This lecture will be recorded

064-0026-00L: COMPAS II

Introduction to Computational Methods for Digital Fabrication in Architecture

slides + code

*https://***dfab.link/fs2022**

TODAY

graphs

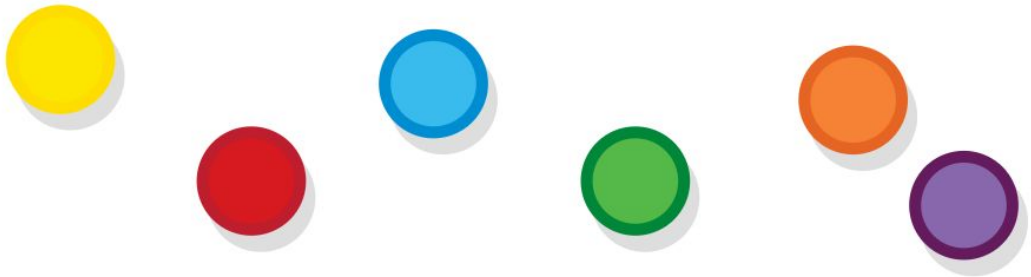
assemblies

sequencing

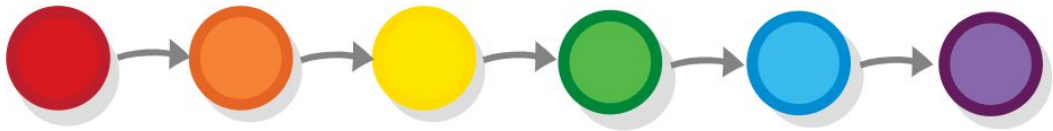
Use the **assembly data structure** and **assembly sequencing**

graphs

Sets

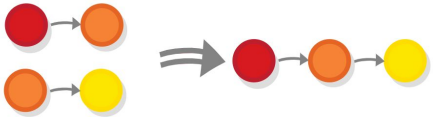


Linear order



```python
@functools.total_ordering
class BoxComparer(object):
    def __init__(self, box, *args):
        self.box = box

    def __eq__(self, other):
        return self.box.data == other.box.data

    def __lt__(self, other):
        return self.box.dimensions < other.box.dimensions
```

Reflexivity
Each object has to be bigger or equal to itself.

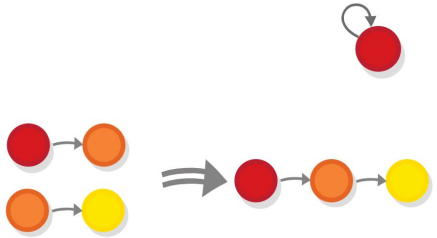


Transitivity
If A is bigger than B, and B is bigger than C, then A is bigger than C.

Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. `x <= y` and `y <= x` only iff `x == y`

Totality
All elements should be comparable to each other

Source: Boris Marinov, https://github.com/boris-marinov/category-theory-illustrated



Reflexivity
Each object has to be bigger or equal to itself.

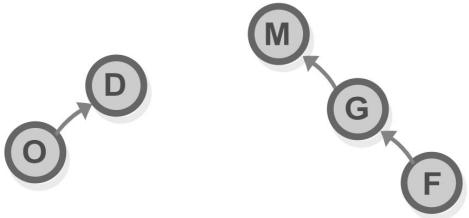
Transitivity
If A is bigger than B, and B is bigger than C, then A is bigger than C.

Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. `x <= y` and `y <= x` only iff `x == y`

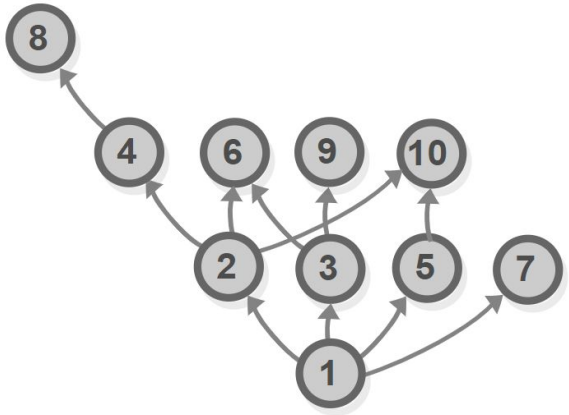
Totality
All elements should be comparable to each other

Source: Boris Marinov, https://github.com/boris-marinov/category-theory-illustrated

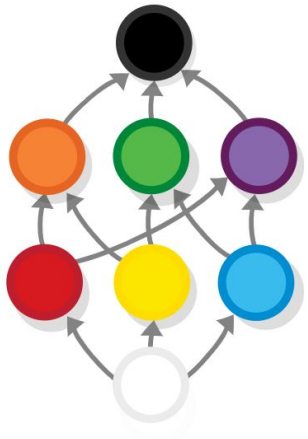
Partial order



Linearly-ordered subsets



Partial order

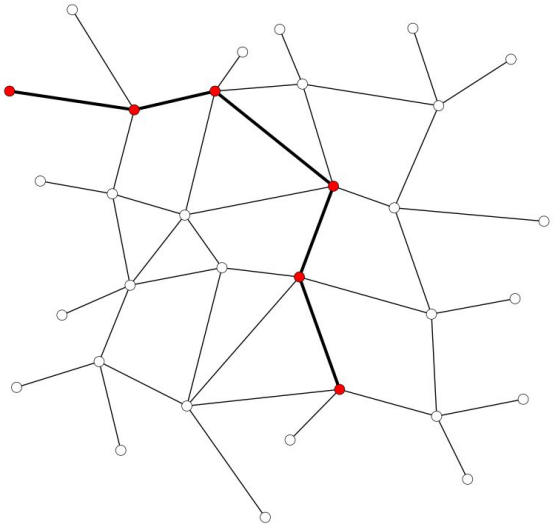


Lattice

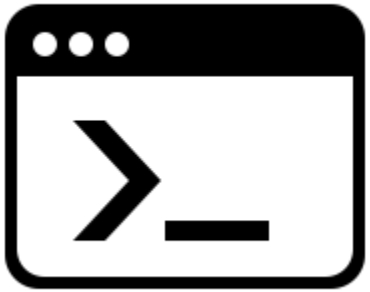
Network

compas.datastructures

- directed edge graph data structure
- graph: topological
- network: geometric implementation of graph
- edge, node, degree, neighbors
- custom attributes
- networkx lossless conversion

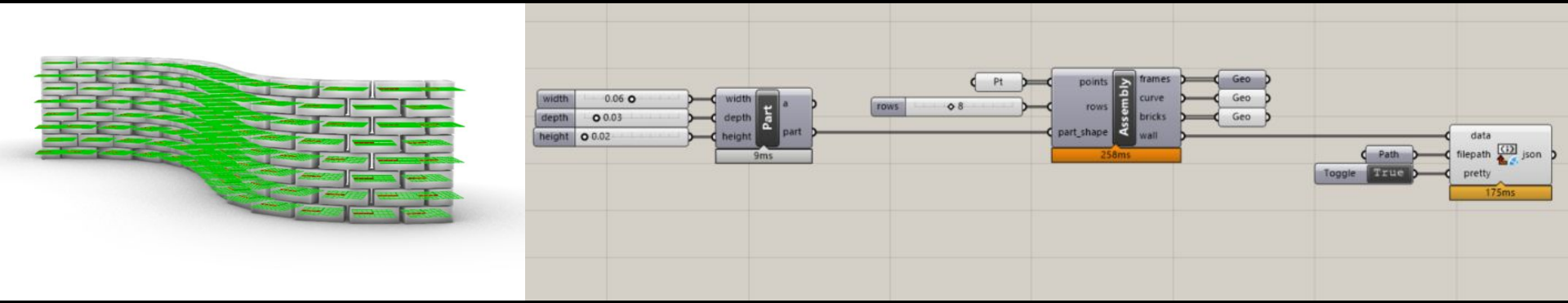


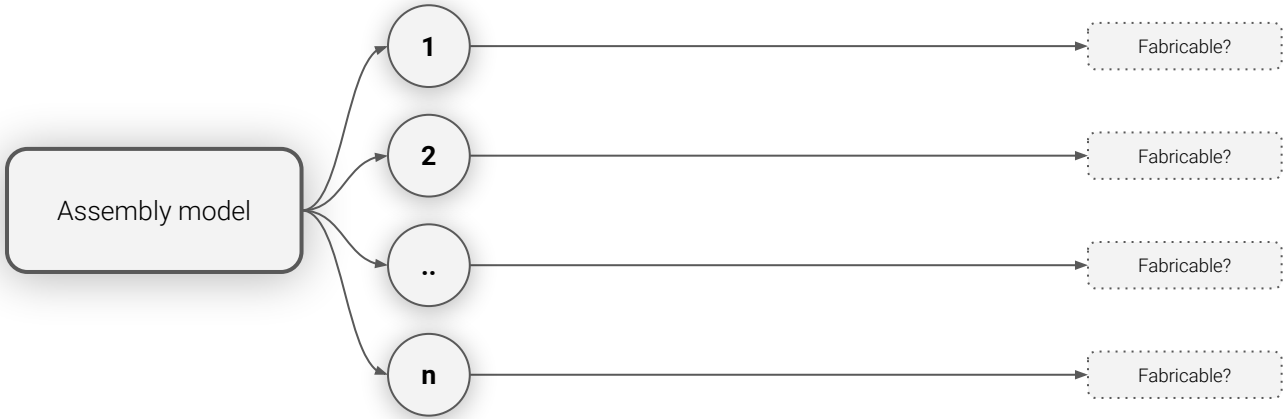
```
network = Network()

s = network.add_node(x=11, y=30, z=0, color=(000, 000, 000), text='black')

o = network.add_node(x=1., y=20, z=0, color=(255, 128, 000), text='orange')
g = network.add_node(x=11, y=20, z=0, color=(000, 255, 000), text='green')
p = network.add_node(x=21, y=20, z=0, color=(128, 000, 128), text='purple')

r = network.add_node(x=1., y=10, z=0, color=(255, 000, 000), text='red')
y = network.add_node(x=11, y=10, z=0, color=(255, 255, 000), text='yellow')
b = network.add_node(x=21, y=10, z=0, color=(000, 000, 255), text='blue')

w = network.add_node(x=11, y=00, z=0, color=(255, 255, 255), text='white')

network.add_edge(w, r)
network.add_edge(w, y)
network.add_edge(w, b)

network.add_edge(r, o)
network.add_edge(r, p)

# [..]
```

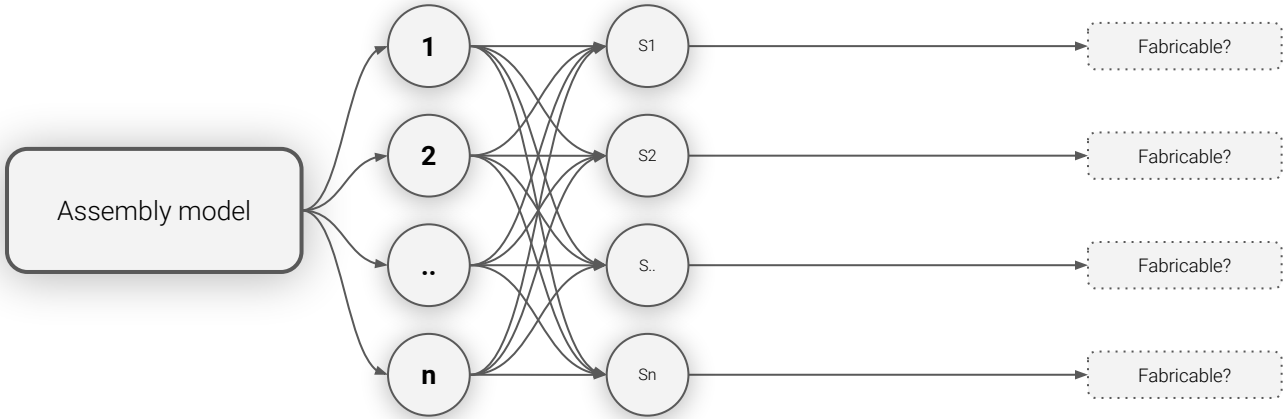


`docker-compose up -d`

Right-click → Compose Up


width
0.06
depth
0.03
height
0.02
Part
a
part
9ms
Pt
rows
8
points
rows
part_shape
Assembly
frames
curve
bricks
wall
258ms
Geo
Geo
Geo
Path
Toggle
True
data
filepath
json
pretty
175ms

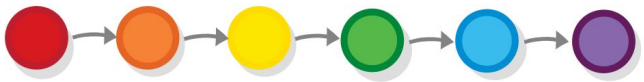

Assembly model
1
2
..
n
Fabricable?
Fabricable?
Fabricable?
Fabricable?


Assembly model
1
2
..
n
S1
S2
S..
Sn
Fabricable?
Fabricable?
Fabricable?
Fabricable?

Sequence types

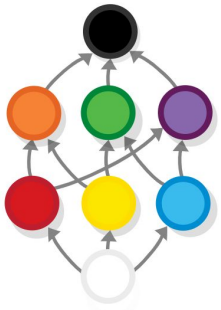
Total orders (fully linear sequences)

- Simple to describe
- Work for simple processes



Partial orders (e.g. dependency graph).

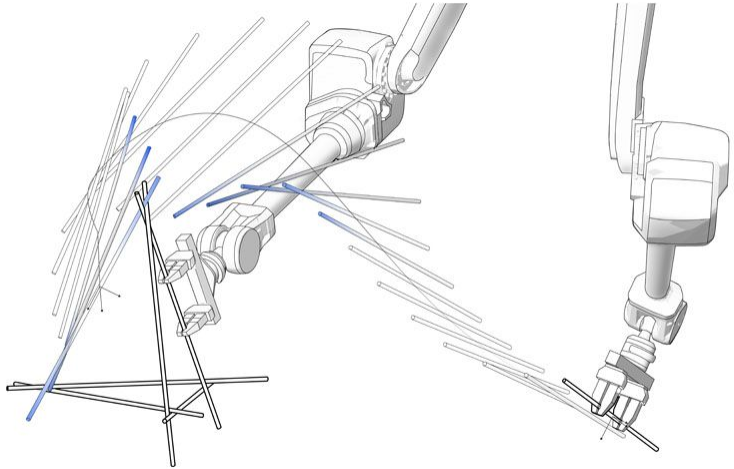
- Allow to express more advanced process (e.g. multiple robots in parallel)
- More involved to describe
- Broader selection of algorithms available



Impact of building sequence

Sequence affects fabricability in multiple ways:

- Stability during fabrication
- Tolerance build-up
- Robotic accessibility
- Material behavior

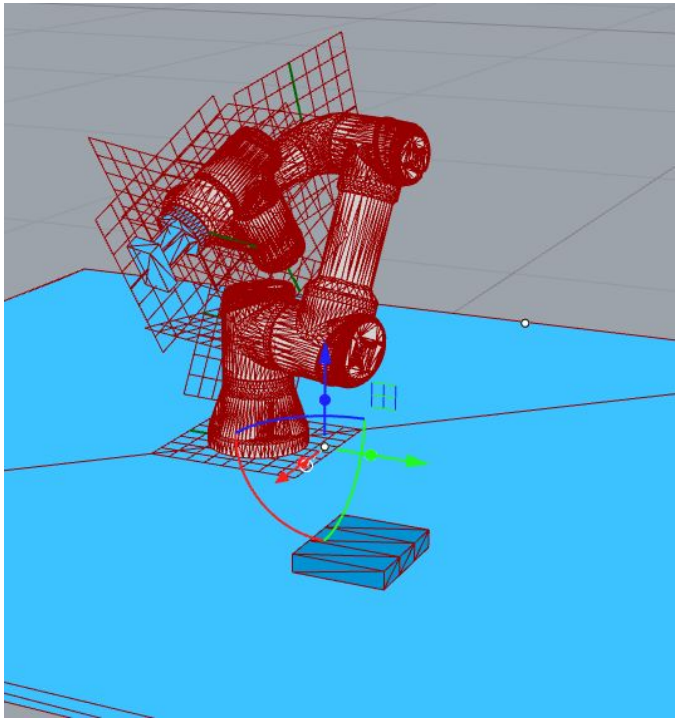


Sequenced assembly planning

```
assembly = compas.json_load(filename)
# ..
sequence = breadth_first_ordering(assembly.graph.adjacency, root_part_key)

for part_key in sequence:
    prepare_scene(robot, part_key)

    # ..
    freespace_trajectory = robot.plan_motion(goal_constraints, approach_pick_config)
    part.attributes["freespace_trajectory"] = freespace_trajectory
    # ..

compas.json_dump(assembly, "603_assembly_planning.json"))
```

Assignment

- Define a parametric assembly based on examples 601-612.
- Goal 1: Ensure at least 20 parts are independently buildable (ie. there are trajectories for all).
- Goal 2: Ensure at least 20 parts are buildable taking into account previously built parts.



Next week

- Assignment submission due: Wed 27th April, 9AM.
- Ask for help if needed: Slack, Forum, Office Hours (Fridays, request via Slack)
- Prepare your computer for COMPAS RRC exercises:
  - Install ABB RobotStudio
  - Explore getting started repo: https://github.com/compas-rrc/compas_rrc_start
- Next lecture:
  - Robot control with COMPAS RRC

Thanks!