

## ALGOGRAM: AUTOMATED DIAGRAMS FOR AN ARCHITECTURAL DESIGN STUDIO

CHRISTIANE M HERR AND JUSTYNA KARAKIEWICZ  
*University of Hong Kong, Hong Kong*

**Abstract.** Building on a previously presented theoretical model for the integration of cellular automata into the design process, this paper introduces cellular automata as architectural design process support in the form of automated conceptual diagrams. This approach is the outcome of a series of successive software implementations, which are briefly outlined in terms of key features and observations made during their applications. The main part of the paper focuses on *Algogram*, the latest implementation, and its application in a second year design studio. The integrated concept of abstract representations, automated design exploration and individual interpretation is introduced as *automated diagram*.

### 1. Introduction

Among generative design strategies available to architectural computing, cellular automata (CA) have been described as an approach to support design by facilitating the generation of complex patterns (Chase 2005). CA were originally invented as mathematical games by Ulam (Schrandt and Ulam 1970) during the 1940s, and subsequently applied in a variety of fields to simulate complex processes based on the parallel and local interaction of elements. Previous applications of CA in architectural design have emphasized representations of form with cells depicting building volumes in a variety of ways. In the work of Coates et al. (1996), Krawczyk (2002) and Watanabe (2002), CA are used as generative tools to produce variations of building form (see Figure 1). The CA systems employed in these examples are based on uniform Cartesian grids with cubic cells, usually with very few cell states. Binary cell states such as “zero” and “one” or “dead” and “alive” are often translated into volumetric representations of matter or voids, resulting in three-dimensional forms that are then interpreted as building mass.

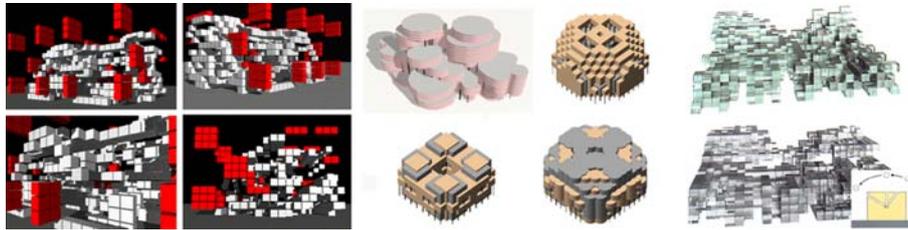


Figure 1. CA as form generators in the work of Coates (left), Krawczyk (middle) and Watanabe (right)

In the early, conceptual design stages, architects typically use visual representations such as sketches and diagrams, which relate more to concepts rather than objects (as discussed in Section 3 below). The direct association of CA-generated shapes with building form as in the examples cited above may thus be premature and limiting. The approach taken here builds upon a previously proposed extended CA model that has been adapted for architectural design purposes (Herr and Kvan 2007). It builds on the design process model proposed by Schön (1983), which is extended to include CA-based design support as part of the design move and offers CA support in form of optional, small-scale steps to an otherwise conventional design process. To explore the implications of this approach and further develop the underlying theoretical assumptions, a series of four software implementations were developed and tested with architecture students in design studio and workshop settings. It was not the purpose of this study to optimize a specific tool. Software implementations instead served as platforms to explore an approach to CA-based design processes and provided mainly qualitative data for this study. As a result of outcomes from the software implementations, this study progressed from an initial focus on CA as generators of building form to an emphasis on rule-based support of diagrammatic representations. This paper first briefly presents the explorative research process that led to the application of CA-based design support in form of automated diagrams. The final implementation, *Algogram*, and its application in a second year design studio are described in more detail in the second half of the paper.

## 2. Exploring CA-based Modeling of Form and Relationships

The first three implementations of the initially proposed extended CA model (Herr and Kvan 2007) indicated several shortcomings of the approach. They are briefly summarized below to provide the background of the subsequent development of *Algogram*, which discussed in more detail in Section 4 and 5. Following a previously published initial pilot study (see Herr and Kvan 2005), the first implementation was used by 23 graduate architecture

students in the initial phase of a design studio at The University of Hong Kong in late 2005. Implemented in *3DStudio MAX Script*, the tool offers its users a graphical interface to CA-based modeling functions to complement conventional three-dimensional modeling in *3DStudio MAX*. A wide variety of geometrical objects can be defined as cells, and then be assigned user-defined rules. The main purpose of this studio test was to explore how students would use and adapt CA-based software in their individual design processes, and how this would affect their design outcomes. While students readily experimented with the use of rules and incorporated the forms generated by the software into their projects, some have questioned the disruption rule definitions caused in their thinking process as well as the lack of predictability in the generated outcomes. What was on the one hand perceived as an interesting new method also discouraged students who previously had used computers only in deterministic ways to illustrate previously developed ideas. Most students however eventually engaged in an exploratory mode of design experimentation instead of a goal-driven one (see Figure 2). As a consequence, students expressed their wish for generative functions that would require less detailed rule definitions but would generate more complex and less predictable form.

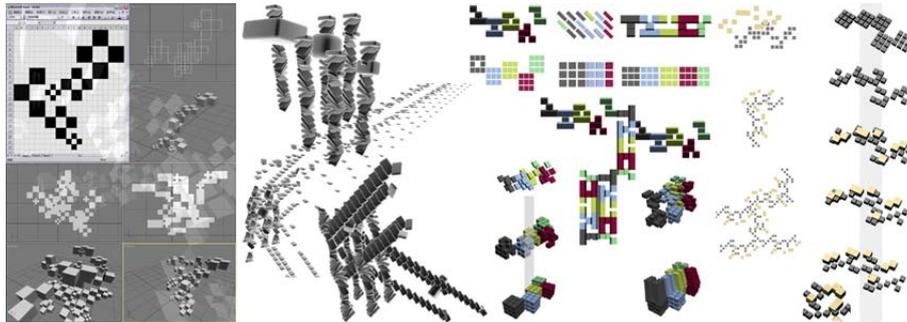


Figure 2. Exploratory design results generated with CA-based design support

In the following software implementation, named *Tofu Automata Generator*, rules were encapsulated into a set of predefined functions, which were geared towards quickly and more easily generating a variety of intricate geometric configurations. For this purpose, the available basic geometries were limited to box shapes only, which could be defined as “matter”, “void”, “neutral” or “context”. Instead of complex neighborhood definitions, the software gives users the option of manipulating geometries by relating geometric transformations to the different object types mentioned above. The *Tofu Automata Generator* was tested in a design workshop with 17 graduate architecture students at National Cheng Kung University in Taiwan. Results from the workshop indicated shortcomings resulting from a

focus on immediate form representation (see Fischer and Herr 2007). To achieve their design intentions, students often used the software in ways unintended by the developer, or re-interpreted forms or functions in the *Tofu Automata Generator* into new concepts and meanings. This kind of misappropriation, however, tended to fit individual design concepts better than the intended modes of software use, and produced a greater variety of design outcomes than expected. It thus indicated the necessity of encouraging individual interpretations in CA-based design support software. As a result of this implementation, flexibility in design support tools emerged as a main concern, as designers are likely to diverge significantly from the mode of use anticipated by software developers.

The third software implementation was used to support an individual graduate student of architecture at The University of Hong Kong. Her design project included a rule-based way to generate city models, which had much in common with CA systems. The software developed to support her design enabled the student to generate varieties of options by first setting up a set of neighborhood rules. These rules could then be modified by manipulating several parameters relating to the composition of land use functions in the given city model (see Figure 3).

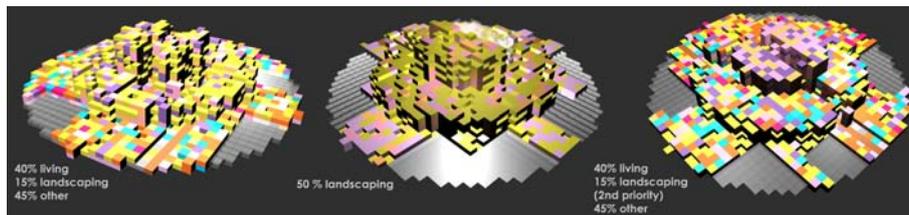


Figure 3. Peggy Louie: KCRC station new town proposals

As in the previous implementations, the use of CA-based design support was combined with manual modeling to allow for exceptions from rule-generated form if desired. Outcomes generated by the software could be read in different ways. While the student's intention was to generate diagrams of urban context to be further interpreted when translating them into form, the cube-based volumes of the graphical representation invited the misconception of the three-dimensional models as directly representing physical form. This however led to new insights: it demonstrated how form generated through CA could play a more abstract and conceptual role in the early design process. Aspects central to this implementation were diagrammatic representations that were intended to provoke further interpretation as well as a focus on element relationships rather than form. These features provided the core of *Algogram*, the following software implementation, and a new focus of the study.

### 3. Sketches, Diagrams and Automated Diagrams

In architectural design, the established design aid of choice for generating novelty is sketching (Fish 1996), which is used at all stages of the design process, from initial ideas to refinements of the final shape. Sketches work by enabling designers to externalize their thinking into graphic representations that provide visual cues for revision and refinement of design (Suwa and Tversky 1997). Schön (1983) describes the design process as “reflection in action”, with the designer engaged in a conversation with the design problem. This conversation consists of recurring cycles of framing, moving and reflection, in which the designer “sees” and then “moves” design objects in a representation of the design problem in question.

To support this creative reasoning process, architects employ visual representations such as sketches and diagrams that generate multiple interpretations by enabling different modes of seeing (Wiggins and Schön 1992). Though sketches are frequently described as “talking back” to the designer (Schön 1983, Goldschmidt 2006), in traditional sketching it is indeed the designer himself who interprets what he sees, perceiving this perspective change as “backtalk” of an active conversation partner. In the proposed CA-supported design process, the role of the computer during the early design stages is to enrich this dialogue between designer and visual representations (see Glanville 1999, p.88) by actively producing feedback that invites reinterpretation by the designer. In this study, CA are used to activate computer-based representations employed in conceptual design.

Drawings produced in the early design stages have been described as typically pertaining to two different types: sketches that aim to capture as well as generate design concepts and guiding principles, and sketches depicting physical form (Do and Gross 2001). Sketches that are used in the development of early design concepts frequently employ symbolic representations and focus more on relationships of basic design components than on building shape. This type of sketch is commonly termed conceptual diagram, recognized for its explanatory power as well as for its versatility in developing design ideas at a conceptual level (Phillips 2006). As drawings frequently exhibit characteristics of both types of design representation however, the distinction between diagrams and sketches remains blurry.

The emphasis on relationships of elements typically found in diagrams relates to a similar emphasis in CA systems. When introducing CA as design process support, this emphasis on relationships may be used to apply CA to provide rule-based diagrammatic representations to aid in the early, conceptual design stages. Such CA-based diagrams – or *automated diagrams* – are not limited to representing physical shape but potentially allow for multiple interpretations and mappings. Automated diagrams may be best suited for idea development during the early design process, when concept

framing and re-framing are more important than thinking about detailed building form (see Goel 1995, p. 193 ff). Two aspects are central to the concept of automated diagrams: Abstract and somewhat ambiguous diagrammatic representations can suggest issues and perspectives for consideration without imposing solutions (Gaver et al. 2003, p. 240). The automated component of such diagrams may be used to give such diagrammatic representations the capability to actively respond to configurational changes initiated by the designer. In this approach, the computer is not utilized merely as a tool to execute predetermined and deterministic commands but as medium in the sense of Glanville (1992).

#### **4. Algogram**

*Algogram* provides design support for the conceptual design of architectural programmes. It has been developed as a design aid that explores the potential of linking generative design, diagrams and cognitive research, resulting in the concept of automated diagrams. Ambiguity of representations and open-ended interpretation by the designer are recognized as central to early, conceptual design (Dogan and Nersessian 2002). In this context, the role of the computer is primarily to provide automated representations that support thinking and reflection in terms of relationships. *Algogram* aims to give designers an opportunity to work with generative CA-based processes while avoiding determining outcomes in terms of form too early in the design process. *Algogram* provides a generic way of representing architectural functions and their relationships, which is related to both bubble diagrams - not so much in their functional, but in their generative capacity as described by Emmons (2006), and Venn diagrams. Architectural functions are represented as spheres instead of cubes or squares to avoid premature visualizing of diagrams as possible building forms. Spheres relate more to diagrammatic representations of functions already known to students, such as bubble or Venn diagrams, while box shapes could easily be interpreted as physical building form. Representations in *Algogram* were intended to be somewhat ambiguous, such that students would be encouraged to reinterpret the diagram while developing their design concepts. The diagrammatic representations can be manipulated either manually or by establishing automated relationships between elements by defining rules. Rules in *Algogram* are used to numerically control relative positions of spheres and their intersections in three dimensions. Within the framework of the diagram, meaning is established by adding user-defined text labels and colours to generic spheres. To translate representations into architectural form, users need to interpret the resulting diagrams, which remain at a relatively abstract level to encourage individual design solutions.

*Algoqram* was developed for a studio project at The University of Hong Kong. The architectural approach taken in this studio project - and implemented in *Algoqram* - seeks to evade the limits of prescriptive typologies developed from the modernist motto of “form follows function”, exemplified by the examples given in AJ Metric or Neufert books. Following these examples likely results in buildings based on preconceptions, which hampers innovation and prevents re-interpretation of architectural programmes. Instead of solving design problems as quickly as possible, students were encouraged to re-think the initial problem statement, with the intention to lead students to re-frame the given problem.

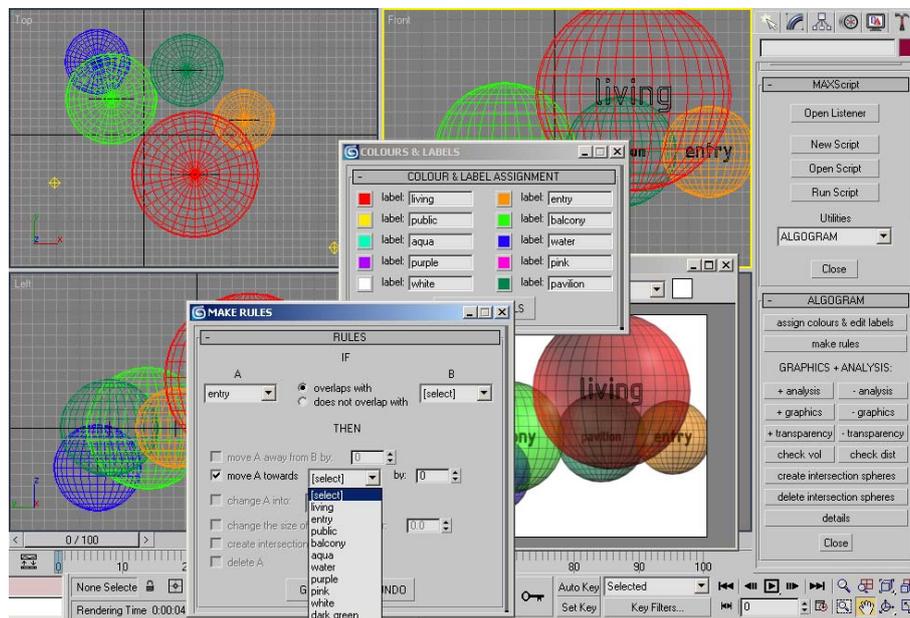


Figure 4. *Algoqram* user interface

*Algoqram* builds on the concept that architectural programmes are rarely stable and often need to be changed or reconfigured throughout the design process. The software supports experimentation with combinations of functions that may co-exist and develop in unusual ways to create different and often unpredictable results. To heighten students’ awareness of the potential of interactions between functions, we introduced the term “hybrid”. Hybrids in this context are opportunities for creating spaces that can be adapted for different uses and allow space to be reinterpreted for different functions. Such spaces are not typically found in traditional architectural programmes and can be generated by adjoining architectural functions that create unconventional synergy. In this way, a typical hybrid may for

example be created by two overlapping spheres labeled “performance” and “water”. In diagrams produced with *Algogram*, hybrids are a source of ambiguity intended to stimulate individual interpretation by the students, both during the conceptual design stages and also when translating into architectural form at a later stage in the design process.

*Algogram* is implemented as a utility in 3DStudio MAX (Figure 4), where it is accessible as an alternative to manual three-dimensional modeling. The three main components of the user interface are graphical representation functions, analysis functions and rules. To compose an “algogram”, users start by drawing spheres, which are then assigned colors from a predefined selection. Colors can be labeled with customizable names, usually indicating architectural programme items. Once assigned with colors, diagrams consisting of spheres can be analyzed for their properties such as composition, sphere volumes or distance between spheres. Diagrams can also be decomposed and presented as a chart overview. If spheres intersect, hybrids can be generated automatically. Hybrids are expressed as newly created spheres located in the centre of the overlap of two spheres. They are of mixed colors, according to the two original overlapping spheres. Sphere positions and properties can be manipulated either manually or with the help of custom configurable rules, which enable users to transform spheres in size or position according to their neighbors. Rules can be configured by choosing from a set of basic selections following an if / then logic, with additional, more detailed parameters.

### **5. Algogram in the Design Studio**

*Algogram* has been tested in a studio project at The University of Hong Kong with a group of ten second year undergraduate students, who had no previous CAAD software training. The purpose of testing the software in an applied design studio setting was to investigate two aspects of using *Algogram* in the early design stages: first, how students integrated the software into their individual design processes, and second, how this affected the outcomes. The project brief for the studio, which included several other student groups as well, called for the design of a school for a given site in Hong Kong, with particular consideration of urban and landscape context. The students were introduced to *Algogram* and *3DStudio MAX* during the early, conceptual design phase, which lasted for about two weeks. At this point, most students switched to traditional sketching and drawing tools for developing both programme and physical form of their buildings in greater detail. After the concept development phase, students were interviewed for their experiences during the design process as well as for their opinions on the usefulness of the software and suggestions for improvements. Students responded well to the software and readily

developed their concepts using the abstract representation in *Algogram*, even without prior CAAD knowledge. As *Algogram* provides only abstract sphere geometries, students had to conceptualize their school in terms of programme, without considering form (see Figure 5).

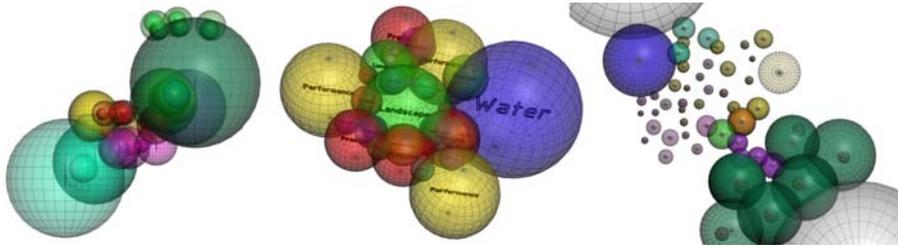


Figure 5. “algorithms” by Finnie Yu (left), Ho Wing Ho (middle), and Claire Fu (right)

The intention behind this strategy was to extend the conceptual design stage in order to allow a more thorough exploration of ideas and to prevent premature decisions. In terms of design process, this extended period of conceptual diagramming was aimed at encouraging the initial framing and re-framing of design problems instead of solving a problem as quickly as possible. In terms of architectural design approach, this strategy aimed to increase students’ awareness of interactions between functions and their role in conceptualizing new and context-specific architectural typologies. The expected outcome in the studio test was for students to engage in an explorative design process, negotiating between definitions of element relationships through rules on the one hand and individual interpretation of ambiguous graphical representations on the other hand. Data on the studio test was collected in form of student results, field notes, samples from other second year students who did not use *Algogram*, interview feedback as well as student sketches produced during the studio.

## 6. Studio Observations and Discussion

The studio test of *Algogram* aimed at open-ended qualitative feedback on how the software integrated with the students’ design processes and aims. Design outcomes of student projects in the group using *Algogram* were slightly different in nature than results from other student groups in the same studio project. This is likely an effect of the particular design approach encouraged, with an extended conceptual design phase and further enforced by the use of *Algogram*. Since student results tend to be shaped by the varying teaching styles and preferences of design tutors, however, this assessment of student work from groups working with or without *Algogram* is merely indicative. It is based on samples of student outcomes from three

other groups as well as field notes from these groups' design critiques. Design proposals from other groups tended to separate functions rather than considering their potential interactions within a building. Students instead focused on developing unconventional exterior building form. Reviewers of other groups' work often criticized the lack of reflection of urban context, which students tended to consider only in terms of composition, with functional aspects frequently ignored. The focus on functions in the student group that used *Algogram* seems to have encouraged students to integrate their design concepts more closely with surrounding urban and landscape context, with many diagrams emphasizing sphere labels such as water, hill, planting, public or private. Most students in the group using *Algogram* had strong individual design concepts that came out of a reconsideration of conventional building programmes. Students subsequently used these programme-oriented concepts to generate unique and site specific building typologies.

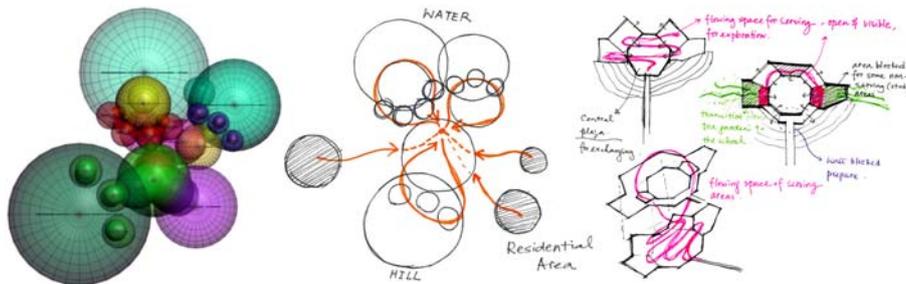


Figure 6. Translation of “algorithms” into building form by Finnie Yu

When switching to other design tools and media to develop building form, most students used the latest stages of their “algorithms” to guide their form development processes (Figure 6). In many cases, buildings closely resemble the three-dimensional concept diagrams in terms of composition, not however in terms of building form. It appears that after the conceptual diagramming phase, concepts were regarded as sufficiently fixed, and students focused mainly on the translation of diagrams into form. In the interviews, all students described using the software as generally helpful for their design processes, while the reasons given varied. “Algorithms” were seen as useful in finding and organizing relationships between functions within a building programme. They were also described as reminders of design potential, especially when considering and interpreting hybrids. The simplicity of the graphical representation was also perceived as helpful, since it allowed for conceptual clarity in the beginning of the design process. Most students appreciated the hybrids as source of inspiration, often describing it as the most important feature of *Algogram*.

It seems that students were happy with the ambiguity of hybrids when interpreting entire diagrams, but when hybrids were separated from their context, such as in the analysis chart overview, students perceived them as confusing. Sphere sizes in *Algogram* can be interpreted in several ways. Initially, students were introduced to sphere sizes as indicators of physical volumes allocated to a particular function. As their design progressed, however, students increasingly developed their own understandings. Many students switched to an interpretation that saw sphere sizes as a sign of importance and relative sphere distances as indicator of mutual relevance in a building programme. Students also reported frequent use of mixed representations, interpreting sphere sizes in some cases as representing physical size, in some cases as representing importance and in yet other cases in yet other ways - all within the same “algogram”. In this sense, ambiguity in “algograms” encourages and supports individual construction of meaning and design thinking. Of the analysis functions, the chart overview was described as especially useful, as it allows a visual overview of the number and type of spheres contained within a diagram.

The rules function received mixed comments: While some students regarded rules as very important, especially in the early diagramming stages, most students considered the interface for rule configuration in *Algogram* too restrictive and exact. They suggested providing rules that could be used in more intuitive ways in the early design stages. This echoes a description by Schön (1988), who characterized rules in design as largely implicit, diverse, contextually dependent and subject to exceptions and modifications. The most frequently mentioned drawback to the use of rules, however, was that similar to rules in classic CA systems, rules in *Algogram* affected all spheres of the same type in the same way. After an initial diagramming stage, in which only one or very few spheres of a given type exist at the same time, “algograms” quickly become more differentiated as relationships between architectural functions are further developed. In a more differentiated diagram, however, uniform rules do not apply any more.

In the final design critique, the majority of students explained their design process as well as their concepts using “algograms” in their presentations, with additional conceptual diagrams used in translating diagrams into building form. Students often used vocabulary introduced through *Algogram* to explain their design concepts. Similarly, student sketch books often showed hand-drawn diagrams using circles and labels similar to those in the software. In some cases, students reverted back to this notation even later in the design process when revising their design schemes at a fundamental level. These results seem to indicate that working with *Algogram* provided students with a notation and vocabulary that they felt comfortable to use in conceptual design development. Students typically described “algograms” as

helpful in the three-dimensional development of their schemes, and as more appropriate to concept development than bubble diagrams. The latter were thought of as appropriate for functional planning issues where architectural programme elements are considered in terms of numbers rather than spatial qualities.

According to the interviews, none of the students felt limited by the software in pursuing individual design ideas. Most positive comments on *Algogram* relate to its potential to provide students with representations of their design concepts while at the same time giving enough room for individual interpretations. In comparison to the utility of immediate visual representations, though, CA-based rules as implemented in *Algogram* seem to be marginalized, in particular past the initial diagramming stages. During the final design critique, it became apparent that the same properties of “algorithms” that are perceived useful by students during their design processes can result in ambiguous and vague elements in presentations that other students, tutors or critics not familiar with the particular design project find difficult to follow. *Algogram* teaches students to recognize the value of ambiguity in design representations during the design process, but also requires students to consciously reduce ambiguity in presentation materials intended to communicate the design to others. In summary, the processes and outcomes of the design studio differed from initial expectations in several ways. The students’ explorative design processes depended heavily on representations inviting individual interpretations and much less on the use of formalized rules. They instead preferred intuitive manipulation of geometry and avoided numeric parameter settings. Students embraced the open-ended aspects of the software, namely the ambiguity of hybrids as well as individual interpretation of sphere sizes and positions in abstract graphic representations. In comparison to the previous software implementations, students working with *Algogram* focused more on concept development than building form. The step of translating conceptual diagrams into form, however, appeared to be of considerable difficulty to the students.

## 7. Conclusion

From an initial focus on supporting design with CA-based form generators, software implementations in this study have developed to embrace diagrammatic representations. This emphasis on CA-based design support as providing automated diagrams for conceptual design resulted in the latest implementation, *Algogram*. Abstract diagrammatic representations were found to be helpful in the early design stages, as they encourage students to develop multiple individual interpretations and allow for an extended process of framing and re-framing design problems. This leads to a prolonged conceptual development stage and prevents a premature transition

to the level of physical building shape. Comments in student interviews indicate that students embraced the ambiguity inherent in the abstract spheres in *Algogram* diagrams to develop a variety of individual interpretations. While aiding students' design processes in this way, the ambiguity in even relatively simple configurations of spheres rendered outcome presentations to outsiders surprisingly challenging.

The generative potential of the proposed automated diagram model was found to be more a consequence of the abstract and partly ambiguous representations employed than it was a consequence of using CA-based rules. Uniform rule sets were perceived as helpful in the early setup of conceptual diagrams, but students criticized uniform rules as inadequate for more differentiated diagrams produced later in the design process. Students commented that rules should be implemented to be used in a more intuitive way, in particular without manual setting of parameters in numeric form, as this interrupts the design process. This leads to a question for further research: How can rules be integrated in a somewhat less precise and more intuitively controllable way, such that relationships between elements can be manipulated without requiring detailed definitions or numerical input? An open question also remains on how initially uniform rule sets can become more local and specific to allow for processes of differentiation in automated diagrams. Overall, *Algogram* supported students not only in developing individual design solutions, but also in exploring architectural typologies beyond their initial preconceptions.

### Acknowledgements

We would like to acknowledge the support and feedback by our colleagues and students at The University at Hong Kong, in particular Peggy Louie, Yunyan Jia and Janice Affleck, and at the Department of Architecture at the National Cheng Kung University. We are particularly grateful to the following students who have used *Algogram* in their studio work during the fall 2006 semester: Wu Tsz Ching, Zhang Jia Heng, Kwong Yan Kit, Ho Wing Ho, Yu Yuk Fan, Fu Hui Yan, Lee Man Yeng, Luk Lai Fun, Lo Yee Cheung and Kwan Chun Sing. We also thank Prof. Thomas Kvan for his feedback and guidance over the course of this study. This research is supported by a PhD scholarship granted by The University of Hong Kong.

### References

- Chase, S: 2005, Generative design tools for novice designers: Issues for selection, *Automation in Construction* **14** (6): 689-698.
- Coates, P, Healy, N, Lamb, C and Voon, WL: 1996, The Use of Cellular Automata to Explore Bottom-Up Architectonic Rules, paper presented at *Eurographics UK Chapter 14th Annual Conference held at Imperial College London/UK*.
- Do, EY-L and Gross, MD: 2001, Thinking with Diagrams in Architectural Design, *Artificial Intelligence Review* **15**(1-2): 135-149.

- Dogan, F and Nersessian, N: 2002, Conceptual diagrams: representing ideas in design, in M. Hegarty, B. Meyer and N.H. Narayanan (eds), *Diagrammatic representation and inference*, Springer, Berlin, pp. 353-355.
- Emmons, P: 2006, Embodying networks: bubble diagrams and the image of modern organicism, *The Journal of Architecture* **11**(4): 441 – 461.
- Fischer, T and Herr, CM: 2007, The architect as toolbreaker? Probing tool use in the practice of generative design, *CAADRIA 2007*, forthcoming.
- Fish, JC: 1996, *How Sketches Work - A Cognitive Theory for Improved System Design*, PhD thesis, Loughborough University of Technology.
- Gaver, WW., Beaver, J and Benford, S: 2003, Ambiguity as a resource for design, *Proceedings of the Conference on Human Factors in Computing Systems*, 5-10 April 2003, Fort Lauderdale, FL. ACM Press, New York, pp. 233-240.
- Glanville, R: 1992, CAD Abusing Computing, in *Proceedings eCAADe 1992*, Polytechnic University of Catalonia, Barcelona, pp. 213-224.
- Glanville, R: 1999, Researching Design and Designing Research, *Design Issues* **15**(2): 80-91.
- Goel, V: 1995, *Sketches of Thought*, MIT Press, Cambridge, MA.
- Goldschmidt, G: 2006, The Backtalk of Self-Generated Sketches, *Design Issues* **19**(1): 72-88.
- Herr, CM: 2003, Using Cellular Automata to Challenge Cookie-Cutter Architecture, in C Soddu (ed), *The 6th International Conference on Generative Art 2003*, Generative Design Lab, DiAP, Politecnico di Milano University, Milano, Italy, pp. 72-81.
- Herr, CM and Kvan, T: 2005, Using Cellular Automata to Generate High-Density Building Form, *Proceedings of the 11th International Conference on Computer Aided Architectural Design Futures, Vienna*, pp. 249-258.
- Herr, CM and Kvan, T: 2007, Adapting cellular automata to support the architectural design process, *Automation in Construction* **16**(1): 61-69.
- Krawcyk, R: 2002, Architectural Interpretation of Cellular Automata, in C Soddu (ed), *The 5th International Conference on Generative Art 2002*, Generative Design Lab, DiAP, Politecnico di Milano University, Milano, Italy, pp. 7.1-7.8.
- Phillips, A: 2006, Lines of Inquiry: diagrams as catalysts in creating memorable architecture, *Architectural Review* **219**(1307): 68-73.
- Schön, DA: 1983, *The reflective practitioner: how professionals think in action*, Basic Books, New York.
- Schön, DA: 1988, Designing: Rules, types and worlds, *Design Studies* **9**(3): 133-43.
- Schrandt, R and Ulam, S: 1970, On Recursively Defined Geometrical Objects and Patterns of Growth, in A Burks (ed), *Essays on Cellular Automata*, University of Illinois Press, Urbana, pp. 232-243.
- Suwa, M and Tversky, B: 1997, What do architects and students perceive in their sketches? A protocol analysis. *Design Studies* **18** (4): 385-403.
- Watanabe, MS: 2002, *Induction design: a method for evolutionary design*, Birkhäuser, Basel.
- Wiggins, G and Schön, DA: 1992, Kinds of seeing and their function in designing, *Design Studies* **13**(2): 135-156.