LAR-ABC, a representation of architectural geometry

From concept of spaces, to design of building fabric, to construction simulation *

April 26, 2014

Abstract

This paper discusses the application of LAR (Linear Algebraic Representation) scheme [**?**] to the whole architectural design process, from initial concept of spaces, to the additive manufacturing of design models, to the meshing for CAE analysis, to the detailed design of components of building fabric, to the BIM processing of quantities and costs. LAR (see, e.g. [1]) is a novel general and simple representation scheme for geometric design of curves, surfaces and solids, using simple, general and well founded concepts from algebraic topology. LAR supports all topological incidence structures, including enumerative (images), decompositive (meshes) and boundary (CAD) representations. It is dimension-independent, and not restricted to regular complexes. Furthermore, LAR enjoys a neat mathematical format, being based on chains, the domains of discrete integration, and cochains, the discrete prototype of differential forms, so naturally integrating the geometric shape with the supported physical properties. The LAR representation find his roots in the design language Plasm [**?**], and is currently embedded in python and javascript, providing the designer with powerful and simple tools for a geometric calculus of shapes. In this paper we introduce the motivation of this approach, discussing how it compares to other mixed-dimensionality representations of geometry and is supported by open-source software projects. We also discuss simple examples of use, with reference to various stages of the design process.

*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [**?**]. April 26, 2014

Contents

1 Introduction

Looking for simplicity. Form and function. Who comes first? CAD is evolving with the net. Algebraic geometry on graphics processors.

2 Linear Algebraic Representation

2.1 Representation scheme

A foundational concept. Mapping from mathematical models to computer representations. Taxonomy: decompositive, enumerative, boundary, procedural representations. The de-facto standard of PLM systems: non-manifold data structures + NURBS. Old, expensive, and inflexible. Towards big geometric data. The need for rethinking the foundations. Algebraic geometry is here to stay. Sparse matrices and GPGPU.

2.2 Topological operations

Space decompositions. Chains as subsets of spaces. Cochains as fields over chains. Geometric integration: pairing of chains and cochains. Homology and cohomology. Boundary and coboundary operators. A single SpMV multiplication to compute the boundary of any chain of spaces. Transposition and coboundary. Coboundary and operators of vector analysis: gradient, curl, divergenze, and Laplacian. Integration of geometry and physics through topological structures. Topological queries. The d-star of every cell. Decomposition of big structures and distributed computing.

2.3 Models and structures

Model = Topology + Geometry. Model as embedded topology. Topology as the list of higher-dimensional cells of a space decomposition. The classical representation of cells via the list of their vertices. Composition of linear operators as the product of their matrices. Fast transposition and product operations via GPGPU and OpenCL. Structures as hierarchical aggregation of space-instanced models and/or other structures. The representation of structures as ordered sequences of strictres, models and operators. Data Base of structures. The traversal of structures at run-time.

3 LAR for Architecture, Building and Construction

3.1 Architectural structures: the organisation of spaces

Organic or rational architecture? Form from function or vice-versa? In either case, space units produce organised assemblies of spaces with more or less specialised functions. In set-theoretical terms, they provide either a partitioning or a covering of the building space.

At the very end, an architectural design defines a topology of the space, as a collection of subsets (of space), closed with respect to finite intersection and union.

In practical terms, the design concept will result in a cellular decomposition of the space, where elementary space units, either open or closed or partially open, establish pairwise adjacency relations; even before of any concrete embedding, i.e. even before that a global shape is envisioned by the architect.

In the majority of cases, the geometric embedding of the design topology will produce a partitioning of space with plane or curved surfaces, and some horizontal or vertical composition of use patterns. In common building and construction, most of the separating surfaces will be either horizontal or vertical, but different arrangements of space cells are possible.

LAR-ABC requires that the topology is first defined hierarchically, producing finally a space plan subdivided by layers, where the junctions of at least three separating surfaces are identified and numbered. The simplest data definition is, of course, via the production of planar architectural drawings embedded in a common reference system.

3.2 Building objects: components and assemblies

When the project plan is accepted by the client, giving a definite shape to the first architectural concept, it is usually a 2.5D model, made by opaque or transparent 2D surfaces embedded in 3D space.

In this stage, LAR-ABC allows for the computation of every topological or geometrical property of interest, including the evaluation of the surface of the building envelope and its partitioning into subsets with different thermal requirements, as well as the computation of the internal volume, and its partitioning into any classes of internal space, and will grant any other geometric computation or simulation (for example of the thermal behaviour) of possible interest for the architect or the client.

The LAR description of the topology and its geometric embedding, defined by the position vectors of vertices or control points of the surfaces, makes possible to (mostly) automatically generate a first 3D model of the physical construction, i.e. of the concrete instances of building components.

This (semi-)automatic transformation from a 2.5D model formed by surfaces to a 3D model formed by assemblies of solid objects, is obtained using the boundary operator, that allow to discriminate between the various subsystems of the building fabric, i.e. between the horizontal and vertical enclosures, the horizontal and vertical partitions of the interior, the elements of horizontal and vertical communications, and so on, as we show in Section **??**.

3.3 Construction process: computer simulation

Some general information about the technologies to be used in the construction allows to visualise as a computer animation the construction process embedded in time. Starting to

the specification of the hierarchical assemblies and from some additional information about the precedence relation between the construction activities, a construction PERT, giving the time schedule of the building erection. In the final subsection of Section **??** we discuss a simplified example. Of course, a preliminary but informed guess of quantities and costs ban be also produced from the 3D solid model.

4 Examples

4.1 Housing design

@D First model input @ V = [[3,-3], [9,-3],[0,0],[3,0],[9,0],[15,0], [3,3],[6,3],[9,3],[15,3],[21,3], [0,9],[6,9],[15,9],[18,9],[0,13], [6,13],[9,13],[15,13],[18,10],[21,10], [18,13],[6,16],[9,16],[9,17],[15,17], [18,17],[0,25],[6,25],[15,25]] FV = [[22,23,24,25,29,28], [15,16,22,28,27], [18,21,26,25], [13,14,19,21,18], [16,17,23,22], [11,12,16,15], [9,10,20,19,14,13], [2,3,6,7,12,11], [0,1,4,8,7,6,3], [4,5,9,13,18,17,16,12,7,8],[17,18,25,2 model = [V,FV] @

From LAR model to list of polylines @D From LAR model to list of polylines @def lar2polylines (model): """ From LAR model to list of polylines """ V,FV = model return [[V[v] for v in cell]+[V[cell[0]]] for cell in FV] @

From faces to list of edges @D From faces to list of edges @def face2edge(FV): """ From faces to list of edges """ edges = AA(sorted)(CAT([TRANS([face, face[1:]+[face[0]]]) for face in FV])) return AA(eval)(set(AA(str)(edges))) @

Figure 1: example caption

Concept design @O test/py/architectural/test01.py @@¡ Initial import of modules @¿ from architectural import * @¡ First model input @¿ VIEW(STRUCT(AA(POLYLINE)(lar2polylines (model)))) VIEW(EXPLODE(1.2,1.2,1)(AA(POLYLINE)(lar2polylines (model)))) VIEW(EXPLODE(1.2,1.2, (model))))) EV = face2edge(FV) VIEW(EXPLODE(1.2,1.2,1)(MKPOLS((V,EV)))) @

4.2 Quick BIM

1 column

4.3 Introducing time

1.5 column

5 The computational framework

5.1 ABC: a set of classes

1 column

5.2 Some specialised methods

2 column

5.3 Zero install: working in the browser

0.5 column

5.4 Exporting the library

@O lib/py/architectural.py @@¡ Initial import of modules @¿ @¡ From LAR model to list of polylines @¿ @¡ From faces to list of edges @¿ @

6 Conclusion

6.1 The state of the art

0.5 column

6.2 What next

0.5 column
1.5 column = 10 pages

A Appendix

B Utility functions

Initial import of modules @D Initial import of modules @from pyplasm import * from scipy import * import os,sys """ import modules from larcc/lib """ sys.path.insert(0, 'lib/py/') from lar2psm import * from simplexn import * from larcc import * from largrid import * from mapper import * @