be seen in the generalization/abstraction phase) as well as transforming the elements based on this framework.

Translation (move & copy/duplicate)

Another repetition that happens between crucial states is translation in the form of movement. This movement occurs along with some form of duplication and union/subtraction operation. Between the initial cube and 3x3 matrices, this translation-movement + duplication occurs on both planes and lines. The vertical planes are being moved and copied in horizontal direction or axes (x and y), while the lines are being moved and duplicated in horizontal and vertical direction (x, y, and z). For the volumes, this translation and duplication are combined with a subtraction operation, which reduces and divides the initial cube into several smaller parts.

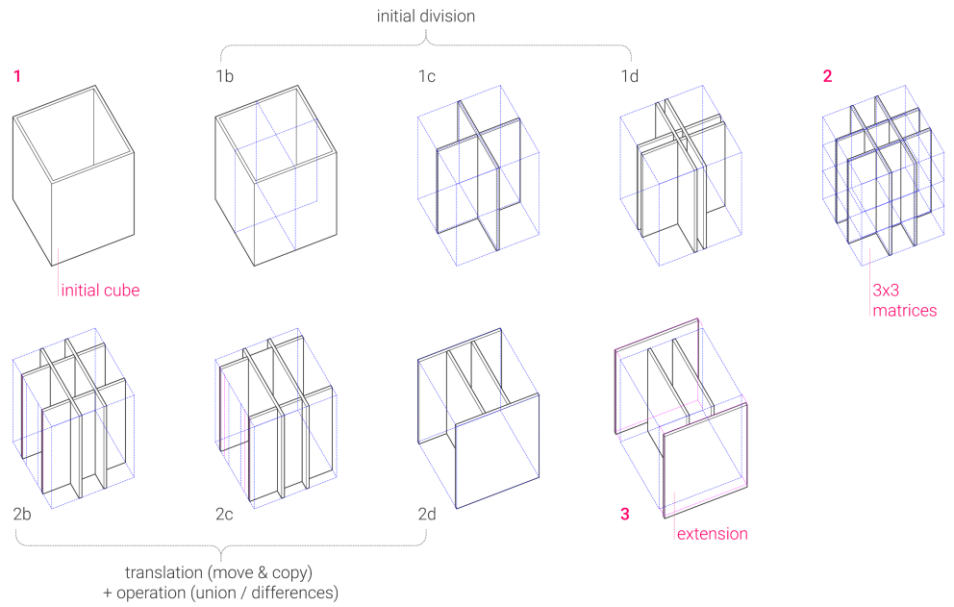


Figure 5. Transformation between Crucial State: Planes
Source: Reproduced and edited from Eisenman Architect Diagram, 1971

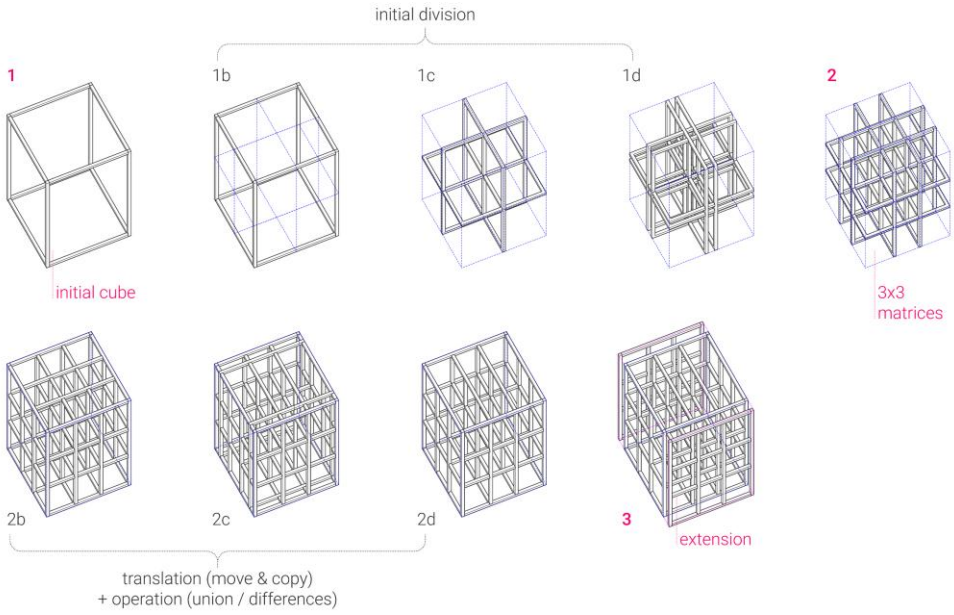


Figure 6. Transformation between Crucial State: Lines
Source: Reproduced and edited from Eisenman Architect Diagram, 1971

A similar transformation process occurs between the 3 x 3 matrices state and the extension state. Overall, it still involves the combination of movement, re-

initiation, and operation (union and/or subtraction). For planes, the movement of elements continues in one horizontal direction (x-axis) until it extends a bit further from the construction framework. For lines, the movements stop at the edge/perimeters of the construction framework, but then some elements were re-initiated and moved in the x direction until the extension position. For volumes, a series of movements combined with union and/subtraction, which result in a pair of rectangular boxes with different orientations on the top and bottom parts. One of these rectangular volumes is then moved on the x and y axes until it reaches the extension state.

Relation between Translation and (Sub)Division

Based on the previous analysis, this study argues that there is a connection between the translation-movement of the elements and the division of the framework. This argument is based on a repeated range of movement that occurs across all elements (such as after the initial division, before reaching the edge/perimeters, and the extended movement). This pattern of repetition indicates the necessity to further divide the 3 x 3 matrices into smaller subdivisions (Figure 7). Even though this subdivision is not as obvious as the 3 x 3 matrices, it is still an important pattern that needs to be generalized/abstracted into a set of values and rules.

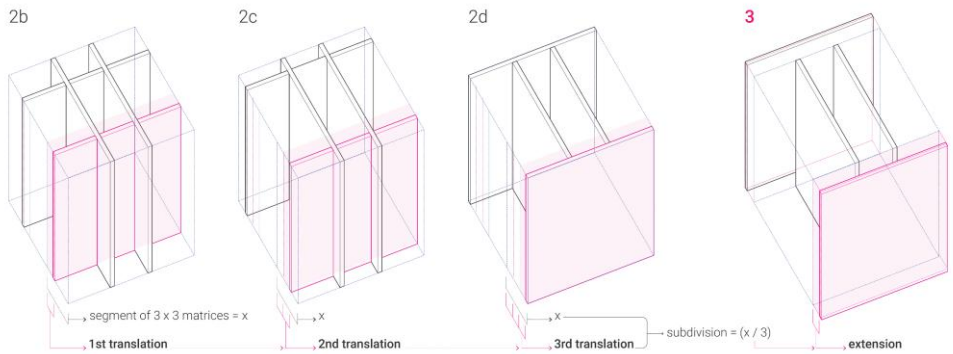


Figure 7. Relation between Translation and (Sub)division

Combination through Union/Addition and Difference/Subtraction

The last pattern or repetition that occurs in the state of transformation is the combination of elements after the extension state (Figure 8). At this stage, the transformation presents in the form of a Boolean operation – specifically union and difference/subtraction (or combination of both) – and involves different elements (hence the terms 'combination' instead of 'composition/configuration). This transformation results in a complex architectural form, a unity that still retains properties and characteristics of its parts (thin accentuation of a plane, frame-like expressions of lines, and volumetric impression of the box) (Figure 9).

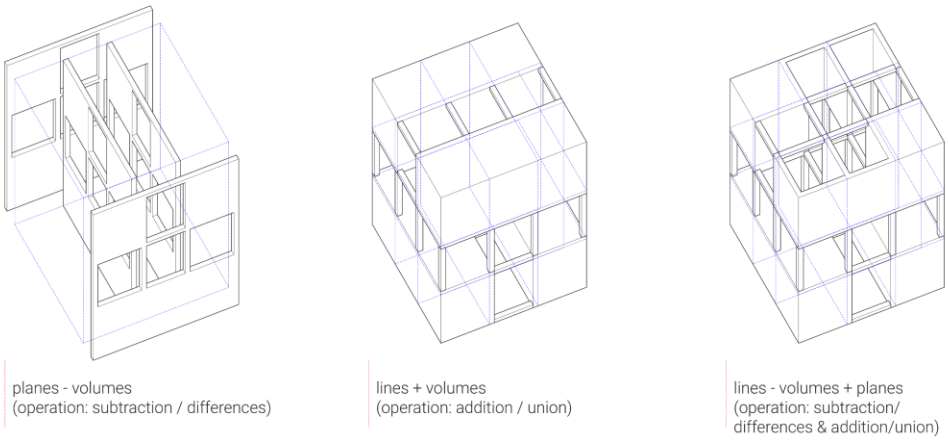


Figure 8. Combination Across Different Elements
Source: Reproduced and edited from Eisenman Architect Diagram, 1971

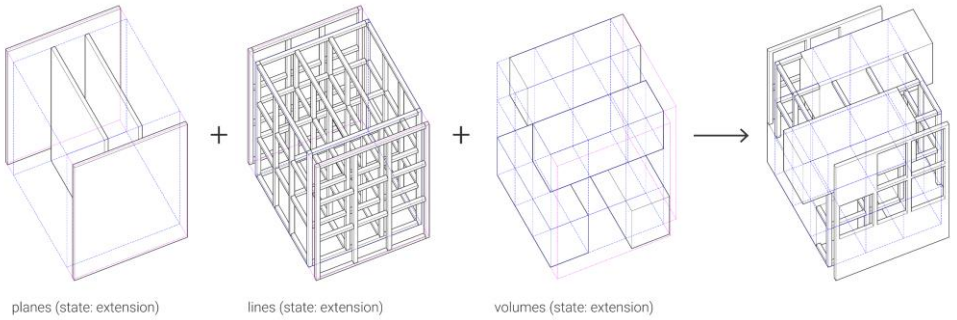


Figure 9. Combination Across Different Elements (Based on the Extension State)
Source: Reproduced and edited from Eisenman Architect Diagram, 1971

Analysis in this section shows the importance of understanding the underlying structure and rules that govern the initiation and configuration of the whole architectural form. This structure acts as an anchor that contains the quality and/or character of the formal/spatial expression, while still open for further abstraction (to extract a set of generalized data and explicit logic to develop the algorithm)

Analytical Phase – Generalization/Abstraction: Variables, Expressions and Conditions

This section further analyzes the properties of the framework (which has been mentioned in the decomposition phase), based on understanding acquired from the pattern recognition phase. The aim is to simplify the series of transformations that occur into a set of data/value and rules that are reusable for different elements and scenarios. The same set of data/value will be used to build the algorithm in the generative phase. Based on the analysis, the process to extract this set of data and rules

is divided into two parts: 1) Extracting variables, coefficients and constants; 2) formulating the expression.

Extracting relevant variable

This phase aims to define relevant variables as data containers where required values for the transformation can be assigned. Based on the identified pattern, there are three sets of variables required to be included in the algorithm, which are:

1. Variables related to the length of the framework.
2. Variables related to selection and (sub)division of the framework.
3. Variables related to translation and operation**.**

The first set determines the size of the construction framework. The second set related to the number of segments produced through the division process (related to the pattern/repetition of the transformation), and the selection of objects/elements in a data set required for further articulation. The last set determines the distance of translation (e.g. move) and/or operation (e.g. extrusion)**.**

These variables will take values that are required to perform a certain state of transformation. The process will rely more on relational values instead of 'hard-coded' values (values that are directly assigned to the variable). For example, the only size-related value that will be directly assigned to the variable is the main edges of the construction framework (length, width and height of the box). While others will take these values and calculate them along with a set of constants (in the form of an expression and/or a condition). This decision aims to create an abstraction in the form of a set of rules that consists of relation/dependency, expression and condition. This set of rules works with a different range of input values, while keeping a certain constraint related to the properties and characteristics of the architectural form.

Formulating the expression

As mentioned in the previous section, an expression is required to determine or calculate the values required in two parts of the transformation: selection and translation/operation. Selection is related to the result of the (sub)division process. It is crucial to choose and keep a specific set of elements while allowing a change in the number/amount of (sub)division elements. Based on the identified pattern, especially the 3 x 3 matrices state of transformation, two expressions – along with a set of constants – are formulated to select a set of points on vertical and horizontal lines.

The first expression is: (x / 2) + 1. This expression is related to the division of the main vertical line of the construction framework, in which x is a variable for the division/segment. This expression works along with two sets of constants: [0, 2] and [-1]. This rule selects a certain item in the data as follows:

1. [0, 2] select the first (index: 0) and third (index: 2) in the data
2. (x / 2) + 1 select the item after the middle item (in which the index is calculated using the x/2 expression).
3. [-1] select the last item (index: -1) in the data

These constants and expressions work as a rule to select a specific set of items (points in a vertical line) related to the division value. This selection is crucial to form the rows for the 3 x 3 matrices. It keeps the number of rows while allowing variation for the distance between the lines that construct the rows (which depend on the input values for the main edges, and the number of divisions) (Figure 10).

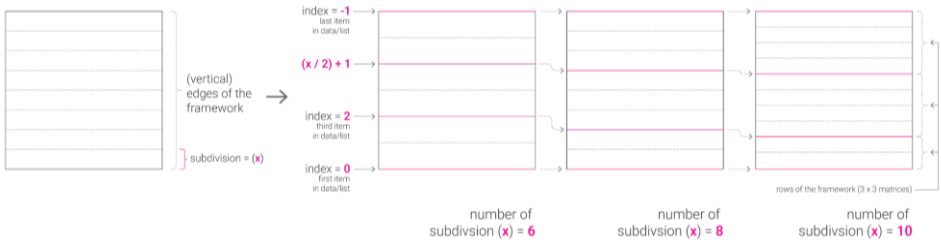


Figure 10. Abstraction for Division and Selection (Related to Rows of the Framework)

The second set of expressions is similar to the first, but with additional coefficients and constants. This set contains two expressions, which are: (x/2) – 1 and (x/2) + 1. These expressions are related to the division of the main horizontal line, and work with two sets of constants: [0, 1] and [-2, -1] (with identical index-based selection method). The selection results in the columns of the 3 x 3 matrices (Figure 11). It also stores the value of the division to be used for the extension process (which makes the extension state still based on the construction framework). The extension itself is a transformation that combines selection and translation, with similar approaches in connecting the values, expressions and constants with previous processes. This section shows that it is possible to formulate an explicit and specific mathematical abstraction from a formal/spatial configuration. This abstraction contains variable, constant, and expression that act as data, logic, and constraint to develop the algorithm.

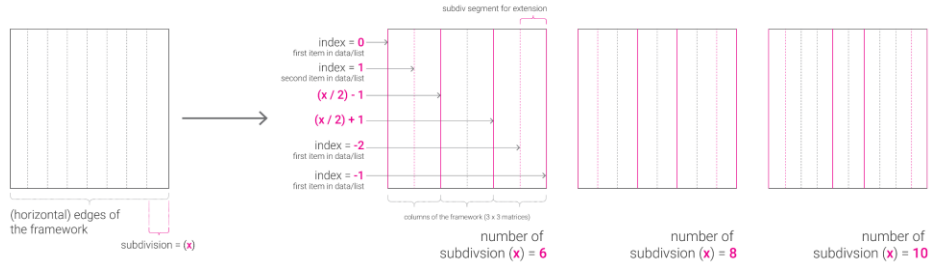


Figure 11. Abstraction for Division and Selection (Related to Columns of the Framework)

Generative Phase – Algorithm: Reconstruction of Framework and Elements

The generative phase takes understanding and information of House IV from the analytical phase and uses it to reconstruct new forms. The aim is not to replicate the same form of House IV. Instead, it adopts the set of values and rules from the analysis and uses them to develop and explore architectural forms that resemble the characteristics of House IV.

This phase started with building the base structure of the algorithm using pseudocode. As mentioned in the method section, pseudocode helps in clarifying and structuring the algorithm before it is further developed in GH. In general, the reconstruction process is divided into three main parts.

1. Remaking of the construction framework. This first part consists of three sub-parts, which are: 1) construction of the main edges; 2) division; 3) subdivision.
2. Reconstruction of elements based on the construction framework. This second part also contains three sub-parts, each for the reconstruction of the planes, volumes, and line elements.
3. Speculation/exploration of architectural form iterations through combination across elements in various states of transformation.

Due to the explicit nature of the process, describing all parts of the process requires very long explanations. Therefore, this section will focus on explaining the processes in the first part (especially the first and second sub-parts), the construction of the planes, and some examples of exploration in the third part. This selection is arguably sufficient to properly demonstrate both the essential features of each part and the comprehensiveness of the whole reconstruction process.

Construction framework: main points and lines/edges

The objective of this part is to create the main edges of the construction framework, which is identified as the initial cube/box. Figure 12 illustrates the set of pseudocode, GH definition and reconstruction progress of the framework as follows, this set of points and lines/edges is the foundation and/or references for further processes and developments.

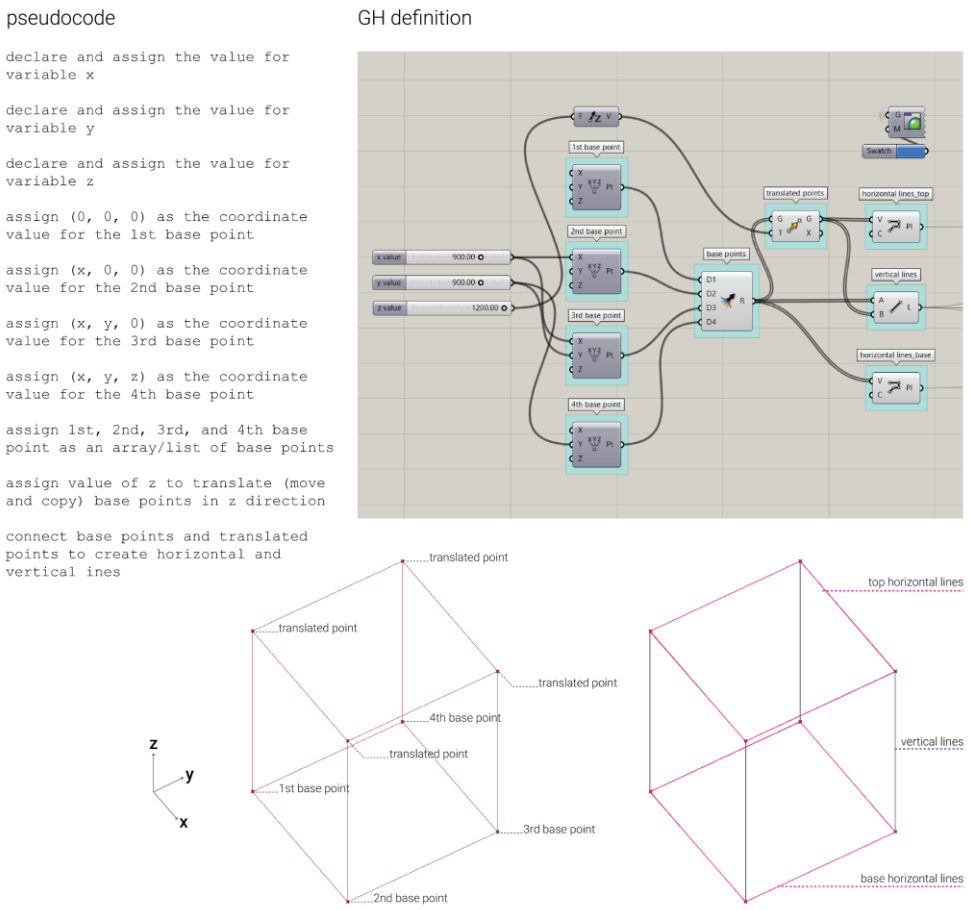


Figure 12. Pseudocode, GH Definition for the Main Points and Lines/Edges

Construction framework: (sub)division

This part aims to articulate the horizontal (rows) and vertical lines (columns) of the initial cube/box to form the '3 x 3 matrices/division'. This process only needs to be performed on one side (out of six sides of the cube), since articulation for the other side can be a replication or projection of the processed side. The reconstruction process was started with the subdivision. The lines for 3 x 3 matrices were then selected from the result of this subdivision. These subdivision and selection processes are done based on the set of rules (variables, constants, expressions) from the generalization/abstraction phase. Figures 13 and 14 illustrate the set of pseudocodes, GH definition and reconstruction progress for this phase as follows. An important note regarding the expression is that the input value should be an even number (since it will be divided by two). In other words, there is a domain of values required as input for the expression. The result of this division acts as the basis to reconstruct and/articulate the forms/elements (planes, lines, and volumes).

pseudocode

construction of horizontal division lines (rows of the framework)

use [0] as a constant to select the first item of the vertical lines

use [1] as a constant to select the second item of the vertical lines

assign value to divide both lines into an array of points and segments

connect the points to form an array of horizontal lines

use [0, 2] as a constant to select the first and third item in array of horizontal lines

use (x / 2) + 1 as an expression to select line after the middle index in array of horizontal lines

use [-1] as a constant to select the last item in array of horizontal lines

display the selected lines and hide the others

construction of horizontal division lines (columns of the framework)

use [0] as a constant to select the first item of the base horizontal lines

use [0] as a constant to select the first item of the top horizontal lines

assign value to divide both lines into an array of points and segments

connect the points to form an array of vertical lines

use [0, 1] as a constant to select the first and second item in array of vertical lines

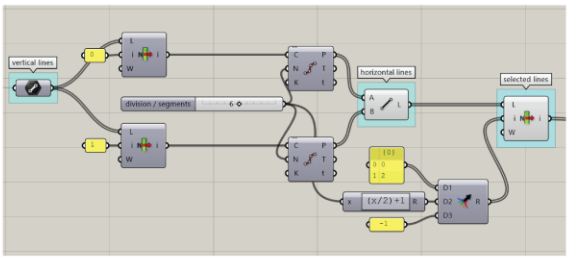
use (x / 2) - 1 as an expression to select line before the middle index in array of vertical lines

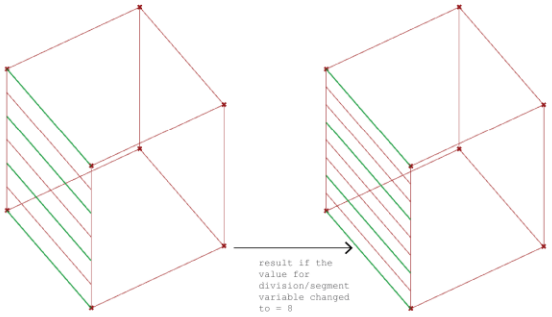
use (x / 2) + 1 as an expression to select line after the middle index in array of vertical lines

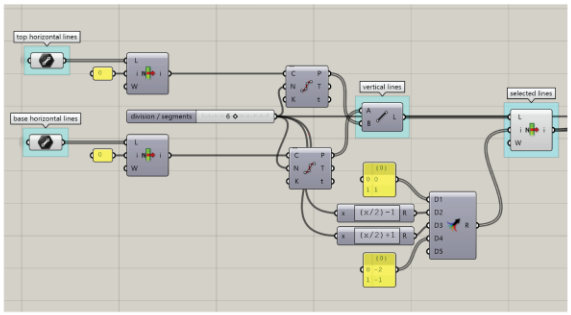
use [-1, -2] as a constant to select the last and second-last item in array of vertical lines

display the selected lines and hide the others

GH definition







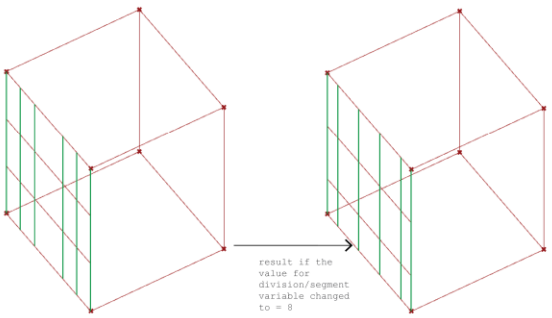


Figure 13. Pseudocode, GH Definition for Division of the Framework

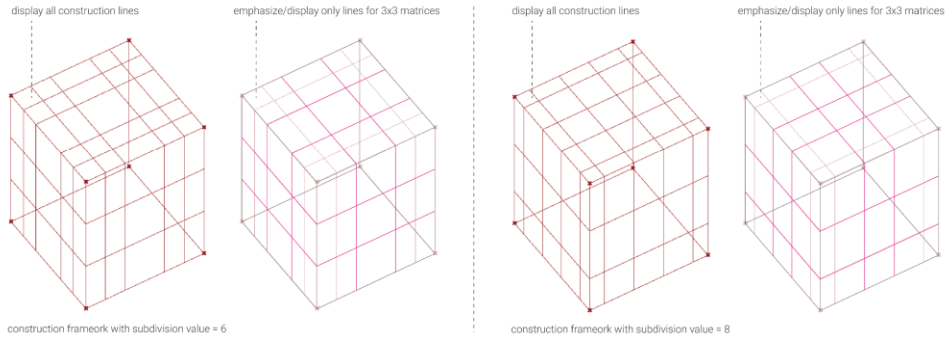


Figure 14. Reconstruction Result and Possible Adjustment of the Framework

Reconstructing the elements (planes)

This part aims to reconstruct the architectural form/elements based on the framework that was created in the previous stage. Specifically, this part focuses on explaining the reconstruction process of the planes. The set of pseudocode, GH definition, and reconstruction progress for this phase is shown in Figure 15.

Similar conditional structures are also used to reconstruct the other elements (lines and volumes). At this stage, the algorithm has been able to reconstruct all essential parts (framework + elements), which can be articulated to achieve similar configurations to the original House IV. However, the aim is not to replicate house IV, but rather to speculate/explore other possibilities of formal configurations.

Speculation through combination

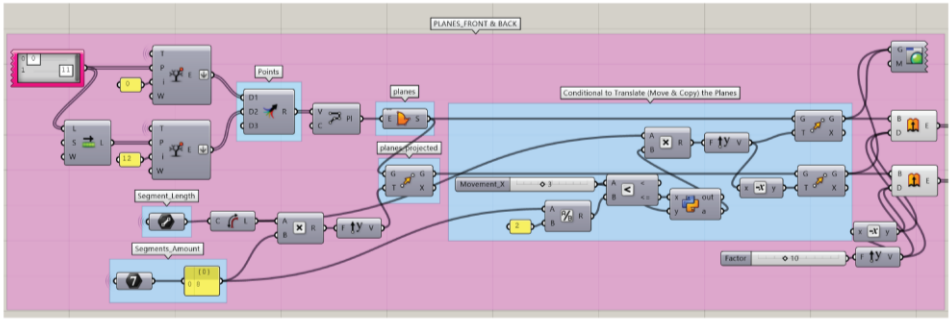
After the reconstruction of elements (planes, volumes, lines) is completed, there are two options for further development, which are: 1) tweak and adjust the values or parameters to achieve a similar configuration to House IV; 2) explore the possibilities for different iterations/variations. It is important to emphasize that the exploration process is still based on the construction framework, which is the abstraction of the House IV transformation process. Therefore, exploration can result in various iterations but still resemble some characteristics of House IV.

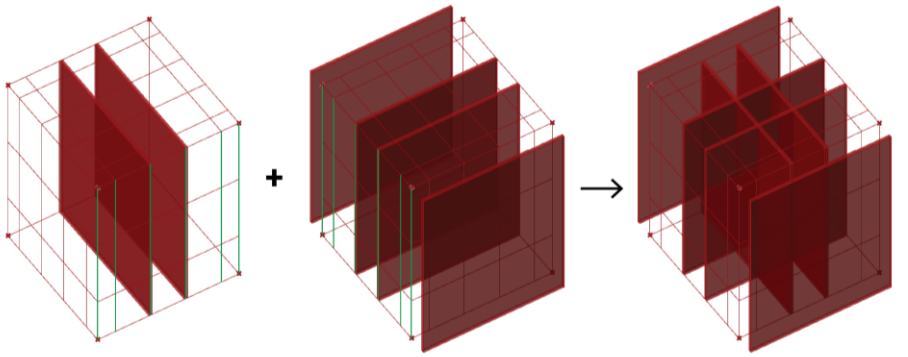
The framework provides access for different states of transformation, which allows for combining elements in different configurations (Figure 16). Hence, even though the idea of Boolean operations is quite simple, this access to the 'recording/trace' of the state of transformation allows us to speculate by combining elements in different configurational states. This process can be further enhanced by changing the parameters of the framework (size and amount of division). This speculation results in complex and unique architectural forms (but still resembles the properties and characteristics of House IV). For example, in the first and second row, we can see a new formal iteration in which the volume does not present as an actual geometry, it instead exists as an 'invisible' element that subtracted the planes and the lines.

pseudocode

```
use [0, 11] as a constant to select the first and last branch in array of sub-division points

use [0, -1] as a constant to select the first and last item on both branch

compile all selected items as an array of points

create a closed curve by connecting all points

create a plane from the closed curve

use (segment_length x segment_amount) as an expression to project the plane in y direction

divide the segment_amount by two 2 and assign it as translation_limit

assign range of values to movement_x

if movement_x is less than or equal to translation_limit:
       get the segment_length
       multiply the segment_length with movement_x and assign it as translation value
       use translation value to move and copy the plane in y direction
       inverse the translation value
       use the inversed value to move and copy the projected plane
```

GH definition





result from above pseudocode & GH definition

result for different side (based on code/definition with similar logic /structure)

combination between the two

Figure 15. Pseudocode, GH Definition for the Reconstruction of Elements (Planes)

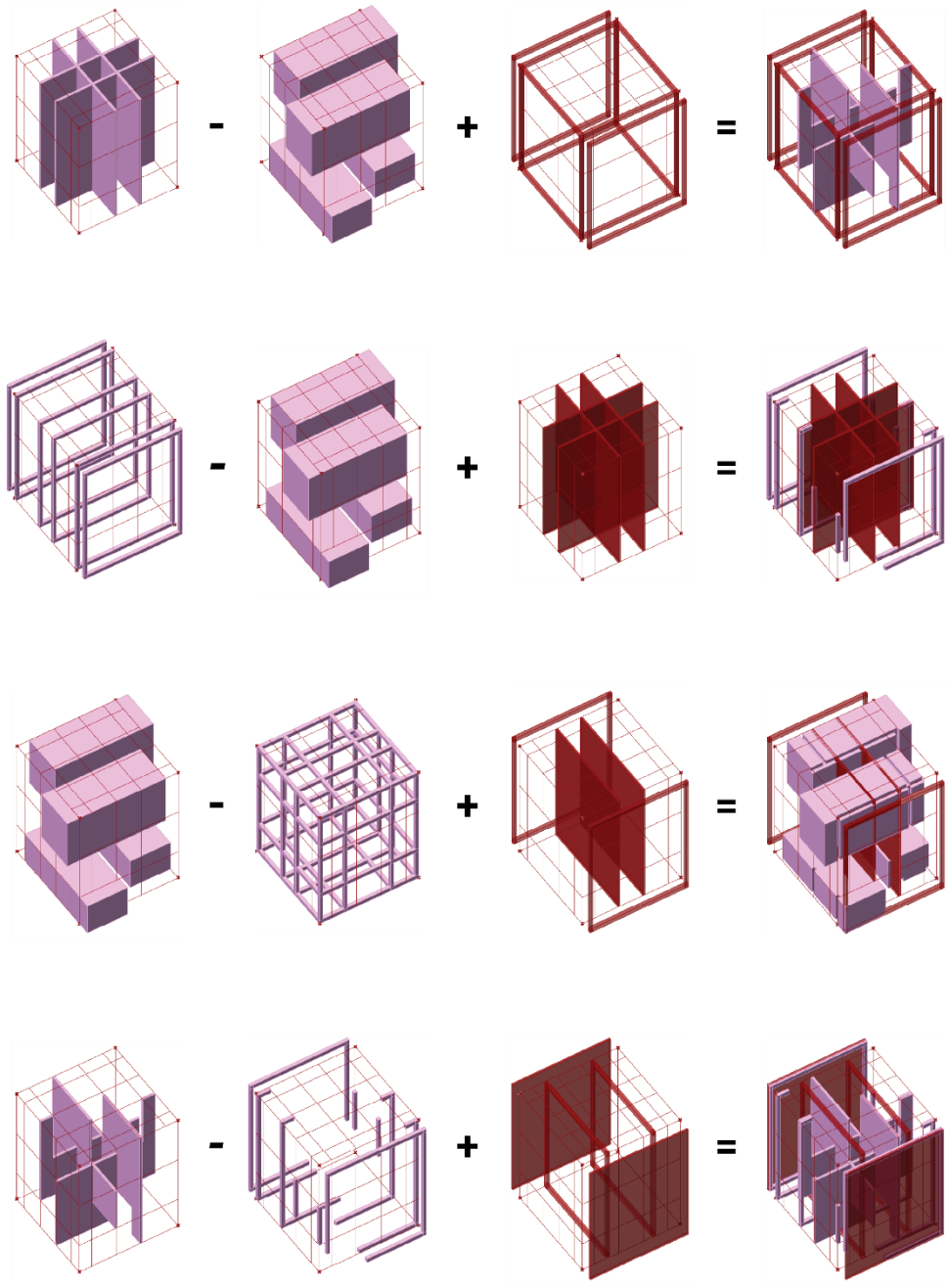


Figure 16. Speculation of Various Combinations of Elements

Reflection: Explorative Analysis and Structured Speculation

A CT-based approach in deconstructing and reconstructing House IV results in two main findings, which are: 1) The idea of explorative analysis; 2) the possibility of structured exploration. Explorative analysis highlights the importance of the analytical process in a CT-based design process, which usually emphasizes the generative process (especially algorithms). The deconstruction phase shows how CT provides a set of concepts (decomposition, pattern recognition, and generalization/abstraction) that can be used as a holistic analytical tool. This CT-based architectural analysis not only allows for a detailed breakdown of the architectural elements (through the decomposition and pattern recognition), but also unfolding of the potential exploration through generalization and abstraction.

Structured speculation emphasizes the traceability and explicitness of the form-generative process in architectural design. As a continuation of the thorough and explorative analysis, it is equipped with an explicit set of elements and a framework (as the rules to configure these elements). It approaches exploration as a declarative and deliberate process (Çolakoğlu and Yazar, 2007; Burry, 2011). This construction framework works as a scaffolding for elements' initiation, placement and articulation. It also provides a 'record' for different states of configuration and transformation for each element, which can be accessed and combined to speculate and produce different iterations of architectural form. It is an *exploratory creativity* process that involves a deep understanding of what we have created, and how improvement and/or innovation could emerge from there (Du Sautoy, 2019). It introduces an idea about exploration that is structured and traceable, but still creative and speculative.

CONCLUSIONS

This study examines the potential adoption of four computational thinking principles – decomposition, pattern recognition, generalization/abstraction, and algorithm – as an approach in the architectural digital form-making process. This study seeks the integration of a structured and deliberate CT-based process with a speculative architectural form exploration. Reflection from the process results in two main findings, which are: 1) the importance of explorative analysis; 2) structured speculation.

Exploratory analysis highlighted the role of the analytical process in discovering elements and structures that can be used for constructing the architectural form. It shows that exploration needs to be initiated in the analytical stage instead of something that is exclusive to the generative phase. Exploration of parts, patterns, and their possible generalization/abstraction in analysis is a crucial foundation for the generative phase.

Structured speculation emphasizes the importance of having an exploration process that is deliberate, structured and traceable. Exploration is seen as a process that is executed with an explicit set of elements and rules, rather than an intuitive act that starts from scratch. This approach does not negate intuition, but intuition is seen as something that emerges from a deep understanding of the design context. It is seen

as a potential improvement based on the record and trace of what has been created along the process.

This study has two main limitations. First, this study involves House IV as an existing case, which means some of the methods, discussion, and findings may be less relevant in a scenario without an existing architectural form. Second, this study solely focuses on architectural form exploration, without any consideration for other elements such as space, context and program/function. Further study with different scenarios and/or more complex factors and considerations is crucial to further develop the understanding of the CT-based architectural design process. With different scenarios, the role and significance of the explorative analytical phase can be further explored with an actual problem such as programmatic function or urban context, and how the structure speculation could be scaled to generate explorative formal iterations from such situation instead of pre-existing architectural form.

REFERENCES

Ahlquist, S. and Menges, A. (2011) "Introduction," *Computational Design Thinking*. Wiley and Sons.

Beecher, K. (2017) *Computational Thinking: A beginner's guide to problem-solving and programming*. BCS, The Chartered Institute.

Buitrago-Flórez, F. et al. (2021) "Fostering 21st century competences through computational thinking and active learning," *International Journal of Instruction*, 14(3), pp. 737–754. https://doi.org/10.29333/iji.2021.14343a

Burry, M. (2011) *Scripting Cultures: Architectural Design and Programming*. Wiley and Sons.

Caetano, I., Santos, L. and Leitao, A. (2019) "Computational design in architecture: Defining parametric, generative, and algorithmic design," *Frontiers of Architectural Research*, 9(2), pp. 287–300. https://doi.org/10.1016/j.foar.2019.12.008

Carpo, M. (2023) "Digitally Intelligent Architecture Has Little to Do with Computers (and Even Less with their Intelligence)," *ARQ (Santiago)*, (113), pp. 18–31. https://doi.org/10.4067/S0717-69962023000100018

Chytas, C. et al. (2022) "Youth's Perspectives of Computational Design in Making-based Coding Activities," *6th FabLearn Europe*. https://doi.org/10.1145/3535227.3535231

Çolakoğlu, B. and Yazar, T. (2007) "An Innovative Design Education Approach: Computational Design Teaching for Architecture," *ODTÜ Mimarlık Fakültesi Dergisi*, 24(2), pp. 159–168

De Boissieu, A. (2022) "Introduction to Computational Design," *Industry 4.0 for the Built Environment*, pp. 55–76

De Boissieu, A. and Watts, A. (2021) "Interoperability and collaboration in Computational Design," *Nouvelles perspectives du BIM*

De Jesus, S. and Martinez, D. (2020) *Applied Computational Thinking with Python*. Packt Publishing

Du Sautoy, M. (2019) *The Creativity Code*. Harvard University Press

Dunne, A. and Raby, F. (2013) *Speculative Everything: Design, Fiction, and Social Dreaming*. MIT Press

Gopsill, J. et al. (2023) "A sustainable computational design concept using web service methods," *Proceedings of the Design Society*, 3, pp. 425–434. https://doi.org/10.1017/pds.2023.43

Haghnazar, R. (2024) "A computational design integrated digital fabrication framework," *Results in Engineering*, 23, p. 102400. https://doi.org/10.1016/j.rineng.2024.102400

Eisenman Architects (1971) *House IV*. Available at: https://eisenmanarchitects.com/House-IV-1971 (Accessed: 3 May 2025).

Kelly, N. and Gero, J.S. (2021) "Design thinking and computational thinking," *Design Science*, 7, p. e8. https://doi.org/10.1017/dsj.2021.7

Knippers, J. (2021) "Integrative computational design and construction," *Civil Engineering Design*, 3(4), pp. 123–135. https://doi.org/10.1002/cend.202100027

Koyama, Y. and Igarashi, T. (2018) *Computational Design with Crowds*. Oxford University Press

Kyratsis, P. (2020) "Computational Design and Digital Manufacturing Applications," *International Journal of Modern Manufacturing Technologies*, 12, pp. 82–91

Lynn, G. (1999) *Animate Form*. Princeton Architectural Press

Maksoud, A. (2024) "Computational Design for Multi-Optimized Geometry," *Sustainability*, 16(7), p. 2750. https://doi.org/10.3390/su16072750

Mamoli, M. (2020) "A Shape Grammar for the Building-Type Definition," *AI EDAM*, 34, pp. 191–206. https://doi.org/10.1017/S0890060420000189

McGill, M.C. (2001) *A Visual Approach for Exploring Computational Design*

Mikaelsson, R. (2022) *Computational Design in the AEC industry*

Mills, K. et al. (2021) *Computational Thinking for an Inclusive World*. https://doi.org/10.51388/20.500.12265/138

Önalan, B. et al. (2025) "Computational methods for circular design," *International Journal of Architectural Computing*. https://doi.org/10.1177/14780771251316125

Oxman, R. (2017) "Thinking difference," *Design Studies*, 52, pp. 4–39. https://doi.org/10.1016/j.destud.2017.06.001

Partridge, E. (1958) *Origins: A Short Etymological Dictionary of Modern English*. Routledge

Ploennigs, J. and Berger, M. (2024) "Automating Computational Design with Generative AI"

Riley, D.D. and Hunt, K.A. (2014) *Computational Thinking for the Modern Problem Solver*. CRC Press

Sun, Q. et al. (2024) "A systematic literature review on computational design methods for non-uniform found objects". https://doi.org/10.2139/ssrn.4952086

Sunarya, W. (2022) "The Implementation of Parametric Design Practice," *Journal of Architecture and Environment*, 21(2), pp. 123–138

Thomsen, M.R. (2022) "Computational design logics for bio-based design," *Architectural Intelligence*, 1, pp. 1–15. https://doi.org/10.1007/s44223-022-00015-8

Wang, J. (2024) "Application of shape grammar to vernacular houses," *Journal of Asian Architecture and Building Engineering*, 23(3), pp. 843–859. https://doi.org/10.1080/13467581.2023.2247465

Wing, J. (2006) "Computational Thinking," *Communications of the ACM*, 49(3), pp. 33–35

Wu, T.-T. et al. (2024) "Identification of Problem-Solving Techniques in Computational Thinking Studies," *Sage Open*, 14(2). https://doi.org/10.1177/21582440241249897