On the correlation of design and computational techniques in architectural education

Alexander Koutamanis

Factilty of Architecture, Delft University of Technology
NL-2628 CR Delft, The Netherlands

tel. +31-15-784957 • fax +31-15-784727 • e-mail KOUTAMANIS@BK.TUDELFT.NL

Abstract

Many studies employ analyses of human intelligence as justification or guideline for the development of machine intelligence. The main benefit brought on by such studies has been the improvement of our understanding of both human and machine intelligence. In teaching architecture with computers the same approach can make explicit design techniques architects use by means of equivalent or similar computational techniques. Explicitation of design techniques leads to a better understanding of architects' activities, as well as to which computer tools can offer automated support to these activities.

In the curriculum of the Faculty of Architecture, Delft University of Technology, relations and correspondences between computational and design techniques form a major underlying theme in computer-aided design courses. The purposes of this theme are (i) comprehension of the computational structure of a computer design tool, and (ii) explanation of how such computational structures relate to architectural design. Correspondences between the computational principles of computer programs and design techniques are instrumental in defining the scope of each computer tool in architectural design while improving the students' understanding of architectural design as a cognitive process and thus promoting automation as a natural extension of established conventional practices.

The paper outlines the correlation of computational and design techniques in the case of electronic spreadsheets. Spreadsheets are introduced through a thorough presentation of the various kinds and aspects of constraint propagation, their underlying computational principle. Numerical constraint propagation is explained by means of spreadsheet applications for simple numerical calculations. Symbolic constraint propagation is presented in the framework of machine perception. Both forms are then linked to architectural design through parametric design and the recognition of spaces in floor plans. Exercises linked to spreadsheets and constraint propagation include the parametric calculation of stairs and making parametric variations of a building on the basis of floor area calculations.

Introduction

The development of computer applications is traditionally justified and motivated by either the need for new, sharper tools or the necessity to understand better human intelligence. According to a certain line of thought the world is becoming too complex and too fast for our cognitive capabilities. The complexity of tasks such as the flying of fighter aircraft or the designing of an integrated circuit and the speed with which we often have to react under critical conditions suggest that we need external supports that assist in the execution of these tasks and thus improve our overall performance. Other tasks, such as the sorting of garbage or the control of dimensioning in engineering drawings, are too tedious, dangerous or demeaning. Advocates of automation suggest that these too should be computerized so as to free human capacity for other, more suitable tasks.

The other line of thought has little to do with the practical, short term benefits of computerization. Instead, it focuses on our lasting quest for the deeper structures of human functionality and performance. It proposes that, drawing from analyses of human intelligence, we can make computers simulate human abilities such as reasoning, recognition and language. The structure and performance of computer intelligence can then lead to a re-evaluation and improvement of our assumptions concerning human intelligence. The theoretical significance of this improvement is significant enough by itself but the practical advantages of machines that simulate litiman intelligence in a largely man-made world should not be ignored.

The area where the above duality has been more evident is Artificial Intelligence. There we encounter systems that attempt to reproduce human intelligence in order to either complement or analyse it, together with an explicit justification of the approach chosen. Computer-aided design systems are similarly either clear alternatives to established 'manual' practices or computerized versions of such practices. The difference is that practically all systems have been presented as a renewal of conventional architectural design, thereby leaving implicit the two underlying approaches and their consequences with respect to both theory and practice. Architectural education reflects this implicit duality with exercises that aim at familiarizing the student with specific types of computer hardware and software and other exercises that attempt to provide a more explicit picture of architecture and design through the use of the computer.

Rather than choosing between the two justifications and approaches on the basis of advantages and disadvantages the computer curriculum of the Faculty of Architecture, Delft University of Technology, attempts to integrate both, towards a better understanding of architectural design and of computer-aided design. The lesson to be learned from Artificial Intelligence and computer science in general is that the correlation of machine and human intelligence has been instrumental for the improvement of our understanding of the structure of human intelligence and of the possibilities of machine intelligence. Design viewed as a cognitive process remains largely unexplored, especially with respect to the architects' activities in practice. Quite often what is presented as "design methods" or "design techniques" is little more than opinionated prescriptive formulas with a simplistic attitude towards the complexity and gravity of design decision making in practice. Computer-aided design has similarly fallen victim to misconceptions and preconceptions stemming from the same lack of deep, comprehensive knowledge of architectural design and of its needs in automation.

By explaining the architects' cognitive activities through computational techniques we arrive at an explicit, quantifiable formulation of discrete, well-defined parts or aspects of the design process and, eventually of architectural design as a whole. This formulation can be tested through computer implementations, evaluated using criteria from both architectural practice and computer science, and progressively developed into a comprehensive and consistent framework for the analysis and automation of architectural design. In practical terms correlation of design and computational techniques and the resulting explicitation of the structure of architectural design and of the computational principles of computer-aided design means that it is possible to

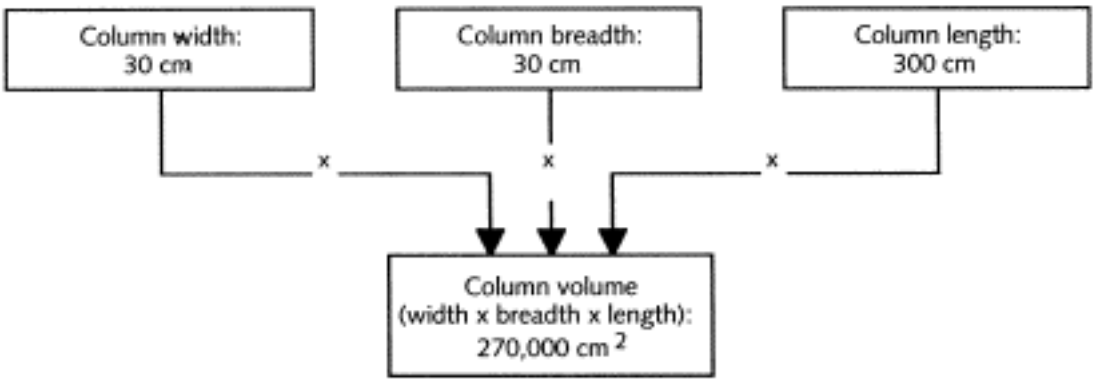
identify techniques and principles suitable for the partial or total automation of design tasks. Consequently, we can also identify existing computer software that is based on these techniques and principles and specify the ways that it should be used or adapted.

One clear example of the practical possibilities offered by the correlation of design and computational techniques is found in the use of electronic spreadsheets in computer-aided architectural design. Spreadsheets seldom appear in computer-aided design textbooks, especially outside chapters on office organization. In practice spreadsheets are used for various indifferent tasks, such as database management, simple calculations and charts, or even making tables. The limited use of spreadsheets can be attributed to a superficial understanding of their computational structure and of its relevance to particular design techniques. The following sections explain the significance and relevance of this computational structure and outline the way this is presented to and used by students in the framework of design computing exercises.

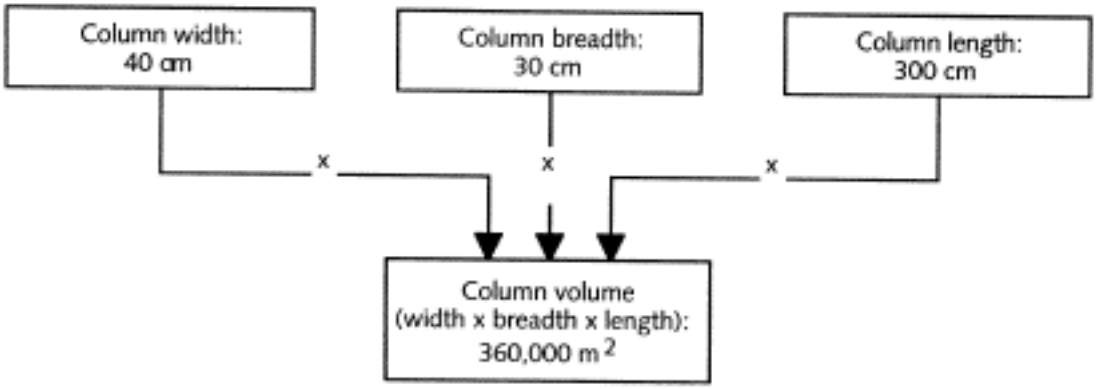
Constraint propagation

The main difference between electronic spreadsheets and other types of computer software also capable of numerical and logical calculations is the underlying principle of *constraint propagation*. Constraint propagation connects local computations made on the basis of local constraints into a comprehensive system defined by the relationships between constraints. Constraint propagation is particularly attractive for the resolution of large, complex problems by reducing them into smaller, simpler local problems and then correlating the local results until global consistency is achieved [Winston, 1984, Chapter 3].

In a spreadsheet constraint propagation takes place in networks of cells. A spreadsheet cell typically contains a verbal annotation (text), an input value (variable) or a formula that returns an output value. A formula usually contains references to other cells. For example, in calculating the volume of a rectangular column, we allocate input (i.e., the column width, breadth and length) to three separate cells and output, the column volume, to a fourth cell. This fourth cell contains a formula indicating that the volume can be obtained by multiplying the three dimensions of the column, or rather, the content of the cells which accommodate these dimensions. In the calculation of this formula the values of the necessary cells are propagated to the formula cell.

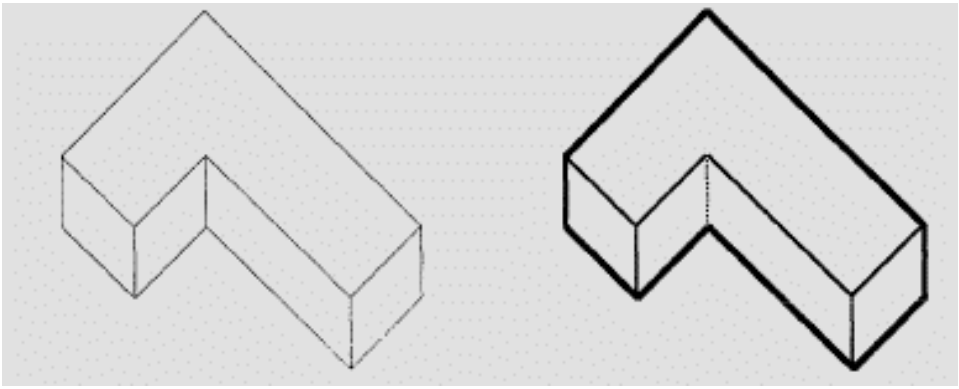


The implementation of the calculation in a network of four cells distinguishes the variables from the value returned by their computation. The practical advantage of the distinction is that a change in the variables is directly reflected in the output without any modification of the formula. If, for example, the width of the column increases to 40 cm the volume of the column increases accordingly, as the new value is automatically propagated to the formula when making the calculation.



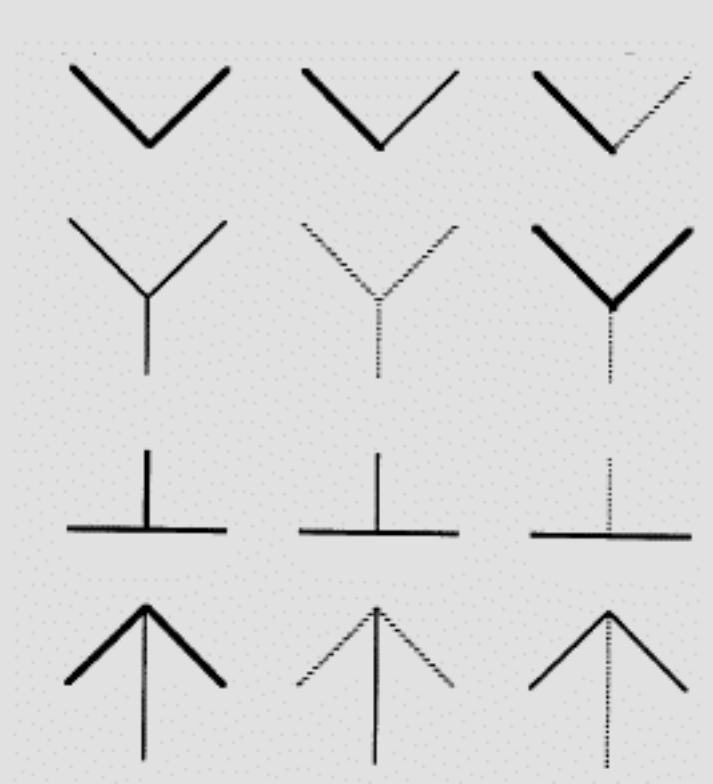
Constraint propagation is not restricted to numerical cases like the above example. *Symbolic* constraint propagation underlies an acknowledged approach to computer recognition of line drawings of axonometric projections of polyhedral objects [Guzmán, 1966; Clowes, 1971; Huffman, 1971; Waltz, 1975]. This approach has close correspondences with human recognition of similar scenes [Green and Curtis, 1966; Attneave, 1972; Walters, 1986]. In these axonometric projections we encounter lines of the following types:

- Occluding edges which indicate the visible outline of an object (thicker lines in the illustration).
- Lines which indicate the outline of a surface of an object. These lines can be either convex or concave edges of a polyhedron (respectively thinner and broken lines in the illustration).



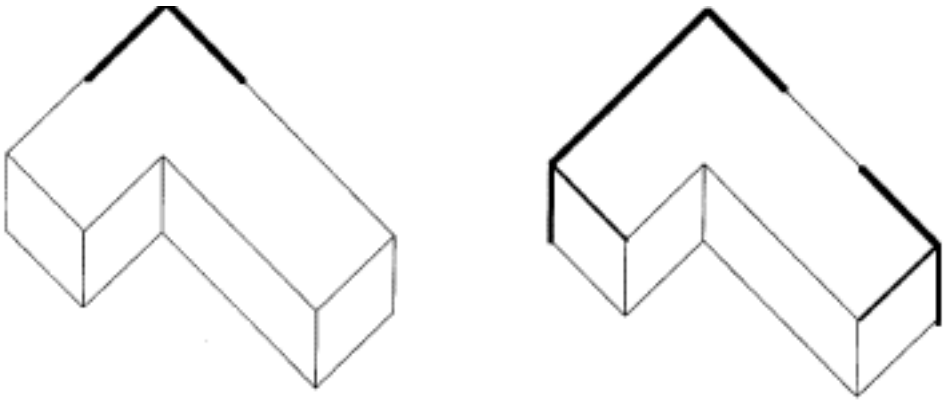
These lines are connected to each other at junctions which correspond to polyhedral corners of an object or to an edge of an object that is occluded by another object (i.e., to the junction of an occluding edge with an edge it occludes). If we exclude a small number of infrequent or exceptional cases, in an axonometric projection of rectangular polyhedra there are just four possible junction types:

By labelling the edges of each type as occluding, convex or concave we arrive at the following twelve possible configurations:



Using the above as the natural constraints of the problem we can recognize the surfaces and volumes in an axonometric projection of rectangular polyhedra through constraint propagation. Recognition starts at an arbitrary edge junction in the drawing, preferably one that supplies enough clues as to the label of its edges. For example, the top junction in such a drawing contains by definition two occluding edges. In the following example the top junction is L-shaped and therefore both its edges must be occluding. As the resulting edge configuration is acceptable in the typology of possible edge configurations, the two edges of the junction are accepted and labelled as occluding edges.

Recognition proceeds with the junctions at the other end of the two already labelled edges. Both new edges are arrow-shaped and contain one occluding edge (as the already recognized edges obviously retain their labelling). In the typology of possible junction configurations there is only one arrow configuration with occluding edges, meaning that the outer edges in both junctions can only be occluding and the third edge convex.

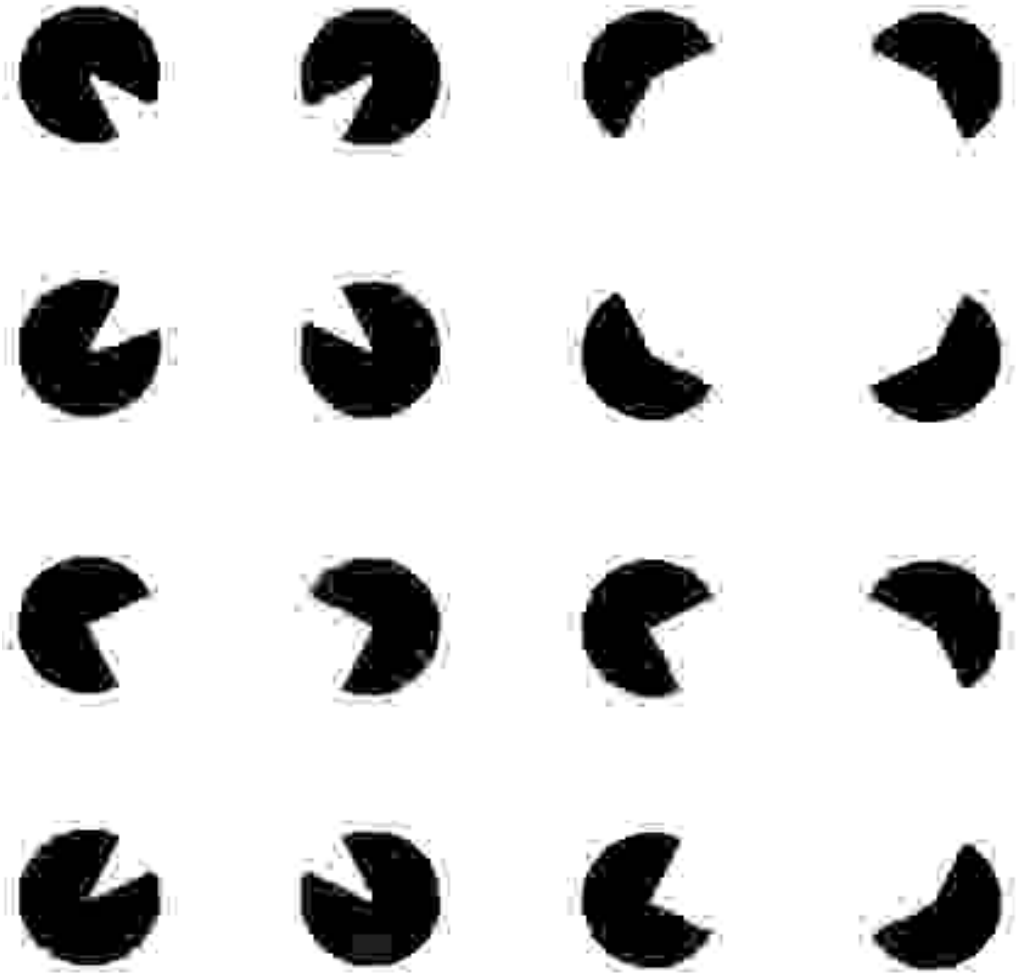


The newly recognized edges are labelled and recognition proceeds with the junctions at the other end of already recognized edges until every edge in the drawing is labelled. it is possible that at a certain step more than one possible junction configurations are acceptable for a junction in the drawing. In such a case edges are labelled with all

possible alternatives and all labels are propagated in parallel. An alternative is rejected when it leads to an impossible or otherwise unacceptable configuration at an edge junction (reductio ad absurdum). In large, complex drawings a number of iterations may be required before all but one alternatives are rejected.

Once all edges are uniquely and unambiguously labelled, surfaces can be recognized by the configuration of edges that surround them. The labels of edges around a surface are sufficient for the identification of orientation and of spatial relationships between surfaces. These relationships in turn are sufficient for the recognition of volumes in the drawing and of their relative position in the drawing [Waltz, 1975].

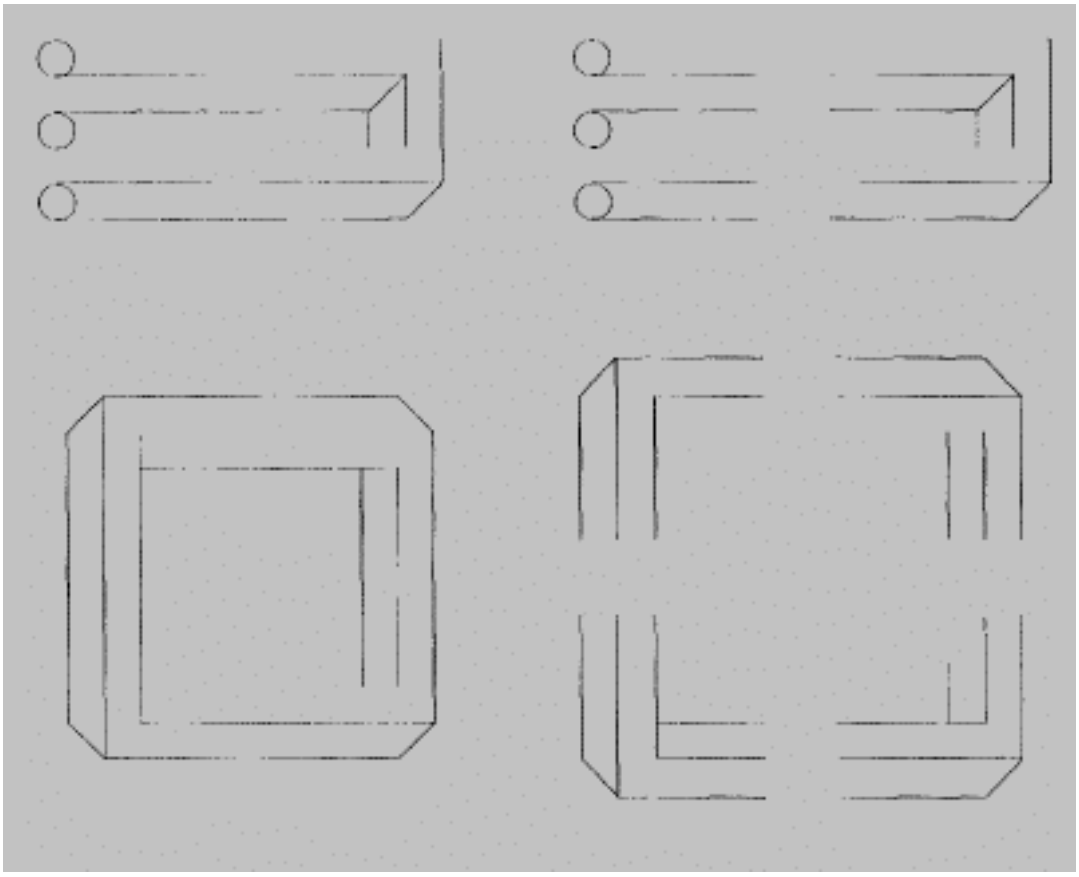
A similar distinction between local constraints and global consistency characterizes human perception of certain optical illusions, such as subjective contours. In the following illustration we perceive two-dimensional white forms partially occluding black disks and not simply black sectors. In fact we perceive the illusory forms as being lighter than their background, as one would expect in cases when one white object is occluding another white object [Gregory, 1970; Gregory, 1973; Kanizsa, 1979; Rock, 1983].



One plausible explanation for the perception of such subjective contours is that, as the black sectors are reminiscent of conditions, we attempt to verify the resulting constraints by correlating the local conditions into a consistent whole: if the

expectations we form at one illusory corner concerning the location and orientation of the other corners of a subjective contour are supported by the other illusory corners in the image, then we perceive the illusory form.

Another type of optical illusions that also appears to relate to constraint propagation is impossible figures composed out of possible parts [Penrose and Penrose, 1958]. Here again expectations formed at one part are verified by relationships of closure and continuity with the other parts but the figures derived by linking the parts on the basis of these relationships are not logically acceptable. One could say that constraint propagation continues in several iterations without satisfactory conclusion, thus resulting in the restlessness experienced when viewing such impossible objects.



Constraint propagation in archltectural design

The above forms of constraint propagation have equivalents in architectural design. Numerical constraint propagation underlies parametric design, where abstract generic specifications of standarized objects such as columns or stairs are refined and adapted to particular conditions or wishes. The *TopDown* program provides a good example of constraint propagation in a tree structure defined by relationships of relative size and position [Mitchell, Liggett et al., 1990]. Another example is the calculation of a staircase where constraint propagation takes place in a network of variables (sizes for the liuman step length and threshold values for treads and risers), as in the following spreadsheet file that calculates straight single flight staircases on the basis of the preferred riser height and the preferred tread length as input by the user.

Calculation of a straight single flight staircase

Input

A. Higher floor level	300
B. Lower floor level	100
C. Human step length (normally 60-65 cm)	60
D. Maximum riser height (normally 20 cm)	20
E. Minimum riser height (normally 10 cm)	10
F- Maximum tread length (normally 40 cm)	40
G. Minimum tread length (normally 25 cm)	25
H. Preferred riser height	17
I. Preferred tread length	35

Output

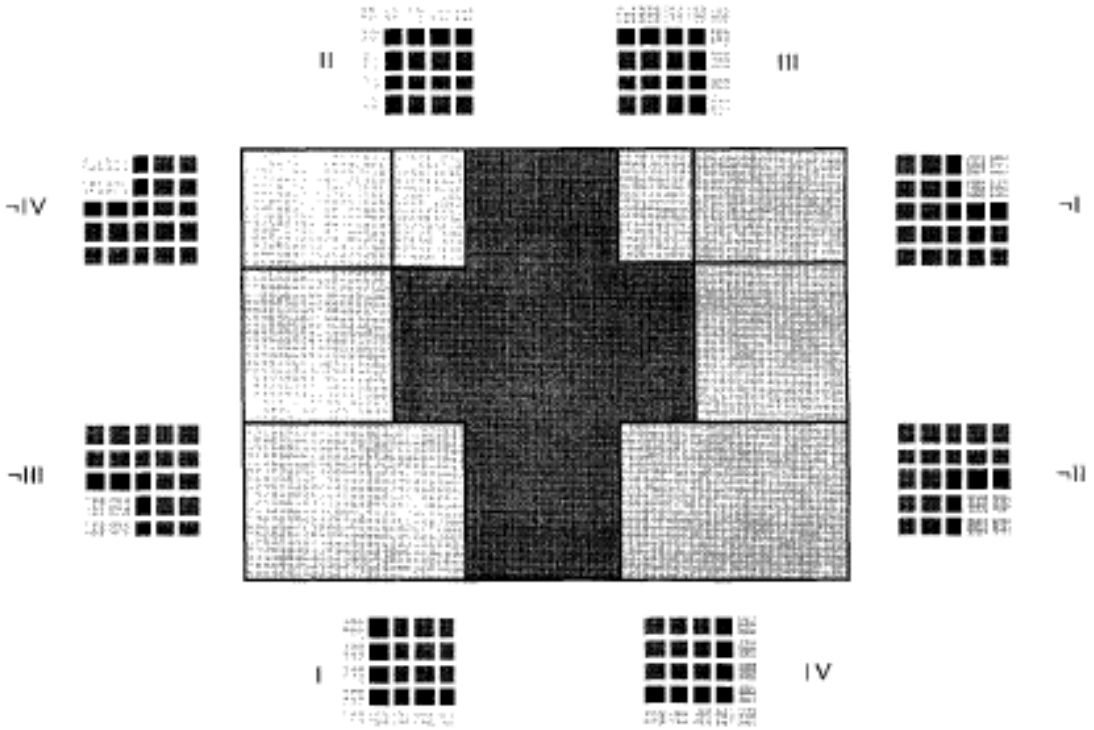
J. Floor to floor height (A-B)	200

On the basis of preferred riser height (H)		*On the basis of preferred tread length (I)*	
K. Riser height (D, E or H --see formula)	17	K. Riser height ([C- L]/2)	12,5
L. Tread length (C-2*K)	26	L. Tread length (F, G or I -see formula)	35
M. Number of risers WK -rounded off)	12	M. Number of risers (J/K -rounded off)	16
N. Precise riser height (J/M)	16,67	N. Precise riser height (J/M)	12,50
0. Number of treads (M-1)	11	0. Number of treads (M-1)	15
P. Staircase length (0*L)	286	P. Staircase length (0*L)	525

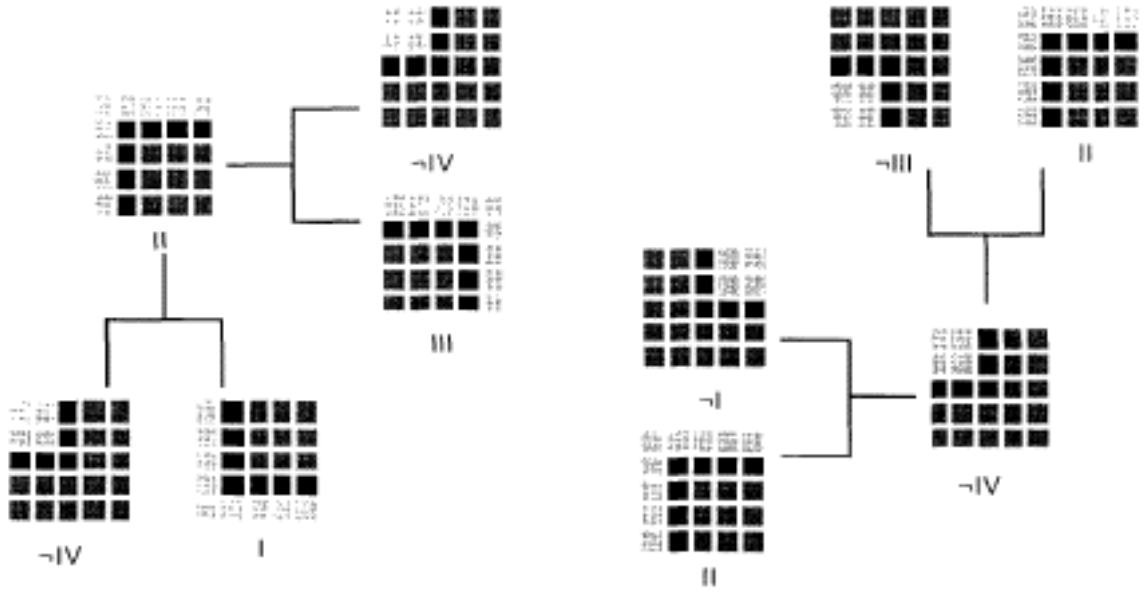
Symbolic constraint propagation forms the basis for the recognition of spaces in floor plans, in a manner similar to the recognition of axonometric projections and to the perception of illusory figures. In floor plans we indicate the shape and position of spaces only implicitly, through the building elements that bound the spaces. Nevertheless, human perceivers can readily and accurately recognize spaces in a floor plan and their interrelationships (i.e., the spatial organization of the building).

For the recognition of spaces we require the constraints of the problem and the relationships between these constraints. These take the form of (i) a typology of space corners and (ii) connectivity relationships between instances of space corner types [Koutamanis and Mitossi, 1992; Koutamanis and Mitossi, 1993].

In a rectangular floor plan we encounter only eight space corner types (indicated in the illustration with respect to the central cross-shaped space).



Connectivity between instances of the above types in rectangular floor plans is quite straightforward: for any given types there are just two possible candidates in the horizontal and two in the vertical direction. The corner type also specifies where exactly (up or down, right or left) one should look for the possible candidates.



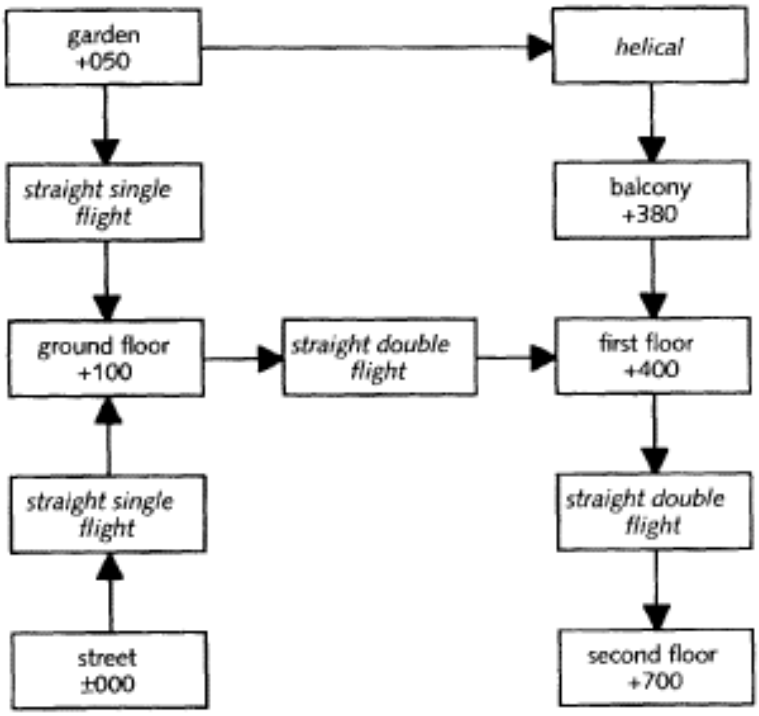
Recognition of spaces is performed by propagating the above constraints in the floor plan image, starting at an arbitrary corner. This corner is matched to the typology of possible corner in order to identify its type. Once the type is known, the computer searches for corner types acceptable to the connectivity constraints in the directions indicated by the connectivity of the type, and from the there found corners for new

connected corners until the initial corner is encountered again. The corners identified thus far are the corners of a single space. Recognition concludes with success when all corners in the floor plan have been recognized.

Design exercises on constraint propagation

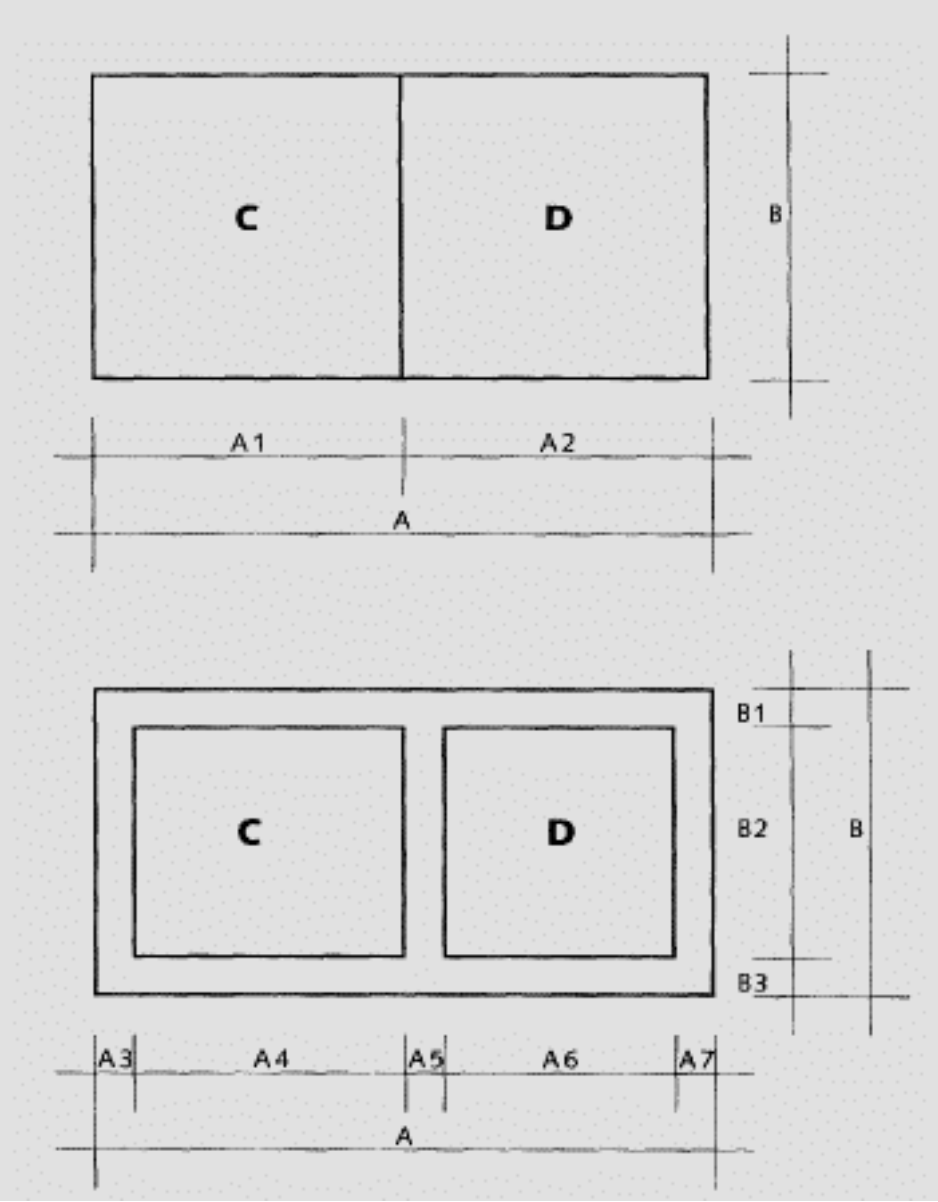
Armed with the above knowledge on the computational structure of electronic spreadsheets and on the use of constraint propagation in architecture students are required to take a number of exercises through which they acquire a working experience of certain computer programs (drawing programs and spreadsheets) and, more significantly, of the ways they can use these in designing. All exercises relate to design practice and present a methodical approach to the use of constraint propagation for design problem solving.

The first exercise aims at deepening the students' understanding of constraint propagation in design. The goal is to develop a network of floors and staircases for a particular building using the spreadsheet file previously mentioned as basis. This spreadsheet file can calculate the essential dimensions of a straight single flight staircase. Students are expected to expand the possibilities of the calculation to other staircase types that exist in the building and then create a new file constraining a model of all floor levels and staircases, as in the following flowchart.



This model is used for the evaluation of the form and size of stairs in the building, for suggesting modifications and for developing alternatives on the basis of these modifications. It is also possible to link the spreadsheet model to a drawing program and automatically adapt a schematic visual representation of the building (parametrization). Detailed two-dimensional ai-id three-dimensional representations can also be used but their complexity may hamper students with a limited knowledge of computer-aided design.

A second exercise that also links drawing programs and spreadsheets is the analysis of floor areas in a design. This analysis is a comparison with either the building programme or another design.



Dimensions

Input		Output	
Dimension A:	600		
Dimension B:	300		
		Area: (total)	180000
Dimension Al:	300		
		Dimension A2:	300
Dimension A3:	30		
		Dimension A4:	265
Dimension A5:	10		
		Dimension A6:	265
		Dimension A7:	30
		Dimension Bl:	30
		Dimension B2:	240
		Dimension B3:	30

Floor areas

Gross			Net		
Space C:	90000	50%	Space C:	63600	35%
Space D:	90000	50%	Space D:	63600	35%
			Partitions:	50400	28%
			Envelope:	2400	l%
Total:	180000	100%	Total:	180000	100%

Parameterization and abstraction play an important role in the analysis, as students are required to organize floor plan drawings in a suitable way using layers to modularize information and links to connect dimensions on the same or different layers before transferring the dimensions of the floor plan and their interrelationships to a spreadsheet model, as in the above illustration. Here again it is possible to make parametric variations and develop alternatives automatically by connecting the spreadsheet to the drawing program.

Conclusions

One complaint that has been frequently voiced in recent years is that, unlike in previous decades, academic education and research in schools of architecture are unable or incapable of leading practice and industry with respect to design computing. In particular, few academic products are produced and even fewer find their way into practice. While this is undeniably true, it should be stressed that the main product of academic research and education is not papers or computer programs but the people who provide and receive education and who conduct research.

Given the structure of the computer and architectural markets today it is naive to suggest that schools of architecture could influence practice by delivering new computer programs. The structure and resources of software and hardware companies are such that the usually abstract or limited academic products do not qualify even as demonstrations, let alone prototypes. Moreover, it can be argued the hardware and software that is currently available at affordable prices offers more that what architectural practice can assimilate in the near future.

Architectural education can assume a leading role with respect to the application of computer-aided design in practice through the production of well-informed generations of new architects who are aware of the possibilities and limitations of computerization in general and of currently available systems. These generations can alter the structure of practice by means of a critical and inventive attitude towards the computationally outdated systems that proliferate practice today, thus creating the necessary specifications to which the computer industry must conform in the future.

The correlation of design and computational techniques is seen as an essential ingredient in the education of these generations. Explicitation of the cognitive structure of designing and active exploration of its possible computational implementations provides the necessary background for understanding and evaluating the applicability of existing computer systems and for specifying future incarnations of their underlying computational principles.

References

Attneave, *F. (1972).* "Some informational aspects of visual perception." *Psychological review* **61**: 183-193.

Clowes, *M. (1971).* "On seeing things." *Artificial Intelligence* **2**: 79-116.

Green, R. T. & M. C. Curtis *(1966).* "Information theory and figure perception: The metaphor that failed." *Acta Psycologica* **25**: 12-36.

Gregory, R. L. *(1970). The intelligent eye.* New York, McGraw-Hill.

Gregory, R. *L. (1973).* The confounded eye. in R. L. Gregory & E. H. Gombrich (ed) *Illusion in nature and art.* London, Duckworth.

Guzmán, *A. (1966). Computer resolution of three-dimensional objects in a visual scene. [report MAC-TR-59].* Ph.D. thesis. MIT, Cambridge, Massachusetts.

Huffman, D. (1971). Impossible objects as nonsense sentences. in B. Meltzer & D. Michie (ed) *Machine Intelligence.* Edinburgh University Press, Edinburgh.

Kanizsa, G. (1979). *Organization in vision. Essays on Gestalt perception.* New York, Praeger.

Koutamanis, A. & V. Mitossi (1992). Automated recognition of architectural drawings. in (ed) *Proceedings of the 11th International Conference on Pattern Recognition. Los* Alamitos, IEEE Computer Society Press.

Koutamanis, A. & V. Mitossi (1993). "Computer vision in architectural design." *Design Studies* **14** (1): 40-57.

Mitchell, W.J., R.S. Liggett & M. Tan (1990). Top-down knowledge-based design. in M. McCullough, W. J. Mitchell & P. Purcell (ed) *The electronic design studio. Architectural knowledge and media in the computer era.* Cambridge, Massachusetts, MIT Press.

Penrose, L.S. & R. Penrose (1958)."Impossible objects: A special type of visual illusion." *British Journal of Psychology* **49**: 31-33.

Rock, I. (1983). *The logic of perception.* Cambridge, Massachusetts, MIT Press.

Walters, D. (1986). Selection and use of image features for segmentation of boundary images. in (ed) *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* Washington, D.C., IEEE Computer Society Press.

Waltz, D. (1975). Understanding line drawings of scenes with shadows. in P. H. Winston (ed) *The psychology of computer vision.* New York, McGraw-Hill.

Winston, P. H. (1984). *Artificial Intelligence.* Reading, Massachusetts, Addison-Wesley.