A STUDIO EXERCISE IN RULE BASED COMPOSITION

SOTIRIOS KOTSOPOULOS AND HALDANE LIEW
Massachusetts Institute of Technology, USA,
skots@mit.edu, haldane@mit.edu

Abstract. This is the outline of a studio exercise that incorporates rule-based methods in design synthesis using analogue and digital means.

1. Introduction

One of the challenges in using a rule-based system is determining its appropriateness and applicability in the synthesis of form. How can rules be used to perform goal-driven design tasks? And how can analogue and digital rule-based systems coexist as part of the studio teaching activity? In order to examine these questions a studio exercise was developed on the basis of a design competition for low cost housing. The exercise aims to become a starting point for the introduction of rule-based methods, in synthesis.

The general strategy, given the building program, is to construct a method of producing a variety of 2-dimentional plan arrangements in response to a variety of functional demands and conditions. The objective is to gradually establish spatial elements, relationships, and rules for the generation of designs.

First, possible sets of spatial elements and rules are formulated with analogue means, as a hypothesis. Then, they are tested using a digital parametric shape grammar interpreter. The interpreter requires the conversion of the rules into a scripting format and provides computer aid in clarifying the ramifications of the hypothesis. Using the interpreter the designer determines if a rule-set produces any desired outcomes. If not, the rule-set is modified and re-tested. The decision process involves a selection among alternative rule-sets, where the designer explores the possible outcomes. The digital interpreter offers fast broad exploration of the products of the rules. The digital 2-d representations generated by the rules could readily be used in solid modeling allowing alternative representations to participate in the evaluation. The process is organized in three general, interdependent levels of abstraction. The first is dedicated to the *formation*

of *partis*, the second to the *transformation* of a *parti*, and the production of variations, and the third to the description of the *tectonic* details.

2. The Exercise

The exercise was based on a housing competition sponsored by the *Habitat For Humanity* (HFH) the summer of 2002 in Boston, Massachusetts. The HFH described the goal of the competition as: "*the building of simple, decent, affordable houses*". The program called for the design of adaptable types of 2, 3 and 4-bedroom houses without determining the square-footage of rooms or house types. All houses included: primary covered entrance, circulation, dining area, living area, at least one full bathroom, kitchen, and bedrooms.

Figure 1. Examples of HFH housing in East Boston, Dorchester, and Roxbury

A minimum living space limit for all house types was suggested: 900 s. f. for 2-bedroom apartments, 1050 s. f. for 3-bedroom apartments, and 1150 s. f. for 4-bedroom apartments. Further, the organizers did not designate specific sites, but offered several possible ones. Small, quadrilateral lots less than 5000 s. f. were an option, but lots larger than 20000 s. f. with complex shapes were also typical.

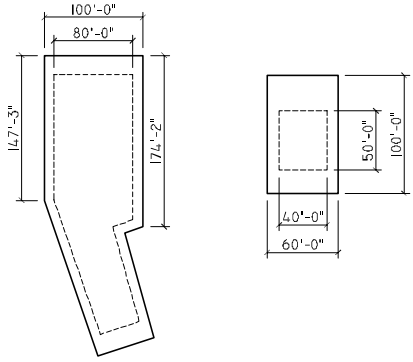


Figure 2. Two examples of typical HFH sites

3. Design Concept and Method

The design approach is influenced by three factors: a) the absence of a designated site, b) the building program, and c) the provision for low construction-cost. The design concept is to develop rule-systems able to generate flexible house arrangements of variable size and morphology. The systematization of the ground plan is the method for the attainment of this objective. Analogue and digital means are both used in the design process.

The computational framework defined in Stiny 1980 within which shapes that belong in some algebra U_{ij} are composed with rules of the form $x \rightarrow y$, is employed in the production of design descriptions. Each design description is treated as a product of a finite device (*grammar*) that includes a finite number of rules and a finite vocabulary of spatial elements.

The proposed rule-based process generates descriptions in a "top-down" fashion. Similar models, referring to the construction of rule-based systems for 0-dimentional languages can be found in Carnap 1912, and Chomsky 1957. The possibility of establishing analogous methods in the analysis and synthesis of 2-d design descriptions was first discussed in Stiny and Mitchel 1978 in the production of Palladian villa plans. The grammar of Stiny and Mitchell captures the generation uniaxial Palladian villa plans in eight stages. Numerous papers have followed describing the generation of Frank Loyd Write's prairie houses (Koning and Eizenberg 1981), Japanese tea-room designs (Knight 1981), Queen Ann houses (Flemming 1987), traditional Taiwanese houses (Chiou and Krishnamurti 1995), Yingzao fashi houses (Li 2000), and Alvaro Siza's houses (Duarte 2001).

The novelty of the proposed approach is that it does not focus on an existent corpus of designs, but attempts to capture the exploratory effort of an intuitive creative process. A process of this kind involves selection among several candidate rule-sets, where the designer explores their possible outcomes using analogue and digital means. The proposed view is that first, the candidate sets of spatial elements and rules are formulated with analogue means, and then, they are tested digitally.

The heuristics of the process are organized in three general levels of abstraction. Each level contains finite sets of rules, the interaction of which is characterized by interdependence. At the top more abstract level of *formation*, the rules produce *partis* for possible designs. At the middle level of *transformation*, a specific *parti* is selected and transformed to a more detailed spatial arrangement. At the lower level of abstraction, that of *refinement*, the rules apply on the transformed arrangement to determine its *tectonic* details such as doors windows etc. All three levels of the process make use of analogue and digital means. The digital aid applies more drastically on the first two more abstract levels of *formation* and *transformation*.

The proposed framework can be described as follows:

Σ : { finite set of spatial elements }

R: { *Formation* $A_1 \rightarrow F_1$: $A_n \rightarrow F_n$ $\Leftrightarrow$ *Transformation* $G_1 \rightarrow M_1$: $G_k \rightarrow M_k$ $\Leftrightarrow$ *Refinement* $N_1 \rightarrow W_1$: $N_r \rightarrow W_r$ }

where $A_1,\ldots, A_n, F_1,\ldots, F_n$ are elements in Σ.

The analogue part of the exploration involves the articulation of candidate rules, while the digital part the rule-testing. Through an iterative process of *formation*, *transformation* and *refinement* similar to the *see-move-see* concept (Schon and Higgins 1992), the rules are evaluated and redefined according to their compliance to programmatic, intuitive, and construction criteria. Finally, the rules are organized in *grammars*.

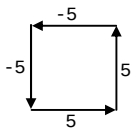
A shape grammar digital interpreter (Liew 2003) is used for the digital part of the exercise. The interpreter, written in VisualLISP, uses a scripting language based on LISP to describe a rule. Each rule has four parts: left-hand schema, right-hand schema, transformation mapping, and variable mapping. A vector description format (Nagakura 1995) is used to describe the geometry and variables of a schema.

The transformation mapping determines any transformation changes between the left-hand schema and the right-hand schema. The variable mappings define a relationship between the parameters of both schemata. A schema is composed of two parts, the geometry and the constraints on the geometry variables. The geometry of a schema is described using a series of vector displacements. Each vector has 3 components: action, vector and label. The action component determines if the shape is a line or a point. The vector component describes the x and y displacement of the shape. The label component determines the name. For example, a horizontal *parti* line that is 5 units long is described as:

```
((action “line”) (vector 5 0) (label “parti”))
```

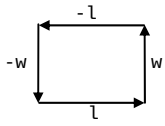
A shape is described as a series of vector displacements that are connected from end to end. For example, the following describes a “*parti*” square that is 5 units by 5 units in size.

```
(((action “line”) (vector 5 0) (label “parti”))
 ((action “line”) (vector 0 5) (label “parti”))
 ((action “line”) (vector -5 0) (label “parti”))
 ((action “line”) (vector 0 -5) (label “parti”)))
```



To describe a parametric shape, the numbers in the vector displacement description are substituted with variables. The following example describes a schema that finds all *parti* rectangles.

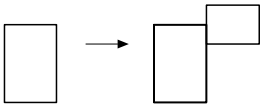
```
(((action “line”) (vector l 0) (label “parti”))
 ((action “line”) (vector 0 w) (label “parti”))
 ((action “line”) (vector (- l) 0) (label “parti”))
 ((action “line”) (vector 0 (- w)) (label “parti”)))
```



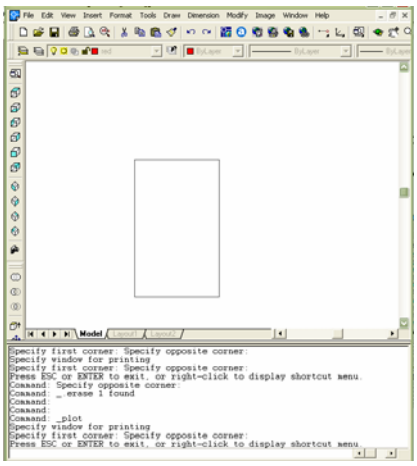
Restrictions can be set on the geometry variables to limit the type of sub-shapes found. Theses restrictions are added in the binding-constraints component of the schema. The following example restricts the size of the square to be less than 10 units.

```
((binding-constraints
 (l (< l 10))
 (w (< w 10)))
```

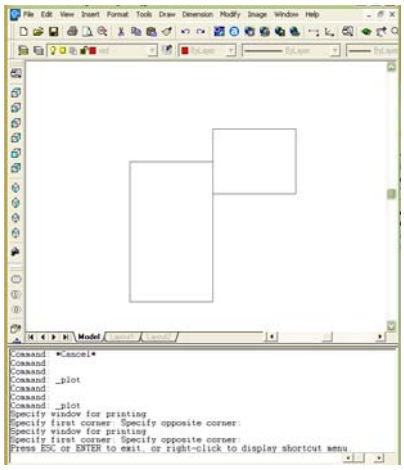
To apply a rule of the form x → x + y, on rectangles



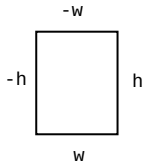
the program recursively searches the input shape for all instances of the left-hand schema and presents the possibilities through an interactive menu that highlights the embedded schemata. Once the user selects an embedded schema the rule application is completed by subtracting the selected schema from the input shape and adding the right-hand schema of the rule.



⇒

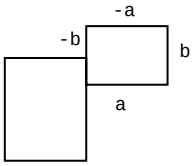


The previous additive rule of the form x → x + y, applying on *parti* rectangles x and y, can be expressed in the symbolic meta-language as follows (for simplicity, all the arrows are omitted from the shapes). First, the description of the left side of the rule,




```
(setq schema-left-rule
   '((geometry
    ((action "line") (vector w 0) (label "parti"))
    ((action "line") (vector 0 h) (label "parti"))
    ((action "line") (vector (- w) 0) (label "parti"))
    ((action "line") (vector 0 (- h)) (label "parti"))
  )
  (parameter-constraints
   (w (> w 0))
   (h (> h w))
  ))
)
```

Second, the description of the right side of the rule,



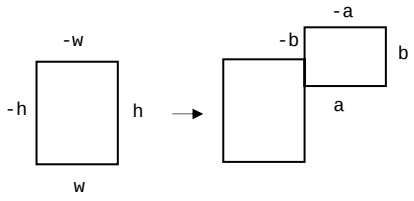

```
 (setq schema-right-rule
       '((geometry
     ((action "line") (vector w 0) (label "parti"))
     ((action "line") (vector 0 h) (label "parti"))
     ((action "line") (vector (- w) 0) (label "parti"))
     ((action "line") (vector 0 (- h)) (label "parti"))
     ((action "move") (vector w (- h (* 0.375 w))))
     ((action "line") (vector a 0) (label "parti"))
     ((action "line") (vector 0 b) (label "parti"))
     ((action "line") (vector (- a) 0) (label "parti"))
     ((action "line") (vector 0 (- b)) (label "parti"))
    )
```

```
   (parameter-constraints
    (w (> w 0))
    (h (> h w))
    (a (> a 0))
    (b (> b 0)))
  )
 )
```

The right side of the rule, also includes the description of the transformation and parameter mapping,

```
(setq tmap-rule
   '((delta-xo . 0)
     (delta-yo . 0)
     (delta-ro . 0)
     (delta-za . 0))
)
(setq pmap-rule
  '((w w)
    (h h)
    (a w)
    (b (* 0.75 w)))
)
```

And third, the connection between the left and the right sides of the rule,




```
 (setq housing-rule
  '((left . schema-left-rule)
    (right . schema-right-rule)
    (tmap . tmap-rule)
    (pmap . pmap-rule)
    (success . nil)
    (failure . nil)
    (applymode . "single")
    (rulename . "housing-rule"))
  )
```

3. Results

How can rules be used to perform goal-driven design tasks in composition? The aim of rules in composition is to project a finite set of properties to a large set of compositions. But in order to identify the appropriate rules in synthesis, one needs to examine what they produce. Then, the rules can be organized to generate compositions with the desired properties. A grammar serves as a memory device: the rules are recorded and gradually classified with respect the attainment of the objective at view. The grammar and the designs are the output of a broad discovery process, where spatial entities and rules are first distinguished and stated as a hypothesis, and then tested. They are transformed, and refined to achieve the purpose at view.

On what basis can we evaluate the products of rules? This is the practical problem regarding the use of rules in both design synthesis and the activities of the studio. In the case of grammars for speaking languages, a test of adequacy (Chomsky 1957) is to have the native speakers accept the produced sentences and to identify the false ones that are generated by this grammar. Chomsky assumes intuitive knowledge of the grammatical sentences of English, and asks "*what sort of grammar is able to produce these sentences in some effective and illuminating way?*"

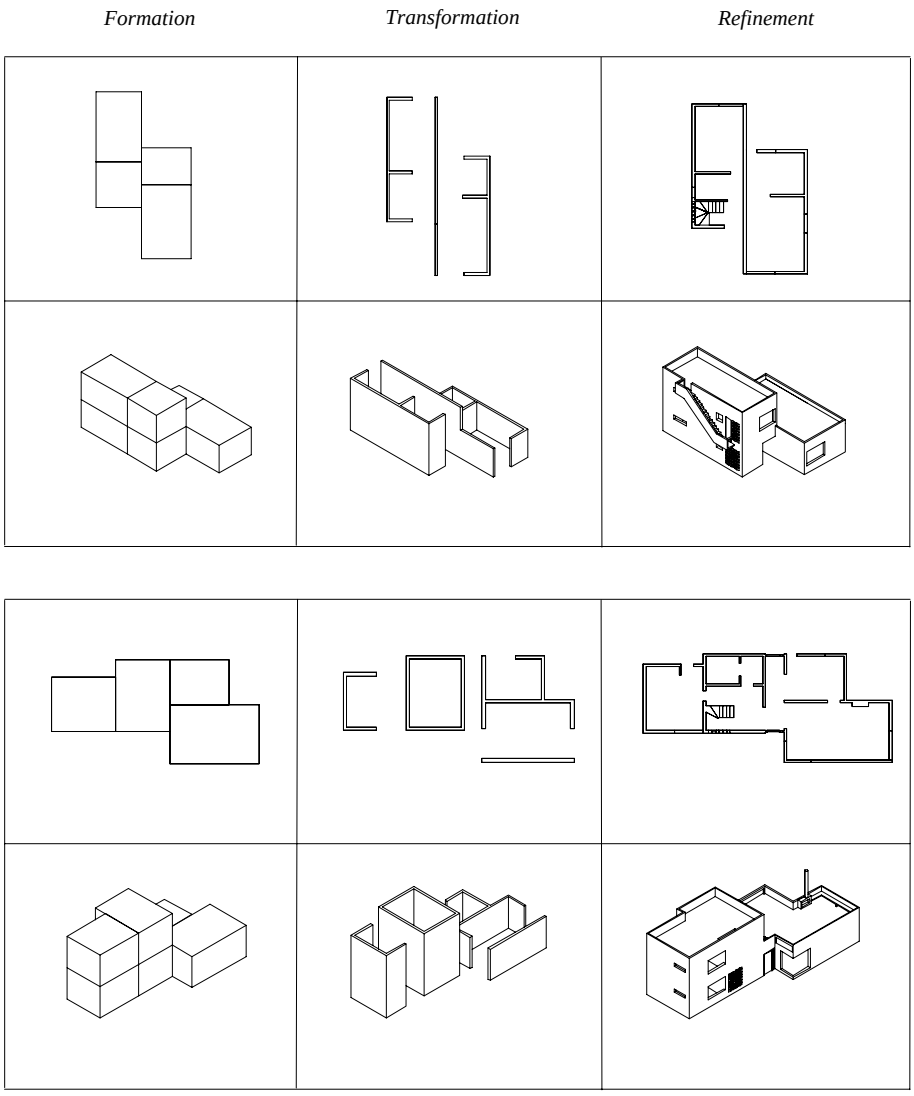
In a similar fashion, in the analysis of a corpus of designs, the required initial properties and the rules that produce it, can be extracted from some original, previously analysed, instance (i.e. Palladian Villas, or Queen Ann houses, or Frank Loyd Write houses etc.) But, there is no predetermined criterion of evaluation in the synthesis of original compositions. The designer has to set the objective and the test at each step. The objective and the test remain open for re-evaluation as the testing proceeds.

In this study the choice of rules and designs is approached on the basis of programmatic-functional distinctions. Spatial relationships and rules that form different room-adjacencies are distinguished, and the values that determine the room sizes are gradually established. Large numbers of arrangements can be produced in this way. The need to develop more focused methods to control the generation of designs leads to the restriction of the rules.

How does analogue and digital rule-based systems coexist as part of the studio teaching activity? The spatial elements and rules are initially formulated with analogue means, because the ambiguity of analogue representation works in favour of the exploratory process. The absence of discreetness prohibits the early designation of values. The description remains ambiguous, and yet useful for the time that questions of value remain uncertain.

The digital representation is more efficient in clarifying the ramifications of a candidate rule-set, by allowing the mechanical execution of large number of tests. In this way, the digital exploration helps to determine if a

particular rule-set produces any desired results. If not, the rules can be modified and re-tested. The digital descriptions of rules require translation of the depicted shapes in symbolic form. Therefore, the values of variables within the rules need to be determined before the rules can take their digital form. Sample descriptions from two working examples, at the three general levels of abstraction (*formation*, *transformation* and *refinement*) are exhibited in the next *Figures 1a* and *1b*,



Figures 1a, 1b. Two working examples (up: *Fig.1a*, down: *Fig.1b*) in 2-d and 3-d

4. Conclusions

In the proposed studio exercise the design descriptions are treated as products of a finite rule based device and generated in "top-down" fashion by a process that involves analogue and digital media. At the top level (*formation*) the rules produce *partis*. At the middle level (*transformation*) a chosen *parti* is transformed to a design. At the lower level (*refinement*) the rules determine the tectonics. Rule-sets are formed with analogue means and are tested digitally to determine their outcomes.

A shape grammar digital interpreter is used for the purpose. The digital generative tool is particularly useful in the exploration of 2-d *partis*, at the stage of *formation*. Further, the *transformation* and *refinement* stages require the juxtaposition of several descriptive layers. For the juxtaposition of 2-d descriptions the digital tool used multiple Auto-CAD layers, and the symbolic expression of rules became increasingly complex. All the necessary 3-d descriptions were executed manually in Auto-CAD without using the interpreter. Computations in 3-d that also require the juxtaposition of multiple descriptive layers will be the subject of future work.

References

Carnap R: 1937 (2002), *The Logical Syntax of Language*, Open Court, pp. 1-52

Chomsky N: 1957 (1976), *Syntactic structures*, Mouton, The Hague, Paris, pp. 34-48

Chiou S and Krishnamurti R: 1995, 'The grammar of the Taiwanese traditional vernacular dwellings', *Environment and Planning B: Planning and Design* 22 689-720

Duarte J P: 2001, *Customizing mass-housing: A discursive grammar for Siza's Malagueira houses*, Ph.D. Dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Flemming U: 1967, 'More than the sum of parts: the grammar of Queen Anne houses' *Environment and Planning B: Planning and Design* 14 323-350

Knight K: 1981, 'The Forty-one Steps: the languages of Japanese tea-room designs', *Environment and Planning B: Planning and Design* 8 97-114

Koning H and Eizenberg J: 1981, 'The language of the prairie: Frank Loyd Wright's prairie houses' *Environment and Planning B: Planning and Design* 8 295-323

Li A: 2000, *A teaching grammar of the Yingzao fashi*, Ph.D. Dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Liew H: 2003, *SGML: A Meta-Language for Shape Grammars* PhD Dissertation, Department of Architecture, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Mitchell J W: 1990, *The Logic of Architecture*, MIT Press, see diagrams 10.15, 10.16, p. 233

Nagakura T: 1995, *Form-processing: A system for architectural design*, Ph.D. Dissertation, Harvard University, Massachusetts

Schon D and Higgins A: 1992, 'Kinds of seeing and their functions in designing', *Design Study*, volume 13, n 2, pp. 135-156

Stiny G: 'Two exercises in formal composition' *Environment and Planning B*, 1976, vol. 3, pp. 187-210

Stiny G: 1980, 'Introduction to shape and shape grammars', *Environment and Planning B*, volume 7, pp. 343-351

Stiny G: 1991, 'The algebras of design', *Research in Engineering Design*, 2, pp. 171-181
Stiny G and Mitchell W J: 'The Palladian grammar', *Environment and Planning B: Planning and Design*, volume 5, pp. 5-18

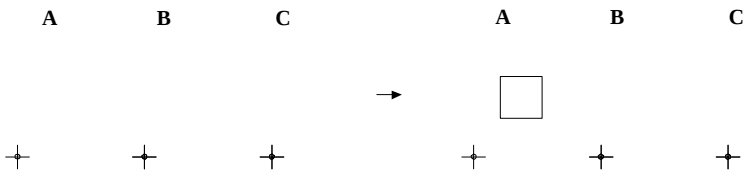
Appendix

The following example shows the derivation of the *formation* and *transformation* samples of *Figure 1a* (p. 9).

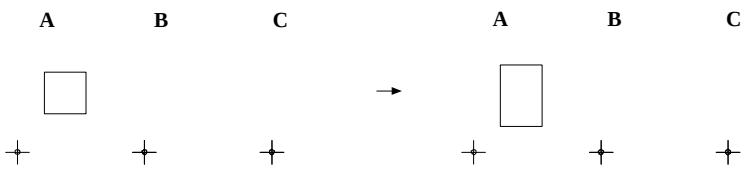
The *formation* includes the derivation of the 2-d *parti*, and the *transformation* the introduction of wall-layout, and openings. The example is part of the digital testing, where coded versions of the shape rules are used in the digital interpreter.

The example involves descriptions made out of lines that belong to three different Auto-CAD layers: Layer A includes only *parti* lines. Layer B includes only the wall-layout. Layer C includes only secondary shapes, of auxiliary character. The setting can be modeled in $<U_{12} \times U_{12} \times U_{12}>$ algebra. The three different layers are indicated by the (0, 0) cross-point of their coordinate system, and by the letters A, B and C, respectively.

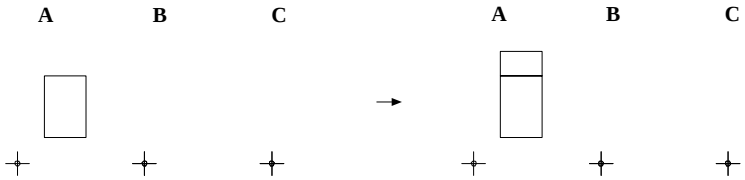
The shape rule R1 generates a *parti* square in layer A. Layers B and C remain unchanged.



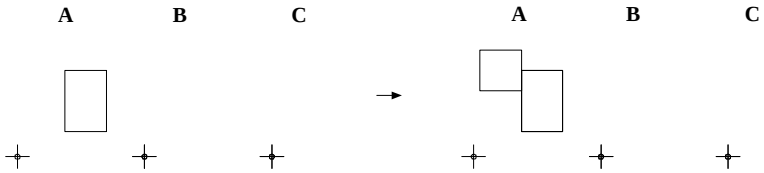
The rule R2 creates a *parti* rectangle from a given *parti* square in layer A, according to a specific proportion. Layers B and C remain unchanged.



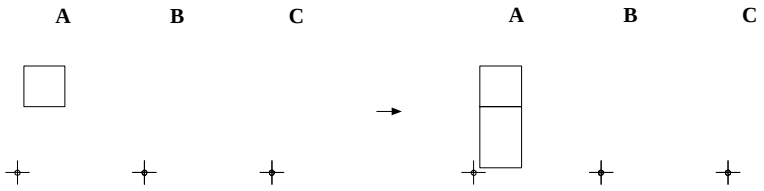
The rule R3 also acts in layer A. The rule is of the form $x \rightarrow x + y$. It adds a *parti* rectangle to an existing *parti* rectangle. The length of the added rectangle is equal to the width of the existing rectangle. The Layers B and C remain unchanged.



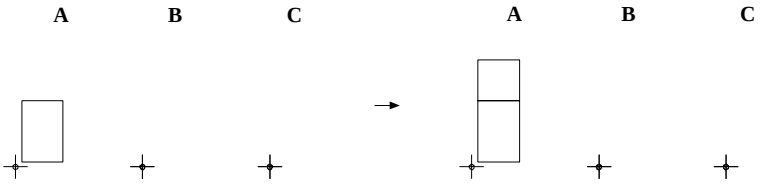
The rule R4 also adds a *parti* square to an existing *parti* rectangle in layer A. Again, the side of the added square is equal to the width of the existing rectangle. The layers B and C remain unchanged.



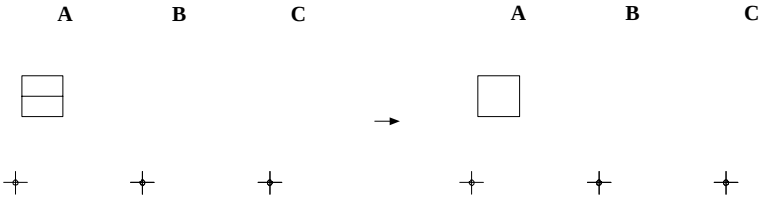
The rule R5 adds a *parti* rectangle to an existing *parti* square, in layer A. The layers B and C remain unchanged.



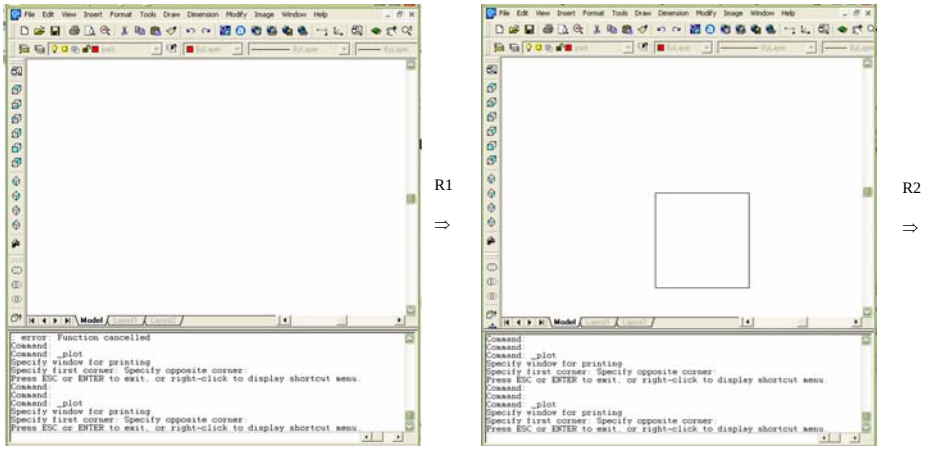
The rule R6 adds a *parti* square to an existing *parti* rectangle, in layer A. The side of the added square is equal to the width of the existing rectangle. The layers B and C remain unchanged.

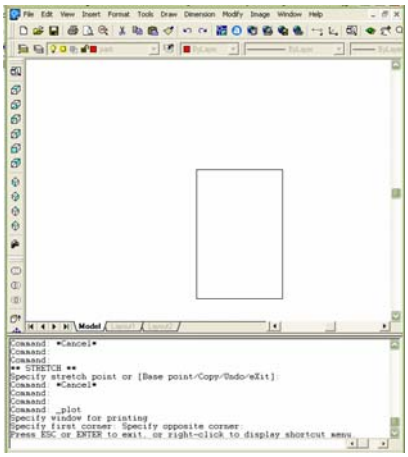


The rule R7 erases a *parti* line that lies inside an existing *parti* rectangle, in layer A. The layers B and C remain unchanged.

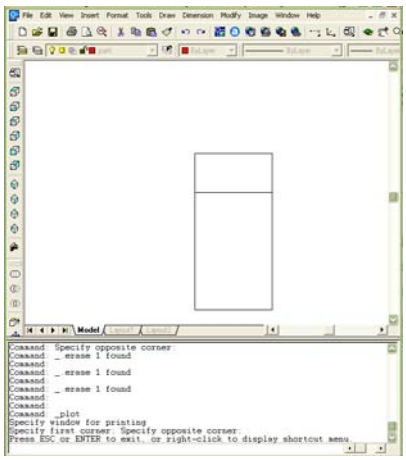


The *formation* of the *parti* is shown in the derivation,

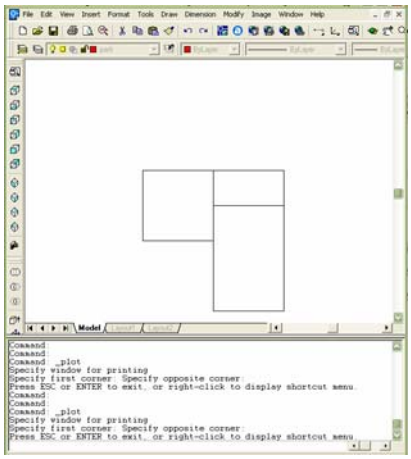




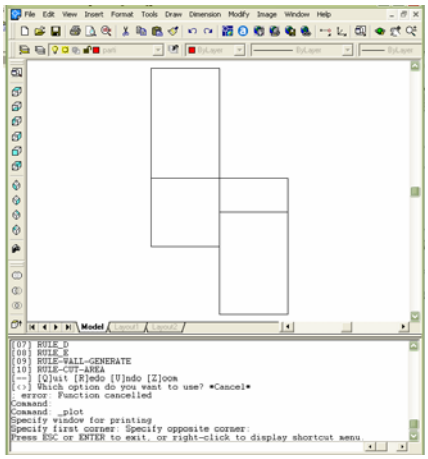
R3 ⇒



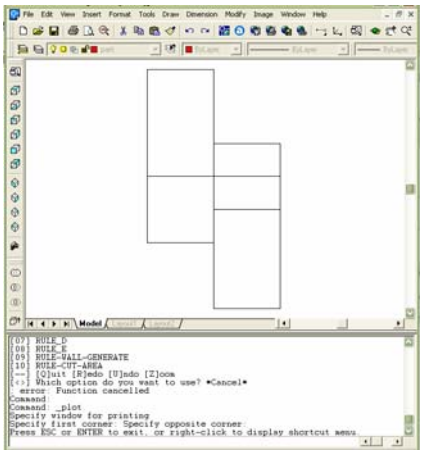
R4 ⇒



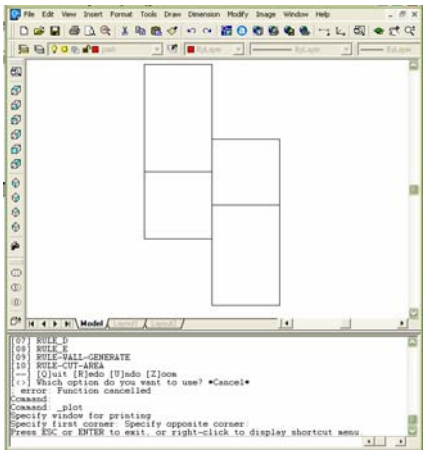
R5 ⇒



R6 ⇒

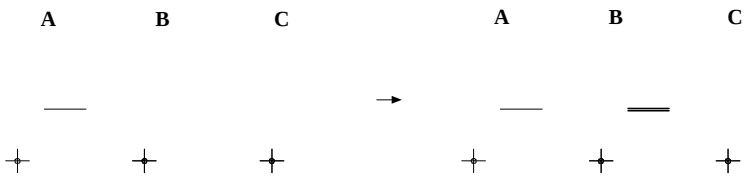


R7 ⇒

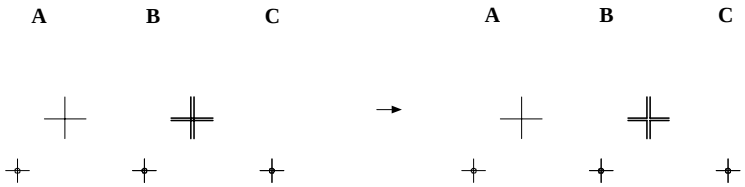


An example of a possible *transformation* is exhibited next. The example shows the creation of a wall-layout. The presented rules follow the conventions of the digital interpreter. The following four shape rules are coded as one action, in the digital interpreter. That is, the interpreter executes all four rules each time the user picks the creation of a wall layout.

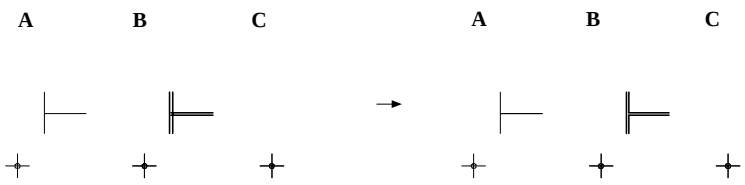
Rule R8 draws the wall-layout in layer B, in correspondence to a parti line in layer A. The existence of a *parti* in layer A is therefore necessary. The parti in layer A remains intact, and layer C remains unchanged



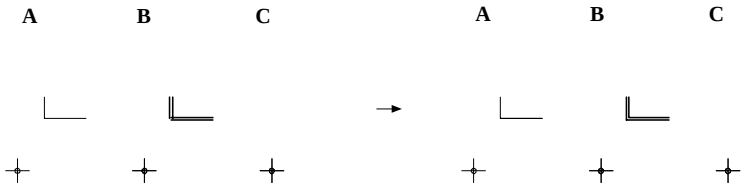
The rule R9 executes the trimming of cross + intersections



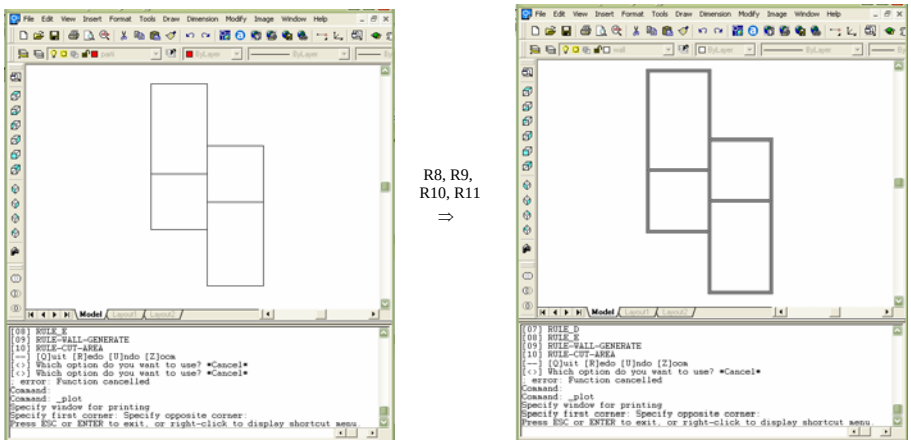
The rule R10 cleans all T intersections,



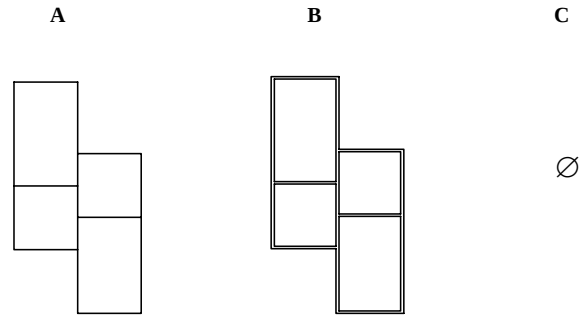
Finally, the rule R11 executes the trimming in L intersections,



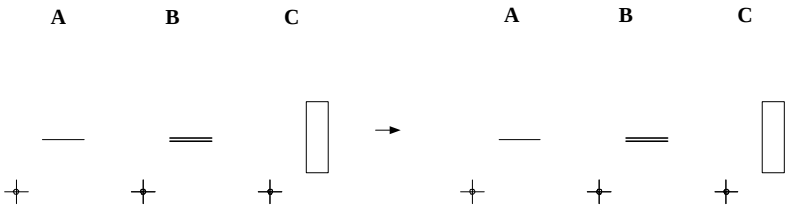
All four rules (R8, R9, R10, R11) are coded so that they are executed at once, one after the other. Therefore, in the derivation, the digital interpreter executes the transformation in one step



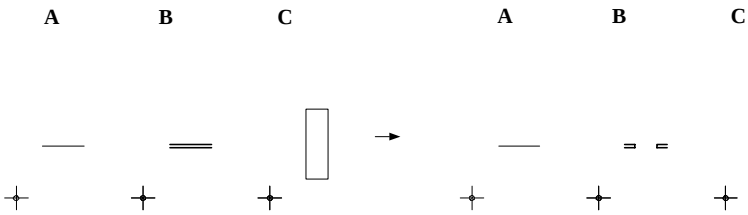
At the end of the derivation, the layer A includes the *parti*, layer B includes the wall layout, and layer C the empty shape,



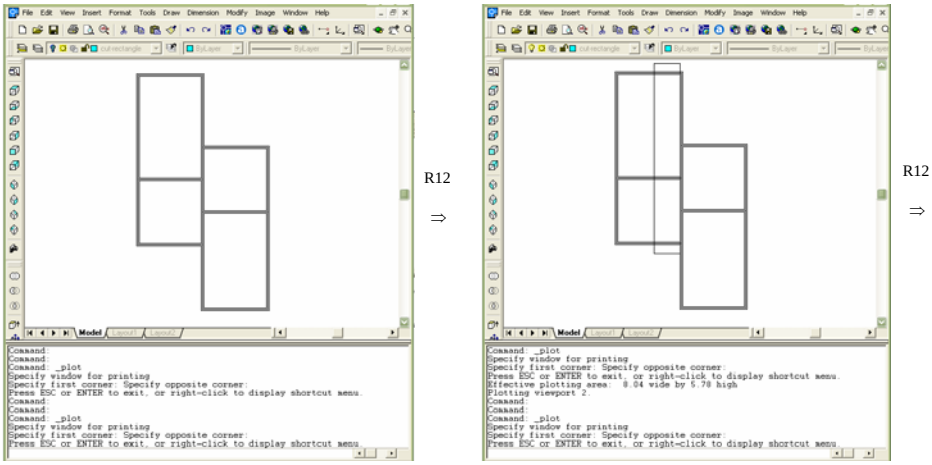
Finally, in the next transformation two openings are created on the wall-layout, in layer B. For this purpose two rectangles are drawn manually in the third layer C of secondary lines. The rectangles indicate the positions of the openings. The addition of rectangles, in layer C, is expressed by the following shape rule R12,

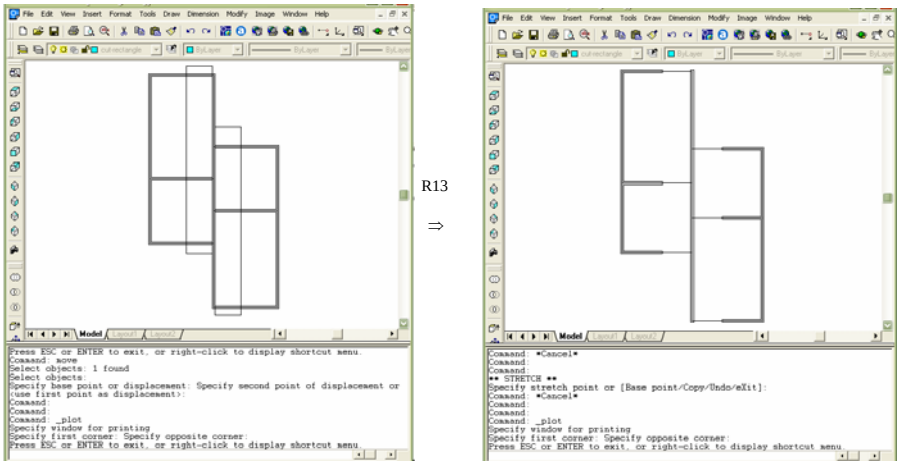


The secondary rectangles are erased from layer C, during the application of the next rule R13, which creates openings in the wall layout (at layer B), and leaves the *parti* intact



The derivation,





At the end of the derivation, the layer A includes the *parti*, layer B includes the transformed wall layout, and layer C the empty shape. The shapes in layers A and B are the *formation* and *transformation* arrangements of *Figure 1a* (p. 9).