This lecture will be recorded

ETH zürich

COMPAS

064-0026-00L: COMPAS II

Introduction to Computational Methods for Digital Fabrication in Architecture slides + code

https://dfab.link/fs2022

```
(fs2022) conda env update -f https://dfab.link/fs2022.yml
```

discussion

`https://metroretro.io/board/LBCY66ABUEJQ`

TODAY

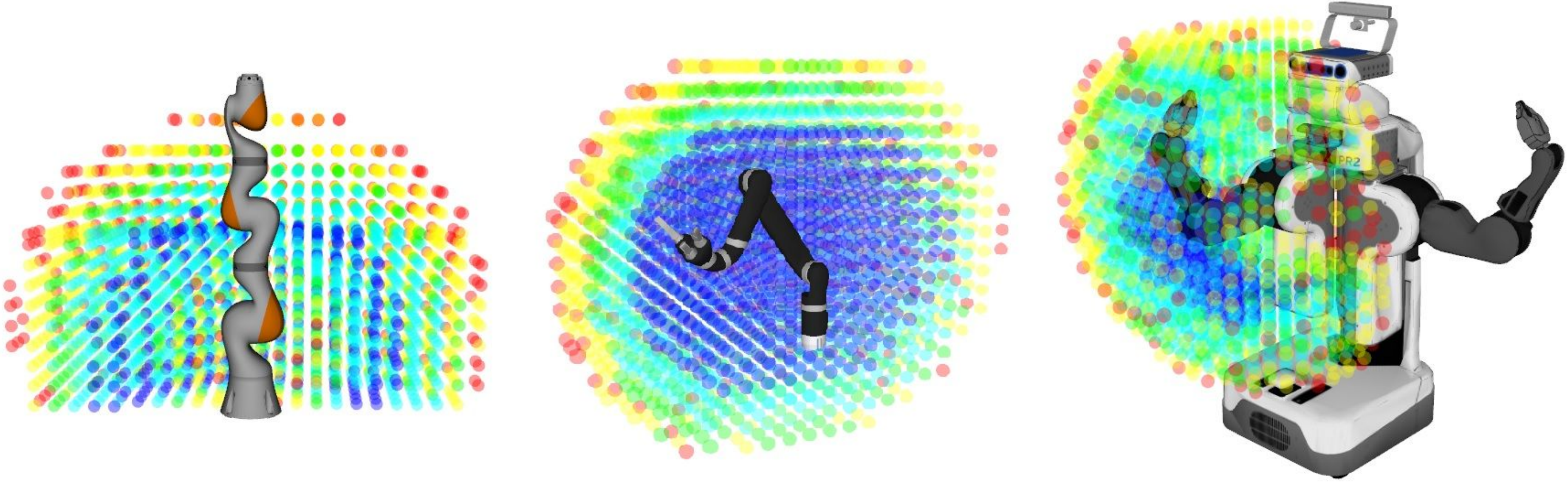
reachability map

graphs

assemblies

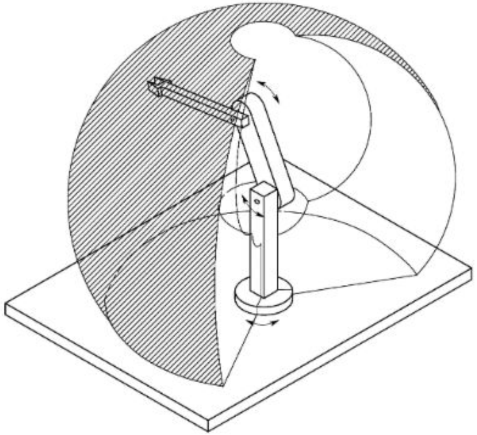
Today's goal

Understand **data structures** to plan a **discrete assembly process** reachability map

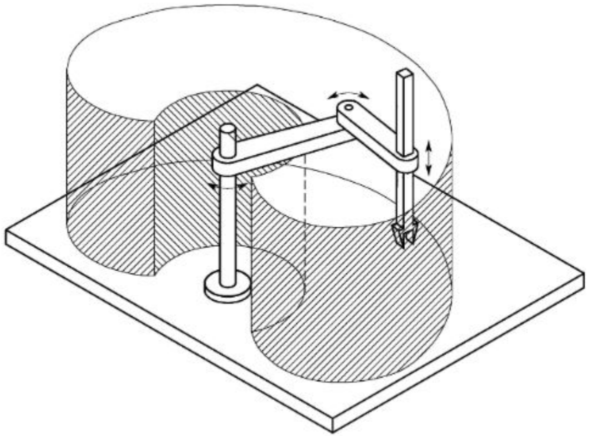


images from http://wiki.ros.org/reuleaux

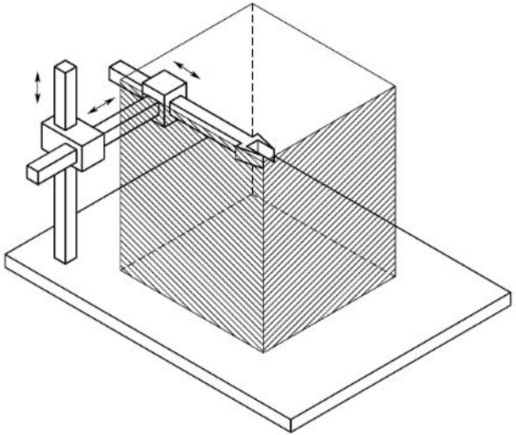
Reachability map != Robot workspace



Anthropomorphic manipulator

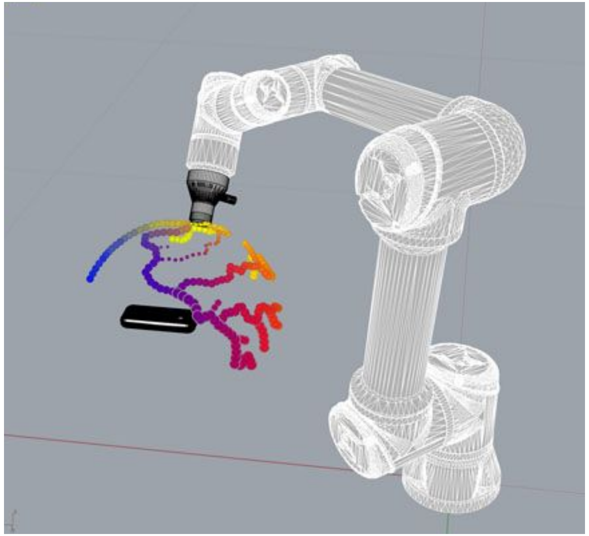


SCARA manipulator



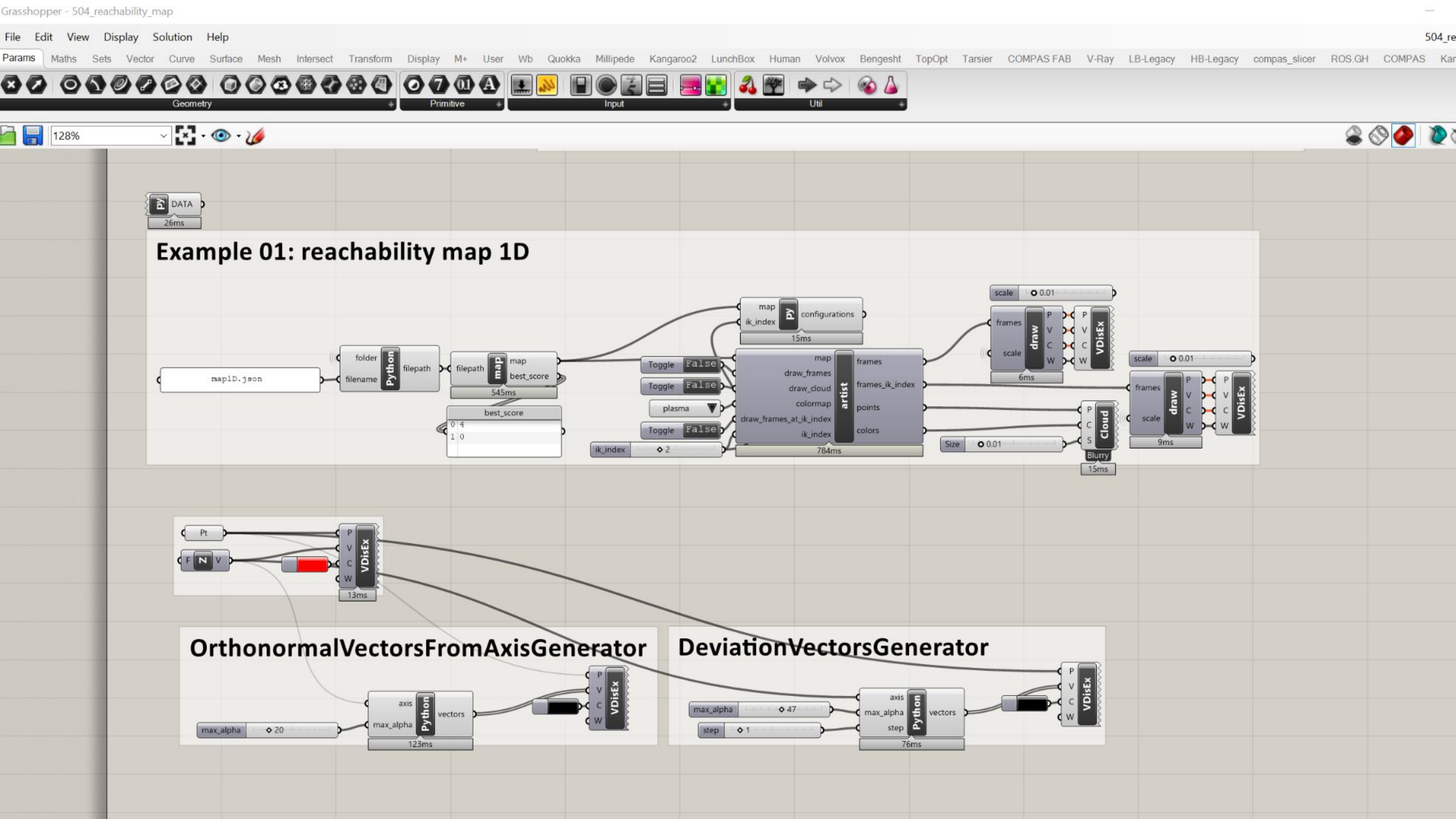
Cartesian manipulator

Robotic 360° Lightpainting



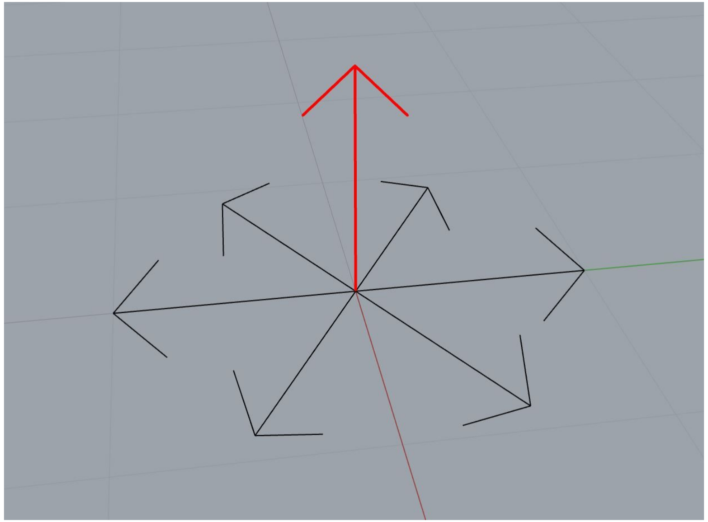
[Example 360° Image]

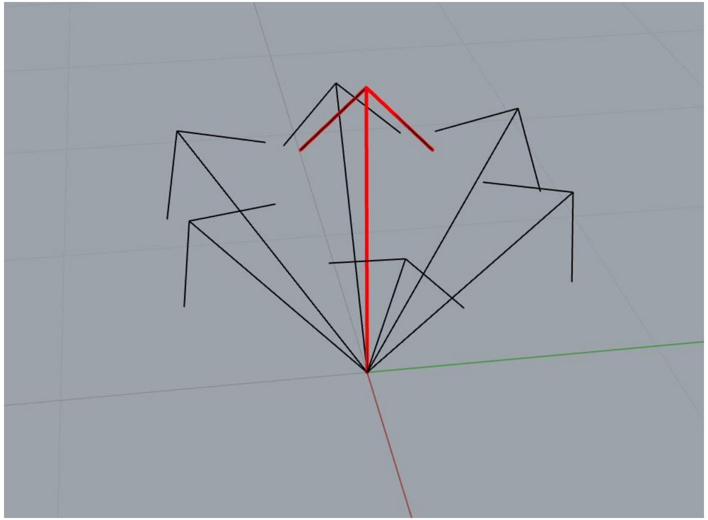
```python
# 1. Define frames on a sphere
sphere = Sphere((0.4, 0, 0), 0.15)


def points_on_sphere_generator(sphere):
    for theta_deg in range(0, 360, 20):
        for phi_deg in range(0, 90, 10):
            theta = math.radians(theta_deg)
            phi = math.radians(phi_deg)
            x = sphere.point.x + sphere.radius * math.cos(theta) * math.sin(phi)
            y = sphere.point.y + sphere.radius * math.sin(theta) * math.sin(phi)
            z = sphere.point.z + sphere.radius * math.cos(phi)
            point = Point(x, y, z)
            axis = sphere.point - point
            plane = Plane((x, y, z), axis)
            f = Frame.from_plane(plane)
            # for UR5 is zaxis the xaxis
            yield [Frame(f.point, f.zaxis, f.yaxis)] # 2D frame generator
# [..]
```



Grasshopper - 504_reachability_map
File Edit View Display Solution Help
Params Maths Sets Vector Curve Surface Mesh Intersect Transform Display M+ User Wb Quokka Millipede Kangaroo2 LunchBox Human Volvox Bengesht TopOpt Tarsier COMPAS FAB V-Ray LB-Legacy HB-Legacy compas_slicer ROS.GH COMPAS
Geometry
Primitive
Input
Util
128%
DATA
26ms
Example 01: reachability map 1D
map1D.json
folder
filename
Python
filepath
map
best_score
545ms
best_score
0 4
1 0
map
ik_index
configurations
15ms
Toggle False
Toggle False
plasma
Toggle False
ik_index 2
draw_frames
draw_cloud
colormap
draw_frames_at_ik_index
artist
frames
frames_ik_index
points
colors
784ms
scale 0.01
frames
scale
draw
6ms
VDisEx
Cloud
Size 0.01
Blurry
15ms
9ms
Pt
13ms
OrthonormalVectorsFromAxisGenerator
axis
max_alpha
vectors
max_alpha 20
123ms
DeviationVectorsGenerator
max_alpha 47
step 1
step
76ms

Vector Generators



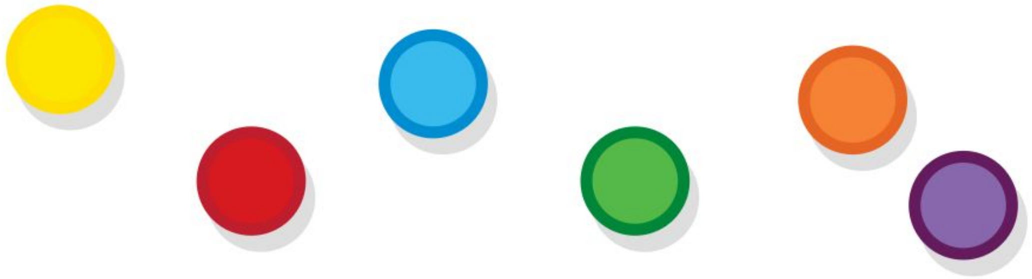


```python
# 1. Define frames on a sphere
sphere = Sphere((0.4, 0, 0), 0.15)


def points_on_sphere_generator(sphere):
    # [..]

def deviation_vector_generator(frame):
    for xaxis in DeviationVectorsGenerator(frame.xaxis, math.radians(40), 1):
        yaxis = frame.zaxis.cross(xaxis)
        yield Frame(frame.point, xaxis, yaxis)

# [..]
```

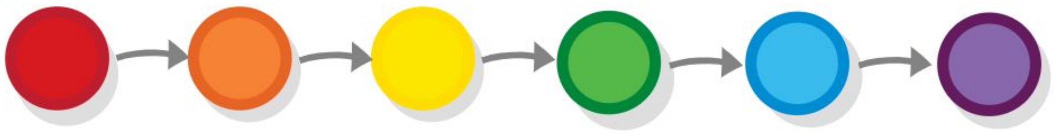
```python
# 1. Define frames on a sphere
sphere = Sphere((0.4, 0, 0), 0.15)


def points_on_sphere_generator(sphere):
    # [..]

def sphere_generator():
    sphere = Sphere((0.4, 0, 0), 0.15)
    for x in range(5):
        for z in range(7):
            center = sphere.point + Vector(x, 0, z) * 0.05
            yield Sphere(center, sphere.radius)

# [..]
```

graphs

Sets

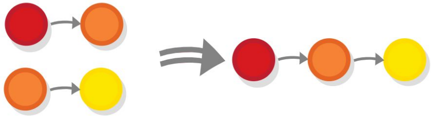


Linear order

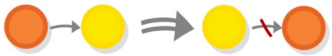


```
@functools.total_ordering
class BoxComparer(object):
    def __init__(self, box, *args):
        self.box = box

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

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

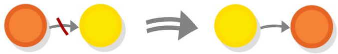
Reflexivity
Each object has to be bigger or equal to itself.



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



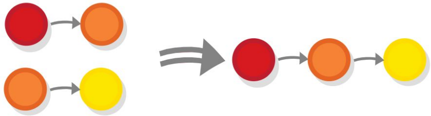
Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. **`x <= y`** and **`y <= x`** only iff **`x == y`**



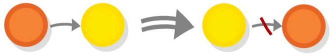
Totality
All elements should be comparable to each other

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

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



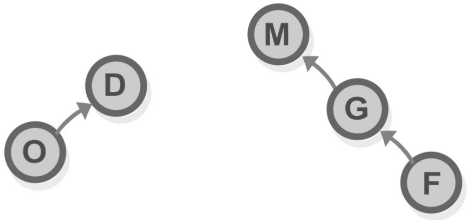
Transitivity
If A is bigger than B, and B is bigger than C, then A is bigger than C.



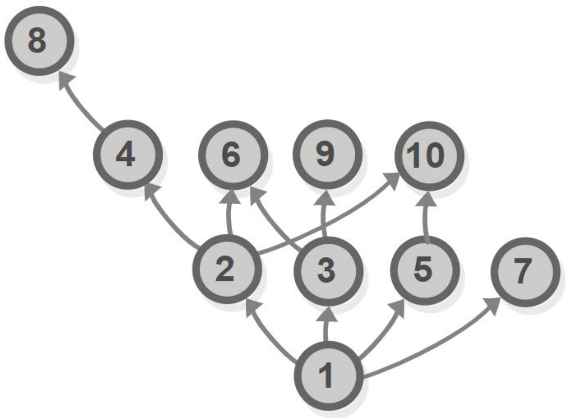
Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. `x <= y` and `y <= x` only iff `x == y`

~~**Totality**~~
~~All elements should be comparable to each other~~

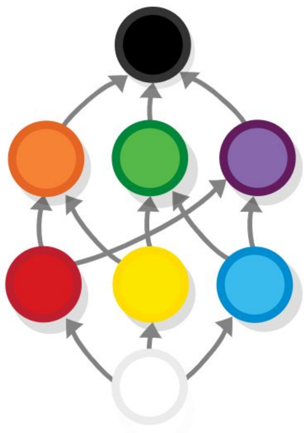
Partial order



Linearly-ordered subsets



Partial order

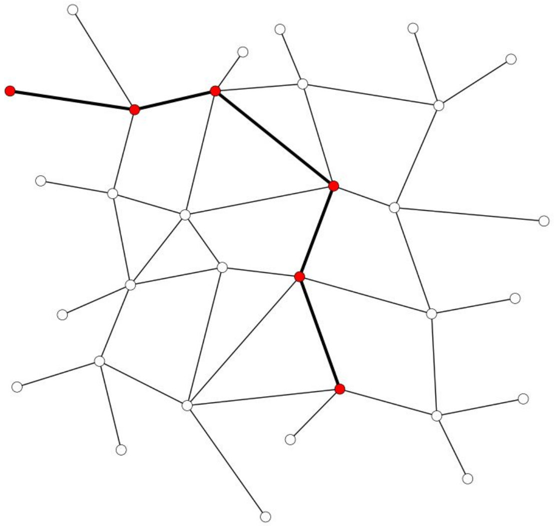


Lattice

Network

compas.datastructures

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

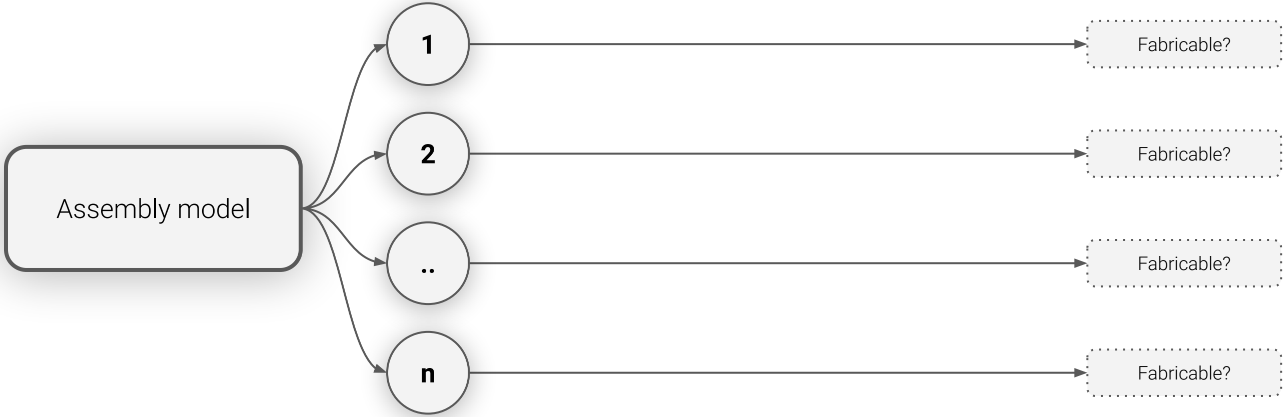


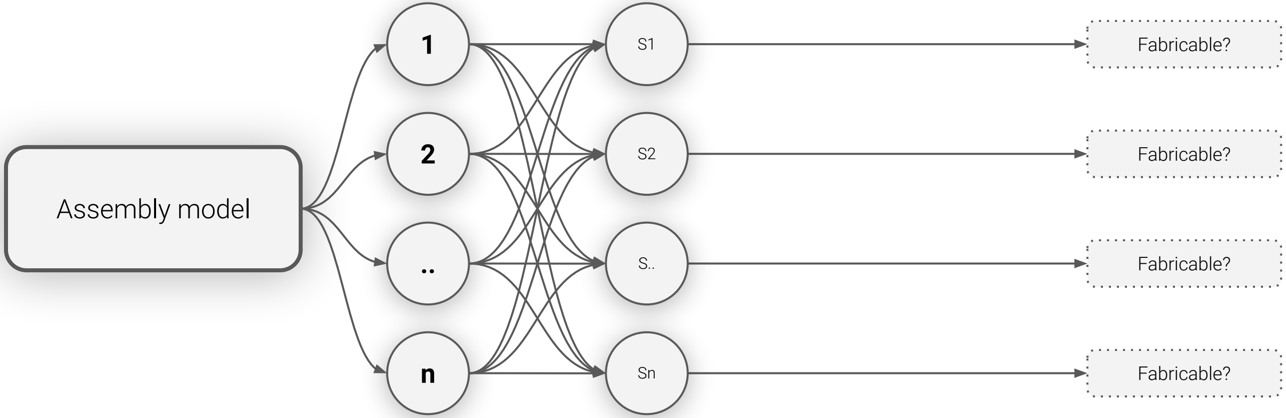
```
network = Network()

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

```python
assembly = Assembly('picknplace')
assembly.attributes['start_configuration'] = start_configuration.data
assembly.attributes['tool'] = tool.data

part = Part(part_key, shape=get_part_shape(part_key), frame=get_place_frame(part_key))
assembly.add_part(part, key=part_key)

compas.json_dump(assembly)
```

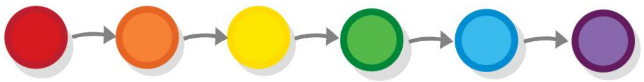

Assembly model
1
2
..
n
Fabricable?
Fabricable?
Fabricable?
Fabricable?


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

Sequence types

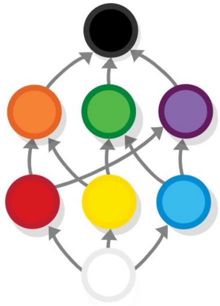
Total orders (fully linear sequences)

- 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

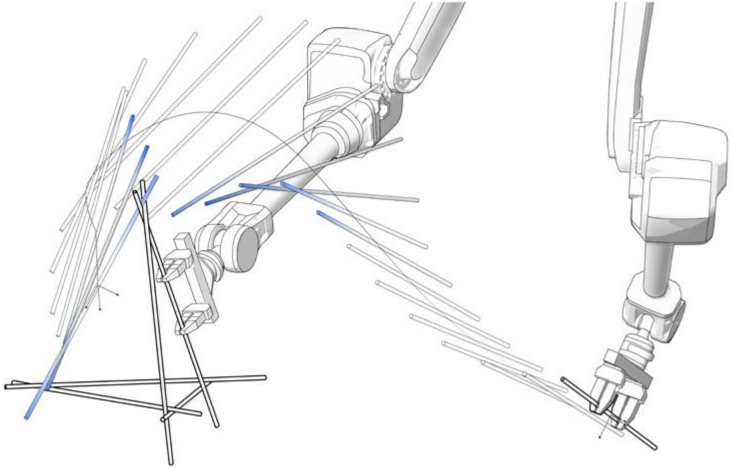


Fabrication-aware design

Impact of building sequence

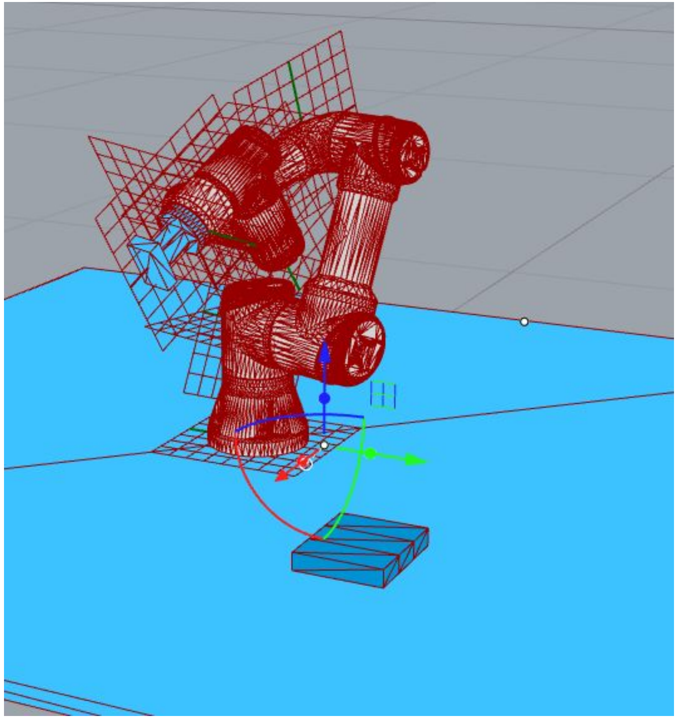
Sequence affects fabricability in multiple ways:

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



Assignment

- Create a simple assembly tweaking the functions of example 530.
- Ensure all parts are independently buildable (ie. there are trajectories for all).



Next week

- Assignment submission due: Wed 13th April, 9AM.
- Ask for help if needed: Slack, Forum, Office Hours (Fridays, request via Slack)
- Next lecture:
  - Building up assembly scene
  - Assembly sequencing

Thanks!