# Draft design of a hospital building *

Alberto Paoluzzi

September 6, 2015

## Contents



**Abstract**

This module follows the concept and the preliminary building program of a hospital of medium size, given in module `hospital`, using as source the document [AM13] of the World Health Organisation.

# 1 The hospital meta-modeling

## 1.1 Project illustration and testing

### Hospital draft design

`"test/py/hospital2/test01.py"` 1 ≡

*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [CL13]. September 6, 2015

```
""" An hospital draft design """
from pyplasm import *

""" import modules from larcc/lib """
sys.path.insert(0, 'lib/py/')
from hospital import metric
from iot3d import polyline2lar
from larstruct import Struct,t,s,struct2lar,evalStruct
from architectural import lar2lines
from lar2psm import MKPOLS,EXPLODE
from mapper import larTranslate,larScale,vcode
from bool import MKTRIANGLES
import largrid
```

⟨ Coding utilities 18b ⟩
⟨ Storey input 14b ⟩
⟨ Structural frame 5b ⟩
⟨ Storey structure 6a ⟩
⟨ Floor visualization 10 ⟩
⟨ Design review 14a ⟩
⟨ Hospital structure 9c ⟩
⟨ Sub-project indexing 2a ⟩
⟨ SVG files printing 5c ⟩
◇

## 1.2 Sub-projects definition and indexing

⟨ Sub-project indexing 2a ⟩ ≡

```
""" Project definitions """
```

⟨ Project 1 definition 2b ⟩
⟨ Project 2 definition 2c ⟩
⟨ Project 3 definition 3a ⟩
⟨ Project 4 definition 3b ⟩
⟨ Project 5 definition 3c ⟩
⟨ Project 6 definition 4a ⟩
⟨ Project 7 definition 4b ⟩
⟨ Project 8 definition 4c ⟩
⟨ Project 9 definition 4d ⟩
⟨ Project 10 definition 5a ⟩
◇

Macro referenced in 1.

### Project 1 definition

⟨ Project 1 definition 2b ⟩ ≡

```
""" Project definition """
Project_1 = Struct([RadioDiagnosticImaging, EmergencyDepartment, Endoscopy,
                StructuralFrame], "Emergency")
Vp1,FVp1,EVp1 = struct2lar(Project_1)
VIEW(STRUCT([SOLIDIFY(STRUCT(MKPOLS((Vp1,EVp1)))), COLOR(CYAN)(STRUCT(MKPOLS((V0,EV0)))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
◇
```

Macro referenced in 2a.

### Project 2 definition

⟨ Project 2 definition 2c ⟩ ≡

```
""" Project definition """
Project_2 = Struct([OutPatientDepartment10, OutPatientDepartment20, RenalDialysis,
                ChemiotherapyUnit,PhysicalMedicineDept, StructuralFrame], "OutPatient")
Vp2,FVp2,EVp2 = struct2lar(Project_2)
VIEW(STRUCT([SOLIDIFY(STRUCT(MKPOLS((Vp2,EVp2)))), COLOR(CYAN)(STRUCT(MKPOLS((V0,EV0)))),
COLOR(RED)(STRUCT(MKPOLS((Vp,EVp)))) ]))
◇
```

Macro referenced in 2a.

### Project 3 definition

⟨ Project 3 definition 3a ⟩ ≡

```
""" Project definition """
Project_3 = Struct([MainEntrance, MedicalWaste, StructuralFrame], "InputOutput")
Vp3,FVp3,EVp3 = struct2lar(Project_3)
VIEW(STRUCT([SOLIDIFY(STRUCT(MKPOLS((Vp3,EVp3)))), COLOR(CYAN)(STRUCT(MKPOLS((V0,EV0)))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
◇
```

Macro referenced in 2a.

### Project 4 definition

⟨ Project 4 definition 3b ⟩ ≡

```
""" Project definition """
Project_4 = Struct([CentralStores, StaffDining, CSSD, HouseKeeping,
      CentralStaffChanging11, CentralStaffChanging21, StructuralFrame], "StaffServices")
Vp4,FVp4,EVp4 = struct2lar(Project_4)
VIEW(STRUCT([ COLOR(MAGENTA)(STRUCT(MKPOLS((Vm,EVm)))), STRUCT(MKPOLS((Vp4,EVp4))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
◇
```

Macro referenced in 2a.

### Project 5 definition

⟨ Project 5 definition 3c ⟩ ≡

```
""" Project definition """
Project_5 = Struct([Pharmacy, CentralWorkshop, Laundry, StructuralFrame], "PatientServices")
Vp5,FVp5,EVp5 = struct2lar(Project_5)
VIEW(STRUCT([SOLIDIFY(STRUCT(MKPOLS((Vp5,EVp5)))), COLOR(MAGENTA)(STRUCT(MKPOLS((Vm,EVm)))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

### Project 6 definition

⟨ Project 6 definition 4a ⟩ ≡

```
""" Project definition """
Project_6 = Struct([MainLaboratories, StructuralFrame], "Laboratories")
Vp6,FVp6,EVp6 = struct2lar(Project_6)
VIEW(STRUCT([SOLIDIFY(STRUCT(MKPOLS((Vp6,EVp6)))), COLOR(MAGENTA)(STRUCT(MKPOLS((Vm,EVm)))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

### Project 7 definition

⟨ Project 7 definition 4b ⟩ ≡

```
""" Project definition """
Project_7 = Struct([AdministrationSuite11, MedicalLibrary, MedicalRecords,
          AdministrationSuite21, MeetingRooms, DataCenter, ServerRoom, StructuralFrame],
          "Administration")
Vp7,FVp7,EVp7 = struct2lar(Project_7)
VIEW(STRUCT([ COLOR(MAGENTA)(STRUCT(MKPOLS((Vm,EVm)))),STRUCT(MKPOLS((Vp7,EVp7))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

### Project 8 definition

⟨ Project 8 definition 4c ⟩ ≡

```
""" Project definition """
Project_8 = Struct([Surgery, CatheterizationLab, CoronaryCareUnit, StructuralFrame],
                "Surgery")
Vp8,FVp8,EVp8 = struct2lar(Project_8)
VIEW(STRUCT([ STRUCT([SOLIDIFY(STRUCT(MKPOLS((V,EV)))) for V,FV,EV in evalStruct(Project_8) ])
   COLOR(ORANGE)(STRUCT(MKPOLS((V1,EV1)))), COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

### Project 9 definition

⟨ Project 9 definition 4d ⟩ ≡

```
""" Project definition """
Project_9 = Struct([DeliveryAndNicu, IntensiveCareUnit, StructuralFrame], "Delivery")
Vp9,FVp9,EVp9 = struct2lar(Project_9)
VIEW(STRUCT([ STRUCT([SOLIDIFY(STRUCT(MKPOLS((V,EV)))) for V,FV,EV in evalStruct(Project_9)]),
   COLOR(ORANGE)(STRUCT(MKPOLS((V1,EV1)))), COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

### Project 10 definition

⟨ Project 10 definition 5a ⟩ ≡

```
""" Project definition """
Project_10a = Struct([SurgicalWard1, StructuralFrame], "floor2Ward")
Project_10b = Struct([GeneralWard1, SurgicalWard2, StructuralFrame], "floor3Ward")
Project_10c = Struct([PediatricWard1, PediatricWard2, StructuralFrame], "floor4Ward")
Project_10d = Struct([GeneralWard2,GeneralWard3, StructuralFrame], "floor5Ward")
Vp10a,FVp10a,EVp10a = struct2lar(Project_10a)
Vp10b,FVp10b,EVp10b = struct2lar(Project_10b)
Vp10c,FVp10c,EVp10c = struct2lar(Project_10c)
Vp10d,FVp10d,EVp10d = struct2lar(Project_10d)
VIEW(STRUCT([COLOR(YELLOW)(STRUCT(MKPOLS((V2,EV2)))), STRUCT(MKPOLS((Vp10a,EVp10a))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
VIEW(STRUCT([COLOR(YELLOW)(STRUCT(MKPOLS((V3,EV3)))), STRUCT(MKPOLS((Vp10b,EVp10b))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
VIEW(STRUCT([COLOR(YELLOW)(STRUCT(MKPOLS((V4,EV4)))), STRUCT(MKPOLS((Vp10c,EVp10d))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
VIEW(STRUCT([COLOR(YELLOW)(STRUCT(MKPOLS((V5,EV5)))), STRUCT(MKPOLS((Vp10d,EVp10c))),
COLOR(RED)(STRUCT(MKPOLS((metric(Vs),EVs)))) ]))
```

◇

Macro referenced in 2a.

⟨ Structural frame 5b ⟩ ≡

```
""" Structural frame """
Xs = range(11); Ys = range(15)
gridPoints = set(AA(tuple)(CART([Xs, Ys]))).difference(AA(tuple)(CART([[4,5,6],[5,6,7]])+[[3,1
Vp,[_,EVp,FVp] = largrid.larCuboids([1,1],True)
Vp = larTranslate([-1./40,-1./40])(larScale([1./20,1./20])(Vp))
Pillar = Struct([(Vp,FVp,EVp)],"Pillar")
structuralFrame = Struct( [Struct([t(*point),Pillar]) for point in gridPoints], "StructuralFra
Vs,FVs,EVs = struct2lar(structuralFrame)
StructuralFrame = Struct( [(metric(Vs),FVs,EVs)], "StructuralFrame" )
VIEW(STRUCT(MKPOLS((metric(Vs),EVs))))
```

◇

Macro referenced in 1.

⟨ SVG files printing 5c ⟩ ≡

```
""" SVG files printing """
def printProject(path,struct):
    filename = struct.__name__()
    theFile = open(path+filename+".svg", "w")
    print >> theFile, '<?xml version="1.0" encoding="utf-8"?>'
    print >> theFile, '<!-- Generator: Adobe Illustrator 16.0.0, SVG Export Plug-In .'+ \
    ' SVG Version: 6.00 Build 0)  -->'
    print >> theFile, '<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" '+ \
    '"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">'
    print >> theFile, '<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" '+ \
    ' xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" width="595.28px" '+ \
    ' height="841.89px" viewBox="0 0 59.528 84.189" '+ \
    'enable-background="new 0 0 59.528 84.189" xml:space="preserve">'

    V,FV,EV = struct2lar(struct)
    for v1,v2 in EV:
        [x1,y1],[x2,y2] = V[v1],V[v2]
        print >> theFile, '<line fill="none" stroke="#000000" stroke-miterlimit="10" '+ \
        'x1="'+str(x1)+ '" y1="'+str(y1)+'" x2="'+str(x2)+'" y2="'+str(y2)+'"/>'

    print >> theFile, '</svg>'
    theFile.close()

Projects = [Project_1, Project_2, Project_3, Project_4, Project_5, Project_6, Project_7,
         Project_8, Project_9, Project_10a, Project_10b, Project_10c, Project_10d ]

for project in Projects:
    printProject("./",project)
```

◇

Macro referenced in 1.

## 2 Hospital structure

⟨ Storey structure 6a ⟩ ≡

```
""" Storey structure """
```

⟨ Ground floor structure 6b ⟩
⟨ Mezanine floor structure 7a ⟩
⟨ First floor structure 7b ⟩
⟨ Second floor structure 8a ⟩
⟨ Third floor structure 8b ⟩
⟨ Fourth floor structure 9a ⟩
⟨ Fifth floor structure 9b ⟩

◇

Macro referenced in 1.

### Ground floor structure

⟨ Ground floor structure 6b ⟩ ≡

```
"""Ground floor """
Ground = [OpenCourt10, RadioDiagnosticImaging,
    ServiceCore10, ServiceCore20,  EmergencyDepartment, Endoscopy,
    OutPatientDepartment10, OutPatientDepartment20, RenalDialysis,
    OpenCourt20, ChemiotherapyUnit, Service, PhysicalMedicineDept,
    MainEntrance, Unknown, Corridor0, Corridor0a, Corridor0b]

Ground_names = ["OpenCourt10", "RadioDiagnosticImaging",
    "ServiceCore10", "ServiceCore20", "EmergencyDepartment", "Endoscopy",
    "OutPatientDepartment10", "OutPatientDepartment20", "RenalDialysis",
    "OpenCourt20", "ChemiotherapyUnit", "Service", "PhysicalMedicineDept",
    "MainEntrance", "Unknown", "Corridor0", "Corridor0a", "Corridor0b"]

for struct,name in zip(Ground,Ground_names): struct.set_name(name)
Ground_floor = Struct(Ground, "Ground_floor", "level")
◇
```

Macro referenced in 6a.

### Mezanine floor structure

⟨ Mezanine floor structure 7a ⟩ ≡

```
"""Mezanine floor """
Mezanine = [MedicalWaste, CentralStores,
    StaffDining, CSSD, HouseKeeping,  CentralStaffChanging11,
    CentralStaffChanging21, Pharmacy, CentralWorkshop, Laundry,
    AdministrationSuite11, MainLaboratories, MedicalLibrary, MedicalRecords,
    AdministrationSuite21, MeetingRooms, DataCenter, ServerRoom, PublicCore,
    ServiceCore11, ServiceCore21, Corridor1, GroundRoof]

Mezanine_names = ["MedicalWaste", "CentralStores",
    "StaffDining", "CSSD", "HouseKeeping", "CentralStaffChanging11",
    "CentralStaffChanging21", "Pharmacy", "CentralWorkshop", "Laundry",
    "AdministrationSuite11", "MainLaboratories", "MedicalLibrary", "MedicalRecords",
    "AdministrationSuite21", "MeetingRooms", "DataCenter", "ServerRoom", "PublicCore",
    "ServiceCore11", "ServiceCore21", "Corridor1", "GroundRoof"]

for struct,name in zip(Mezanine,Mezanine_names): struct.set_name(name)
Mezanine_floor = Struct(Mezanine, "Mezanine_floor", "level")
◇
```

Macro referenced in 6a.

### First floor structure

⟨ First floor structure 7b ⟩ ≡

```
"""First floor """
First = [Surgery, CatheterizationLab,
    ServiceCore32,  CoronaryCareUnit, DeliveryAndNicu, ServiceCore31,
    IntensiveCareUnit, ServiceCore33, PublicCore3, Corridor3, MezanineRoof]

First_names = ["Surgery", "CatheterizationLab",
    "ServiceCore32", "CoronaryCareUnit", "DeliveryAndNicu", "ServiceCore31",
    "IntensiveCareUnit", "ServiceCore33", "PublicCore3", "Corridor3", "MezanineRoof"]

for struct,name in zip(First,First_names): struct.set_name(name)
First_floor = Struct(First, "First_floor", "level")
◇
```

Macro referenced in 6a.

### Second floor structure

⟨ Second floor structure 8a ⟩ ≡

```
"""Second floor """
Second = [ ObstetricGinecologicWard, SurgicalWard1,
    PublicCore4, Filter1, Filter2,
    ServiceCore14, ServiceCore24, FirstRoof,  Corridor4a, Corridor4b,
    Corridor4b1, Corridor4b2, Corridor4c, Corridor4c1, Corridor4c2]

Second_names = [ "ObstetricGinecologicWard", "SurgicalWard1",
    "PublicCore4", "Filter1", "Filter2",
    "ServiceCore14", "ServiceCore24", "FirstRoof", "Corridor4a", "Corridor4b",
    "Corridor4b1", "Corridor4b2", "Corridor4c", "Corridor4c1", "Corridor4c2"]

for struct,name in zip(Second,Second_names): struct.set_name(name)
Second_floor = Struct(Second, "Second_floor", "level")
◇
```

Macro referenced in 6a.

### Third floor structure

⟨ Third floor structure 8b ⟩ ≡

```
"""Third floor """
Third = [GeneralWard1, SurgicalWard2, PublicCore4, ServiceCore14, ServiceCore24,
    Filter1, Filter2, Corridor4a, Corridor4b, Corridor4b1, Corridor4b2, Corridor4c,
    Corridor4c1, Corridor4c2]

Third_names = [ "GeneralWard1", "SurgicalWard2", "PublicCore4", "ServiceCore14",
```

```
    "ServiceCore24", "Filter1", "Filter2", "Corridor4a", "Corridor4b", "Corridor4b1",
    "Corridor4b2", "Corridor4c", "Corridor4c1", "Corridor4c2"]

for struct,name in zip(Third,Third_names): struct.set_name(name)
Third_floor = Struct(Third, "Third_floor", "level")
◇
```

Macro referenced in 6a.

### Fourth floor structure

⟨ Fourth floor structure 9a ⟩ ≡

```
"""Fourth floor """
Fourth = [PediatricWard1, PediatricWard2, PublicCore4, ServiceCore14, ServiceCore24,
    Filter1, Filter2, Corridor4a, Corridor4b, Corridor4b1, Corridor4b2, Corridor4c,
    Corridor4c1, Corridor4c2]

Fourth_names = [ "PediatricWard1", "PediatricWard2", "PublicCore4", "ServiceCore14",
   "ServiceCore24", "Filter1", "Filter2", "Corridor4a", "Corridor4b", "Corridor4b1",
   "Corridor4b2", "Corridor4c", "Corridor4c1", "Corridor4c2"]

for struct,name in zip(Fourth,Fourth_names): struct.set_name(name)
Fourth_floor = Struct(Fourth, "Fourth_floor", "level")
◇
```

Macro referenced in 6a.

### Fifth floor structure

⟨ Fifth floor structure 9b ⟩ ≡

```
"""Fifth floor """
Fifth = [GeneralWard2, GeneralWard3, PublicCore4, ServiceCore14, ServiceCore24,
    Filter1, Filter2, Corridor4a, Corridor4b, Corridor4b1, Corridor4b2, Corridor4c,
    Corridor4c1, Corridor4c2]

Fifth_names = [ "GeneralWard2", "GeneralWard3", "PublicCore4", "ServiceCore14",
   "ServiceCore24", "Filter1", "Filter2", "Corridor4a", "Corridor4b", "Corridor4b1",
   "Corridor4b2", "Corridor4c", "Corridor4c1", "Corridor4c2"]

for struct,name in zip(Fifth,Fifth_names): struct.set_name(name)
Fifth_floor = Struct(Fifth, "Fifth_floor", "level")
◇
```

Macro referenced in 6a.

### Hospital structure

⟨ Hospital structure 9c ⟩ ≡

```
"""Hospital structure """
from larstruct import embedStruct, larApply
floors = [ Ground_floor, Mezanine_floor, First_floor,
    Second_floor, Third_floor, Fourth_floor, Fifth_floor ]

Floors = AA(embedStruct(1))(floors)

Floor_names = ["Ground_floor", "Mezanine_floor", "First_floor",
               "Second_floor", "Third_floor", "Fourth_floor", "Fifth_floor"]

for struct,name in zip(Floors, Floor_names): struct.set_name(name)

Hospital = Struct( CAT(TRANS([Floors, 7*[t(0,0,4)]])), "General_Hospital", "building")

print "\nstructCochain(0)(Ground_floor) =",structCochain(0)(Ground_floor),"\n"
print "structCochain(1)(Ground_floor) =",structCochain(1)(Ground_floor),"\n"
print "structCochain(2)(Ground_floor) =",structCochain(2)(Ground_floor),"\n"

W,WF,WE = struct2lar(Hospital)
lone = struct2lar( Struct( AA(embedStruct(1))([OpenCourt10, OpenCourt20]) ))
Z,FZ,EZ = struct2lar( embedStruct(1)(Fifth_floor) )
ZZ = AA(SUM)(DISTR([Z,[0,0,6*4]]))

VIEW(STRUCT( MKTRIANGLES(W,WF,WE) + AA(COLOR(CYAN))(MKTRIANGLES(ZZ,FZ,EZ)) + [T(3)(.2)] + [COL
◇
```

Macro referenced in 1.

⟨ Floor visualization 10 ⟩ ≡

```
"""Floor visualization """
V0,FV0,EV0 = struct2lar(Ground_floor)
Vm,FVm,EVm = struct2lar(Mezanine_floor)
V1,FV1,EV1 = struct2lar(First_floor)
V2,FV2,EV2 = struct2lar(Second_floor)
V3,FV3,EV3 = struct2lar(Third_floor)
V4,FV4,EV4 = struct2lar(Fourth_floor)
V5,FV5,EV5 = struct2lar(Fifth_floor)

VIEW(STRUCT(MKPOLS((V0,EV0))))
VIEW(STRUCT(MKPOLS((Vm,EVm))))
VIEW(STRUCT(MKPOLS((V1,EV1))))
VIEW(STRUCT(MKPOLS((V2,EV2))))
VIEW(STRUCT(MKPOLS((V3,EV3))))
VIEW(STRUCT(MKPOLS((V4,EV4))))
```

```
VIEW(STRUCT(MKPOLS((V5,EV5))))
```
◇

Macro referenced in 1.

### 2.1 Data sources

The starting point of the modelling developed here is the paper [AM13], about Hospital Planning and Design, downloadable from here, and in particular the two images shown in Figure 1 and relative to the functional zoning of floors, and providing an axonometric view of the vertical organisation of the hospital.

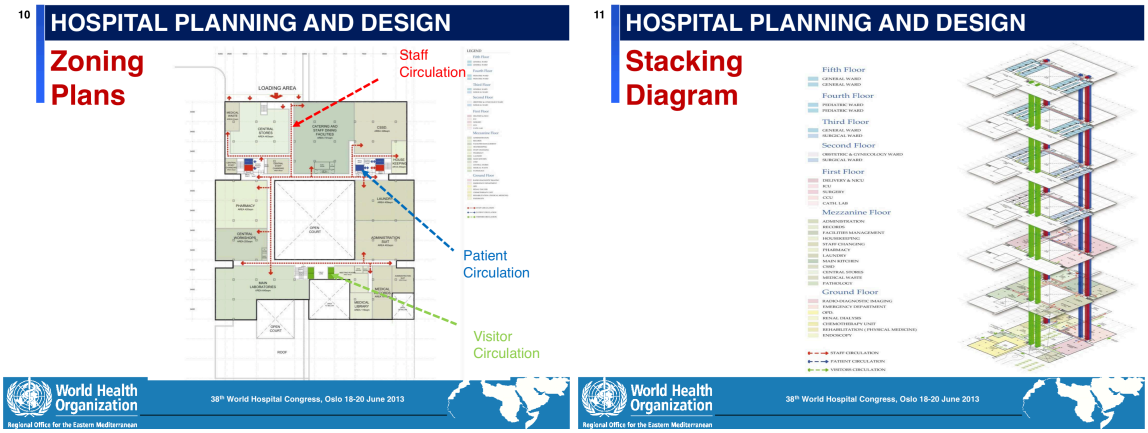


Figure 1: Two images of the example model for hospital planning and design used in this module: (a) functional zoning of the mezzanine floor; (b) axonometric view of the vertical design organisation.

## 3 Design review

The design review is here intended as the computation of the discrete fields of interest, above the space decomposition generated in the previous design steps.

As stated in the beginning of this document, the design processing proceeds by step-wise refinement of topological (and geometrical) constraints on the planned spaces, aimed to satisfy a set of design requirements. In case of non-satisfaction, some design changes are introduced, producing a different space decomposition, and repeating the tests.

The discrete field values within the cells of the decomposition provide an evaluated *cochain* over a chain (subset of cells) defined above the design decomposition. Both the chain of interest, and the cochain evaluated over it may vary, depending on the design task at hand. In any case the *evaluation* of a cochain $\phi$ above a chain $\rho$, denoted as either $\phi(\rho)$

or the pairing $\langle \phi, \rho \rangle$, will produce a field value, in our case a number, corresponding to the integral of the field (discrete differential form) above the discretized integration domain (set of cells).

### 3.1 Integration and cochains computation

Very often, the discrete field value associated to every cell of a cellular decomposition results from the integration of some differential $k$-form on the given $k$-cell. In general, a $k$-form can be described as an entity to be integrated on a $k$-dimensional (sub)region [DKT05]. For this purpose, we are going to (a) approximate the given form (function $\mathbb{R}^k \to \mathbb{R}$) with a multivariate polynomial; (b) use a finite integration method for $k$-variate monomials in 2D or 3D Euclidean space [CP90]. The result is a (hierarchical) mapping between the (hierarchy of) cells of the decomposition, and the integral of an approximating polynomial on the cells.

**Computing a surface cochain via Struct traversal** The function `structCochain`, given in the script below, returns the hierarchical (surface) cochain corresponding to the given `depth` of the `struct` input. In particular, every value in the output dictionary `cochain` returns the evaluated (i.e. summed) cochain associated to the `struct` subgraph rooted in every node at `depth` distance from the root node of `struct`. Just notice that the keys of the output `cochain` dictionary result from the *string-join* of the names of the `struct` nodes along the *current* path from the root to the node (at level `depth`) associated with the key.

⟨ Computing a surface cochain via Struct traversal 12 ⟩ ≡

```
""" Computing a surface cochain via Struct traversal """
from collections import defaultdict
from larstruct import checkStruct
import scipy
from larstruct import Model
from integr import surfIntegration

⟨ Traversing a hierarchical surface cochain 13 ⟩

def structCochain(depth=1):
    def structCochain0(struct):
        cochain = defaultdict(int)
        dim = checkStruct(struct.body)
        CTM, stack = scipy.identity(dim+1), []
        cochainMap = structCochainTraversal(CTM, stack, struct, [], [], [])
        print cochainMap
        cochainMap = [(key,[sum(value)]) for key,value in cochainMap]
        for cell,cochainValue in cochainMap:
```

```
            nameArray = cell.split(".")
            cochain[".".join(nameArray[:depth])] += cochainValue[0]
        return cochain
    return structCochain0
```

⟨ Example of hierarchical surface cochains ? ⟩
◇

Macro referenced in 14a.

**Traversing a hierarchical surface cochain** The `structCochainTraversal` function given below executes a standard traversal of a hierarchical structure, consisting in relocating all encountered objects from local coordinates to the root coordinates.

While executing the traversal, a set of pairs (corresponding to each traversed node) is accumulated in the `cochainMap` list, initially empty. Every such pair contains the joined names of nodes along the current path, and the surface integral evaluated on the traversed node, cast to an integer value.

Notice that if a `struct` node contains more than one instance of the same son, then the names of such instances are joined with the counter value associated to the son's `name` within a dictionary `repeatedNames`, in order to make individually identifiable the various object instances.

⟨ Traversing a hierarchical surface cochain 13 ⟩ ≡

```
""" Traversing a hierarchical surface cochain """
from larstruct import Mat, Model
def structCochainTraversal(CTM, stack, obj, cochainMap=[], names=[], nameStack=[]):
    repeatedNames = defaultdict(int)

    def map(model):
        V,FV,EV = larApply(CTM)(model)
        print "eccomi!"
        print "V,FV,EV =",V,FV,EV
        return AA(int)(surfIntegration((V,FV,EV)))

    for i in range(len(obj)):
        if isinstance(obj[i],Struct):
            repeatedNames[obj[i].name] += 1
            if repeatedNames[obj[i].name]==1: theName = obj[i].name
            else: theName = obj[i].name + str(repeatedNames[obj[i].name]-1)
            names.append(theName)
            nameStack = nameStack+[names]

            stack.append(CTM)
            structCochainTraversal(CTM, stack, obj[i], cochainMap, names, nameStack)
            CTM = stack.pop()
```

```
                theName = names.pop()

            elif isinstance(obj[i],Model):
                cochainMap += [( ".".join(names), map(obj[i]) )]
            elif (isinstance(obj[i],tuple) or isinstance(obj[i],list)) and (
                  len(obj[i])==2 or len(obj[i])==3):
                cochainMap += [( ".".join(names), map(obj[i]) )]
            elif isinstance(obj[i],Mat):
                CTM = scipy.dot(CTM, obj[i])
        return cochainMap
```
◇

Macro referenced in 12, 14a.

### Design review

⟨ Design review 14a ⟩ ≡
```
"""  Surface cochain review """
```
⟨ Computing a surface cochain via Struct traversal 12 ⟩
⟨ Traversing a hierarchical surface cochain 13 ⟩
◇

Macro referenced in 1.

## 4 Model input

⟨ Storey input 14b ⟩ ≡
```
"""  Storey input """
```
⟨ Ground floor 14c ⟩
⟨ Mezanine floor 15a ⟩
⟨ First floor 15b ⟩
⟨ Second floor 16 ⟩
⟨ Third floor 17a ⟩
⟨ Fourth floor 17b ⟩
⟨ Fifth floor 17c ⟩
◇

Macro referenced in 1.

### Ground floor input

⟨ Ground floor 14c ⟩ ≡
```
"""  Ground floor """
OpenCourt10 = mpoly2struct([TRANS([[3,3,4,4,6,6,6.65,6.65],[4,8,8,7.8,7.8,8,8,4]])])
RadioDiagnosticImaging = mpoly2struct([TRANS([[7,7,9,10,10,8.7],[4,8,8,8,4,4]])])
ServiceCore10 = mpoly2struct([TRANS([[1.15, 1.15, 1.3,2.55, 2.55,2], [2.85, 3.7,3.7,3.7, 2.85,
ServiceCore20 = mpoly2struct([TRANS([[7,7,8.7,8.8,8.8],[2.8,3.7,3.7,3.7,2.8]])])
```

```
EmergencyDepartment = mpoly2struct([TRANS([[4.7,4.7,7,7,8.8,8.8,9.65,9.65],[0,3.7,3.7, 2.8,2.8
Endoscopy = mpoly2struct([TRANS([[3,3,3,4.4,4.4],[0,2.5,3.7,3.7,0]])])
OutPatientDepartment10 = mpoly2struct([TRANS([[4./7.5, 4./7.5,1.15,1.15,2,2,3,3], [0,3.7,3.7,2
OutPatientDepartment20 = mpoly2struct([TRANS([[0,0,2.65,2.65,1.3],[4,5.85,5.85,4,4]])])
RenalDialysis = mpoly2struct([TRANS([[0,0,1,2.65,2.65],[5.85,8,8,8,5.85]])])
OpenCourt20 = mpoly2struct([TRANS([[2,2,2,2,4,4,4,4],[10,11,11.35,12,12,11.35,11,10]])])
ChemiotherapyUnit = mpoly2struct([TRANS([[0,0,4.5,4.5,4,4,2,2,1], [11.35,14,14,11.35,11.35,12,
Service = mpoly2struct([TRANS([[0,0,1,1,2,2,2,1],[8.35,10,10,9,9,8.5, 8.35,8.35]])])
PhysicalMedicineDept = mpoly2struct([TRANS([[2,2,1,1,0,0, 1,2,2,4,4,4.5,4.5,4,4], [8.5,9,9,10,
MainEntrance = mpoly2struct([TRANS([[4,4,4,4.5,4.75,4.75,6.65,6.65,6,6], [8.4,8.5,9,9,9,11,11,
Unknown = mpoly2struct([TRANS([[7.25,7.25, 6.65,6.65,6.65,10,10,9,8.2], [8.35,8.5,8.5,9,11,11,8
#Mortuary = mpoly2struct([TRANS([[],[]])])
Corridor0 = mpoly2struct([[[4.4,0],[4.4,3.7],[3,3.7],[3,2.5],[2,2.5],[2,2.85],[2.55,2.85], [2.5
Corridor0a = mpoly2struct([TRANS([[1, 1, 2, 2], [11, 11.35, 11.35, 11]])])
Corridor0b = mpoly2struct([TRANS([[4.5, 4.5, 4, 4, 4.5, 4.5, 4.75,4.75, 4.75], [9, 11, 11, 11.3
◇
```

Macro referenced in 14b.

### Mezanine floor input

⟨ Mezanine floor 15a ⟩ ≡

```
""" Mezanine floor """
MedicalWaste = mpoly2struct([TRANS([[4./7.5,4./7.5,.8,1.25,1.25],[0,1.5,1.5,1.5,0]])])
CentralStores = mpoly2struct([TRANS([[1.25,1.25,.8,.8,3.7,3.7,2.55,2.55,2.2,2.2],[0,1.5,1.5, 2
StaffDining = mpoly2struct([TRANS([[3.95,3.95,6.7,6.7,6.95,6.95],[0,3.7,3.7,2,2,0]])])
CSSD = mpoly2struct([TRANS([[6.95,6.95,6.95,8.8,8.8,9.65,9.65],[0,2,2.65,2.65,2,2,0]])])
HouseKeeping = mpoly2struct([TRANS([[8.8,8.8,8.8,8.8,8.8,9.65,9.65],[2,2.65,2.8,3.7,3.7,2]])])
CentralStaffChanging11 = mpoly2struct([TRANS([[4./7.5,4./7.5,1.15,1.15],[2.85,3.7,3.7,2.85]])])
CentralStaffChanging21 = mpoly2struct([TRANS([[2.55,2.55,3.7,3.7],[2.85,3.7,3.7,2.85]])])
OpenCourt11 = mpoly2struct([TRANS([[3,3,7,7,7],[4,8,8,6,4]])])
Pharmacy = mpoly2struct([TRANS([[0,0,2.65,2.65,1.3],[4,6.45,6.45,4,4]])])
CentralWorkshop = mpoly2struct([TRANS([[0,0,1,2.65,2.65],[6.45,8,8,8,6.45]])])
Laundry = mpoly2struct([TRANS([[7,7,10,10,8.7],[4,6,6,4,4]])])
AdministrationSuite11 = mpoly2struct([TRANS([[7,7,9,10,10],[6,8,8,8,6]])])
MainLaboratories = mpoly2struct([TRANS([[1,1,0,0,2,2,5,5,4,4,4],[8.3,8.4,8.4,11,11,10,10,9, 9,8
MedicalLibrary = mpoly2struct([TRANS([[6.7,6.7,8,8,7.75],[9.7,11,11,9.7,9.7]])])
MedicalRecords = mpoly2struct([TRANS([[8,8,8,8.85,8.85,8.85],[8.3,9.7,11,11,9.75,8.3]])])
AdministrationSuite21 = mpoly2struct([TRANS([[8.85,8.85,10,10,9,9],[8.3,9.75,9.75,8.4,8.4,8.3]
MeetingRooms = mpoly2struct([TRANS([[6,6,6,6.7,6.7,7.75,7.75,7.45,7,7],[8.3,8.4,9,9,9.7,9.7, 8
DataCenter = mpoly2struct([TRANS([[7,7,7.45,7.45],[8.3,8.7,8.7,8.3]])])
ServerRoom = mpoly2struct([TRANS([[7.45,7.45,7.75,7.75],[8.3,8.7,8.7,8.3]])])
PublicCore = mpoly2struct([TRANS([[4,4,5,6,6],[8.4,9,9,9,8.4]])])
ServiceCore11 = mpoly2struct([TRANS([[1.15,1.15,1.3,2.55,2.55],[2.85,3.7,3.7,3.7,2.85]])])
ServiceCore21 = mpoly2struct([TRANS([[7,7,8.7,8.8,8.8],[2.8,3.7,3.7,3.7,2.8]])])
Corridor1 =mpoly2struct([[[2.2,0],[2.2,0.65],[2.55,0.65],[2.55,0.35],[3.7,0.35],[3.7,2.65],
[0.8,2.65],[0.8,1.5],[0.5333,1.5],[0.5333,2.85],[1.15,2.85],[2.55,2.85],[3.7,
```

```
2.85],[3.7,3.7],[2.55,3.7],[1.3,3.7],[1.3,4],[2.65,4],[2.65,6.45],[2.65,
8],[1,8],[1,8.3],[4,8.3],[4,8.4],[6,8.4],[6,8.3],[7,8.3],[7.45,8.3],
[7.75,8.3],[7.75,8.7],[7.75,9.7],[8,9.7],[8,8.3],[8.85,8.3],[9,8.3],[9,8],
[7,8],[3,8],[3,4],[7,4],[8.7,4],[8.7,3.7],[7,3.7],[7,2.8],[8.8,2.8],
[8.8,2.65],[6.95,2.65],[6.95,2],[6.7,2],[6.7,3.7],[3.95,3.7],[3.95,0]]])
GroundRoof = mpoly2struct([TRANS([[4,4,2,2,1,1,0,0,4.75,4.75],[10,12,12,11,11,11.35,11.35,14,
◇
```

Macro referenced in 14b.

### First floor input

⟨ First floor 15b ⟩ ≡

```
""" First floor """
OpenCourt3 = mpoly2struct([TRANS([[3.,3.,7.,7.],[4.,8.,8.,4.]])])
Surgery = mpoly2struct([TRANS([[4.15,4.15,7.,7.,8.8,8.8,9.65,9.65],[0,3.7,3.7, 2.8,2.8, 3.7,3.
CatheterizationLab = mpoly2struct([TRANS([[3,3,4.15,4.15],[0,3.7,3.7,0]])])
ServiceCore32 = mpoly2struct([TRANS([[7.,7.,8.7,8.8,8.8],[2.8,3.7,3.7,3.7,2.8]])])
CoronaryCareUnit = mpoly2struct([TRANS([[7.,7.,8.3,9.,10.,10.,8.7],[4.,8.,8.,8.,8.,4.,4.]])])
DeliveryAndNicu = mpoly2struct([TRANS([[0,0, 1.7,2.65,2.65,1.3],[4.,8.,8.,8.,4.,4.]])])
ServiceCore31 = mpoly2struct([TRANS([[1.15, 1.15, 1.3,2.65, 2.65], [2.85, 3.7,3.7, 3.7, 2.85]]
IntensiveCareUnit = mpoly2struct([TRANS([[4./7.5, 4./7.5,1.15,1.15,2.65, 2.65,1.95,1.95], [0.,
ServiceCore33 = mpoly2struct([TRANS([[1.95,1.95,2.65, 2.65],[0,.6,.6,0]])])
PublicCore3 = mpoly2struct([TRANS([[1.7,1.7,4.,4.,6.,6.,8.3,8.3,7,3,2.65], [8,8.4,8.4,9,9,8.4,8
Corridor3 = mpoly2struct([TRANS([[2.65,2.65,2.65,2.65,1.3,1.3,2.65,2.65,3.0,3.0,7.0,8.7,8.7, 7
MezanineRoof = mpoly2struct([TRANS([[1,1,0,0,2,2,4.75,4.75,10,10,9,9,8.3,8.3, 6,6,4,4 ,1.7,1.7]
◇
```

Macro referenced in 14b.

### Second floor input

⟨ Second floor 16 ⟩ ≡

⟨ Ward sections 18a ⟩

```
""" Second floor """
PublicCore4 = mpoly2struct([TRANS([[1.7,1.7,4,4,6,6,8.3,8.3, 8,7+2./3, 7, 3, 2+1./3,2], [8,8.4
Filter1 = mpoly2struct([TRANS([[1,1,1.35,1.35,1.15],[3.7,4,4,3.7,3.7]])])
Filter2 = mpoly2struct([TRANS([[8.65,8.65,9,9,8.8],[3.7,4,4,3.7,3.7]])])
ServiceCore14 = mpoly2struct([TRANS([[1.15, 1.15, 1.35,2.55, 2.55], [2.8, 3.7,3.7, 3.7, 2.8]])]
ServiceCore24 = mpoly2struct([TRANS([[7,7,8.65,8.8,8.8],[2.8,3.7,3.7,3.7,2.8]])])
FirstRoof = mpoly2struct([TRANS([[4./7.5, 4./7.5,1.15,1.15,2.55,2.55,7,7,8.8,8.8,9.65,9.65], [
Corridor4a = mpoly2struct([[[1.35,3.7],[1.35,4],[2,4],[2.3333,4],[3,4],[7,4],[7.6667,4],[8,4],
Corridor4b = mpoly2struct([[[1,4.0],[1,4.25],[1,4.5],[1,4.75],[1,5.0],[1,5.25],[1,5.5], [1,5.75
Corridor4b1 = mpoly2struct([[[1.3,4.3],[1.3,4.6],[1.3,4.9],[1.3,5.3],[1.3,5.7],[1.5,5.7],[1.7,5
Corridor4b2 = mpoly2struct([[[1.3,6.3],[1.3,6.7],[1.3,7.1],[1.3,7.4],[1.3,7.7],[1.7,7.7],[1.7,7
```

```
Corridor4c = mpoly2struct([[[8,4.0],[8,4.25],[8,4.5],[8,4.75],[8,5.0],[8,5.25],[8,5.5], [8,5.75
Corridor4c1 = mpoly2struct([[[8.3,4.3],[8.3,4.6],[8.3,4.9],[8.3,5.3],[8.3,5.7],[8.5,5.7],[8.7,5
Corridor4c2 = mpoly2struct([[[8.3,6.3],[8.3,6.7],[8.3,7.1],[8.3,7.4],[8.3,7.7],[8.7,7.7],[8.7,7

ObstetricGinecologicWard = Struct([t(0,4), Ward],'ObstetricGinecologicWard')
SurgicalWard1 = Struct([t(7,4), Ward],'SurgicalWard1')

V,FV,EV = struct2lar(ObstetricGinecologicWard)
ObstetricGinecologicWard = Struct( [(metric(V),FV,EV)], "ObstetricGinecologicWard" )
V,FV,EV = struct2lar(SurgicalWard1)
SurgicalWard1 = Struct( [(metric(V),FV,EV)], "SurgicalWard1" )
◇
```

Macro referenced in 14b.

### Third floor input

⟨ Third floor 17a ⟩ ≡

```
""" Third floor """
GeneralWard1 = Struct([t(0,4), Ward])
SurgicalWard2 = Struct([t(7,4), Ward])

V,FV,EV = struct2lar(GeneralWard1)
GeneralWard1 = Struct( [(metric(V),FV,EV)], "GeneralWard1" )
V,FV,EV = struct2lar(SurgicalWard2)
SurgicalWard2 = Struct( [(metric(V),FV,EV)], "SurgicalWard2" )
◇
```

Macro referenced in 14b.

### Fourth floor input

⟨ Fourth floor 17b ⟩ ≡

```
""" Fourth floor """
PediatricWard1 = Struct([t(0,4), Ward])
PediatricWard2 = Struct([t(7,4), Ward])

V,FV,EV = struct2lar(PediatricWard1)
PediatricWard1 = Struct( [(metric(V),FV,EV)], "PediatricWard1" )
V,FV,EV = struct2lar(PediatricWard2)
PediatricWard2 = Struct( [(metric(V),FV,EV)], "PediatricWard2" )
◇
```

Macro referenced in 14b.

### Fifth floor input

⟨ Fifth floor 17c ⟩ ≡

```
""" Fifth floor """
GeneralWard2 = Struct([t(0,4), Ward])
GeneralWard3 = Struct([t(7,4), Ward])

V,FV,EV = struct2lar(GeneralWard2)
GeneralWard2 = Struct( [(metric(V),FV,EV)], "GeneralWard2" )
V,FV,EV = struct2lar(GeneralWard3)
GeneralWard3 = Struct( [(metric(V),FV,EV)], "GeneralWard3" )
◇
```

Macro referenced in 14b.

**Ward sections** Here input by polylines and structure modeling are freely mixed. Just notice that the affine maps included in structures are given in grid coordinates. This fact does not permit an immediate transformation in Cartesian coordinates using the `metric` function.

⟨ Ward sections 18a ⟩ ≡

```
""" Ward sections """
Room = poly2struct([TRANS([[0,0,1,1,2./3,2./3],[0,0.5,0.5,0.25,0.25,0]])])
RestRoom = poly2struct([TRANS([[2./3,2./3,1,1],[0,0.25,0.25,0]])])
Nursing1 = poly2struct([TRANS([[0,0,.2,.2],[0,.4,.4,.0]])])
Nursing2 = poly2struct([TRANS([[.2,.2,.4,.4],[0,.4,.4,.0]])])
Nursing3 = poly2struct([TRANS([[0,0,.4,.4,.2],[.4,.8,.8,.4,.4]])])
Nursing4 = poly2struct([TRANS([[0,0,.4,.4],[.8,1.1,1.1,.8]])])
Nursing5 = poly2struct([TRANS([[0,0,.4,.4],[1.1,1.4,1.4,1.1]])])

room = Struct([Room],"Room")
restRoom = Struct([RestRoom],"RestRoom")
nursing1 = Struct([Nursing1],"Nursing1")
nursing2 = Struct([Nursing2],"Nursing2")
nursing3 = Struct([Nursing3],"Nursing3")
nursing4 = Struct([Nursing4],"Nursing4")
nursing5 = Struct([Nursing5],"Nursing5")

service1 = Struct([nursing1,nursing2,nursing3,nursing4,nursing5],"Service1")
service2 = Struct([t(0,1.4),s(1,-1),service1],"Service2")
wardServices = Struct([t(1.3,.3),service2,t(0,2),service1],"WardServices")
theRoom = Struct([room,restRoom],"TheRoom")
twoRooms =  Struct([theRoom,t(0,1),s(1,-1),theRoom],"TwoRooms")
halfWard = Struct(4*[twoRooms,t(0,1)],"HalfWard")
Ward = Struct([halfWard, wardServices, t(3,0),s(-1,1), halfWard],"Ward")
```

```
#Vw,FVw,EVw = struct2lar(Ward)
#theWard = Struct( [(metric(Vw),FVw,EVw)], "theWard" )
◇
```

Macro referenced in 16.

# A Code utilities

### Coding utilities

⟨ Coding utilities 18b ⟩ ≡

```
""" Coding utilities """
⟨ Filter functions 19a ⟩
⟨ Reference grid 19b ⟩
⟨ From grid to metric coordinates 21 ⟩
⟨ Mapping a grid frame to a Cartesian one 22a ⟩
⟨ From array indices to grid coordinates 22b ⟩
◇
```

Macro referenced in 1.

### Filter functions

⟨ Filter functions 19a ⟩ ≡

```
""" Filter functions """
DEBUG = True
def poly2struct(polylines,name="Name",category="Department"):
    larModel = polyline2lar(polylines)
    return Struct( [larModel], name, category )

def mpoly2struct(polylines,name="Name",category="Department"):
    larModel = polyline2lar(AA(metric)(polylines))
    return Struct( [larModel], name, category )
◇
```

Macro referenced in 18b.

## A.1 Reference grid

Looking at the images of Figure 1, it is easy to notice the presence of a very regular structural frame, providing in the following a reference grid for the numeric input of the geometry of the departments and floors of the hospital model. Some images with evidenced (in blue) the structural frame grid are shown in Figure **??**.

It may be useful to underline that the grid step in the $y$ direction (from top to bottom of the drawings) is constant and equal to $8.4m$, whereas the grid in the $x$ direction (from left to right of the drawings) alternates the $[7.5, 9.5, 7.5]m$ pattern with the step-size used

in the other direction ($8.4m$). the above numeric patterns are actually derived by the architect from the layout of the inpatient wards.

Notice also that both grid directions, and of course the structural frame of the building, are aligned with the *inpatient wards*, that supply one the main ideas of the design concept as a whole.

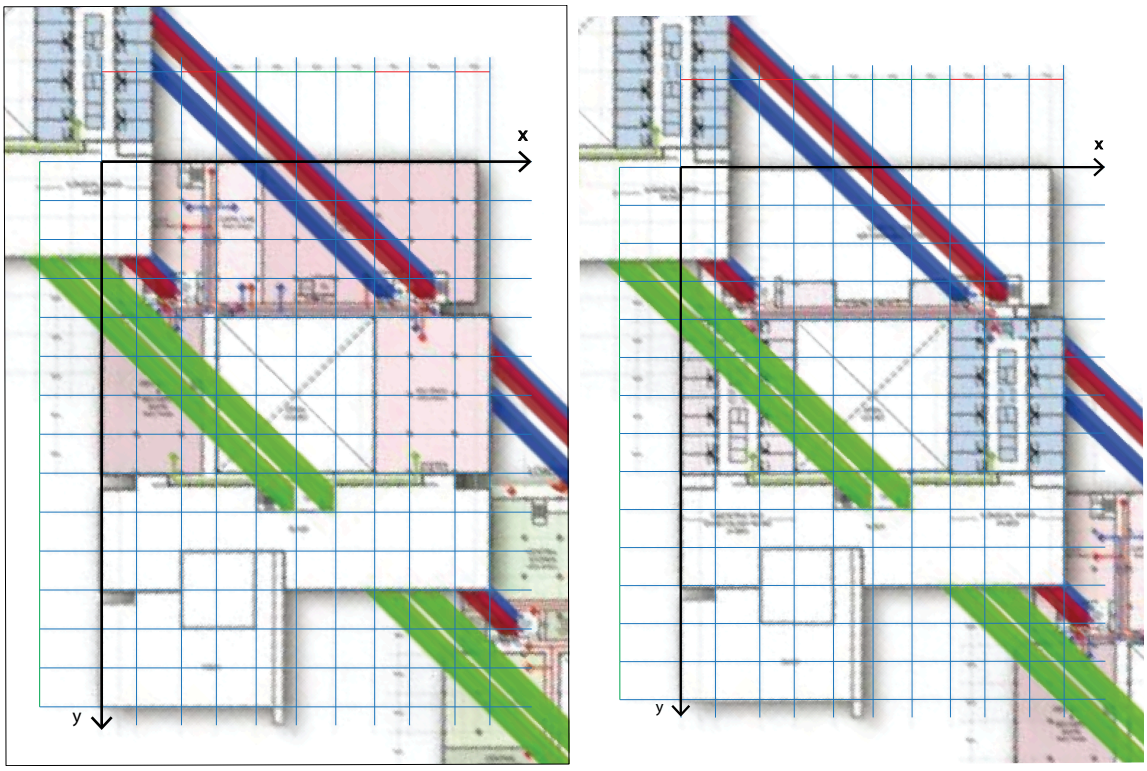


Figure 2: The zooming of two floor plans, with evidentced the structural grid (in blue): (a) first floor; (b) second floor.

**Reference grid** The reference grid is defined as `structuralGrid` in the script below, where `PROD` is the `pyplasm` primitive for Cartesian product of geometric values. The global variable `YMAX` is used in this module to compute (in the `metric` function) a proper coordinate transformation of the model from the reference frame used in the 2D hospital drawings (origin at top-left point, $y$ pointing downwards—see Figure 2) to the standard righthand reference frame (origin at bottom-left point, $y$ pointing upwards—see Figure 3).

⟨ Reference grid 19b ⟩ ≡

```
""" Reference grid """
```

```
X = [0]+[7.5,9.5,7.5]+4*[8.4]+[7.5,9.5,7.5]+[0]
Y = [0]+14*[8.4]+[0]
xgrid = QUOTE(X[1:-1])
ygrid = QUOTE(Y[1:-1])
structuralGrid = PROD([xgrid,ygrid])
YMAX = SUM(Y)
◇
```

Macro referenced in 18b.

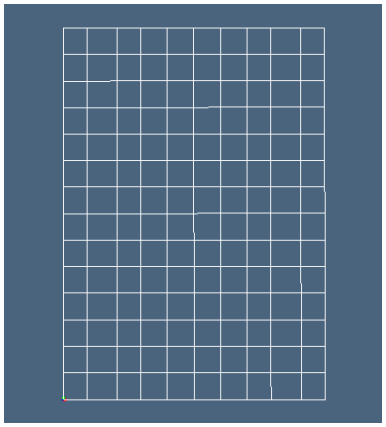

Figure 3: The reference grid used in the model construction. The intersections of grid lines have integer coordinates.

**From grid to metric coordinates** The actual transformation of vertices of geometric data is executed by applying the (partial) function `metric` to a list of 2D points, as shown by the example below.

⟨ From grid to metric coordinates 21 ⟩ ≡

```
""" From grid to metric coordinates """
def grid2coords(X,Y):
    xMeasures = list(cumsum(X))
    yMeasures = list(cumsum(Y))
    def grid2coords0(point):
        x,y = point[0:2]
        xint,yint = int(x), int(y)
        xdec,ydec = float(x-xint), float(y-yint)
        xcoord = xMeasures[xint] + xdec*X[xint+1]
        ycoord = yMeasures[yint] + ydec*Y[yint+1]
        if len(point)==2: return [xcoord, ycoord]
        else: return [xcoord, ycoord, point[2]]
```

```
        return grid2coords0

    def coordMaps(YMAX):
        def coordMaps0(polyline):
            polyline = AA(grid2coords(X,Y))(polyline)
            polyline = vmap(YMAX)(polyline)
            return [eval(vcode(point)) for point in polyline]
        return coordMaps0

    metric = coordMaps(YMAX)
    ◇
```

Macro referenced in 18b.

**Example** A simple example of transformation from grid to metric coordinates is given here:

```
polyline = metric([[3,4],[3,8],[4,8],[4,7.8],[6,7.8],[6,8],[6.65,8],[6.65,4]])
>>> [[24.5,84.0],[24.5,50.4],[32.9,50.4],[32.9,52.08],[49.7,52.08],[49.7,50.4],
    [55.16,50.4],[55.16,84.0]]
```

### Mapping the grid frame to a Cartesian right-hand frame

⟨ Mapping a grid frame to a Cartesian one 22a ⟩ ≡

```
    """ Mapping the grid frame to a Cartesian right-hand frame """
    def vmap(YMAX):
        def vmap0(V):
            if len(V[0])==3: W = [[x,YMAX-y,z] for x,y,z in V]
            else: W = [[x,YMAX-y] for x,y in V]
            return W
        return vmap0
    ◇
```

Macro referenced in 18b.

**From array indices to grid coordinates** The reference grid, as the Cartesian product of two subsets of adjacent integers, will be used both to strongly simplify the input of data, and to assign to such coordinate numbers a more interesting meaning. For example the open space in the middle of the building will so defined as the 2D box with extreme points of integer coordinates $(3, 4)$ and $(7, 11)$. Therefore the whole building will be contained in the 2D interval $[0, 10] \times [0, 14]$ in "*grid coordinates*".

⟨ From array indices to grid coordinates 22b ⟩ ≡

```
    """ From array indices to grid coordinates """
    def index2coords(theArray):
        return CONS(AA(T([1,2]))(CAT((theArray).tolist())))
    ◇
```

Macro referenced in 18b.

## References


[AM13] Adham R. Ismail Abdel-Moneim, *Hospital planning and medical equipment design*, Future Healthcare – The opportunities of new technology (Oslo, Norway), 38th World Hospital Congress, 18–20 June 2013.

[CL13] CVD-Lab, *Linear algebraic representation*, Tech. Report 13-00, Roma Tre University, October 2013.

[CP90] C. Cattani and A. Paoluzzi, *Boundary integration over linear polyhedra*, Computer-Aided Design **22** (1990), no. 2, 130–135.

[DKT05] Mathieu Desbrun, Eva Kanso, and Yiying Tong, *Discrete differential forms for computational modeling*, ACM SIGGRAPH 2005 Courses (New York, NY, USA), SIGGRAPH '05, Acm, 2005.