From Black Box to Generative System

PEDRO VELOSO, DR. RAMESH KRISHNAMURTI
Carnegie Mellon University

Under the umbrella of the digital turn, novel computational workflows and distinct aesthetical principles are becoming an integral part of architectural education. Nonetheless, in current educational settings, there is not much scope for a deep understanding nor the development of custom computational design methods beyond standard toolkits. To fill this gap, we outline an educational framework for the development of new generative systems. The proposed framework combines canonical techniques for generative systems from different fields with recent advancements in Artificial Intelligence. It comprises eight schemas: unstructured constructive, structured constructive, variational, improvement, discrete simulation, continuous simulation, generative learning and behavioral learning. Each schema consists of a different formulation of design space and navigation, providing a knowledge base and a common language for design. Their adoption in design education can potentially expand the boundaries of design both within the agendas of the authorial design, nurtured in the studios, or even expand the boundaries of the profession to address future demands from society.

1. ARCHITECTURE IN A BLACK BOX

Reyner Banham consistently criticized architectural design for its mysterious mode of operation and superficial incorporation of new technologies. In early works, he claimed that the avant-garde only adopted an aesthetic of the industrial revolution, while preserving the academic composition as an implicit design method[1]. Later, he depicted the modus operandi of architecture as a black box[2], "recognized by its output, though unknown in its content"[3]. In this polemic analogy, architecture has nothing to do with the quality of the built environment; it is an exercise of an arcane, privileged and unspoken aesthetic code, inculcated in the studios and glorified in the draftsmanship of architectural drawing.

In this black box hypothesis, computer-aided design and the binary logic of the computers would represent a "probably fatal blow"[4] to the mystique of Architecture. Coincidentally, its original publication in 1990 was followed by a digital turn in architecture[5]. This turn invoked a paradigm shift in design education and practice, marked by the emergence of computational design models and a new conceptual vocabulary[6]. In the 1990s, architects started to manipulate digital geometry with the new CAD, modeling and animation software, challenging the limits of traditional architectural representation. In the following decade, programming was rediscovered by designers with scripting languages and the rise of graph-based parametric editors integrated in CAD systems. The use of algorithms in design required the externalization of the instructions for design generation, moving the designer away from the autographic domain of architectural drawing and fracturing the black box.

However, the incorporation of computational techniques in digital practices did not promote the type of rationality in design and education aimed for by Banham. As in the past, the claims of technological shift were saturated by aesthetical disputes. Associated with digital fabrication and new advancements in architectural geometry, the computational methods supported a new repertoire of non-standard forms that were developed in research pavilions and, eventually, applied to the design of the building surfaces, components, and envelope. Practical architects intertwined design methods and aesthetic for the digital architecture.[7] Even specific technologies, such as animation or parametric modeling, were assimilated as a blend of method and design content.[8] In this sense, the draftsmanship associated with architectural drawing was replaced by digital variations.

This recurrence of techno-ideological feuds in architecture is related to the socialization of the aesthetical code and formation of the professional behavior, in particular with the prominence of the "design crit in the architectural school studio"[9]. The studio culture is reminiscent of the Beaux Arts, where students were educated by the guidance of one or more experienced architects, who critique their solutions [10]. In this setting, design success is measured by satisfying the expectations of the circle of educated "patrons". Despite the aesthetic and technical differences, the first digital educational experiences, such as the paperless studios, also revolved around the exploration of an aesthetic agenda of experienced architects[11].

Notwithstanding, the studio, being a well-established and tested educational setting, in face of new technologies establishes a problematic relation between design method and the expectations of the critic. For instance, it is usual to adopt pre-defined computational workflows and plug-ins as a platform to explore a certain design agenda. This setting comes with the cost of immediacy and bias, as there is no time to investigate the potential of computation for design beyond the agenda and toolbox provided by the studio.[12]

This relation is even more critical in face of design automation and Artificial Intelligence (AI), which have long been encroaching on the territory of architecture. AI acquired new momentum with the recent wave of deep neural networks, which succeeded in automating activities such as painting style transfer, playing go, medical imaging, speech recognition, face recognition and synthesis. Not surprisingly, Mario Carpo's response to this momentum was the announcement of a new digital turn for architecture [13]. This time, the turn is based on the computational logic of search and the availability of big data[14], resulting in a movement from the non-standard forms and smooth surfaces to a focus on voxelization and the generation of form with search and simulation.

The complex relation between the terms computation and digital is crucial for the future of design education. Computation strictly refers to the use of algorithms or models to perform operations on symbolic representations. For example, computational design methods use these algorithms and models to represent, analyze and synthesize design alternatives. In contrast, the term digital has been used fluidly to describe a certain cultural condition or state of being related to the advent of different information technologies. Design benefits from historical and philosophical interpretations of the influence of these technologies on our society and environment. However, pursuing legitimate expressions of the digital with restricted computational schemas prevents the access to a more general knowledge on computational logic and structural changes in design practice.

2. REVIEWING GENERATIVE SYSTEMS

While computation logic can address different aspects of architecture, in this paper we focus on the synthesis of forms and spatial patterns. A proper knowledge base of computational synthesis can be developed based on generative systems (GS) – systems that can generate design alternatives automatically for a certain problem. While current digital studios and programming textbooks focus on parametric, NURBS and mesh modeling, GS comprehends a wider variety of techniques from different fields and domains. Design education would benefit from a systematic and historical understanding of the different computational schemas available for GS.

The definition and systematization of GS have been developed in computational design books[15], courses, articles and research conferences in Computer-aided Architectural Design[16] (CAAD). A brief review is given below.

Christopher Alexander's understanding of "generating system" is as a system composed of a kit of parts and combinatory rules that can generate many variations[17]. In his structuralist view, all natural and artificial phenomena are themselves systems generated by a specific set of interacting forces or rules.[18] However, while natural systems are adaptable to the interaction of forces, the built environment requires artificial methods to capture the existing forces and promote a global behavior based on their equilibrium. In opposition to conventional design, which addresses this task by intuition, Alexander categorized three methods to generate form based on forces[19]: (1) numerical methods, which use linear optimization of design variables to look for the best form; (2) analog methods, which use a physical model to represent the forces of the system and look for a stable configuration; (3) relational methods, which use diagrams to represent the different forces of a system and fuse them to generate the proper form.

William Mitchell wrote one of the first overviews of GS in the field of CAAD[20]. His understanding of GS is as systems that can produce a variety of potential solutions for a problem. He proposed three general categories for GS that are similar to Alexander's methods: (1) analogue GS are composed of analogue elements that enable mechanical operations to change the state of the system; (2) iconic GS are systems that use movies, models, drawings, and geometric operations to generate solutions, (3) symbolic GS use symbols and computational data-structures to represent a solution and rely on arithmetic and logical operations to change it.

While Alexander's work focused on GS based on relational/iconic GS[21], Mitchell's overview focused on symbolic GS to produce spatial solutions that meets certain specified criteria automatically – i.e., space planning. In contrast to Alexander's structuralist approach, Mitchell interprets symbolic GS with classical AI concepts. Each GS operates with discrete steps. Related designs that are visited in the computational process are codified in a directed graph called state-action graph, which structures the space of all possible designs and available operations for navigation.[22] The goal of GS is to navigate the set of solutions in this state-action graph looking for a subset of solutions that satisfy the design goals.

Mitchell[23] and other researchers, for example Henrion[24] and Liggett[25], classified and described the solution procedures of symbolic GS for space planning. Most of their categories are inscribed in two computational schemas: search and optimization[26], which were the main topics in Simon's famous text on a science of design[27]. See Table 1.

Since the 1990s, not only metaheuristics but also other computational concepts, such as cellular automata and swarm algorithms became part of architectural experimentation, which were explored in computational design books[31]. Additionally, architects also incorporated animation, NURBS, mesh and parametric modelers in their design toolbox. While most experimentations departed from classical AI, eventually, techniques, such as search and dissection, are revisited[32].

In Table 2, we organize authors who try to capture the different computational and mathematical concepts associated with design. We combine (1) the pedagogical categories for the teaching of GS proposed by Fischer and Herr[33], (2) the concepts identified by Kolarevic to characterize the digital morphogenesis[34], (3) the digital models of design beyond CAD systems described by Oxman[35] and (4) the mathematical concepts identified by Burry and Burry[36] in contemporary design. We organize them according to the following schemas: complex geometry and topology, packing and tiling, diagramming and data visualization, animation, optimization and performance, parametrics, rule-based systems and complexity.

Recent classifications have moved away from the computational logic to focus on their source or the application domain, indicating the incorporation of GS as a common practice for architectural design. Oxman and Oxman[39] provide six general models of form generation: (1) mathematical: which exploits mathematical formulae for generative procedures; (2) tectonic, which employs tectonic patterns for form generation; (3) material: which uses tectonic and assembly patterns,

Schemas	Simon	Mitchell	Henrion	Liggett
Search	• Heuristic search	• Heuristic search	• Jigsaw (satisficer) • Dissection (satisficer)	• Constructive procedures
Optimization	• Linear programming • Dynamic programming • Geometric programming • Queuing theory • Control theory	• Generate and Test • Improvement • Nonlinear and linear programming	• Additive (optimizer) • Permutational (optimizer) • Hybrid optimizing (optimizer)	• Improvement procedures • Simulated annealing • Genetic algorithm
Hybrid			• "Satisfeasing"[28] • Dissection with dimensional optimizing	• Hybrid approach
Other	• Generate and Test[29]	• Generate and Test[29] • Analytical procedures • (Shape grammars)[30]		

Table 1: Computational schemas and techniques for GS and space planning.

Schemas	Fischer and Herr	Kolarevic	Oxman	Burry and Burry
Complex geometry and topology		• Topology • Non-Euclidean geometry • NURBS	• Topological formation model (Formation)	• Topology • Mathematical surfaces and series
Packing and Tiling				• Packing and Tiling
Diagramming and data visualization	• Algorithmic generation and growth (data mapping)[37]	• Datascapes		• Datascapes and multi-dimensionality
Animation		• Metamorphosis • Dynamics and fields of forces	• Motion-based formation model (Formation)	
Optimization and performance	• Algorithmic (re-) production (e.g. genetic algorithms, selective procedures)	• Genetics (genetic algorithm)[38] • Performative Architecture	• Evolutionary design model (Generative) • Performance-based formation model (Performance) • Performance-based generation model (Performance)	• Optimization
Parametrics	• Algorithmic generation and growth (parametric design)[37]	• Parametrics	• Associative design formation model (Formation)	
Rule-based modeling and complexity	• Algorithmic generation and growth (e.g. fractals, re-writing rules)[37] • Emergent systems, self-organization (e.g. cellular automata and swarm modelling) • Generative grammars (e.g. L-systems and shape-grammars)	• Genetics (L-systems)[38] • Non-linearity, indeterminacy and emergence	• Grammatical transformative design model (Generative)	• Chaos, complexity and emergence

Table 2: Recent concepts associated with GS and computational design.

Schemas	Design space	Design navigation	Examples[42]
1. Unstructured constructive schema	Possible transformations of the initial state by set of rules	Application of rules or procedures	Fractals, L-systems, mesh subdivision, tessellations and shape grammars
2. Structured constructive schema	Possible transformations of the initial state by set of rules structured by graph or tree	Search, traversal or sampling by application of rules or procedures	Partition trees, uninformed search, heuristic search, random search, colorings and dissections
3. Variational schema	Parameter space with properties and relations between entities	Navigation in parameter space	Early algorithmic art (8-corner, Turbulence centered and Schotter) and parametric models
4. Improvement schema	Parameter space with properties and relations between entities with corresponding fitness value.	Navigation in parameter space, guided by fitness landscape	Hill climbing, gradient descent, simulated annealing, particle swarm optimization and evolutionary algorithms
5. Discrete simulation	Possible states of a composite representation, starting from an initial configuration	Local rules applied on the components of the representation over time	Cellular automata, reaction-diffusion, diffusion-limited aggregation, discrete urban simulations, slime mold and ant foraging
6. Continuous simulation	States resulting from interaction between agents and continuous environment	Local rules followed by each agent in the environment	Rigid/soft body simulation, differential growth and swarms
7. Generative learning	Parameter space and generative functions learned by data distribution	Navigation in the learned parameter space	Autoencoder, GAN and PCA
8. Behavioral learning	The policy space, which combines the state and action spaces of the system	Application of the learned policy	The Bucket Brigade, Monte Carlo learning, Temporal-difference learning and Deep Q-Learning
Hybrid	Combine previous design spaces	Combine previous design navigations	Parametric Shape Grammars (1 + 3), Shape Annealing (1 + 4), Shape GAN (1 + 7), optimization of a partition tree (2 + 4) and learning steering behaviors (6 + 8)

Table 3: A GS framework based on design space and navigation

such as folding, braiding, knitting and weaving, to generate form; (4) natural or neo-biological, which employs biological principles to generate form; (5) fabricational, which uses existing patterns of fabrication for design generation; and (6) performative, which models physical data of the context as the input for a generative process that satisfy certain objectives. This type of categorization reflects the ubiquity of GS in different architectural approaches, which is also reinforced by categorization of solution procedures in specific domains, such as digital fabrication with parametric modeling[40].

3.A FRAMEWORK FOR GS

The challenge in categorizing GS is that their underlying techniques originate in different fields, they overlap, or they apply to multiple domains of design. While pioneer classifications focused on the computational logic of the GS, recent ones address specific technologies, design inspiration or application domain in architecture. These recent approaches are very productive for education and bring design to the center of attention. However, they limit the scope of GS to the status quo and design instantiation – i.e., to a set of solutions with its own tested workflows and tools.

In the opposite direction, our framework recovers the idea of computation not as a tool for design, but as an alternative logic of design. It does not focus on the representation of the design elements but on the high-level concepts that mediate design and computational generation: design spaces and navigational strategies. Broadly speaking, design space refers to the space of possible alternatives that can be generated given a certain formulation of the problem. Design navigation refers to the operations and control strategies that are available to navigate between these alternatives. In traditional practices, designers use heuristics to formulate and reformulate the problem, building expressive design spaces and navigational strategies to explore solution candidates.[41] In GS, the design spaces and navigation strategies are a consequence of its formulation with specific algorithms and models.

By focusing on computational logic with more abstract categories, this framework comprehends both the existing solution procedures and the potential incorporation of recent advancements in AI. The eight schemas of our proposed framework are in Table 3.

Unstructured constructive schema

The schema is based on the existence of a discrete representation, which can be geometrical or even alphanumerical, that is sequentially constructed by the application of rules or procedures, which might require a certain shape or a certain relation between shapes to be matched. Once the initial

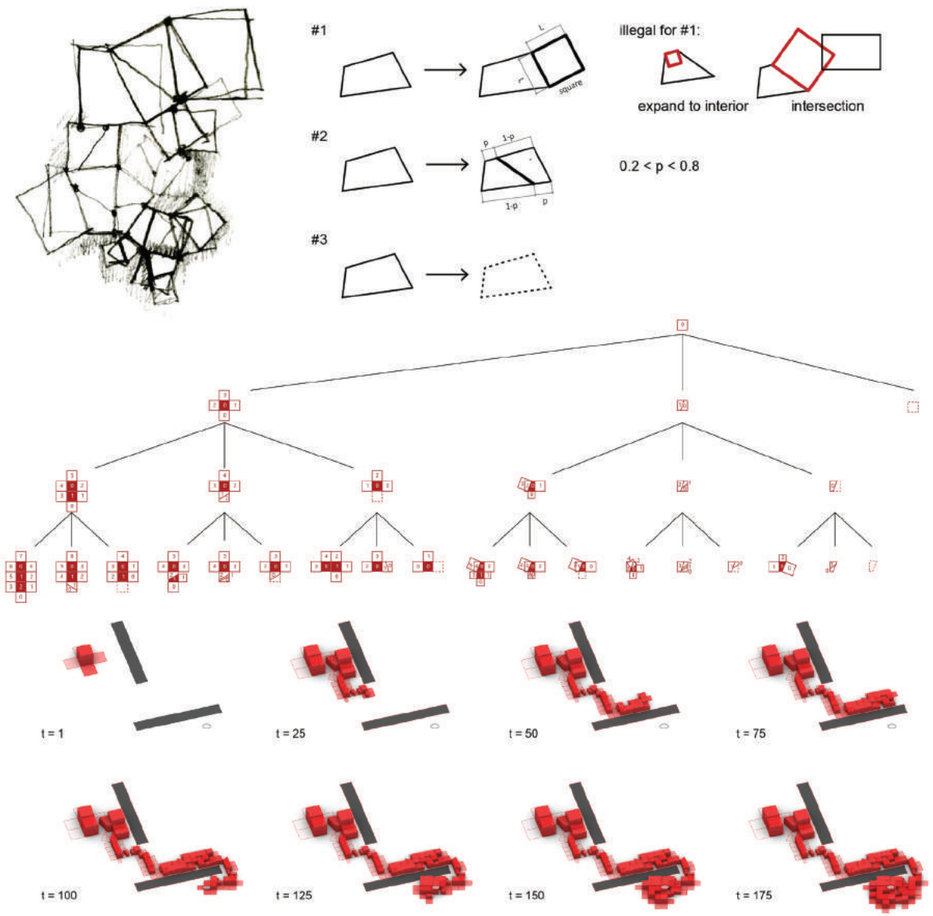

Figure 1: Constructive schema. Top: unstructured constructive schema of the qGrowth grammar. Middle: qGrowth structured in a tree with a greedy best-first search algorithm that orders the frontier based on the Euclidean distance to a target. Bottom: example of qGrowth generated by sampling. It follows the target behind a wall (ellipse).

state of the system is specified, navigation is defined by the selection and application of rules or procedures, manually or automatically, to change the state of the system in discrete steps. The design space is neither explicit nor definitely finite. See example in Figure 1, top.

Structured constructive schema

This schema addresses problems structured as a graph or a tree, which are composed of states (nodes) and available actions (edges). The states are the nodes and the possible constructions or actions are the edges. Navigation combinatorically explores valid states in the design space, which contains "the set of all states reachable from the initial state by any given sequence of actions"[43]. In a search, the solution is a sequence of actions from an initial state at the root to a desired state. In a traversal, the solution is a systematic way to access some or all the states. In a sampling, a solution is generated by following a single path defined by a certain probability or policy. Different algorithms organize the search in different ways by using strategies to order the exploration, to evaluate the costs of the decisions or to check the consistency of the different constraints of the problem. See example in Figure 1, middle and bottom.

Variational schema

This schema defines geometric entities and relations using

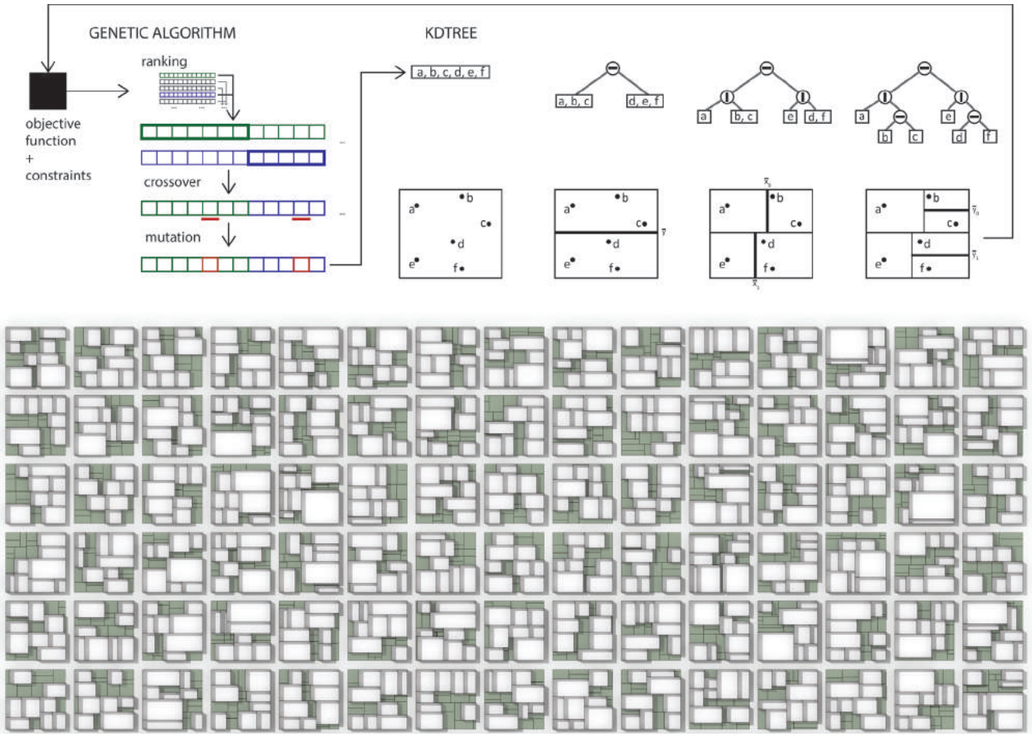

Figure 2: Hybrid (2 + 4). Optimization of floorplans with Genetic algorithm and KDTree.[50]

parameters and constraints through explicit functions defined by the designer. The resulting design space is explicit in the parameter space. It comprehends all the geometrical and numerical variations resulting from all possible combination of the parameter values, which can be done manually or algorithmically, using stochastic or deterministic procedures.

Improvement schema

This schema is based on the transformation of a state by a search for alternatives that perform better according to a metric. The design space is the parameter space, where parameter values describe all possible design solutions. The function space contains the possible results of a single or many evaluative functions applied to a solution. The fitness space is a one-dimensional space that translates the results of the functions to a single measurement of success. The combination of these spaces results in a representation called a fitness landscape that contains all the fitness value for all the solutions in the parameter space. Improvement procedures can be solved by the application of calculii, optimization strategies or metaheuristics. See hybrid example in Figure 2.

Discrete Simulation

This schema is based on a discrete representation, such as a grid or a graph. In combination, the states of the local units characterize the global state. Once an initial global state is defined, the rules or procedures affect some elements of the collection, based on local states and neighborhoods, whilst preserving the global characteristics of the representation. The design space comprehends all possible variations of the global representation, considering the initial state and the set of rules applied over time. The examples of discrete simulations are generally associated with mathematical models of urban or natural phenomena. See example in Figure 3.

Continuous simulation

This schema is based on a collection of agents that sense, act and interact. Each agent interweaves local evaluation of its goals and actions on the environment – the space, fixed elements and the other agents – by the application of local rules or procedures. Agents can have differing levels of autonomy. While simpler systems use basic reflex agents, more complex systems have a program to evaluate and act. The design

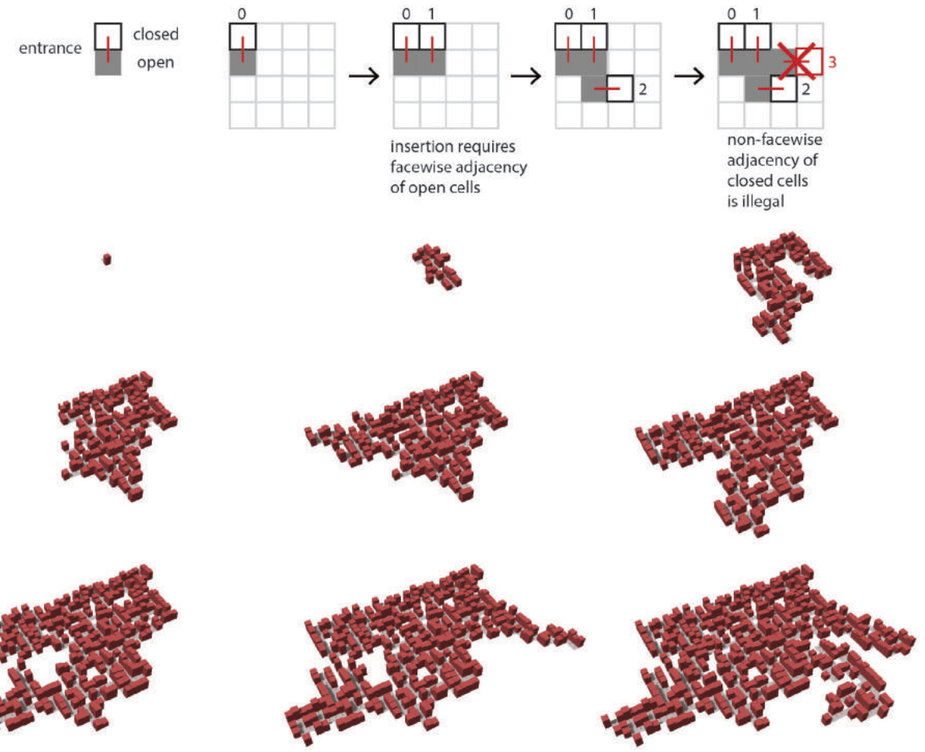

Figure 3: Discrete simulation. Rules and example of Beady Ring.[51]

space is defined by all possible configurations of a simulation, given its initial settings and the agents' policies

The next two categories employ Machine Learning – a multidisciplinary field concerned with programs that automatically improve with experience – to search for the best hypothesis about a given data, behavior or knowledge.[44] They open the possibility of learning GS from data.

Generative Learning

To solve many learning tasks, researchers employ generative models, which after having been exposed to a dataset, "explicitly or implicitly model the distribution of inputs as well as outputs"[45]. More than simply learning how to perform a certain task, they model how the data has been generated. Thus, a generative model enables the sampling of synthesized data based on the data distribution that it learned for the task. The design space is the space and domain of the input vector, which is translated to a result by the learned mapping. See example in Figure 4.

Behavioral Learning

This schema is based on learning the actions of agents to customize a GS. Among other approaches, it includes adaptive agents[46] and reinforcement learning[47]. The system learns a policy (the probability of choosing the available actions in a state) by combining simulation with evolutionary algorithms or by exploring and exploiting actions to maximize a reward in an environment.

In practice, the schemas presented above can be combined to form hybrid GS. By combining the different categories, designers can develop a custom GS with a proper design space and navigation for their problems.

4. A POST-DIGITAL COMPUTATION?

The post-digital should not be understood as an emerging era, but as a critical attitude towards the fascination or denial of the digital. New formulations of a digital zeitgeist[48] or re-mystifications of the architectural representation[49] are both short-sighted acts. In contrast, if we accept the ubiquity of the digital, we can use computation to explore complex spatial patterns and interactions, addressing previously ungraspable aspects of society and environment.

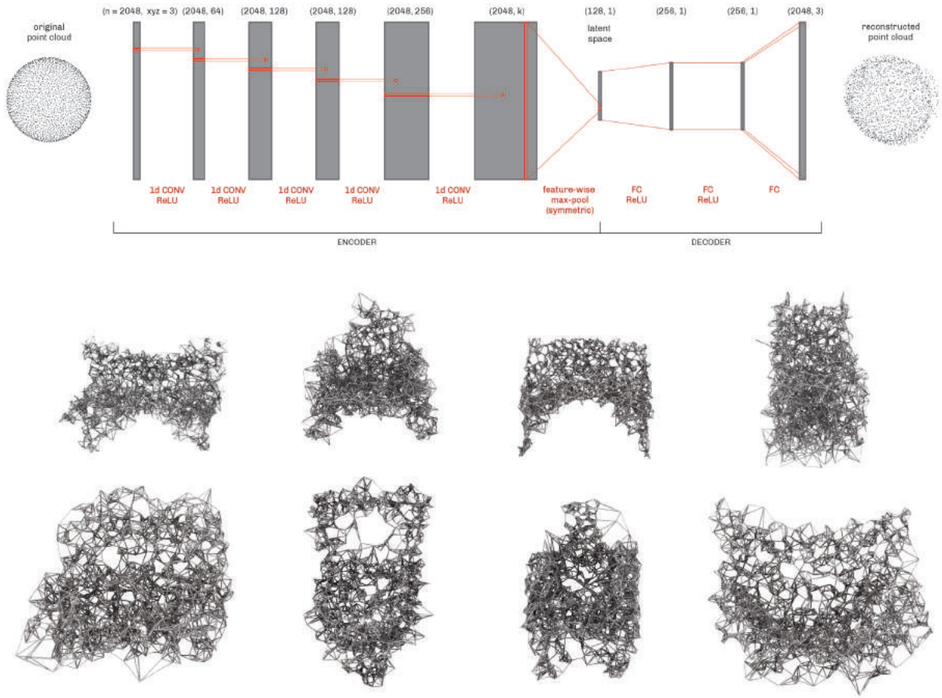

Figure 4 Generative learning: Deepcloud. Top: architecture of auto-encoder trained on a dataset of chairs. Bottom: chair series generated by navigation on the resulting feature space.[52]

Our own approach in this paper situates the logic and history of computation as a way of designing and not as a tool to express a digital condition. In the same way that an algorithm course in computer science might refer to canonical algorithms as a base for new developments – without claims for an algorithmic era –, canonical computational schemas can provide a common background to support research on GS. By exploring and hybridizing the different schemas, the designer can navigate in the complex territory of computation to look for novel design logics.

We introduced this framework in a mini course in our institution, addressing the first five schemas. In the end of the course, the students had to choose a problem in their domain (game, building, landscape, urban design, etc.) and develop a GS to produce alternatives of solutions. We still plan to extend it to a full course where we can discuss all the schemas, incorporating the recent advancements in AI. For a future objective, we intend to refine and formalize the categories and their relations into a book.

ACKNOWLEDGEMENTS

We would like to express our gratitude to the Brazilian National Council for Scientific and Technological Development (CNPq) for granting Pedro Veloso a PhD scholarship (grant 201374/2014-5)

ENDNOTES

1 *Reyner Banham, Theory and Design in the First Machine Age (The MIT Press, 1980).*

2 *A black box is a sealed device originally depicted in an electrical engineer problem. The observer must deduce the behavior of a sealed box by applying different disturbances to the input terminals and by observing the results.*

3 *Reyner Banham, "A Black Box: The Secret Profession of Architecture," in A Critic Writes: Selected Essays by Reyner Banham (Berkeley: University of California Press, 1999), 293.*

4 *Ibid., 298.*

5 *Mario Carpo, The Digital Turn in Architecture 1992-2012 (Chichester: John Wiley & Sons, 2013).*

6 *Rivka Oxman, "Theory and Design in the First Digital Age," Design Studies 27, no. 3 (2006): 229–265; Rivka Oxman, "Digital Architecture as a Challenge for Design Pedagogy: Theory, Knowledge, Models and Medium," Design Studies 29, no. 2 (2008): 99–120.*

7 *Joseph Rosa, Next Generation Architecture: Folds, Blobs, and Boxes (New York: Rizzoli, 2003); Peter Zellner, Hybrid Space: Generative Form and Digital Architecture (New York: Rizzoli, 1999); Branko Kolarevic, "Digital Morphogenesis," in Architecture in the Digital Age: Design and Manufacturing (London: Spoon Press, 2003), 12–28.*

8 *Greg Lynn, Animate Form (New York: Princeton Architectural Press, 1999); Patrik Schumacher, "Parametricism as Style - Parametricist Manifesto," 2008, https://www.patrikschumacher.com/Texts/Parametricism%20as%20Style.htm.*

9 *Banham, "A Black Box: The Secret Profession of Architecture," 294.*

10 *Joan Draper, "The Ecole Des Beaux-Arts and the Architectural Profession in the United States: The Case of John Galen Howard," in Architect: Chapters in the History of the Profession, ed. Spiro Kostof (Berkeley: University of California Press, 2000), 211; John F. Haberson, The Study of Architectural Design (New York: Norton, 2008), 182.*

11 *Stan Allen, "The Paperless Studios in Context," in When Is the Digital in Architecture? (Montreal: CCA and Sternberg Press, 2013), 383–404; Bernard Tschumi, "The Making of a Generation: How the Paperless Studios Came About," in When Is the Digital in Architecture? (Montreal: CCA and Sternberg Press, 2013), 405–19.*

12 *For example, parametric design can be associated with NURBS and mesh modeling to explore complex geometry. If performance is part of the agenda, plug-ins for optimization or physics simulation are added. Whenever some additional customization is necessary, the students try to "hack" the available toolbox to satisfice the design task.*

13 *Mario Carpo, The Second Digital Turn: Design Beyond Intelligence (Cambridge: The MIT Press, 2017).*

14 *Carpo's perception of technological change is legitimate, but his definition of a digital zeitgeist based on the duality of "don't sort; search" (Ibid., 23.) is fragile. In his interpretation, the discrete logic of computers and its capacity to operate with big data – which he refers to as posthuman complexity (Ibid., 48.) – is opposed to the formulaic and compressive logic of classical science. However, many of the techniques that Carpo uses to represent this logic of computation in design (heuristic search, optimization, cellular automata and simulation associated with FEM) are not directly related to big data. Additionally, computation is not opposed to compression. The difficulty of current (or even future) computers to deal with large and continuous state spaces justify the use of heuristics and function approximators, such as deep neural networks, instead of brute-force search or tabular methods. Carpo also opposes Calculus to the discrete logic of computation and its capacity to search (Ibid., 65–70.). However, not only Calculus can be discretized for geometry processing (see discrete differential geometry), but it is also the base for backpropagation, a main technique to train neural networks for big data analysis.*

15 *Gabriela Celani and Pedro Veloso, "The Intersection of Theory and Technology: Computational Concepts Applied to Architectural Design since Late 1980s - a Literature Review," in Frontiers of Science and Technology: Water Availability, Automation and Sensor Technologies, Digital Fabrication (7th Brazilian-German Conference, Campinas, 2017).*

16 *Gabriela Celani and Pedro Veloso, "CAAD Conferences: A Brief History," Electronic Proceedings of 16th International Conference CAAD Futures 2015. Sao Paulo, July 8-10, 2015., CAAD Futures, 2015.*

17 *Christopher Alexander, "Systems Generating Systems," Architectural Design 38, no. 12 (1968): 605–10.*

18 *Georg Vrachliotis, "How Form Came about from Society: Christopher Alexander or About Architecture as a Form of Culture and Structure," in Structuralism Reloaded: Rule-Based Design in Architecture and Urbanism (Stuttgart: Axel Menges, 2011), 61–68.*

19 *Christopher Alexander, "From a Set of Forces to a Form," in The Man-Made Object, ed. Gyorgy Kepes (New York: George Braziller, 1966), 96–107.*

20 *William J. Mitchell, Computer-Aided Architectural Design (New York: Mason Charter Pub, 1977).*

21 *Alexander's own work emphasized the use of relational methods for the synthesis of form, from the fusion of functional diagrams to the development of a language of semi-autonomous rule sets for design. For a general understanding of this transition, see: Christopher Alexander, Notes on the Synthesis of Form (Harvard University Press, 1964); Christopher Alexander, A Pattern Language: Towns, Buildings, Construction (Oxford University Press, 1977).*

22 *Mitchell, Computer-Aided Architectural Design, 46–48.*

23 *Ibid., 425–74.*

24 *Max Henrion, "Automatic Space-Planning: A Postmortem?," in Conference Proceedings (Artificial Intelligence and Pattern Recognition in Computer Aided Design, New York: North-Holland, 1978), 175–91.*

25 *Robin S. Liggett, "Automated Facilities Layout: Past, Present and Future," Automation in Construction 9, no. 2 (2000): 197–215.*

26 *While optimization is itself a form of search, it is differentiated from classical search methods because it does not explore a systematic path to a solution, but specific nodes/states, based on their fitness. This creates some ambiguities. For more details about classical search and optimization, see chapters 3 and 4 of Russel Stuart J. and Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. (Upper Saddle River: Prentice Hall, 2010).*

27 *"The Science of Design: Creating the Artificial," in The Sciences of the Artificial, 3rd ed. (Cambridge: The MIT Press, 1996).*

28 *Word created by Henrion to describe certain formulations of space planning that combine constraint satisfaction and goal optimization.*

29 *Generate and test is a general search paradigm based on the combination of a random generator and a tester, which filters the satisfying solutions. It can be considered a prototypical optimization where the tester is an objective function that returns a Boolean value or it can be structured as a classical search, if the generator relies on previous generations of solutions. Mitchell describe it as a problem-solving method for space-planning and Simon use it as a general strategy to decompose the design problem.*

30 *Mitchell describes shape grammars as a technique to formulate the problem (a syntactic formulation) and not as a solution procedure. This distinction is subtle, as the shape grammars can be developed with shape interpreters or with search strategies. However, shape grammars are canonical rule-based GS, so we subverted his original classification and added them to this table as a GS.*

31 *John Frazer, An Evolutionary Architecture, Architectural Association (London, 1995); Paul Coates and Robert Thum, Generative Modelling (University of East London, 1995), http://roar.uel.ac.uk/948/; Paul Coates, Programming Architecture (London: Routledge, 2010); Kostas Terzidis, Algorithmic Architecture (Oxford: The Architectural Press, 2006).*

32 *Kostas Terzidis, Permutation Design: Buildings, Texts, and Contexts (New York: Routledge, 2014).*

33 *Thomas Fischer and Christiane M. Herr, "Teaching Generative Design," in Proceedings of the 4th Conference on Generative Art (Generative Art, Politechnico di Milano University, 2001).*

34 *Kolarevic, "Digital Morphogenesis."*

35 *Oxman, "Theory and Design in the First Digital Age."*

36 *Jane Burry and Mark Burry, The New Mathematics of Architecture (Thames & Hudson Londres, 2010).*

37 *Fischer and Herr combine different techniques under the category "Algorithmic generation and growth". We opted to divide them according to our general categories.*

38 *Kolarevic mixes different algorithms (genetic algorithms and L-systems) under the category genetics. We opted to divide them according to our general categories.*

39 *"From Composition to Generation," in Theories of the Digital in Architecture (New York: Routledge, 2014), 55–61.*

40 *Lisa Iwamoto, Digital Fabrication: Architectural and Material Techniques (New York: Princeton Architectural Press, 2009); Nick Dunn, Digital Fabrication in Architecture (Laurence King Publishing, 2012).*

41 *Peter G. Rowe, Design Thinking (Cambridge: The MIT Press, 1987).*

42 *This is an initial attempt to define a framework, so we mixed general methods, models and algorithms from different areas in our examples.*

43 *Russel and Norvig, Norvig Peter, Artificial Intelligence, 67.*

44 *Tom M. Mitchell, Machine Learning (Boston: McGraw-Hill, 1997), 2–15.*

45 *Christopher M. Bishop, Pattern Recognition and Machine Learning (New York: Springer-Verlag, 2006), 43.*

46 *John H. Holland, Hidden Order: How Adaptation Builds Complexity (New York: Basic Books, 1995); John H. Holland, Signals and Boundaries: Building Blocks for Complex Adaptive Systems (Cambridge: The MIT Press, 2012).*

47 *Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction (Draft), 2nd ed. (Cambridge: The MIT Press, 2018), http://incompleteideas.net/book/.*

48 *Carpo, The Second Digital Turn; Mario Carpo, "The Post-Digital Will Be Even More Digital, Says Mario Carpo," Metropolis, July 5, 2018, https://www.metropolismag.com/ideas/post-digital-will-be-more-digital/.*

49 *Sam Jacob, "Architecture Enters the Age of Post-Digital Drawing," Metropolis, March 21, 2017, https://www.metropolismag.com/architecture/architecture-enters-age-post-digital-drawing/.*

50 *Inspired by Katja Knecht and Reinhard König, "Generating Floor Plan Layouts with Kd Trees and Evolutionary Algorithms," in Proceedings of the 13th Generative Art Conference (Generative Art, Milan: Domus Argenia Publisher, 2010), 238–253, http://www.generativeart.com/.*

51 *Bill Hillier and Julienne Hanson, The Social Logic of Space (Cambridge: Cambridge University Press, 1984), 55–64.*

52 *Ardavan Bidgoli and Pedro Veloso, "DeepCloud. The Application of a Data-Driven, Generative Model in Design," in Recalibration: On Imprecision and Infidelity. Proceedings of 38th ACADIA Conference, ed. Phillip Anzalone, Marcella Del Signore, and Andrew J. Wit (ACADIA, Mexico City: Universidad Iberoamericana, 2018), 176, 180.*