Architectural Assemblages as Computational Medium

Introducing Assembler, a tool for the design and study of architectural assemblages

Alessio Erioli
University of Bologna – Co-de-iT
alessio.erioli@unibo.it

Assembler is a computational tool designed for the creation and study of assemblages in architecture, interweaving mereology, combinatorial design, and decision at scale. The tool leverages the potential of automation and repeated parts to generate scalable and spatially heterogeneous assemblages, emphasizing the computational role of both parts and relations in creating emergent qualities. Assembler utilizes iterative, rule-based heuristic, enabling computation across scales via part/assemblage/environment relationhood. The design process is understood as a decision network, where the user has control over the design of parts, connections, and heuristics of the system, and the tool enacts those decisions in space and time. After a theoretical contextualization and an overview of precursors and precedents in architecture and combinatorial design, the tool logic is explained and its current status and potential developments are discussed.

Keywords: *Assemblage, Architecture, Mereology, Computation, Intelligence, Tools.*

INTRODUCTION

Assembler is a computational tool developed for the design and study of architectural assemblages, with particular attention to the generation of heterogeneous space via iterative combinatorial growth from a limited set of parts. It is part of a larger theoretical and operational body of research that mobilises the autonomous computing potential and economies of repeatable parts and/or infrastructures, into spatial assemblages able to support a multiplicity of programmes that can adapt over time.

Drawing from the concept of assemblage in DeLanda's formulation (2016), Assembler focuses on the importance of both parts and their mutual relations in the formation of emergent qualities, allowing computation of spatial structure and topology at the local, regional, and global scales. Parts and connections are conceptualised as computing objects representing respectively the geometry of reserved space and its connectivity capabilities; how that space is occupied by data (both numerical and geometrical in nature - i.e. constructive elements and metadata) involves multiscalar decisions leveraging contextual information about component-assemblage-environment parthood and relationhood.

The design process is thus defined as a decision network, where intentionality can be encoded in granular *procedures* and *criteria*, operative and boundary-like constraints (Snooks, 2012). Assembler performs iterative, heuristics-based decisions, where the computation of parts informs the assemblage growth at each step.

The user has control over the design of parts and connections, the decisional heuristics of the system, and the design of environmental information encoded as a discrete tensor field.

Assembler aims to interweave mereology, combinatorial design, and automated decision at scale; it has been successfully used in several education and research contexts over the past two years.

DESIGN AS AUTOMATED DECISIONAL NETWORK

According to Bratton (2019), automation is not just a shift of agency from man to machine, but the dissolution of decision into networks of biological and prosthetic relays. It propagates across scales, from nanomolecular to planetary, in *trophic cascades* (indirect rippling interactions through the relay network). Cities represent a particular interesting scale as theatres of mutualising automation, the actual ecologies where the personal and communitarian interactions among humans, biodiversity, technology, and architectural space occur.

If any design process can be seen as a network of decisions (Erioli, 2016), automating those decisions requires a different stance on design itself: instead of an overarching conceived idea to be progressively refined, design intention is granularly encoded. Steenson (2017) uses the verb 'architecting' to describe the work of architects who first used this approach, building computational tools to work "from the minutiae to the major, fitting parts and wholes into something greater than the sum".

Abstraction and encoding, from Alberti and Serlio to BIM, enable the embedding of multiple data dimensions into architectural parts and relations; the next step is giving them computational agency, both individually and as part of a connected network, similarly to Von Neumann's automata (Koehler, 2020). Complex computation is achieved through combinations of seven logical gates that regulate electrical flow in spatial networks; architectural assemblages made of parts as spatial computing units can perform complex computation by arranging flows in complex, non-linear spatial networks (Figure 1). The intended approach encodes embedded operating constraints from the start, exploring a space of possibilities engendered by intrinsic coherence for *geometric constructability* (the possibility for an assemblage to exist in a given domain maintaining geometric coherence). Architectural space can thus emerge from the assemblage of computing parts encoded with multidimensional data, as a result of the automated iteration of granular decisions.

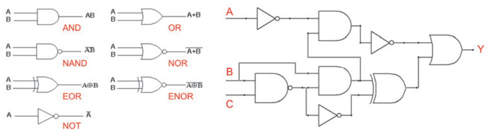

Figure 1
Computation as spatial assemblage of logical gates

But *why* automating? Computation and data have become the most important factors in shaping fundamental aspects of the spaces and objects we interact with. Easterling (2014) calls this dense matrix of details and repeatable formulas *infrastructure space*, a medium of information and immaterial activities governing the arrangement and flow of objects that acts as "an operating system for shaping the city". This *active form* (the operating infrastructure that defines the building: the protocols, routines, schedules, and choices it manifests as space) produces the *object form* (the building). Singularly crafted architectural objects are the exception in an urban reality mostly made of buildings as commodities: reproducible products within similar urban arrangements, engineered more around logistics and paths of minimal resistance defined by the operating economy of means. Patently, in housings developments such as the *Plattenbauten* and Lewittown, a house is not a singularly crafted object but a multiplier of activities, the result of an assembly line of repeatable objects and processes that ultimately produce an urban structure (Easterling, 2014). Habitat '67 (Safdie, 1967) is a relevant case in this context for its intervention on the active form, steering the economies and logistics of prefabrication away from linear blocks and towards a novel spatial arrangement by means of combinatorial design (Figure 2).

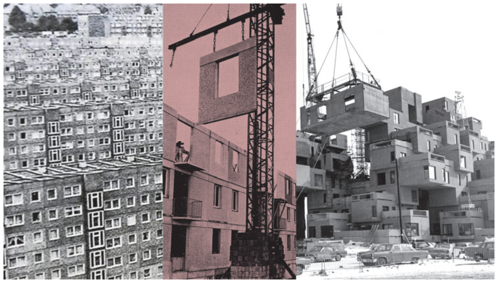
Figure 2
Plattenbauten vs Habitat '67 – same construction logistics, different spatial logic

PRECEDENTS AND PRECURSORS

The proposed approach builds upon theoretical, architectural, and methodological precedents. Theoretical premises are centered on assemblage thinking; architectural precedents on the attempt to work across scales, eroding the boundary between architectural and urban, between building and city and/or use of computational methods in the process. Methodological precedents focus on the computational extension of combinatorial methods.

Assemblage Thinking and Space

Assemblages are first introduced by Deleuze and Guattari (1987), further articulated by DeLanda (2006) and extensively connected to architecture by Dovey (2013, 2017). The concept of assemblage refers to a socio-spatial network, where the identities and functions of both parts and wholes emerge from the mutual flows. It is a tetravalent structure with two intersecting axes that both oppose and connect: materiality and formal expression on the horizontal, and territorialization and deterritorialization (the formation and erasure of territories), on the vertical. Viewing architectural space as an assemblage allows for a reconfiguration of the relation of form to function and avoids reduction to either text or material conditions alone. Territorialization mediates the degree to which an assemblage is stabilized or destabilized, and segmentarity refers to the ways boundaries are used to construct, perform, or inscribe territories, cooperating both vertically (across scales: rooms, buildings, streets, districts, cities cannot be fully understood without their mutual relations) and horizontally (within scales: i.e. room-to-room relations). Such relations should not be framed in a tree-like hierarchy of scales, but, as Alexander (2015) also formulated, more akin to a 'semilattice' structure, allowing for overlaps and interdependency. Assemblage thinking enables a focus on connectivity and flow rather than object and form, and on the crucial connections between multiple scales of assemblage.

The concepts of *smooth* and *striated* (Deleuze and Guattari, 1987, p.474), are not meant as types of architectural space insomuch as its mutually informing *properties*, with smooth as a rhizome-like space of becoming, and striated as the tree-like space of ordering and hierarchy. The act of design, while significantly impacting the ways in which the smooth/striated interactions unfold, is primarily one of striation: a set of decisions on how to program the arrangement of constraints, stabilizing the building form, and not the flows they will enable (Dovey, 2013).

Architectural assemblages resist the definition of overview or an enclosing perspective, they can never be fully captured: "wholes" (or, better yet, multiplicities) are not defined *a priori*, but they can be recognised as such; what Alexander (2011) refers to as "holistic properties", as "systemic" points of view. Then again, wholes or systemic properties are not a function of a point of view, they exist and operate regardless of an observer; an aspect also shared by mereology, capacities, and affordances, all relational in nature.

Assemblages enable an heterogeneous space (Hensel et al, 2009), where difference is neither produced by exception nor by collage of varieties, but enabled by relations. Heterogeneous space permits differentiation and discontinuity of quality and organization across multiple conditions within an overall coherency.

An assemblage is both a theory and a way of thinking through practice, teaching us how to think as much as what to think, and requiring the production of both concepts as tools for thinking, and tools as operational devices to conceptualize.

Architecture

A relevant precedent of computation and system thinking in architecture can be found in Alexander's aforementioned 'semilattice' (2015), which implies mereological notions of parthood, and in his twofold perspective on systems (2011): as expression of relations-enacted emergent qualities and as generative engines made of kits of parts and rules producing the former.

Yona Friedman's Flatwriter (1971), merges computation, combinatorial design and user participation; Negroponte (Architecture by Yourself, The Architecture Machine), and Cedric Price (The Generator Project) enhance the degree of autonomy of the algorithms' decisional process. Steenson (2017) provides an in-depth account of the work of Alexander, Negroponte, Price and Wurman, who pioneered the fields of computational design thinking, information architecture and intelligence in architecture.

Relevant precedents in the exploration of spatially assembled systems are also in Dutch Structuralism: as "essentially concerned with how the individual and the collective are interdependent and able to influence one another" (Hertzberger, 2016), it bears close relations with assemblage thinking. Although not explicitly declared by its purveyors (notably: Hertzberger, Blom, Van Eyck, Bakema), it shares with assemblages qualities such as the attention to part-whole relations, interscalar correlation enabling emergent complexity, and multiplicity of enactment (Hertzberger, 2016).

Chiappone-Piriou (2019) provides a good collection of precedents in the application of combinatorics and/or discrete parts in architecture; another notable example is the Metastadt system by Richard J. Dietrich (1965).

The City in the Space by RBTA (Bofill, 2016) represents a milestone, not so much for its radical political ambitions or for providing through a rigorous, yet open-ended modularity, a spatial structure to flesh out the visions of Lefebvre and Constant, rather for the openly antimetabolist way in which circulation is embedded in the modular growth (Figure 3).

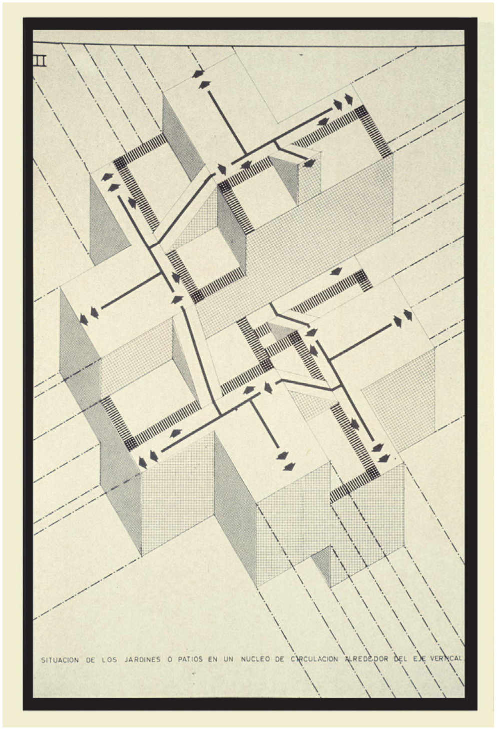
II
SITUACION DE LOS JARDINES O PATIOS EN UN NUCLEO DE CIRCULACION ALREDEDOR DEL EJE VERTICAL

Figure 3
RBTA – The City in The Space – © RBTA – used with permission

Methods and Implementations

As Sanchez (2016) points out, combinatorial design relies on patterning of units; this sets it apart from parametric design, (which relies on flow articulation) but also from permutation design, in which the problem is exhausted by a finite catalogue of possible configurations. Combinatorial design establishes an open-ended network of relations among a set of repeatable parts without the necessity to define a whole. The use of protocols, algorithms and combinatorial techniques in architecture and urban design also predates the diffusion of computers, such as in James Oglethorpe's plan for Savannah (Easterling, 2014).

Since shape grammars (Stiny, 1980), many authors have implemented and evolved upon combinatorial methods using computation for a wide range of application and disciplines (Liu et. al., 2009). In the architectural field, notable contributions have been made by Hillier (1989), Knight (1991), Grasl & Economou (2011), Sanchez (2016), Rossi & Tessmann (2019). Although their methodology took substantial departures from the starting point, computational inheritors of combinatorial design such as Sanchez, Retsin, Koehler et al. (Retsin, 2019), all intersect the matter of producing variety through organization of repeated components in different declinations. Each developed a bespoke toolset as an operative apparatus as an indispensable companion and complement to theoretical speculation in the construction of their discourse.

Assembler comes from an analogous operational necessity that could not be satisfied by the currently available tools, Wasp (Rossi, 2017) and Fox (Vestartas, 2017): a need for deterministic, customizable, and encodable decision-making, the use of void geometries as basic parts, a focus on relations and the ability to produce computable topological structures.

ASSEMBLER

Assembler (Erioli, 2022) is a software plugin for the Rhino-Grasshopper environment that fosters a paradigm of computing parts iteratively forming computing assemblages by means of their arrangement. It is conceived as scale-agnostic to stimulate interscalar exploration; it defines a relationality between parts, without specifying their nature in advance. It is developed as a set of compiled components and a .dll library to provide an entry point from an established platform and sidestep presets limitations for a more versatile, programmable and interoperable architecture.

The system acts like a single agent in an iterative simulation whose operating principle follows the question 'where do I place the next object?', which begets two necessary choices: a location where the next object will be added, and what object should be placed in which orientation. Starting from one or more objects, the logic is applied iteratively adding one object at a time until an end condition is met. The hidden complexity in this logic resides in the criteria (or policy) adopted for the choices made at each iteration.

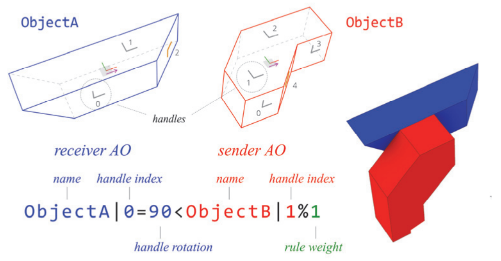

Figure 4
AssemblyObject, Handles and Rules

Parts, connections and data

Repeatable parts (called AssemblyObjects - AO) are encoded as collision volumes with a reference plane, a direction vector, and one or more connectors named Handles - data structures including reference planes and rotation angles (Figure 4). Any other non-essential data (primitive data types, geometry, etc.) is stored in the form of XData (extended data) and associated post assemblage to the relative AO. Both AOs and Handles can act as receiver (when already existing) or sender (when newly added). An Assemblage is a structure of voids and data, selectively populated according to a variety of criteria that can exploit computation on contextual data at the scale of the whole: the same AO kind can host different sets of geometry and data in different instances (Figure 5).

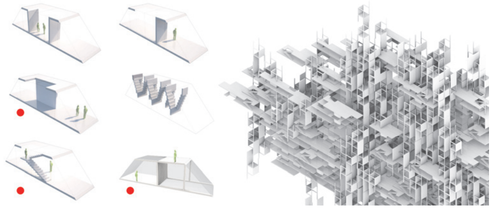

Figure 5
Different variants of the same AO kind with result from selected variants (assigned after assembling) – Urban Eurhythmics – project developed at MAA – IaaC

Parts and whole undergo a process of *mereological individuation*, in which the identities of parts and whole are reciprocally generative: the parts and their relations determine the identity of the whole, which in turn contributes to the identities of the parts themselves. In a matrix of identical rooms, their organization shapes the identity of the whole, while each room can acquire further identity according to, for example, its relative location in the matrix (core, edge, corner).

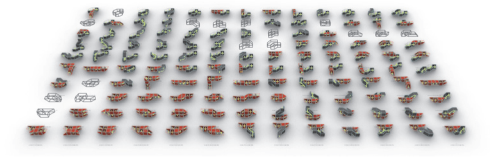

Figure 6
Automatically generated rules table - invalid combinations outlined in black

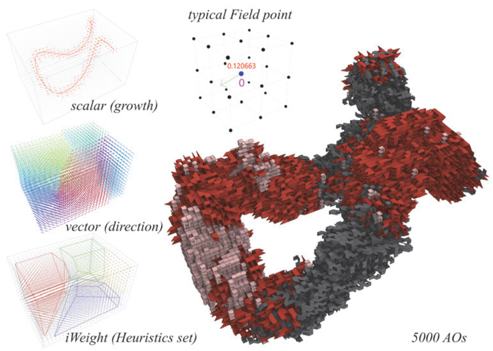

Figure 7
Field-driven Assemblage, 2 AO kinds with orientation-dependent inserted geometry

Heuristics
Heuristics include the set of rules and the criteria for their selection at each assemblage iteration. Rules are automatically generated by permutation of the designed Handles (Figure 6), with the possibility to encode asymmetrical connections, and display the table of combinations, where geometric inconsistencies are highlighted and can be filtered.

Criteria for selecting receiver and sender objects at each iteration can be chosen from available presets, or customized by writing C# code using the classes and methods provided in the .dll library.

Environment & Field
An environment is defined by closed meshes that can be interpreted as solids or voids, limiting growth by collision or inclusion criteria respectively.

A Field is a 3D voxel structure to encode exogenous data. Although inspired by the work of Rossi and Tessmann (2019), its implementation differs substantially: they can be dense or sparse, and encode a multiplicity of data via a Tensor class. Each point can encode floating point numbers (scalars), vectors and integer indexes (iWeights), which can affect the Assemblage growth: scalars generate a signal that the assemblage can follow, vectors can affect the preferred direction for the placement of a new object, and iWeights determine the set of rules used in a particular region of space (Figure 7).

When Field-dependent criteria are used, the Assemblage reads Field local information to compute the location, kind and orientation of the object placed at each growth iteration. Fields can encode physical data (such as stress value distribution, temperature, wind direction) as well as direct design intentions in its data distribution, for example using signed or unsigned distance from designed geometry. There are no preset methods for computing values that populate the Field, as such methods are germane to design strategies more than tool features.

Engine
The engine embodies the operating logic, iterating through a 3-stages loop: select a viable receiver among the assembled AOs, select a criteria-coherent, geometrically consistent sender AO kind among possible candidates, and update the assemblage status. Criteria involve comparison of values computed using Field's and AOs' data alike.

Post-Processing
Assembler enables access to connectivity and computed data of each object, allowing multiscale computation for condition-based arrangement of constructed elements or the analysis and data extraction for the study of specific conditions. Example files include some cases, although these too are design strategies rather than tool features.

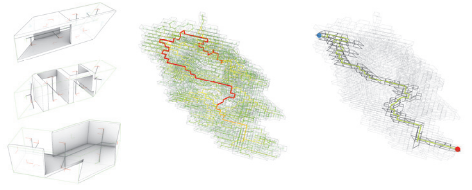
Figure 8
Computing circulation frequency and shortest path based on circulation graph of individual AOs

Circulation-Flow

A key factor for architectural assemblages and their computation that sets them apart from aggregates, is the capability to access connectivity to compute circulation. Assembler preserves topological connectivity data among components throughout the process and provides examples on how to approximate the generation of a circulation graph for the produced architectural space from parts geometries, subject to a necessary design consistency of parts and connections placement (Figure 8).

Alternatives to this method have been considered: Puusepp (2014) reviews several agent-based strategies for computing circulation, but those are either decoupled from the geometry of spaces or subservient to circulation forces - thus exposing the flaws of single-feature optimization for complex problems with interdependent variables. Another possible option is using a NavMesh (Mononen, 2009), currently accessible by interfacing with game engines such as Unity.

Although the tool does not implement directly algorithms and strategies to solve graph circulation problems, the connectivity data and/or the generated geometry can easily be interfaced with dedicated tools and libraries.

CONCLUSIONS

Assembler is a practical as well as intellectual toolkit to explore how architectural spaces work to complement assemblage theory. Its structure and parthood are function of its transduction in its current computational implementation, and do not aim at mimicking the theoretical counterpart. Generalisations, objects, relations, graphs, are the result of both design intentions and the path of minimal resistance of certain mathematical paradigms behind CAD tools, as well as the specific platform adopted (Rhino-Grasshopper). The tool is an operative playground, in the understanding that assemblages “emerge from the work and are better defined by their use than their meaning” (Dovey, 2013).

Controlled and articulated flow is obtained through the design of constraints and segmentations (Dovey, 2013), generating heterogeneity and variety from a minimal catalog of parts. Assembler generates spatialized data that can be harnessed in ways and for purposes intentionally left open and up to the designer’s agenda.

Assembler is the result of a desire to create an access point to provide opportunity for researchers, educators and practitioners in the explorations of assemblages and decision-scaling in architecture. It aims to provides the means to pragmatically respond to questions on how buildings and places are assembled and how they work in terms of affordances (signals about use potential), and not prediction. It has been used in the last two years in several educational context such as IaaC (Figure 5) and University of Bologna (Figure 9).

Assembler is developed with the intention of exploring and bringing the user into new territory by means of machinic operations, not just parroting existing knowledge and/or filling in as an executor of “artistic labour” to the “concept” of an architect. This is a deliberate counter current move against the separation between ideas and execution; sometimes the execution *is* the key to the idea.

The tool has an intentional orientation towards architectural space, but is equally intentionally left open and unburdened of theoretical policy over its operations, its origin and motivation do not define its potential. The only coherence it enforces is a geometrical one; its bias are structural, not conceptual. It computes assemblages of parts and their connectivity, regardless of their scale, content, or meaning. As overarching as a theory may be, a tool (or its mindless use) that forces its perspective onto everything turns projects into nails to the proverbial Manslow's hammer (*Wikipedia*, 2023).

Beyond *ars combinatoria*

Hillier (2015, p. 256) warns against the problematic association of architecture to a mere *ars combinatoria* of elements and relations: stable identities at higher scales (elements and spatial patterns) emerge, enabled by laws from global to local that drastically narrow the combinatorial possibilities. This echoes with Alexander's generating systems as kits of parts and rules, and the emergence of 'holistic' (read: emergent) properties as a product of interaction among parts: the system "must be generated by some process which assembles parts according to certain constraints, chosen to ensure the proper interaction of these parts, when the system operates" (Alexander, 2011). It is perhaps not coincidental, and fitting to assemblage thinking, that both authors use language as a working metaphors to make their respective point.

Assembler's heuristics are deliberately designed as deterministic to ensure a reliable response for the designer and act as a probe to navigate the space of possibilities in search of stabilities and meaningful, repeatable results.

Limitations

Assembler's underlying idea is to provide decision at scale, with the end goal of automating the generation of meaningful spatial patterns, but that doesn't suddenly make decisions easy nor meaningfulness abundant.

Born out of research purposes, one of the current tool criticalities is the user experience and interface. Once mastered, it provides a rich set of opportunities, but the learning curve is still steep: for perspective, in a 13-weeks, 11 hours/week academic course (4 dedicated to tools introduction, from the basics of Rhino to advanced techniques with Assembler), students show confidence after week 5.

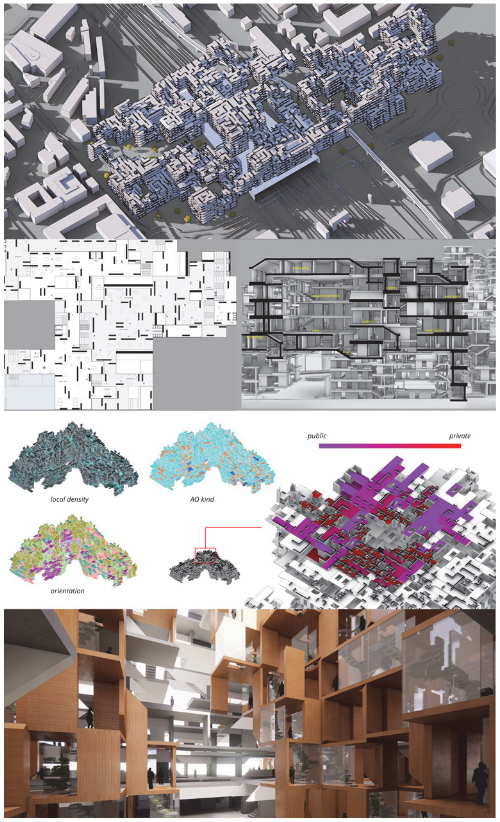

Figure 9
Two example projects developed at University of Bologna

Streamlining for a production environment is also improvable.

Another important criticality is in the mental shift required to understand control and intention in the context of assemblages and iterative systems with interdependent parameters. Coherent machinic operations seldom match a preconceived idea formed elsewhere: this mismatch often leads to overconstraining or baroque settings. Postponing judgement and trial-and-error approach are two things that do not sit well with traditional understandings of conceptualization, authoriality and intuition in architecture. Assessment and critical filters are necessary, but with a degree of open-mindedness to avoid falling into pre-existent categorizations.

Future developments

Decisions are at the core of Assembler, but its choice criteria is at the moment limited to fixed heuristics-based policies. To extend and improve its functionalities, this aspect could be integrated with an artificial neural network that acts as a brain that takes decisions, trained in a given environment via Deep Reinforcement Learning or with the aid of real-world data.

REFERENCES

Alexander, C. (2011). 'Systems generating systems', in Menges, A. and Ahlquist, S. (eds.) *AD Reader: Computational Design Thinking*, Chichester, U.K.: Wiley, pp. 58-67.

Alexander, C. (2015). *A city is not a tree*. 50th anniversary edition. Edited by M.W. Mehaffy. Portland, Oregon: Sustasis Press.

Bofill, R. (2016) The City in the Space [Online]. Available at: https://ricardobofill.com/projects/city-in-the-space/ (Accessed: 20 March 2023)

Bratton, B.H. (2019). *The Terraforming.* Moskau: Strelka Press.

De Landa, M. (2016). *Assemblage theory*. Edinburgh: Edinburgh University Press (Speculative realism).

Deleuze, G. and Guattari, F. (1987). *A Thousand Plateaus: Capitalism and Schizophrenia*. First edition. Translated by B. Massumi. Minneapolis: University of Minnesota Press.

Dietrich, R.J. (1965) *Metastadt*. Available at: http://www.dietrich-ingenieur-architektur.de/AS-1-start.htm (Accessed: 31 March 2023).

Dovey, K. (2013). 'Assembling architecture', in Frichot, H. & Loo, S. (eds.), *Deleuze and Architecture,* Edinburgh: University of Edinburgh Press, pp. 131-148.

Easterling, K. (2014). *Extrastatecraft: the power of infrastructure space*. London ; New York: Verso.

Erioli, A. (2016). 'Aesthetics of Decision - Unfolding the design process within a framework of complexity and self-organization', in *Herneoja, Aulikki; Toni Österlund and Piia Markkanen (eds.), Complexity & Simplicity - Proceedings of the 34th eCAADe Conference - Volume 1, University of Oulu, Oulu, Finland, 22-26 August 2016, pp. 219-228*. Available at: https://papers.cumincad.org/cgi-bin/works/Show?_id=ecaade2016_062 (Accessed: 20 March 2023).

Erioli, A. (2022). *Assembler* [Software] Available at: https://www.food4rhino.com/en/app/assembler (Accessed 28 March 2023)

Friedman, Y. (1971) 'The Flatwriter : Choice By Computer', in *Progressive Architecture. 1971. vol. 3: [4] p. : ill*. Available at: https://papers.cumincad.org/cgi-bin/works/paper/c844 (Accessed: 17 March 2023).

Grasl, T. and Economou, A. (2011), 'GRAPE: A parametric shape grammar implementation', in: Ramtin Attar, R. (ed.) *SimAUD 2011 Conference Proceedings*, Boston, MA, 4-6 April 2011, pp. 45-52.

Hensel, M., Hight, C. and Menges, A. (2009). 'En route: Towards a Discourse on Heterogeneous Space beyond Modernist Space-Time and Post-modernist Social Geography', in *Space Reader: Heterogeneous Space in Architecture*. 1st edition. Chichester, U.K: Wiley, pp. 09–37.

Hertzberger, H. (2016). *Architecture and Structuralism: The Ordering of Space*, Rotterdam: nai010 publishers.

Hillier, B. (1989) 'The architecture of the urban object', *Ekistics*, 56(334/33), pp. 5–21.

Hillier, B. (2015) *Space is the machine: A configurational theory of architecture*. CreateSpace Independent Publishing Platform.

Koehler, D. (2020). 'From Partitioning to Partaking, or Why Mereologies Matter', *B-Pro Prospectives Journal* [Online]. Available at: https://journal.b-pro.org/article/from-partitioning-to-partaking-or-why-mereologies-matter/ (Accessed 5 January 2022)

Koehler, D. (2019). 'Mereological Thinking: Figuring Realities within Urban Form', *Architectural Design*, 89(2), pp. 30–37. Available at: https://doi.org/10.1002/ad.2409.

Knight, T. (1991) 'Designing with Grammars'. Available at: https://cumincad.scix.net/data/works/att/2559.content.pdf (Accessed: 26 March 2023).

'Law of the instrument' (2023). *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Law_of_the_instrument (Accessed 8 March 2023)

Mononen, M. (2009). *Recast* [Software]. Available at: https://recastnav.com/index.html (Accessed 28 March 2023)

Puusepp, R. (2014). 'Agent-based models for computing circulation', in *ACADIA 2014: Design Agency - Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture, 23-25 October, 2014, Los Angeles, USA, pp. 43-52*. Available at: https://doi.org/10.13140/ 2.1.4184.2882. (Accessed 8 March 2023)

Retsin, G. (ed.) (2019). *Discrete: Reappraising the Digital in Architecture*. Chichester, U.K: Wiley (Architectural Design, 89).

Rossi, A. (2017). *Wasp* [Software]. Available at: https://www.food4rhino.com/en/app/wasp (Accessed 28 March 2023)

Rossi, A. and Tessmann, O. (2019). 'From Voxels to Parts: Hierarchical Discrete Modeling for Design and Assembly', in L Cocchiarella (ed.) *ICGG 2018 - Proceedings of the 18th International Conference on Geometry and Graphics*, *Advances in Intelligent Systems and Computing*, Milan 3-7 August 2018, pp.1001–1012.

Sanchez, J. (2016). 'Combinatorial design: Non-parametric computational design strategies', in, *ACADIA 2016: POSTHUMAN FRONTIERS: Data, Designers, and Cognitive Machines - Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture,* 27-29 October 2016, Ann Arbor, MI, pp. 44-53. Available at: https://papers. cumincad.org/cgi-bin/works/paper/ acadia16_44 (Accessed: 20 March 2023).

Safdie, M 1967, 'Habitat '67 - Towards the Development of a Building System', *PCI Journal*, vol. 12, no. 1, pp. 60–66.

Snooks, R. (2012). 'Volatile Formation', *Log*, vol. 25, pp. 55–62.

Steenson, M.W. (2017) *Architectural Intelligence: How Designers and Architects Created the Digital Landscape*. Illustrated edition. Cambridge, MA: The MIT Press.

Stiny, G. (1980). 'Kindergarten grammars: designing with Froebel's building gifts', *Environment and Planning B: Planning and Design*, 7(4), pp. 409–462. Available at: https://doi.org/10.1068/b070409.

Vestartas, P. (2017). *Fox* [Software]. Available at: https://www.food4rhino.com/en/app/fox (Accessed 28 March 2023)