408

Space Syntax with Prolog

PEDRO AFONSO FERNANDES

Universidade Católica Portuguesa, Católica Lisbon School of Business & Economics, Católica Lisbon Research Unit in Business & Economics (CUBE), Portugal

ABSTRACT

This paper introduces a new way to perform space syntax analyses using Prolog, a Logic Programming language concerned with Artificial Intelligence. Developed in the 1970's to process natural languages, Prolog can deal easily with simple declarations of facts like the connection (or permeability) between convex spaces or axial lines. Readily available on-line through the SWISH platform, in a fancy format inspired by Jupyter Notebooks, Prolog may help to understand the recursive nature of urban processes, given some elementary generators, or to describe or even check the structure (e.g. concentric) of some village. Mostly important, Prolog can compute space syntax measures such connectivity, control or integration in a comprehensive, transparent and attractive way, namely, for students and researchers on space syntax. The experience suggests that Prolog may be appropriate for gamma-analysis of small buildings like the Ashanti's shrine, acting as flexible and easily replicable calculator of syntactic measures. This flexibility is reinforced by the free and open-source nature of the code stored in the SWISH platform, as well as by the declarative nature of Logic Programming, which facilitates the description of the patterns of discrete systems as *social knowables*. In fact, a Prolog program represents a certain amount of knowledge, which is used to answer queries about the social and economic consequences of some spatial design.

KEYWORDS

Space syntax, Gamma-analysis, Concentric structures, Logic Programming, Prolog.

1 INTRODUCTION

Space syntax is a set of techniques for analysing urban settlements and buildings, as well of theories linking space and society founded on Architecture, Engineering, Mathematics, Sociology, Anthropology, Ethnography, Linguistic, Psychology, Biology and Computer Science. It was developed originally by Bill Hillier, Julienne Hanson and colleagues at the Bartlett School

of Architecture and Planning, University College of London (UCL), since the 1970s. Their innovative approach was condensed in three landmark books: *The Social Logic of Space* (Hillier and Hanson 1984), *Decoding Homes and Houses* (Hanson 1998) and *Space is the machine: a configurational theory of architecture* (Hillier 2007).

An important contribution of space syntax is the study of urban form using *discrete systems*, that is, computer-aided recursive techniques based on *elementary generators* such as a pair of open and closed cells. Another key contribution is the concept of *depth* in urban systems and its relation with social phenomena like pedestrian movements or crime incidence, with special focus on different uses of open spaces by inhabitants and strangers, men and women, old and young, adults and children, and so on (Hillier 2007, pp. 146–52). Anyway, space syntax had influenced and contributed to several projects, namely, the redesign of Trafalgar Square or the location of the Millennium Bridge, both in London.

Space syntax is mainly concerned with the study of the relations between *convex spaces*. Convexity exists when straight lines can be drawn from any point in a space to any other point in it without going outside of the space itself (Hillier and Hanson 1984, pp. 97–98). In fact, convex rather that concave spaces stimulate social interaction in the sense that everyone sees everybody within that kind of space. Thus, the starting point of a syntactic analysis is typically a map of the fattest convex and open (or permeable) spaces that cover the settlement (or building) in question. Eventually, this framework can be represented by an axial map with the smallest set of straight lines that pass through all convex spaces or by a map of segments. Then, several syntactic measures can be computed to explore the connectivity between spaces and lines or the angular deviation between segments in order to evaluate the degree of either asymmetry (or integration) or distributeness of each convex space/line/segment or of the whole system (Heitor and Pinelo Silva 2015, pp. 154–64).

Space syntax analyses are typically performed with Depthmap software (Turner 2004) (Heitor and Pinelo Silva 2015, pp. 167–69) having as input a convex/ axial/ segment map with the spatial configuration of the complex in question, previously designed with Computer-Aided Design (CAD) or Geographic Information System (GIS). Depthmap does not import *adjacency lists* of nodes that can be meaningfully represented as strings, that is, in natural language. As we will see, there are very few packages that can do it, namely, NetworkX and Gephy.

Here, we propose an alternative approach to perform syntactic analyses that do not require a previous drawing, shape file, graph or adjacency matrix, but only the declaration in a computer program, written in Prolog, of the connections between the convex spaces (or axial lines) of a settlement or building, using **natural language**.

This innovative approach aims to capture the essence of space syntax. In fact, Prolog is a *Logic Programming* language, developed by Alain Colmerauer, Philippe Roussel, Robert Kowalski and colleagues between Montreal, Marseilles and Edinburgh in the 1970s (Colmerauer and Roussel 1993) exactly to process natural languages, that uses logic to represent knowledge and deduction to solve problems by deriving logical consequences (Kowalski 1988, p. 38). A corollary of using logic is that such knowledge can be understood declaratively. This is the kind of reasoning

embodied in space syntax, in the sense that space syntax represents the spatial arrangements as *fields of knowables*, that is, as systems of possibilities governed by simple and abstract underlying concepts (Hillier and Hanson 1984, p. 66).

Besides, clump, concentric, estate and other syntactic processes defined in the book *The Social Logic of Space* are recursions of simple relations between open and closed primary cells (Hillier and Hanson 1984, 78). Prolog is recursive by construction too and uses relational clausal logic in a straightforward way to declare binary concepts like *containment* or *adjacency*, used to define the elementary generators of space syntax processes. Thus, the social logic of space can, and ought to be, represented by a logic and declarative programming language like Prolog.

This article is organised in the following fashion:

In chapter 2, we briefly describe the theoretical framework of space syntax with a special focus on the concepts of *configuration*, *depth* and *elementary generators*.

In chapter 3, we provide a brief survey of the software commonly used to perform space syntax and network analyses, in order to show that there are very few packages that can treat descriptions of graphs in natural language like Prolog.

In chapter 4, we use Prolog to generate hypothetic recursive settlements from some elementary generators, following the ideography proposed by Hillier and Hanson (1984, pp. 66–81).

Then, we specify the syntax of concentric spaces with Prolog *grammars* developed to parse natural languages, with the application to the villages of Bororo and Omarakana (chapter 5).

In chapter 6, we show how syntactic measures such *integration*, *control* or *ringiness* can be computed with the same programming language in an elegant and straightforward way, having as case study the well-known Ashanti's *abosomfie*, or shrine, also described by Hillier and Hanson (1984, p. 181).

Finally, some conclusions and future developments are pointed out in chapter 7.

The material in this article is complemented by a series of **open-source computer programs** whose paths are indicated in the following footnotes, stored on SWISH (https://swish.swi-prolog.org/), the SWI-Prolog (Wielemaker et al. 2012) web server. For convenience, these (and other) links were centralized in the **Space Syntax with Prolog site**, https://www.sswprolog.net, that provides several resources concerned with programming the logic of space (programs, articles, posters, presentations and a list of references).

2 THEORETICAL FRAMEWORK

Based on graph theory and computer-aided simulations, *space syntax* aims to find and explain the relation between spatial configurations and social activities. *Configuration* is a concept addressed to the whole of a complex (settlement or building) rather than to its parts that captures how the relations between two spaces, say A and B, might be affected by a third space C (Hillier 2007, pp. 23–24). For instance, if A and B are adjacent or permeable, then they have a symmetric configuration in the sense that, if A is neighbour of B, then B is neighbour of A, as illustrated by the left-hand side of Figure 1.

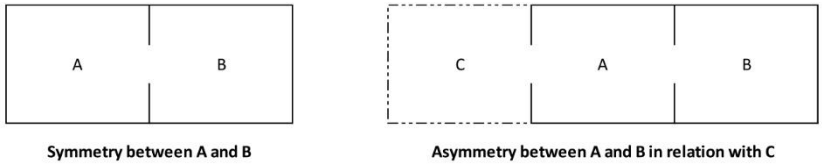

Figure 1: Basic configurations. Adapted from (Hillier 2007, p. 24).

However, if only A is connected with a third space C, as in the right-hand side of the same figure, A and B become asymmetrical in relation with C, because we have to go through A to get to B from C, but we do not have to go through B to get to A from C. Thus, *asymmetry* relates to *depth*, that is, to the number of spaces (or steps) to go from some space, say C, to another space, A (1 step), B (2 steps), and so on.

$$C \rightarrow A \rightarrow B \rightarrow \ldots$$

If we count the number of steps to go from some space to every other space in a complex, we can get a measure of its *total depth* (TD), or of its *mean depth* (MD), by dividing that total by the number of spaces in the complex minus one (Hillier and Hanson 1984, p. 108) (Hanson 1998, pp. 27–28). In the previous example, the total depth of C is 3 which is the sum of steps to reach A and B (1+2) from that origin. This is also the case of B, but the total depth of A is 2 noting that this space is directly connected with both B and C (1+1). Thus, the mean depth is 1 (2/2) for A, and 1.5 (3/2) for either B or C, noting that the number of spaces minus one is 2 (3-1).

This illustration suggests that syntactic measures can be computed for every (convex) space in a settlement or building. Then, we may find that some spaces have lower depth than all other spaces (A in the same example), and others have greater depth (B and C). The former are the most integrated spaces, where social life and/or economic activities such as retail trade might be concentrated in cities (Heitor and Pinelo Silva 2015, p. 162), or the living room (Hillier and Hanson 1984, p. 155–58) (Hanson 1998, pp. 104–5) (Hillier 2007, pp. 25–27). The latter are typically the most segregated spaces in a town, building or house. Thus, *integration* is inversely related with *depth*. It is a *global* measure in the sense that it considers the configuration of one space in relation with all other spaces (or with the spaces at some radius of *n* steps from the original space). Besides, *local* measures such *control* are based on the relations between each space and only the spaces directly connected to it (Turner 2004, p. 14).

As an intelligent critique of Modernism, space syntax stresses the deep tree-like configuration of modern estates (Hillier and Hanson 1984, pp. 129–32, 262–63). A *tree* is a special kind of graph which contain a *root* such that there is a unique path from the root to any other node (Flach 1994, p. 83). Thus, trees are necessarily non-cyclic or *acyclic*, that is, they do not contain paths from a node to itself. The absence of 'rings' in trees give them a *non-distributed* rather than a *distributed* configuration where there is more than one independent route from one space to another including one passing through a third space (Hillier and Hanson 1984, p. 148).

Distributedness can be illustrated by recursive discrete processes where a primary cell is 'glued' together with cells of the same type by the space 'between' them, meaning that the global structure was distributed amongst all primary cells (Hillier and Hanson 1984, p. 11).

3 SPACE SYNTAX AND NETWORK ANALYSIS SOFTWARE: A SURVEY

The software commonly used by the scientific community for syntactic spatial analysis is **Depthmap** (Heitor and Pinelo Silva 2015, p. 167). It is a multi-platform programme to perform a set of spatial network analyses (convex, axial, visibility - VGA, isovist, segment, agent-based and data-driven analyses), originally developed by Alasdair Turner (between 2000 and 2010) and maintained, since 2011, by Tasos Varoudis and Petros Koutsolampros. Depthmap is supported by a comprehensive handbook (Turner 2004), a methodological guide (Al_Sayed et. al. 2014) and diverse on-line tutorial material.[1]

The easiest way to use Depthmap is to import a convex/axial/segment map previously saved in the Autodesk Drawing eXchange File format (extension .dxf), eventually with several layers. Alternatively, Depthmap can import maps (.mif) and data (.mid) compatible with the MapInfo Interchange Format, eventually prepared with the well-known open source geographic information system QGIS (former Quantum GIS), that can export vector drawings into .dxf files too. Additionally, Depthmap accepts the formats for Ordnance Survey 'LandLine' data (.ntf) from United Kingdom, Graph Modelling Language (.gml), SAS/GIS TIGER Basic Data Record (.rt1) and the so-called 'Chiron and Alasdair Transfer Format' (.cat), which is very simple by listing the pair wise (XY) coordinates for just two sorts of data, closed polygon or open polyline (Turner 2004, p. 6).

Table 1: Formats accepted by a selection of space syntax and network analysis software.

EXT	Description	DM	SST	PST	iGraph	NX	Gephy	Pajek	Prolog
.dxf	Autodesk Drawing eXchange File	✓							
.ntf	Ordnance Survey LandLine Data	✓							
.gml	Graph Modelling Language	✓			✓	✓	✓		
.cat	Chiron and Alasdair Transfer Format	✓							
.rt1	SAS/GIS TIGER Basic Data Record	✓							
.mif/.mid	MapInfo Interchange Format	✓							
.tab	MapInfo TAB File			✓					

[1] Available at: https://www.spacesyntax.online/software-and-manuals/depthmap/

EXT	Description	DM	SST	PST	iGraph	NX	Gephy	Pajek	Prolog
.shp	Esri ArcView Shape File		✓			✓			
.sqlite	SQLite/SpatiaLite files		✓				✓		
-	PostgreSQL/PostGIS databases		✓				✓		
-	MySQL database						✓		
-	MS SQL Server database						✓		
.csv	Edge list				✓	✓	✓	✓	
.csv	Data frame/ array/ spreadsheet				✓	✓	✓		
.csv	Adjacency matrix				✓	✓	✓		
.net	Pajek				✓	✓	✓	✓	
.graphml	GraphML (NodeXL)				✓	✓	✓		
.gexf	Graph Exchange XML Format					✓	✓		
.gdf	GUESS						✓		
.dot	GraphViz						✓		
.dl	UCINET				✓		✓		
.tlp	Tulip						✓		
.vna	Netdraw						✓		
.js	JSON					✓			
.leda	LEDA					✓			
.g6	Graph6 / Sparse6					✓			
.txt	Matrix Market					✓			
.txt, .pl	Adjacency or connection list (*)					✓	✓		✓

(*) Declarations in natural language.
EXT – File extension; DM – Depthmap; SST – Space Syntax Toolkit; PST – Place Syntax Tool; NX – NetworkX.

Depthmap's analyses can be performed directly in QGIS platform with a special plugin developed by Jorge Gil (2015), called **Space Syntax Toolkit (SST) for QGIS**. With this useful package, syntactic analyses of Esri ArcView Shape files (.shp), SQLite/SpatiaLite files (.sqlite)

or PostGIS databases become possible with no need to import these maps and data previously into Depthmap, that is, without an exporting operation to either .dxf or .mif/.mid files.

Place Syntax Tool (PST) is another open source plugin for QGIS. It combines the space syntax description of the urban environment with conventional descriptions of attraction into a combined accessibility analysis tool. PST is developed by KTH School of Architecture, Chalmers School of Architecture (SMoG) and Spacescape AB, Sweden. The software was first introduced in a paper by Ståhle, Marcus and Karlström (2005), published in the proceedings of the 5th Space Syntax Symposium held in Delft, the Netherlands. PST runs with MapInfo TAB (.tab) files, but it is easy to save a shape file as Mapinfo TAB directly in QGIS (Stavroulaki et al. 2021, p. 2).

Syntactic analyses may be realised using generic network packages such **igraph** or NetworkX. The former is a collection of open source tools with the emphasis on efficiency, ease to use and portability between input files, operation systems and programming languages. In fact, igraph can be programmed in R, Python, Mathematica or C/C++. For instance, the R package 'igraph' can import comma separated text files (.csv) with edge lists, data frames/spreadsheets and adjacency matrices, eventually sparse, besides other file formats, namely, gml, GraphML or **Pajek**, the specific format used by the homonymous visualization tool that can be exported by Depthmap. Then, igraph functions such degree(), closeness() or estimate_betweenness() can be used to compute, respectively, connectivity, integration or choice (Nepusz 2021, pp. 64, 105 and 149) (Heitor and Pinelo Silva 2015, p. 158-162).

NetworkX is a tool like igraph, but focused on Python users (Hagberg, Schult and Swart 2022). It can import a large set of formats (see Table 1), including very simple *adjacency lists*. This format is useful for graphs without data associated with nodes or edges and for nodes that can be meaningfully represented as strings. The adjacency list format consists of lines with node labels: the first label in a line is the source node and further labels in the line are considered target nodes and are added to the graph along with an edge between the source node and target. For instance, the graph with edges a-b, a-c, d-e can be represented in natural language such that:

```
a b c
c e
```

As we shall see in the next chapters, this kind of natural representation is also privileged by Prolog with small adaptations. Unfortunately, adjacency (or connection) lists are not common among the set of inputs for the space syntax and network analysis software here described. Nevertheless, they can be directly imported in addition to NetworkX by **Gephy**, an open source, easy to use visualization and exploration software for all kinds of graphs and networks, that can compute some relevant metrics too, namely, betweenness (choice) and closeness (integration).

4 ELEMENTARY GENERATORS

In the seminal book *The Social Logic of Space*, Bill Hillier and Julienne Hanson (1984, pp. 66–81) proposed an ideographic language to represent *elementary syntactic generators* such the ones indicated in Figure 2.

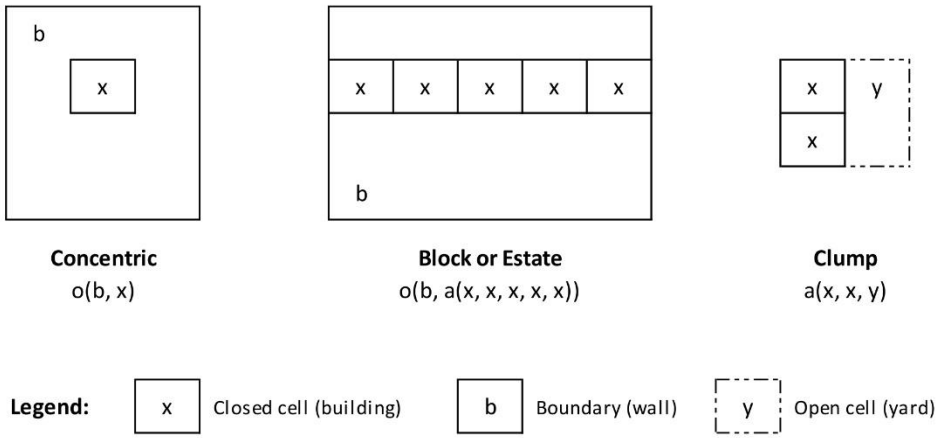

Figure 2: Examples of elementary generators.

Here, the *concentric* type is composed by a building x inside a boundary (wall) b. This is a typical asymmetric generator in the sense that the relation of x with b is different from the relation of b with x. In fact, the boundary b contains the building x, but not the converse, a relationship that can be declared in Prolog with the atom o(b,x) where 'o' represents the relation of *containment*, following the Hillier and Hanson (1984, p. 67) notation.

The *block* or *estate* generator is composed by, say, five buildings combined to form a continuous band, represented here by the atom a(x,x,x,x,x) where 'a' represents the relation of *adjacency*. Then, a boundary was superimposed to each band to form another asymmetric generator that can declared with o(b,a(x,x,x,x,x)) in Prolog. Either block or concentric types generate *depth* in the sense that an additional step should be done to go from the local access (e.g. 'cul-de-sac') to the main entry of each building in order to pass its front wall and yard.

The last type in Figure 2 is a symmetric generator with two buildings attached to an open space or yard y, here described with a(x,x,y). We call it *clump* because it is like the 'beady ring' generator from Hillier and Hanson (1984, pp. 67–68, 78), composed by a closed cell x adjacent to an open cell y.

Now, we will demonstrate how to use Prolog to replicate these elementary generators in a recursive (and pedagogic) way as we found in Hillier and Hanson's book (1984). Firstly, we must define a rule to distribute some generic type *X*:

```
distributed(X,X).
distributed(a(Z,X),X):-distributed(Z,X).
```

The first formula simply declares that a distributed process limited to *X* must contain *X* itself. This is the initial, reflexive condition for the second recursive formula that generates a process *Z* from itself where *Z* is always adjacent to an additional elementary generator *X*.

Secondly, after consulting this simple program with a distribution such SWI-Prolog (Wielemaker et al. 2012), readily available online at the SWISH platform[2], we may ask it about a process *Z* generated from a specific elementary generator, namely

```
?- distributed(Z,o(b,x)).
```

Then, we get a succession of adjacent buildings x, each of them inside a boundary (wall) b:

```
Z = o(b, x)
Z = a(o(b, x), o(b, x))
Z = a(a(o(b, x), o(b, x)), o(b, x))
Z = a(a(a(o(b, x), o(b, x)), o(b, x)), o(b, x))
...
```

Other patterns can be generated at the indicated SWISH web page following, namely, the queries:

```
?- distributed(Z,o(b,a(x,x,x,x,x))).
?- distributed(Z,a(x,x,y)).
```

Thus, we can replicate the same ideography introduced by Hillier and Hanson (1984) with Prolog in a straightforward, suggestive and pedagogic way due to the declarative and recursive nature of Logic Programming.

5 CONCENTRIC GRAMMARS

Among several influences, space syntax was founded on Anthropology, namely, on the works of Claude Lévi-Strauss, Robert Sutherland, Bronislaw Malinowski and Franz Boas (Hillier and Hanson 1984) (Hillier 2007). Besides, Anthropology was influenced by Structural Analysis, namely, by the seminal works of the prince N. S. Troubetzkoy (1949) on Phonology. As stressed by Lévi-Strauss (1963, pp. 31, 33):

> "Linguistic occupies a special place among the social sciences, to whose ranks it unquestionably belongs. It is not a social science like the others, but, rather, the one in which by far the greatest progress has been made. It is probably the only one which can truly claim to be a science, and which was achieved both the formulation of an empirical method and an understanding of the nature of the data submitted to its analysis. (...) Structural linguistics will certainly play the same role with respect to the social sciences that nuclear physics, for example, had played for the physical sciences."

Originally, Prolog was developed to process natural languages, namely, French using linear resolution on definitive clauses (Colmerauer and Roussel 1993, pp. 2, 6). Within this framework, the syntax of a language is specified by a *grammar*, which is a set of rules of the form (Flach 1994, p. 134):

```
sentence      --> noun_phrase, verb_phrase.
noun_phrase --> article, noun.
verb_phrase  --> verb, noun_phrase.
```

[2] This simple Prolog application is available at: https://swish.swi-prolog.org/p/egenerators.pl. This program, as well as other companion programs for this article, can be found on the **Space Syntax with Prolog website**: https://www.sswprolog.net

The former chunk states that a sentence may be structured by a noun phrase followed by a verb phrase, with the former composed by an article and a noun and the later by a (transitive) verb and another noun phrase.

This kind of grammatical analysis can be applied directly to spatial problems, namely, to describe **concentric villages**. Following Lévi-Strauss (1963, p. 141), Hillier and Hanson (1984, pp. 92-93) gave special attention to traditional villages with this pattern like Bororo, Mato Grosso, Brazil, which is characterized by a large amount of convex space invested in a central zone, highly *synchronized* with several objects placed at its periphery (Figure 3).

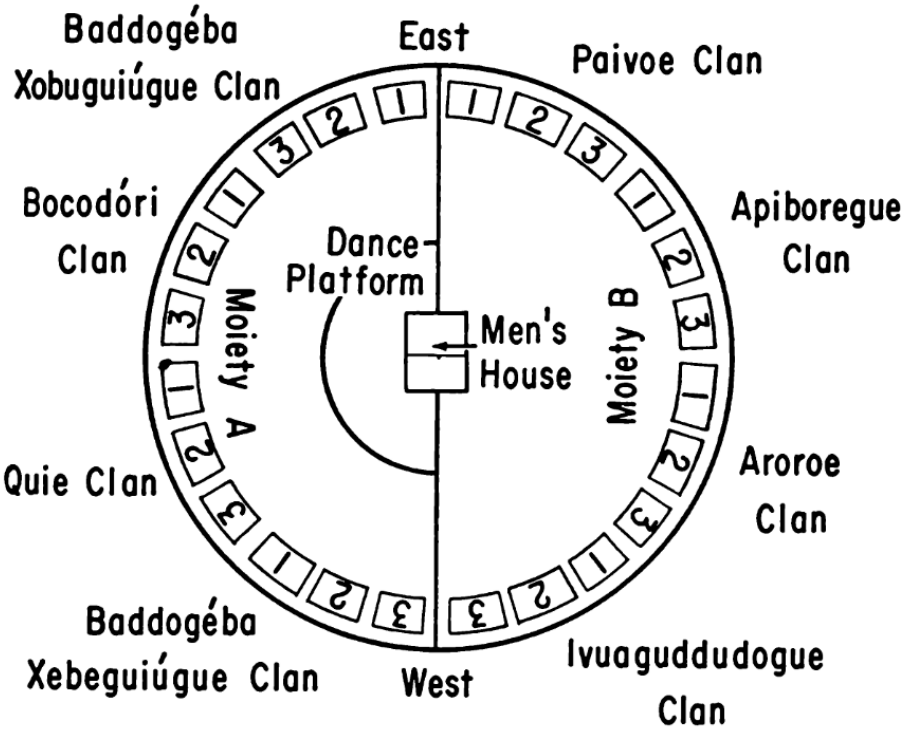

Figure 3: Plan of a Bororo village, Mato Grosso, Brazil (Lévi-Strauss 1963, p. 141).

Thus, we can define a generic concentric village as a centre surrounded by a periphery where the former is composed by facilities and open spaces and the latter by several (e.g. up to three) types of housing estates, using Prolog grammars, that is:

```
conc_village  --> centre, periphery.
centre        --> facility, open_space.
centre        --> facility, facility, open_space.
centre        --> facility, facility, open_space, open_space.
periphery     --> housing.
periphery     --> housing, housing.
periphery     --> housing, housing, housing.
```

Then, we must specify the *terminal* categories (such words in a phrase) for the spaces that structure the Bororo village described in Figure 3: [3]

```
facility      --> [men_house].
facility      --> [dance_platform].
open_space    --> [scrub_land].
```

[3] Program available at: https://swish.swi-prolog.org/p/concentric_bororo.pl

```
housing      --> [upper_class_houses].
housing      --> [middle_class_houses].
housing      --> [low_class_houses].
```

After consulting this program in the SWISH web page, we can ask the query

```
?- conc_village([men_house,dance_platform,scrub_land, upper_class_houses,middle_class_houses,
        low_class_houses],[]).
```

to get an affirmative answer, that is, the Bororo village respects the above-defined syntax of concentric villages because it is structured by a centre with two facilities (men house and dance floor) and an open space (scrub land) surrounded by three types of houses (upper, middle and low or 1/2/3 in Figure 3).

This kind of concentric structure was found in many places, namely, in the village of Omarakana, Trobriand Islands, Melanesia (Malinowski 1929, Fig. I, p. 649, reproduced by Lévi-Strauss 1963, p. 136), and even in modern estates such Portela, near Lisbon, Portugal (Pereira 2017, p. 521).

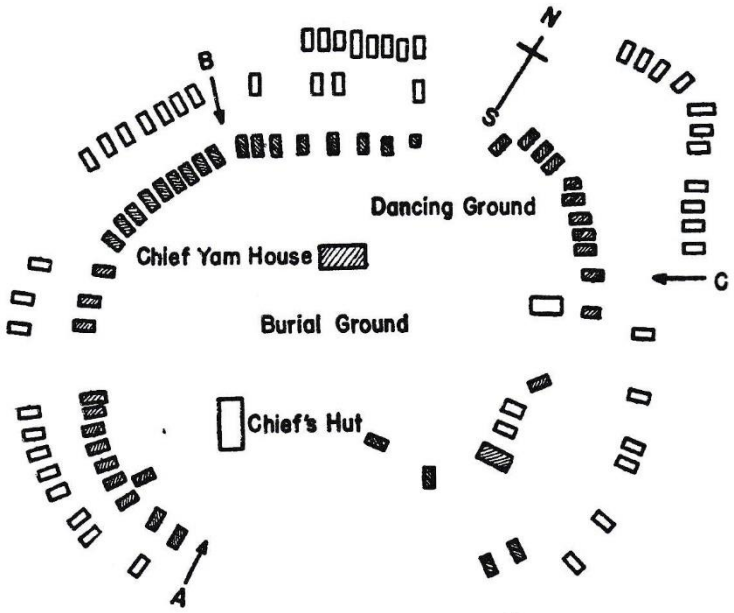

Figure 4: Plan of an Omarakana village, Trobriand Islands, Melanesia (Lévi-Strauss 1963, p. 136).

The terminals for the Omarakana village (see Figure 4, above) may be: [4]

```
facility      --> [chief_yam_house].
facility      --> [chief_hut].
open_space    --> [burial_ground].
open_space    --> [dancing_ground].
housing       --> [yam_storehouses].
housing       --> [dwelling_houses].
```

Now, we will get an affirmative answer (or true statement) from the following query:

```
?- conc_village([chief_yam_house,chief_hut,burial_ground,dancing_ground,yam_storehouses,
dwelling_houses],[]).
```

[4] Program available at: https://swish.swi-prolog.org/p/concentric_omarakana.pl

in the sense that Omarakana can be described as a concentric village with a centre composed by the chief's house and hut on two distinct open spaces (burial and dancing grounds), surrounded by the yam storehouses and the dwelling houses properly.
Recall that we changed only the terminals from Bororo to Omarakana, maintaining the syntactic rules in both cases. Thus, the two villages have the same spatial grammar, that is, they have the same concentric structure, as stressed by Lévi-Strauss (1963, pp. 135-142).
Obviously, Prolog grammars can be applied to a wide range of spatial structures, namely, to the elementary generators introduced by Hillier and Hanson (1984, p. 78) in their seminal book.

6 SYNTACTIC METRICS

6.1 Knowledge base

A Prolog *program* is a knowledge base, or database, composed by a collection of facts and rules which describe a set of relations of interest (Blackburn, Bos, and Striegnitz 2006). *Facts* express unconditional truths (Flach 1994, p. 5), namely, the connections between lines in an *axial map*, the longest and fewest lines that cover the street grid, or between convex spaces in a *gamma map*, namely, of a building or house (Hillier and Hanson 1984, p. 91–92) (Hillier 2007, p. 98).
In the well-known case of the Ashanti's *abosomfie*, or shrine, introduced by Hiller and Hanson (1984, p. 181) after Rutter, we know that the 'outside', represented in the graph of the Figure 5 (above) by the circle with a cross, is directly connected with the courtyard (*gyaase*) through the main entrance, and also with the backyard (*ahemfie*, represented with a filled circle in the same graph), which is, by itself, connected with the courtyard by an exit door. These facts can be declared in a Prolog program (plain text file), say **abosomfie.pl**, in the following fashion: [5]

```
connected(outside, courtyard, 1).
connected(outside, backyard, 1).
connected(courtyard, backyard, 1).
```

These three clauses or *atoms* belong to the same *predicate* because they have the same *name* ('connected') and the same *arity*, that is, the same number of arguments (three) enclosed in parentheses and separated by commas (Flach 1994, pp. 25, 31). Here, the third argument is the topological distance between two adjacent lines, that is, one. In some applications, it may be greater than one in order to incorporate the metric distance or other cost (e.g. stairs or ramp) between two convex spaces, but here we adopt the topological distance introduced in *The Social Logic of Space* (Hillier and Hanson 1984, p. 103).
Clauses of the same predicate must be declared jointly in a Prolog program, so we must add the remaining connections (or permeabilities) inferred from the plant and graph of the Ashanti's shrine (Figure 5, below) to our knowledge base in order to proceed, namely:

```
connected(courtyard, kitchen, 1).
connected(courtyard, drum_room, 1).
connected(courtyard, singers_patio, 1).
```

[5] Program available at: https://swish.swi-prolog.org/p/abosomfie.pl or, alternatively, through the **Space Syntax with Prolog website**: https://www.sswprolog.net

```
connected(courtyard, passage, 1).
connected(passage, waiting_room, 1).
connected(waiting_room, shrine_room, 1).
```

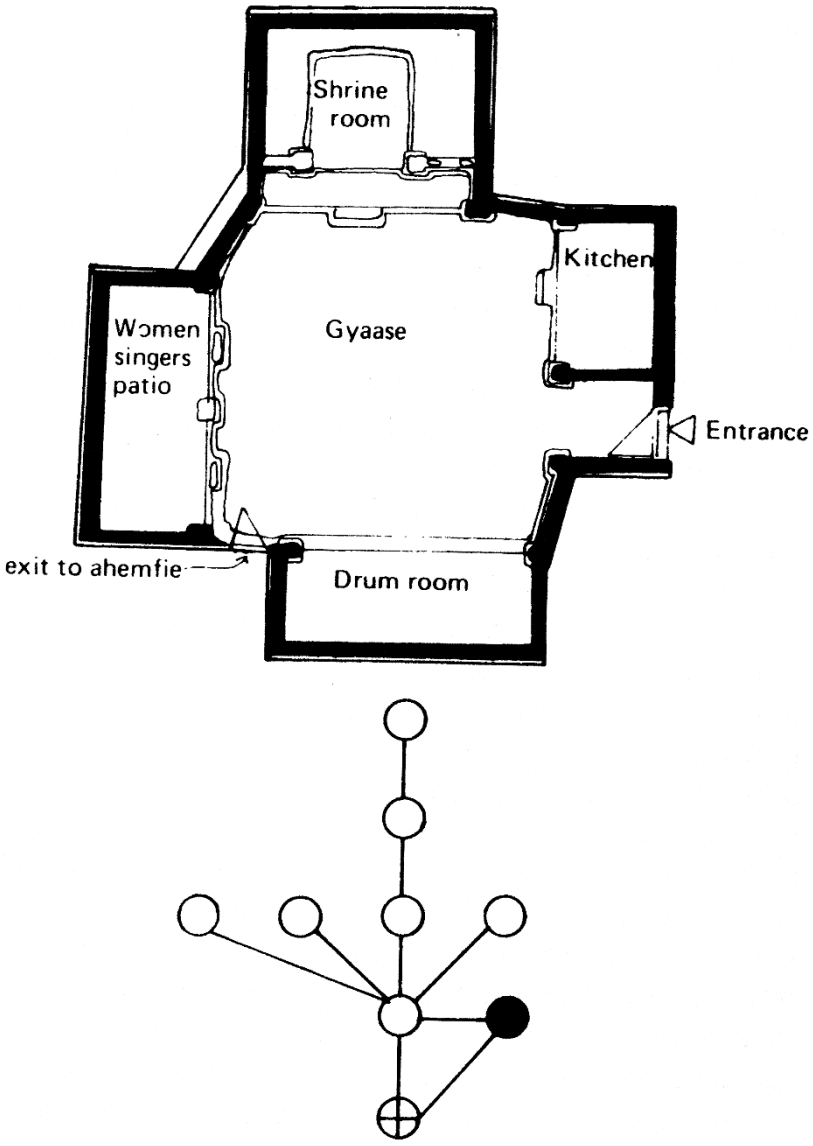

Figure 5: Plan and graph of an Ashanti's *abosomfie*, or shrine (Hiller and Hanson, 1984, p. 181).

Then, after consulting the program **abosomfie.pl** available on the SWISH platform (see the last footnote), we may ask it about the spaces *Y* connected, for instance, with the courtyard:

```
?- connected(courtyard, Y, 1).
```

where the prefix '?-' indicates that this is a query rather than a fact. In Prolog, a *variable* such *Y* is written as a sequence of letters and digits, beginning with a capital letter or underscore. An answer to the previous query, e.g. backyard, will be written {Y -> backyard} following the notation of Flach (1994, p. 5). This means that Prolog found a value for the variable *Y*, that is, a solution given the knowledge base previously loaded and consulted. If asked, Prolog will try to find more values to *Y* given the knowledge base, namely, {Y -> kitchen}, {Y -> drum_room},

and so on. These are the convex spaces 1-step away from the courtyard, that is, at depth 1 from this root.

6.2 Connectivity

For practical purposes, we must create a pair of *rules*, that is, of conditional truths that can only be drawn when their premises are known to be true (Flach 1994, p. 5), that isolates the lines that are *adjacent* with some space *X* in the sense that it can be connected with *Y* or, conversely, *Y* can be connected with *X*:

```
adjacent(X, Y, 1) :- connected(X, Y, 1).
adjacent(X, Y, 1) :- connected(Y, X, 1).
```

where the symbol ':-' should be read as 'if'. These rules mean: 'for any values of *X* and *Y*, *X* and *Y* are adjacent if *X* is directly connected with *Y* or *Y* is directly connected with *X* with a topological distance of 1'.

Now, we can compute the *connectivity* (AC) of some space, which measures the number of spaces immediately connecting that space of origin (Hillier and Hanson 1984, p. 103), using the SWI-Prolog built-in predicate aggregate_all/3 with the template count:

```
connectivity(X, Y, AC) :- aggregate_all(count, adjacent(X, Y, 1), AC).
```

For instance, the following query returns {AC -> 6} because the *abosomfie*'s courtyard is directly connected with six spaces, including the 'outside':

```
?- connectivity(courtyard, Y, AC).
```

6.3 Control

Connectivity is the basic syntactic measure in the sense that the others are based on it (Heitor and Pinelo Silva 2015, p. 158). In particular, the measure of *control* (E) proposed by Hillier and Hanson (1984, p. 109) and implemented below sums up the space given *G* to some space *X* by its immediate (adjacent) neighbours *Z*, where *G* is the reciprocal of the connectivity (AC) of each *Z*, and the comma ',' (between predicates) should be read as 'and':

```
control(X, Y, E) :- aggregate_all(sum(G), (adjacent(X, Z, 1), connectivity(Z, Y, AC), G is 1/AC), E).
```

This specific rule illustrates how useful and pedagogical Prolog can be in describing the syntactic concepts. In fact, it suggests the direct relation of control with the connectivity of the space of origin, that is, with the number of its adjacent neighbours, as well as the inverse relation with the connectivity of the last ones. In fact, spaces with a strong control, that is, with an E greater than 1, are typically connected with several spaces, namely, with terminal rooms (or 'cul-de-sacs' in the context of a settlement analysis).

So, in order to be controlling, a line must see many spaces, but these spaces should each see relatively little (Turner 2004, p. 16). In the *abosomfie*, the case of the courtyard is remarkable with {E -> 4.5}, which can be computed by formulating the query:

```
?- control(courtyard, Y, E).
```

The last figure is a direct consequence of the several rooms and other spaces directly connected with the *gyaase* (see Figure 5). In fact, the courtyard is the bipermeable space of visitors, that synchronises all the significant spaces in the system (Hillier and Hanson 1984, p. 181). As we shall see, the genotype of buildings for religious observance combines a deep sacred space with a large shallow space like the abomsofie's *gyaase*.

6.4 Depth

Control is a *local* measure, since it only considers relations between a space and its immediate neighbours (Hillier and Hanson 1984, 109). Another local measure, *controllability* picks out areas that may be easily visually dominated (Turner 2004, 16). It is simply the ratio of connectivity to the total number of spaces 1 or 2-steps away from the original space. In order to compute this measure, as well as relative asymmetry which is a *global* measure that considers the distance of some space to the others, we must define a predicate for *depth* firstly.
Obviously, connectivity is related with the concept of depth 1 in the sense that it is the number of spaces 1-step away from the space of origin. In fact, we can define a rule such that *Y* is 1-deep from *X* if *X* and *Y* are adjacent:

```
depth(X, Y, 1) :- adjacent(X, Y, 1).
```

Similarly, the spaces 2-steps away from the original space are 1-step away from the spaces adjacent to the root. The following rule explores this recursive nature of depth, that is, the spaces at depth 2 from *X* are the spaces *Y* at depth 1 from the spaces *Z* adjacent to *X*:

```
depth(X, Y, 2) :- adjacent(X, Z, 1), depth(Z, Y, 1), dif(X, Y).
```

where the SWI-Prolog built-in predicate dif/2 introduces a constraint such that every 2-deep space *Y* must be different from the space of origin *X* in order to avoid backward relations. The following query returns a list, indicated by square brackets '[Y]', with the spaces 2-steps away from the 'outside':

```
?- distinct([Y], (depth(outside, Y, 2))).
```

where the built-in predicate distinct/2 eliminates duplicates, that is, 2-deep spaces that can be accessed through two or more spaces directly connected with the space of origin. Depth 3 can be defined in the same recursive way by isolating the spaces 2-steps away from the spaces *Z* directly connected with the original space *X*, depth 4 by isolating the deep 3 spaces, and so on:

```
depth(X, Y, 3) :- adjacent(X, Z, 1), depth(Z, Y, 2), dif(X, Y).
depth(X, Y, 4) :- adjacent(X, Z, 1), depth(Z, Y, 3), dif(X, Y).
...
```

6.5 Justified graph

It is important to note that a space may be accessible at several steps from the same origin. For instance, the backyard is either 1-step away from the 'outside' or 2-steps away because we can go through the courtyard to get there. This occur in every graph with 'rings' or 'islands' like Figure

5. So, we must create a new rule to set the minimum number of steps D to go from *X* to *Y* once again using the Prolog's predicate distinct/2:

```
graph(X, Y, D) :- distinct([Y], depth(X, Y, D)).
```

We named this predicate 'graph' because it analytically generates the *justified graph*, or j-graph, for some root *X*. The resulting j-graph is a 'picture' with the depth of all spaces in a complex from a point in it (Hillier 2007, pp. 22–23, 72–73). For instance, we can analytically describe the j-graph from the 'outside' with the following simple query:

```
?- graph(outside, Y, D).
```

to get:

```
{Y -> courtyard,      D -> 1}
{Y -> backyard,       D -> 1}
{Y -> kitchen,        D -> 2}
{Y -> drum_room,      D -> 2}
{Y -> singers_patio,  D -> 2}
{Y -> passage,        D -> 2}
{Y -> waiting_room,   D -> 3}
{Y -> shrine_room,    D -> 4}
```

Right now, the former output suggests the deepest nature of the most sacred space, the shrine room, which is the domain of the inhabitants (priests), in contrast with the shallowness of the courtyard, the space of visitors. This duality is a characteristic found in religious buildings across many cultures and times, namely, in the English parish church (Hillier and Hanson 1984, p. 181).

6.6 Controllability

The predicates graph/3 and connectivity/3 are the 'building blocks' to compute several measures, namely, the above-mentioned local measure of *controllability* (F):

```
controllability(X,Y,Z,F):-connectivity(X,Y,AC),aggregate_all(count, graph(X,Z,2), D2),F is AC/(AC+D2).
```

6.7 Mean depth

Another measure that can be computed with graph/3 is *mean depth* (MD) referred in the theoretical framework. It is calculated by summing up the depth values D from some root *X* and dividing by the number of spaces in the complex minus one, as explained above (Hillier and Hanson 1984, p. 108). The following implementation introduces two previous (auxiliary) predicates that compute the *total depth* (TD) and the *number of nodes* (N) in the complex except (or minus) the root:

```
totdepth(X, Y, TD)        :- aggregate_all(sum(D), graph(X, Y, D), TD).
nodes(X, Y, D, N)         :- aggregate_all(count, graph(X, Y, D), N).
meandepth(X, Y, D, MD)    :- totdepth(X, Y, TD), nodes(X, Y, D, N), MD is TD/N.
```

Mean depth is an important syntactic measure, as said, and its reciprocal can be used as a straightforward measure of the *integration* (I) of each space in a complex as suggested by Heitor and Pinelo Silva (2015, p. 162):

```
integration(X, Y, D, I) :- meandepth(X, Y, D, MD), I is 1/MD.
```

6.8 Relative asymmetry

The value of total or (even) mean depth can be affected by the number of nodes in a graph. Thus, Hillier and Hanson (1984, p. 108) proposed a normalization of MD which eliminates the bias due to the number of nodes, that is:

```
asymmetry(X, Y, D, RA) :- totdepth(X, Y, TD), nodes(X, Y, D, N), RA is 2*(TD/N-1)/(N-1).
```

This formula will give a value between 0 and 1, with low values indicating a space which tends to integrate the whole complex or system, and high values a space which tends to be segregated from the system. Thus, *relative asymmetry* (RA), also denoted by 'i-value' (Hillier 2007, p. 77), is a normalized measure of integration.[6]

6.9 Real relative asymmetry

The RA measure can be used to compare different systems with (approximately) the same number of spaces. However, if the systems differ considerably in size, a second normalization should be applied because a small system always looks more integrated than a large one (Turner 2004, p. 14).

To deal with this empirical problem, Hillier and Hanson (1984, pp. 109–13) proposed an adjusted measure, the *real relative asymmetry* (RRA), which is the RA value of the space divided by the RA value of the root of a graph shaped like a 'diamond' where there are *k* nodes at middle level, *k*/2 at one level above and below the middle level, *k*/4 at one level above and below the *k*/2 level, and so on, until there is one node at the root and deepest nodes (Teklenburg, Timmermans, and Wagenberg 1993, pp. 350–51). The following implementation uses the formula proposed by Krüger and Vieira (2012, p. 200) to estimate the RA value of the root of that diamond (*d-value*):

```
dvalue(K, DV)     :- DV is 2*(K*(log((K + 2) / 3) / log(2) - 1) + 1) / ((K - 1)*(K - 2)).
rra(X, Y, D, RRA) :- asymmetry(X, Y, D, RA), nodes(X, Y, D, N), dvalue(N + 1, DV), RRA is RA / DV.
```

Finally, the inverse of RRA is itself a measure of integration (IHH), the so-called 'Hillier and Hanson (integration value with) d-value normalization' (Turner 2004, p. 25), that 'captures the extent to which each spatial element contributes to drawing the whole configuration together into a more or less direct relationship' (Hanson 1998, p. 27).

```
integrationHH(X, Y, D, IHH) :- rra(X, Y, D, RRA), IHH is 1 / RRA.
```

6.10 Results

Prolog can compute the metrics described above at once for some space, say, the courtyard with a **multi-condition query** such:

[6] The mentioned program **abosomfie.pl**, available at the SWISH web page, has additional predicates to compute radius-*r* integration and other local measures.

```
?- X = courtyard, connectivity(X, Y, AC), control(X, Y, E), controllability(X, Y, Z, F), totdepth(X, Y, TD),
meandepth(X, Y, D, MD), integration(X, Y, D, I), asymmetry(X, Y, D, RA), rra(X, Y, D, RRA),
integrationHH(X, Y, D, IHH).
```

With this tip in mind, it was very easy (and fast) to compute the relevant syntactic measures for each space of the Ashanti's shrine, using the program **abosomfie.pl**, provided at the SWISH web page.[7] The results were condensed in Table 1, that may provide additional material for a curious student of the book *The Social Logic of Space* (Hillier and Hanson 1984).

Table 2: Syntactic metrics for the Ashanti's *abosomfie*, or shrine, computed with Prolog.

Convex spaces	**AC**	**E**	**F**	**I**	**IHH**	**MD**	**RA**	**RRA**	**TD**
Outside	2	0.67	0.25	0.47	0.99	2.13	0.32	1.01	17
Backyard (*ahemfie*)	2	0.67	0.25	0.47	0.99	2.13	0.32	1.01	17
Courtyard (*gyaase*)	6	4.50	0.67	0.72	2.96	1.38	0.11	0.34	11
Kitchen	1	0.17	0.17	0.44	0.89	2.25	0.36	1.13	18
Drum room	1	0.17	0.17	0.44	0.89	2.25	0.36	1.13	18
Singers patio	1	0.17	0.17	0.44	0.89	2.25	0.36	1.13	18
Passage	2	0.67	0.25	0.57	1.48	1.75	0.21	0.68	14
Waiting room	2	1.50	0.67	0.42	0.81	2.38	0.39	1.24	19
Shrine room	1	0.50	0.50	0.31	0.49	3.25	0.64	2.03	26

AC – Connectivity; E – Control; F – Controllability; I – Integration Heitor and Pinelo Silva (1/TD); IHH - Integration Hillier and Hanson (1/RRA); MD – Mean Depth; RA - Relative Asymmetry; RRA - Real Relative Asymmetry (RA/d-value); TD – Total Depth.

Those metrics confirm the integration and the distribution function, measured by control (E) and controllability (F), of the courtyard, as well the asymmetry of the most sacred space of the shrine, the room where the priest receives the visitors. Thus, a large shallow space, with a direct access from the 'outside', is synchronized with a deep space, following the genotype of religious and similar buildings (Hillier and Hanson 1984, p. 181). The relation between these two spaces is made through an axis of progressive deeper spaces, namely, a waiting room before the shrine room properly, which is a direct by-product of that kind of genotype.[8]

6.11 Bookkeeping

It is quite easy, or even funny, to adapt the previous program to perform a swift analysis of another building, for instance, the well-known farmhouse 'La Bataille' (Normandy, France) described by Julienne Hanson (1998, p. 85). Here are the **practical steps**:

[7] Program available at: https://swish.swi-prolog.org/p/abosomfie.pl, as said.
[8] See also the 3D model, plans and sections of the Besease Ashante Shrine in Ejisu, Kumasi, Ghana, available here: https://www.zamaniproject.org/site-ghana-kumasi-asante-shrine.html

1. Go to the SWI-Prolog homepage: https://www.swi-prolog.org/
2. Select 'Try SWI-Prolog online' (or alternatively download and install SWIPL);
3. In the search box from the SWISH platform, find the **abosomfie.pl** file and pick it in the pulldown callout;
4. At the top of this program (eventually downloaded with 'File'-'Download', if you installed SWIPL in your computer), you will find a knowledge base with the connections for the Ashanti's shrine; please edit them to declare, instead, the connections (or permeabilities) between 'La Bataille' convex spaces, namely,

```
connected(outside, vestibule_1, 1).
connected(outside, salle_commune, 1).
connected(outside, vestibule_2, 1).
connected(vestibule_1, grande_salle, 1).
connected(vestibule_1, couloir, 1).
connected(couloir, salle, 1).
connected(salle, bureau, 1).
connected(couloir, salle_commune, 1).
connected(vestibule_2, salle_commune, 1).
connected(vestibule_2, depense, 1).
connected(depense, laiterie, 1).
connected(laiterie, laverie, 1).
connected(laverie, salle_commune, 1).
```

5. Calculate the syntactic measures, e.g. of *salle commune* with the multi-condition query:

```
?- X = salle_commune, connectivity(X, Y, AC), control(X, Y, E), controllability(X, Y, Z,
F), totdepth(X, Y, TD), meandepth(X, Y, D, MD), integration(X, Y, D, I), asymmetry(X, Y,
D, RA), rra(X, Y, D, RRA), integrationHH(X, Y, D, IHH).
```

6. Eventually save your edited file with 'File' – 'Save …' for future use, given it a different name and description. [9]

7 CONCLUSION

The previous analyses illustrate how does Prolog can be useful and attractive, namely, for students and researchers on space syntax. **Readily available on-line** through the SWISH platform (https://swish.swi-prolog.org/), in a fancy format inspired by Jupyter Notebooks for Python, R or Julia[10], SWI-Prolog can give a precious help to understand the recursive nature of urban processes given some elementary generators, or to describe or even check the structure (e.g. concentric) of some village. Mostly important, Prolog can compute space syntax measures such connectivity, control or integration in a **comprehensive and transparent way**, using natural language and avoiding 'black boxes' and complex software packages.

The short experience with this new tool and approach to space syntax suggests that Prolog may be appropriate for **gamma-analysis of buildings and houses**. In this sense, Prolog can function as a **readily available and flexible calculator** of syntactic measures, given a database with the

[9] In fact, we already made it readily available at: https://swish.swi-prolog.org/p/labataille.pl. Please check the website https://www.sswprolog.net (continuously updated) for other examples.
[10] See: https://jupyter.org/

connections between spaces, like an *adjacency list* in NetworkX or Gephy. Its flexibility is reinforced by the **free and open-source** nature of the code stored in the SWISH platform, including the programs that illustrate this paper.

Nevertheless, Prolog can perform an **alpha analysis of urban settlements**, by declaring the connections between axial lines instead of convex spaces. The only care is to avoid very complex and/or big complexes. An illustrative example is the syntactic analysis of the Areeiro borough in Lisbon, Portugal, with the computation of *relative ringiness*, a metric that Depthmap does not provide (Fernandes, 2022).

In one phrase, the present research shows how Prolog can be used to describe **the structure of buildings or settlements as a 'field of knowables' or 'intelligent system'**. In fact, space syntax was founded on the premise that even complex spatial arrangements, just like **natural languages**, are systems governed by simple concepts and rules. Prolog was founded on the same idea in the sense that it emerged as a research project to process natural languages using *grammars*. Thus, space syntax and Prolog are similar in reasoning, and both represent knowledge in a declarative way. Indeed, the 'problem of knowability' is paramount not only to space syntax but also to Artificial Intelligence (and Prolog), as stressed by Hillier and Hanson (1984, p. 46) in their seminal work.

Future developments of this research foresee the specific website [11] that centralizes the links to the Prolog programmes already stored (or that will be stored) in the SWISH platform. The main goal is to provide a unique central point and companion for the student and/or researcher that want to compute the metrics for a selection of buildings and settlements that illustrates the books *The Social Logic of Space* (Hillier and Hanson 1984), *Decoding Homes and Houses* (Hanson 1998) and *Space is the machine* (Hillier 2007). Among others, the programs will cover the Barnsbury borough and the Maiden Lane Estate, both in London, the English cottages, the Ashanti palace or the Normandy farmhouses. Additionally, the Space Syntax with Prolog site will concentrate the author's research on the field (papers, presentations, posters and programs), a list of relevant references, including tutorials for all that want to learn Prolog, and, eventually, a calculator of syntactic measures for Microsoft Windows, to be develop with Visual Prolog.

Other, rather ambitious, project is the development of a natural language parser that could identify, from a simple description of an urban network (e.g. *adjacency list*), the elementary generators, namely, beady-ring or concentric, embodied on it. This is the kind of 'intelligent task' that Prolog had been made for. The starting point would be the grammars described in chapter 5.

Finally, the inclusion of economic reasoning in space syntax theories and methods, complementing the contributions of Anthropology and other social sciences, is another promising research topic that may use Logic and/or Neuro-Dynamic Programming, following the works of Bertsekas and Tsitsiklis (1996), Sutton and Barto (2020) and Bertsekas (2022).

[11] The **Space Syntax with Prolog website**, https://www.sswprolog.net, is available also from the author's homepage https://paf.com.pt. It will be developed and enriched continuously, namely, with programs for a selection of settlements and buildings in Portugal, e.g. the traditional farmhouses studied by Mário Moutinho (1979).

REFERENCES

Al_Sayed, K., Turner, A., Hillier, B., Iida, S. and Penn, A. (2014) *Space Syntax Methodology*. Bartlett School of Graduate Studies, University College of London.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996) *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific.

Bertsekas, D. P. (2022) *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptative Control*. Belmont, Massachusetts: Athena Scientific.

Blackburn, P., Bos, J. and Striegnitz, K. (2006) *Learn Prolog Now!* College Publications. (Texts in Computing). Available at: http://www.learnprolognow.org/.

Colmerauer, A. and Roussel, P. (1993) 'The birth of Prolog', *ACM SIGPLAN Notices*, 28(3), pp. 37–52. doi: 10.1145/155360.155362.

Fernandes, P. A. (2022) 'Programming the Logic of Space: a new approach to space syntax', in Smaniotto Costa, C. and Aragão, N. (ed.) *Understanding and transforming the territory.* Culture & Territory 05. Lisbon: Edições Universitárias Lusófonas. Forthcoming.

Flach, P. (1994) *Simply Logical: Intelligent Reasoning by Example*. John Wiley and Sons. Available at: https://book.simply-logical.space/.

Gil, J. (2015) *Space Syntax Toolkit for QGIS – User Guide (version 0.1.0)*. Space Syntax Laboratory, The Bartlett School of Architecture, University College of London. Available at: https://github.com/SpaceGroupUCL/qgisSpaceSyntaxToolkit

Hagberg, A., Schult, D. and Swart, P. (2022). *NetworkX Reference (release 2.8.1)*. NetworkX developers. Available at: https://networkx.org/

Hanson, J. (1998) *Decoding Homes and Houses*. Cambridge: Cambridge University Press.

Heitor, T. and Pinelo Silva, J. (2015) 'A Sintaxe Espacial e o Ambiente Construído - Análise Morfológica', in Oliveira, V., Marat-Mendes, T., and Pinho, P. (eds) *O Estudo da Forma Urbana em Portugal*. Porto: Universidade do Porto, pp. 147–189.

Hillier, B. (2007) *Space is the machine: a configurational theory of architecture*. London: Space Syntax.

Hillier, B. and Hanson, J. (1984) *The Social Logic of Space*. Cambridge: Cambridge University Press.

Kowalski, R. A. (1988) 'The early years of Logic Programming', *Communications of the Association for Computing Machinery*, 31(1), pp. 38–43. doi: 10.1145/35043.35046.

Krüger, M. and Vieira, A. P. (2012) 'Scaling relative asymmetry in space syntax analysis', *Journal of Space Syntax*, 3(2), pp. 194–203.

Lévi-Strauss, C. (1963) *Structural Anthropology*. New York: Basic Books.

Malinowski, B. (1929) *The Sexual Life of Savages in North-Western Melanesia*. New York: Readers League of America and Eugenics Publishing Company. Available at: https://archive.org/details/sexuallifeofsava00mali/page/n9/mode/2up.

Moutinho, M. (1979) *A Arquitectura Popular Portuguesa*. Lisboa: Editorial Estampa.

Nepusz, T. (2021) *Package "igraph" - Network Analysis and Visualization*. Available at: https://igraph.org

Pereira, S. M. (2017) 'Mass housing in Lisbon: sometimes it works', *Journal of Housing and the Building Environment*, 32, pp. 513–532. doi: 10.1007/s10901-016-9525-2.

Ståhle, A., Marcus, L. and Karlström, A. (2005) *Place Syntax - Geographic Accessibility with Axial Lines in GIS*. 5th International Space Syntax Symposium, Delft.

Stavroulaki, G., Berghauser Pont, M., Marcus, L. and Fitger, M. (2021) *PST Documentation (version 3.2.3)*. KTH School of Architecture, Chalmers School of Architecture (SMoG) and Spacescape AB. doi: 10.13140/RG.2.2.26263.91043.

Sutton, R. S. and Barto, A. G. (2020) *Reinforcement Learning: An Introduction*. Second edition. Cambridge, Massachusetts and London, England: The MIT Press.

Teklenburg, J. A. F., Timmermans, H. J. P. and van Wagenberg, A. F. (1993) 'Space Syntax: Standardised Integration Measures and Some Simulations', *Environment and Planning B: Urban Analytics and City Science*, 20(3), pp. 347–357. doi: 10.1068/b200347.

Troubetzkoy, N. S. (1949) *Principes de Phonologie*. Paris: Librairie C. Klincksieck. Available at: https://archive.org/details/principesdephono00trub/page/n9/mode/2up.

Turner, A. (2004) *Depthmap 4 - A Researcher's Handbook*. London: Bartlett School of Graduate Studies, University College of London. Available at: https://discovery.ucl.ac.uk/id/eprint/2651/.

Wielemaker, J. et al. (2012) 'SWI-Prolog', *Theory and Practice of Logic Programming*, 12(1–2), pp. 67–96. doi: 10.1017/S1471068411000494.