Functional Programming for Compiling and Decompiling Computer-Aided Design

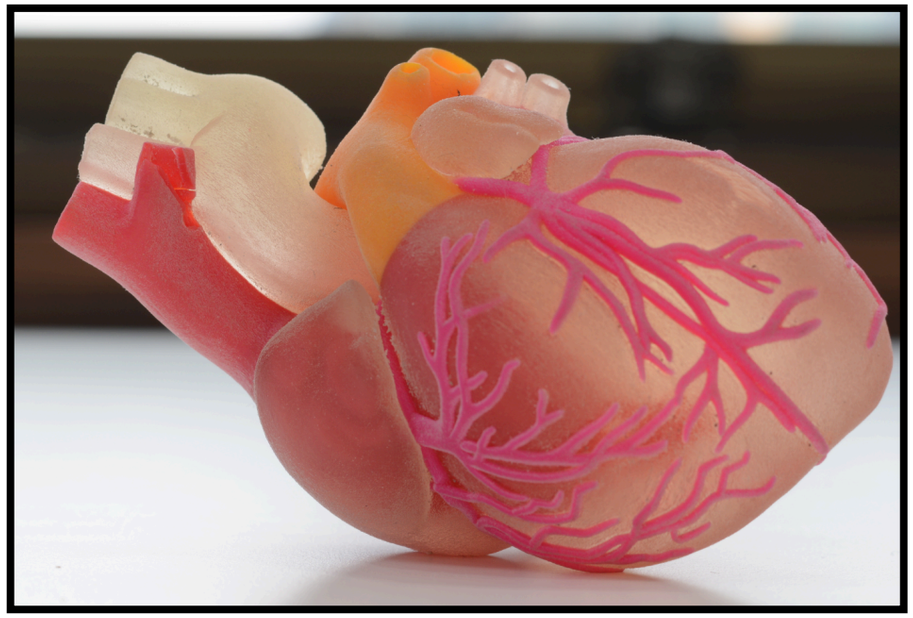
Chandrakana Nandi, James R. Wilcox, Pavel Panchekha, Taylor Blau, Dan Grossman, Zachary Tatlock

ICFP 2018

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

PLSE

The 3D Printing Revolution



Forbes Billionaires Innovation

No Donor Required: 5 Body Parts You Can Make With 3-D Printers

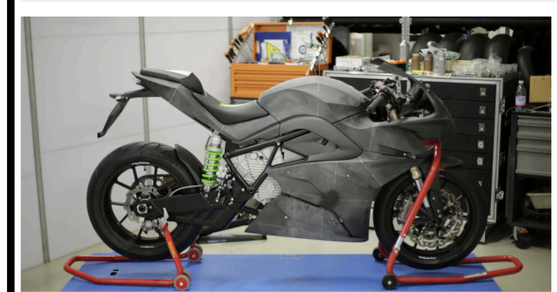
Broke a Glass? Someday You Might 3-D-Print a New One

A pretzel made with a new 3-D printing technique that uses fused silica glass. NeptunLab/KIT

Catching Up to 3D Printing

Design

By ALICE RAWSTHORN JULY 21, 2013



The fairings for this Energica electric motorcycle were made with 3D-printed pieces from Windform.
Windform/CRP Group

| SHORT FILM SHOWCASE |

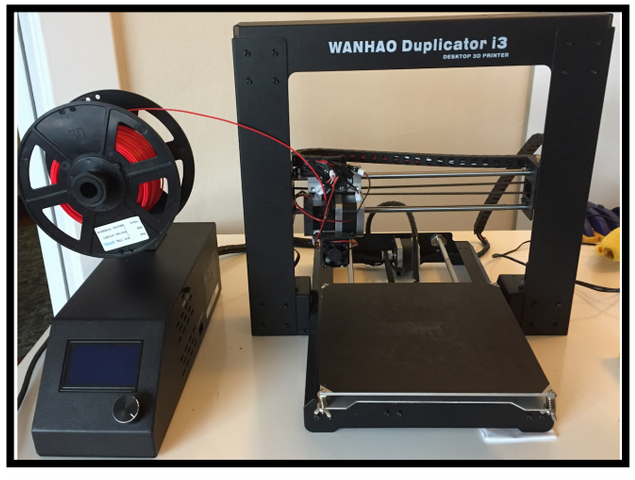
How 3-D-Printed Prosthetic Hands Are Changing These Kids' Lives

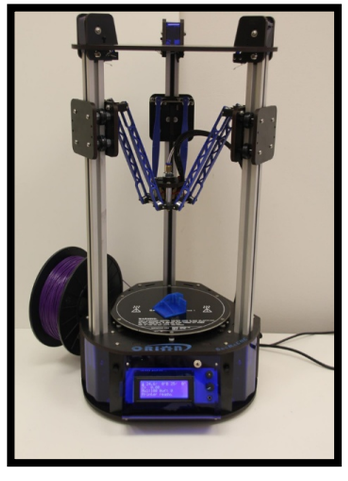
ENABLING THE FUTURE

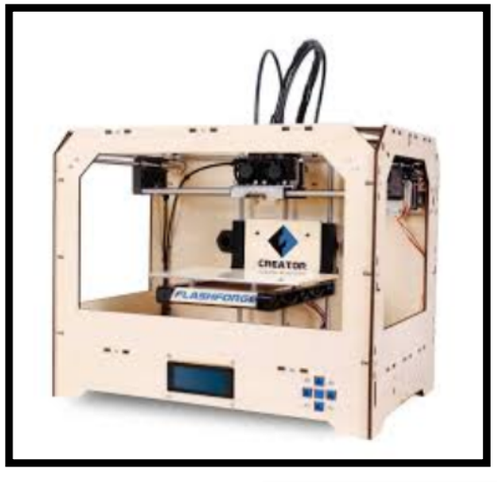
A Global Network Of Passionate Volunteers Using 3D Printing To Give The World A "Helping Hand."

3D-printed prosthetic limbs: the next revolution in medicine

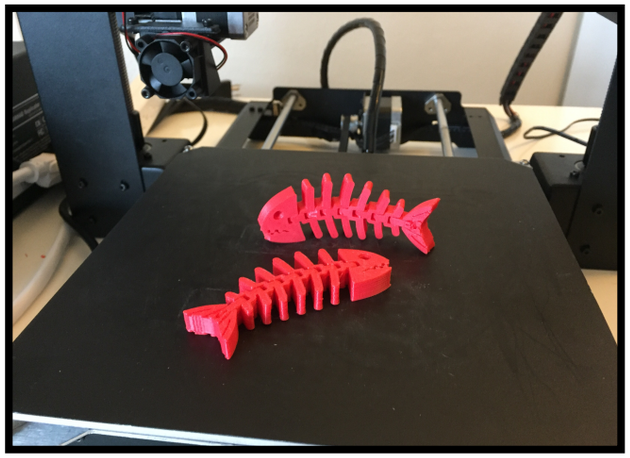
Democratized Fabrication

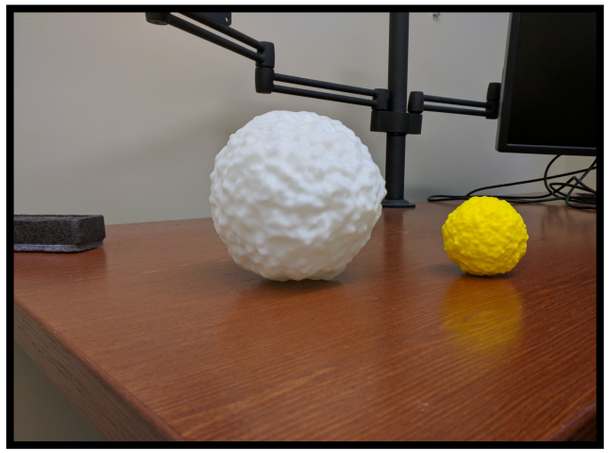


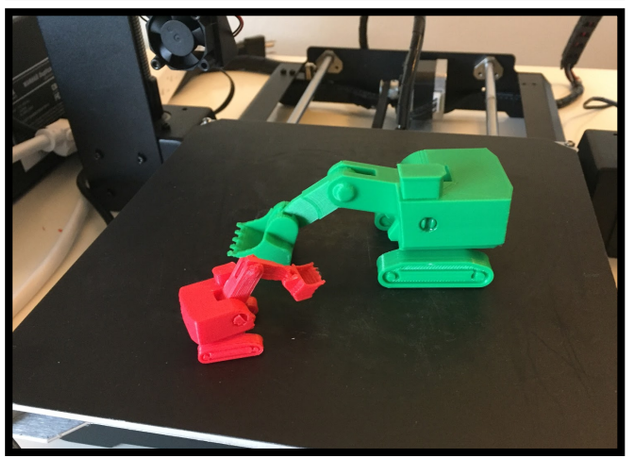


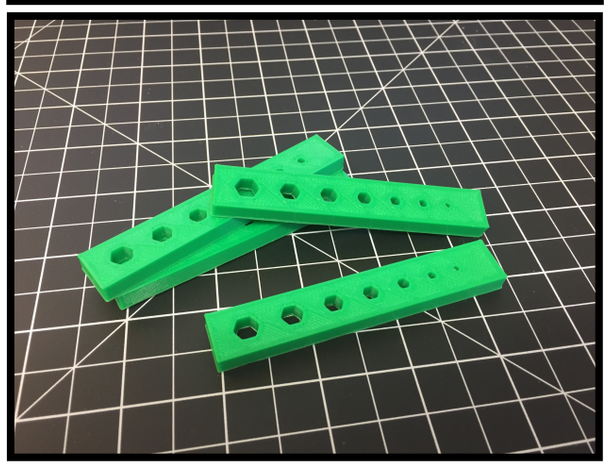


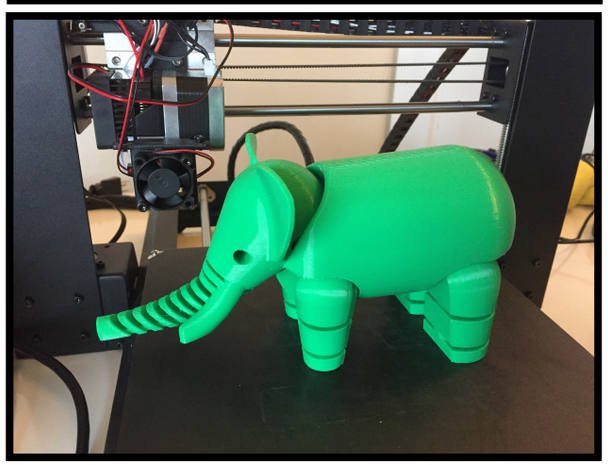
PLSE printed!





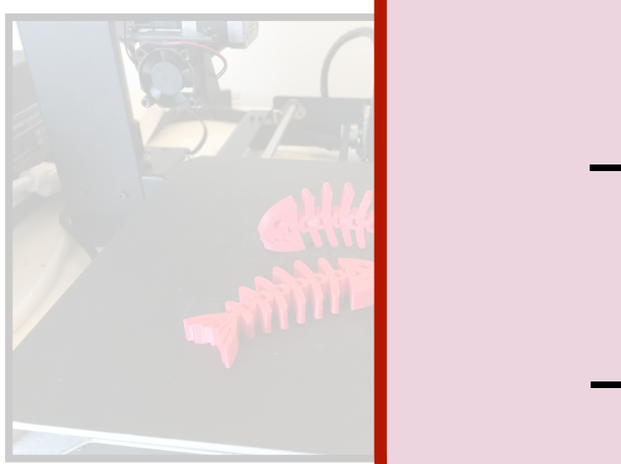


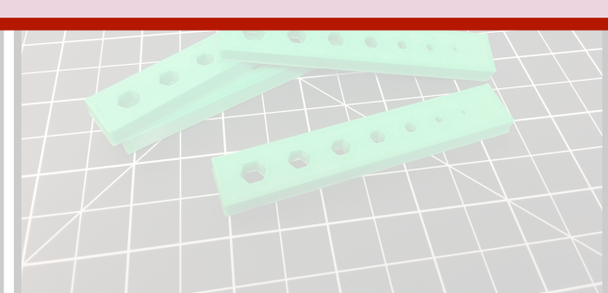




Hardware cost has gone down!

Democratized Fabrication





Design tools challenges

— steep learning curve

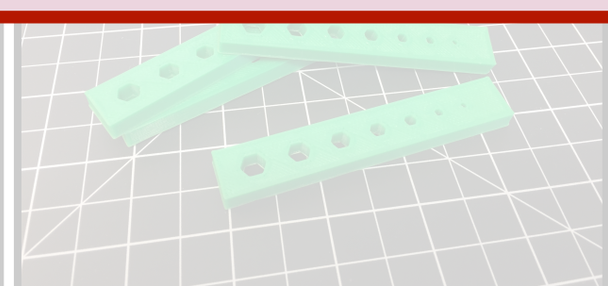
— lack of specifications

— expensive

printed!

rdware
st has
gone
down!

Democratized Fabrication



Design tools challenges

— steep learning curve

— lack of specifications

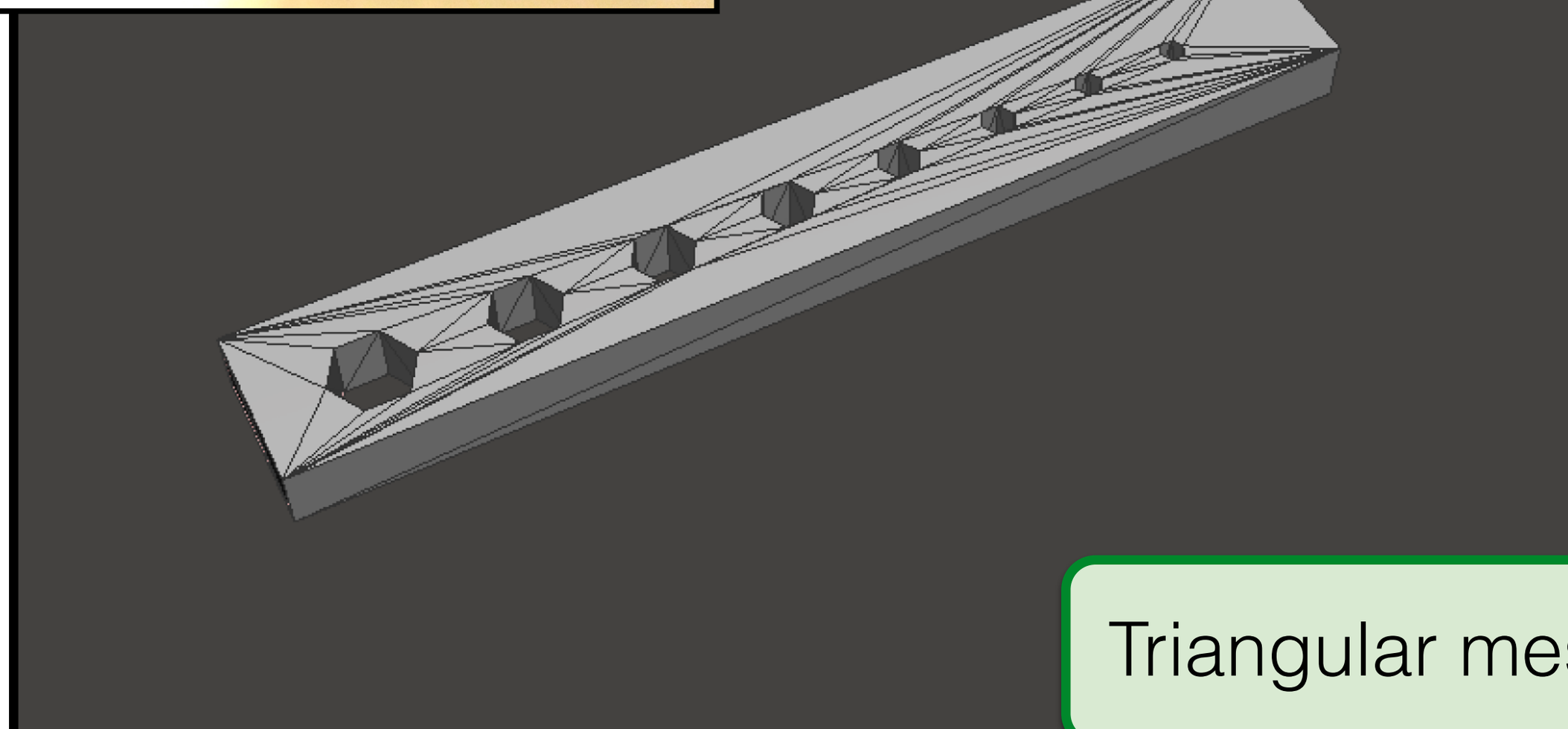
— expensive

Thingiverse

YM GRABCAD

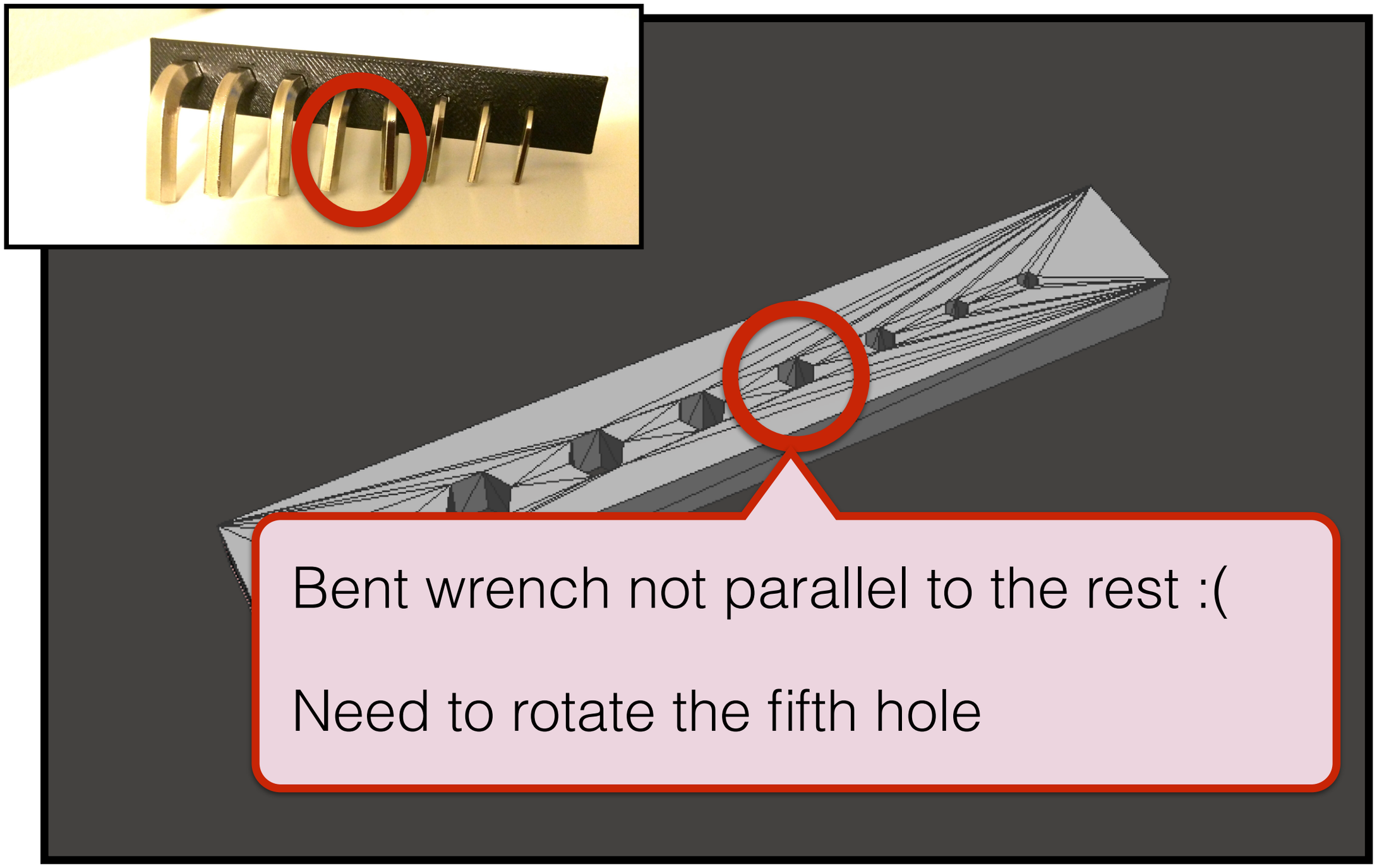
Online repositories

A mesh for a hex holder

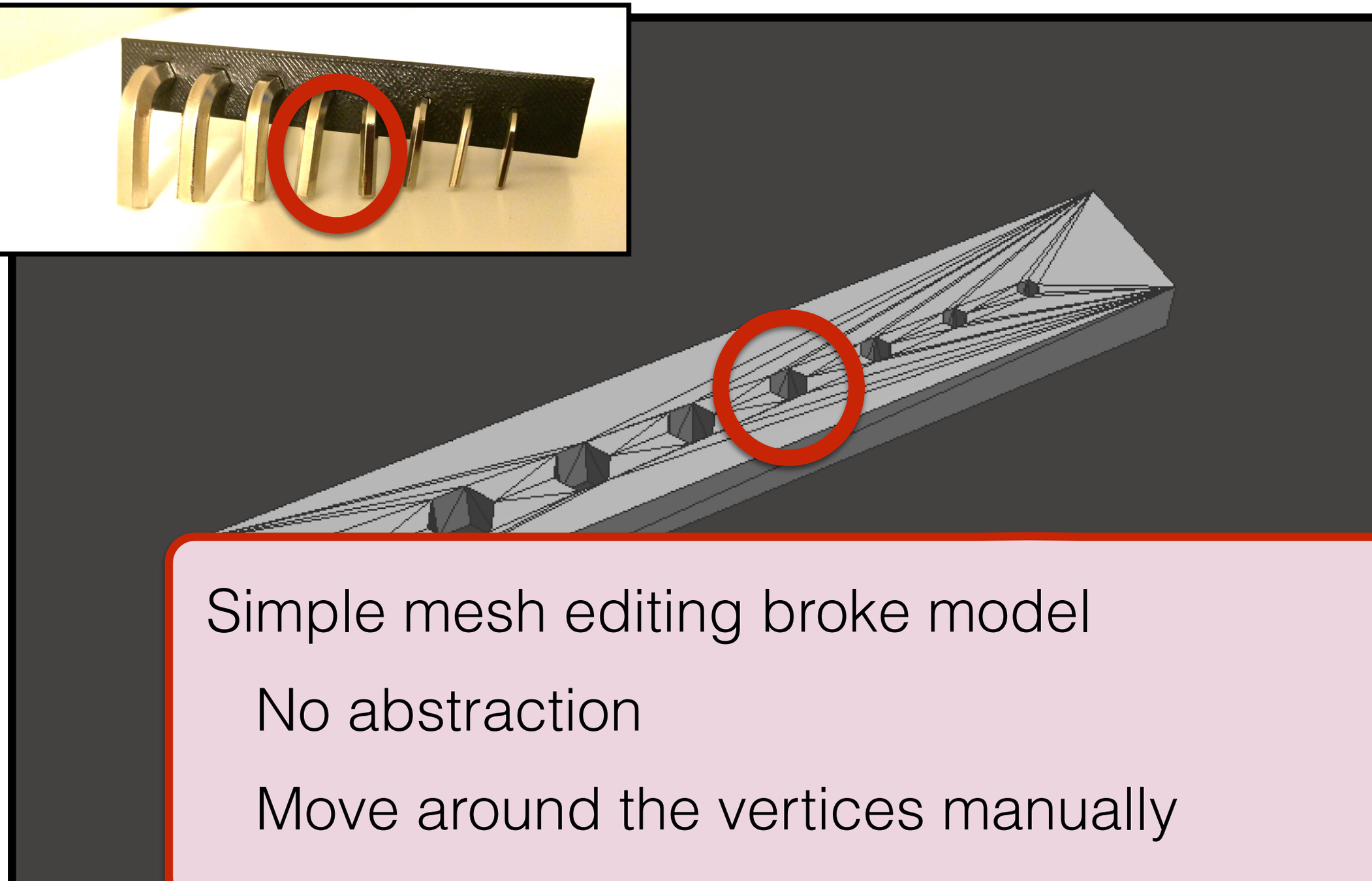


Triangular mesh

A mesh for a hex holder



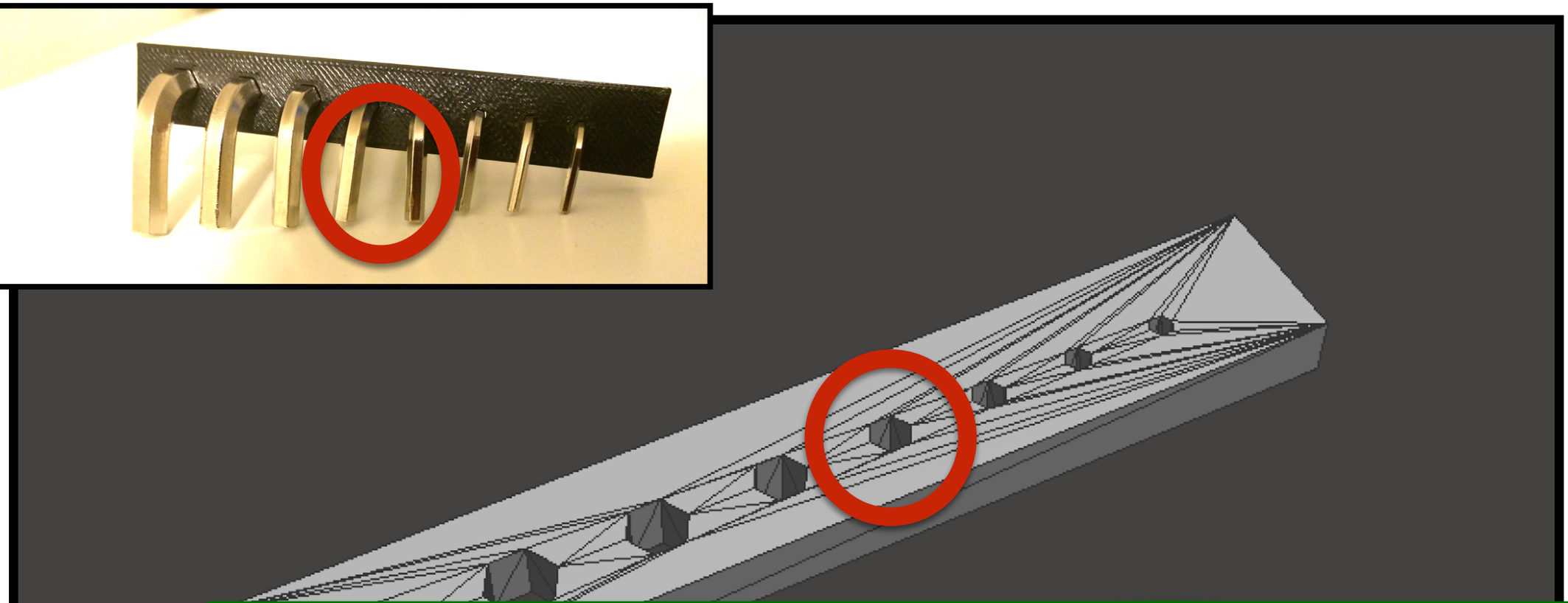
A mesh for a hex holder



Simple mesh editing broke model

- No abstraction
- Move around the vertices manually

A mesh for a hex holder



CAD (Computer Aided Design) editing is easier

- Higher level of abstraction
- Easier to visualize outcome

A mesh for a hex holder

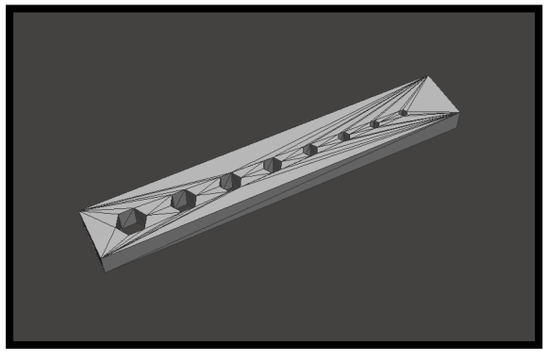
Automatically infer CAD from Mesh!

$: mesh \rightarrow cad$

Higher level of abstraction

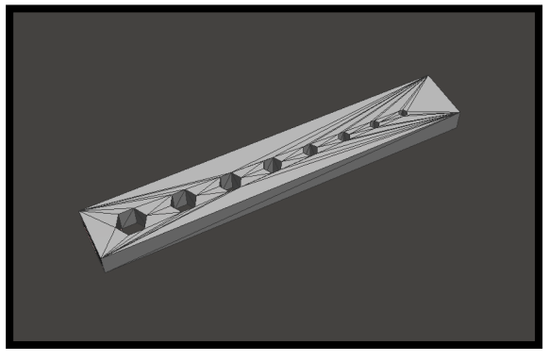
Easier to visualize outcome

Mesh

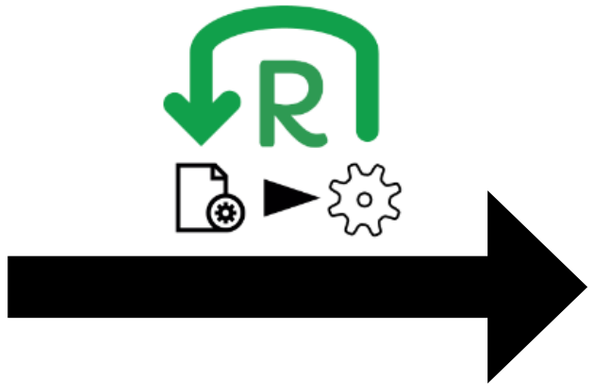


1600 LOC

Mesh



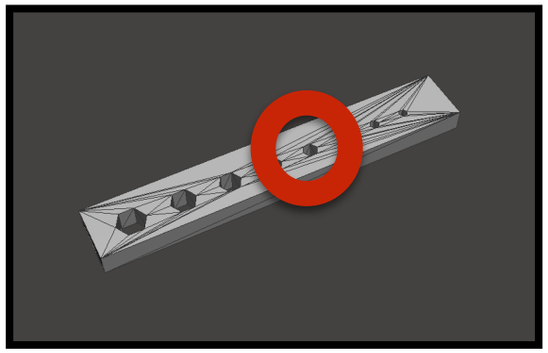
1600 LOC



```
difference (
    scale (97.0, 25.0, 5.0)  cube

     trans (49.0, 13.0, 2.5) (
         scale (7.0, 6.06, 5.0) (
             polyhedron 6 ))
     ...
)
```

80 LOC

Mesh



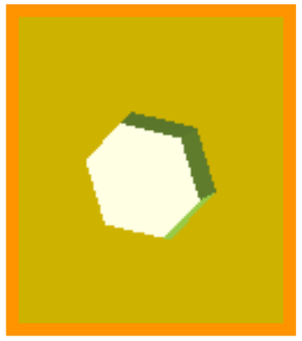
1600 LOC

```
difference (
    scale (97.0, 25.0, 5.0)  cube

    trans (49.0, 13.0, 2.5) (
        scale (7.0, 6.06, 5.0) (
            polyhedron 6 ))
    ...
)
```

80 LOC

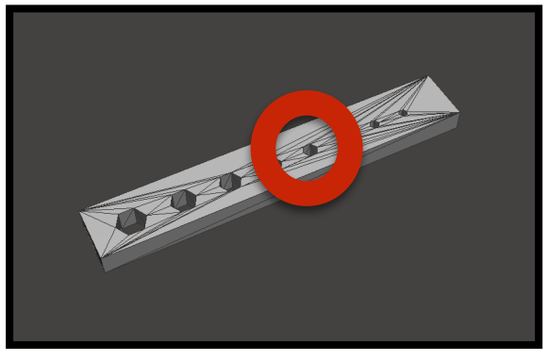
modify

```
difference (
   scale (97.0, 25.0, 5.0)  cube

  trans (49.0, 13.0, 2.5) (
    scale (7.0, 6.06, 5.0) (
     rotateZ (35.0)
        polyhedron 6 )))
...
)
```

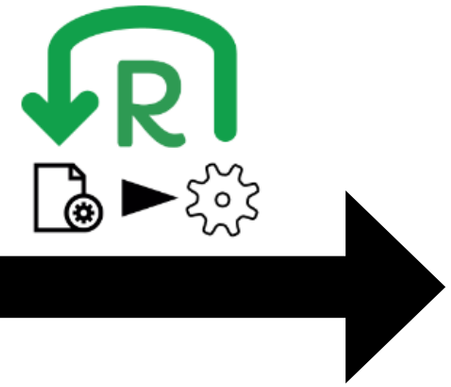


Rotating by 35 degrees solved the problem!

Mesh



1600 LOC

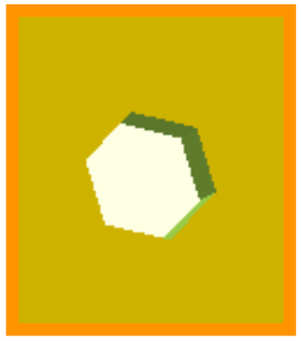


```
difference (
    scale (97.0, 25.0, 5.0)  cube

    trans (49.0, 13.0, 2.5) (
        scale (7.0, 6.06, 5.0) (
            polyhedron 6 ))
    ...
)
```

80 LOC

modify

```
difference (
    scale (97.0, 25.0, 5.0)  cube

    trans (49.0, 13.0, 2.5) (
        scale (7.0, 6.06, 5.0) (
            rotateZ (35.0)
            polyhedron 6 )))
...
)
```



Rotating by 35 degrees solved the problem!

print

success

Mesh


```
difference (
    scale (97.0, 25.0, 5.0)  cube

    trans (49.0, 13.0, 2.5) (
        scale (7.0, 6.06, 5.0) (
            polyhedron 6 ))
```

Automatically infer CAD from Mesh!

$: mesh \rightarrow cad$

print

success

```
trans (49.0, 13.0, 2.5) (
    scale (7.0, 6.06, 5.0) (
        rotateZ (35.0)
            polyhedron 6 )))
...
)
```

How?

Key insight: view the computational fabrication pipeline as a compiler

How?

Key insight: view the computational fabrication pipeline as a compiler

PL foundations applied to computational fabrication to provide *clarity* and *usefulness*

Clarity

Denotational semantics, inductive definitions

Proof of correctness of a compiler from CAD to mesh

Usefulness

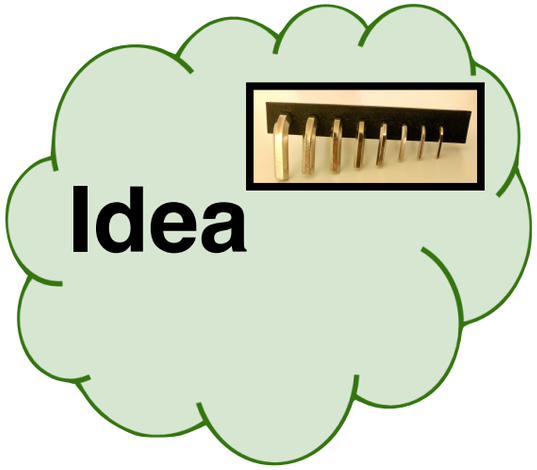
Program synthesis to reverse engineer CAD from mesh

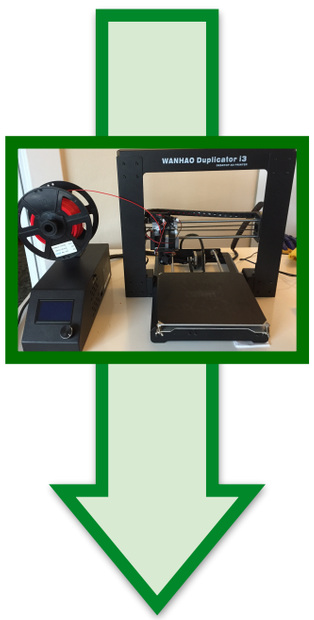
Talk Outline

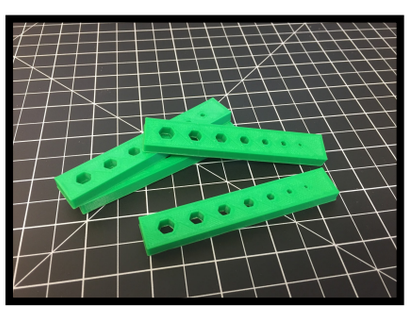
- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

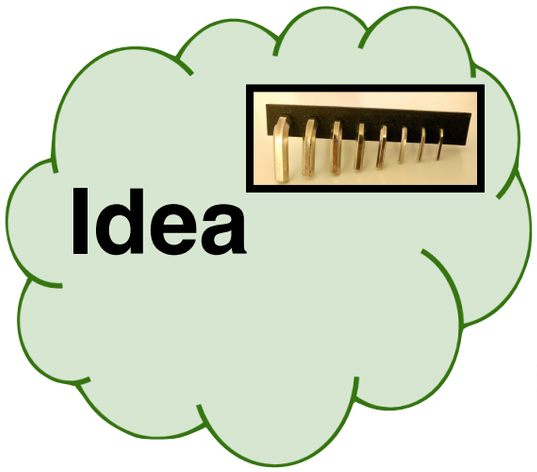




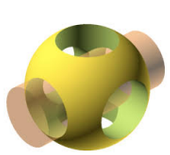


3D printing workflow

Idea



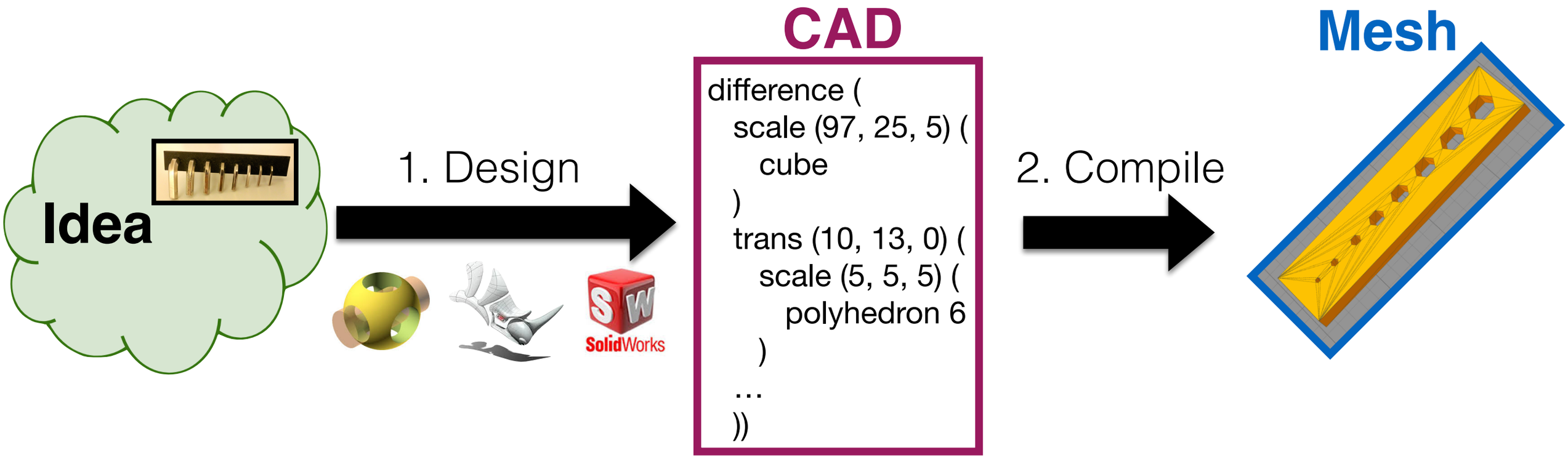
1. Design

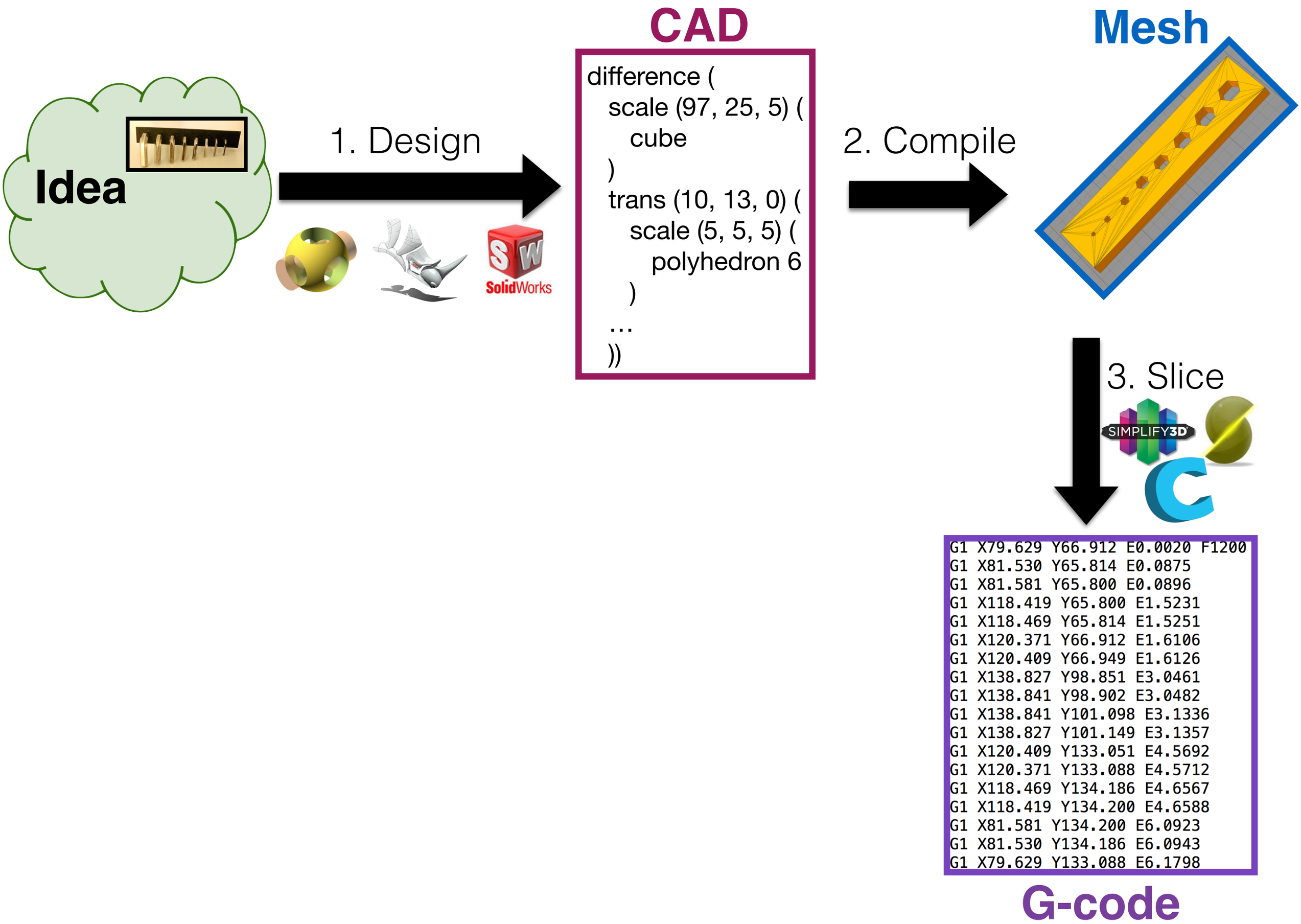


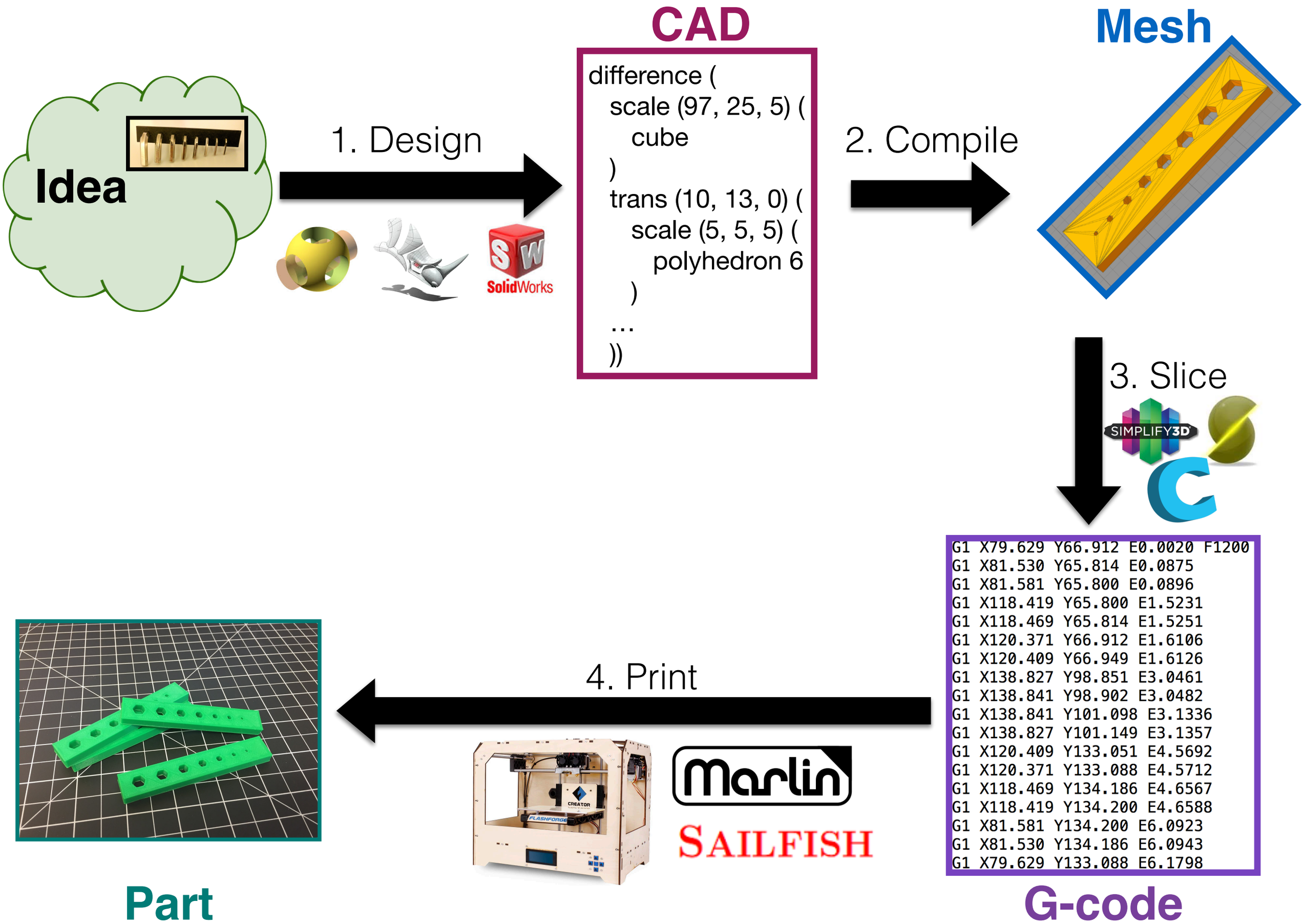
SolidWorks

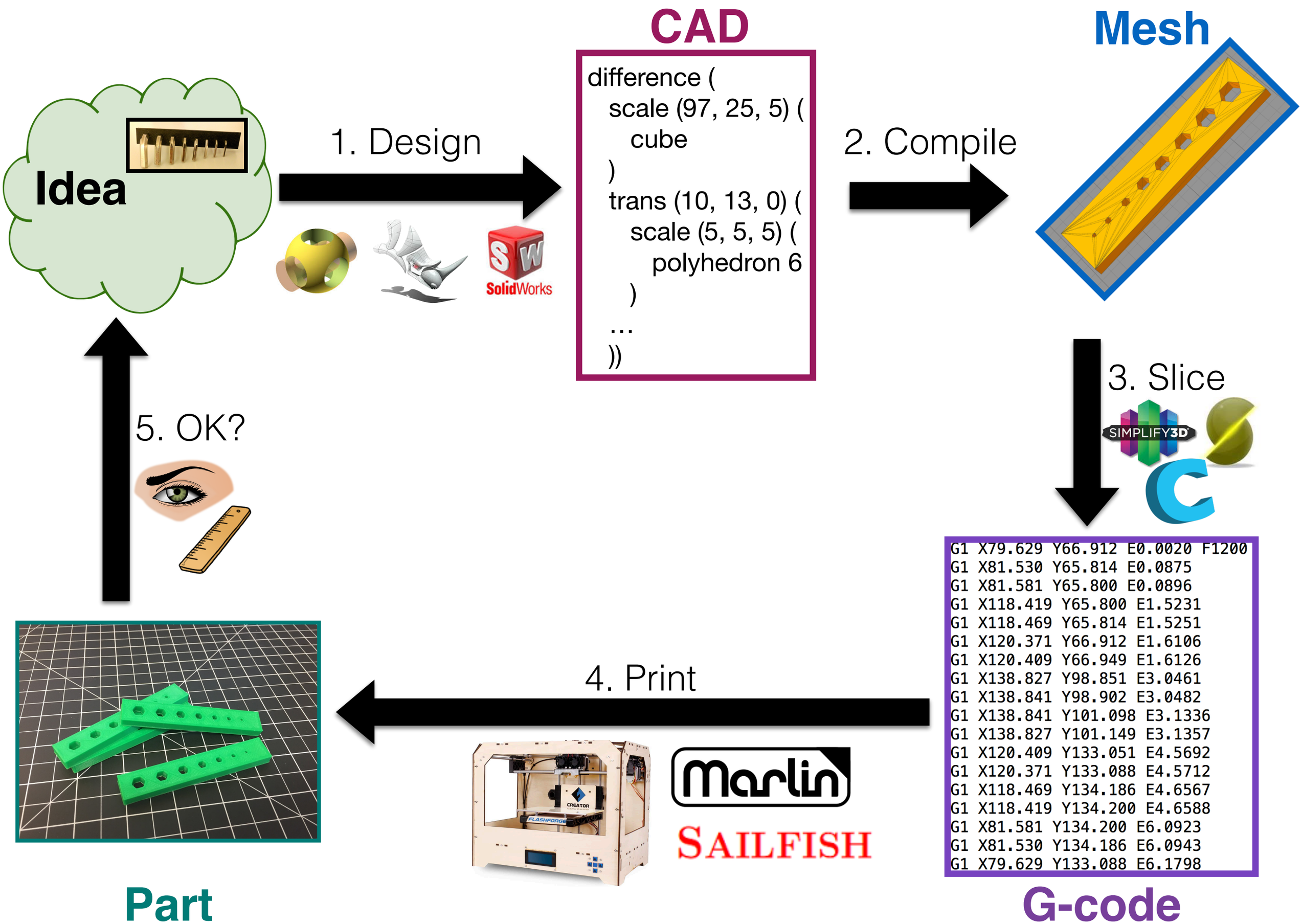
CAD

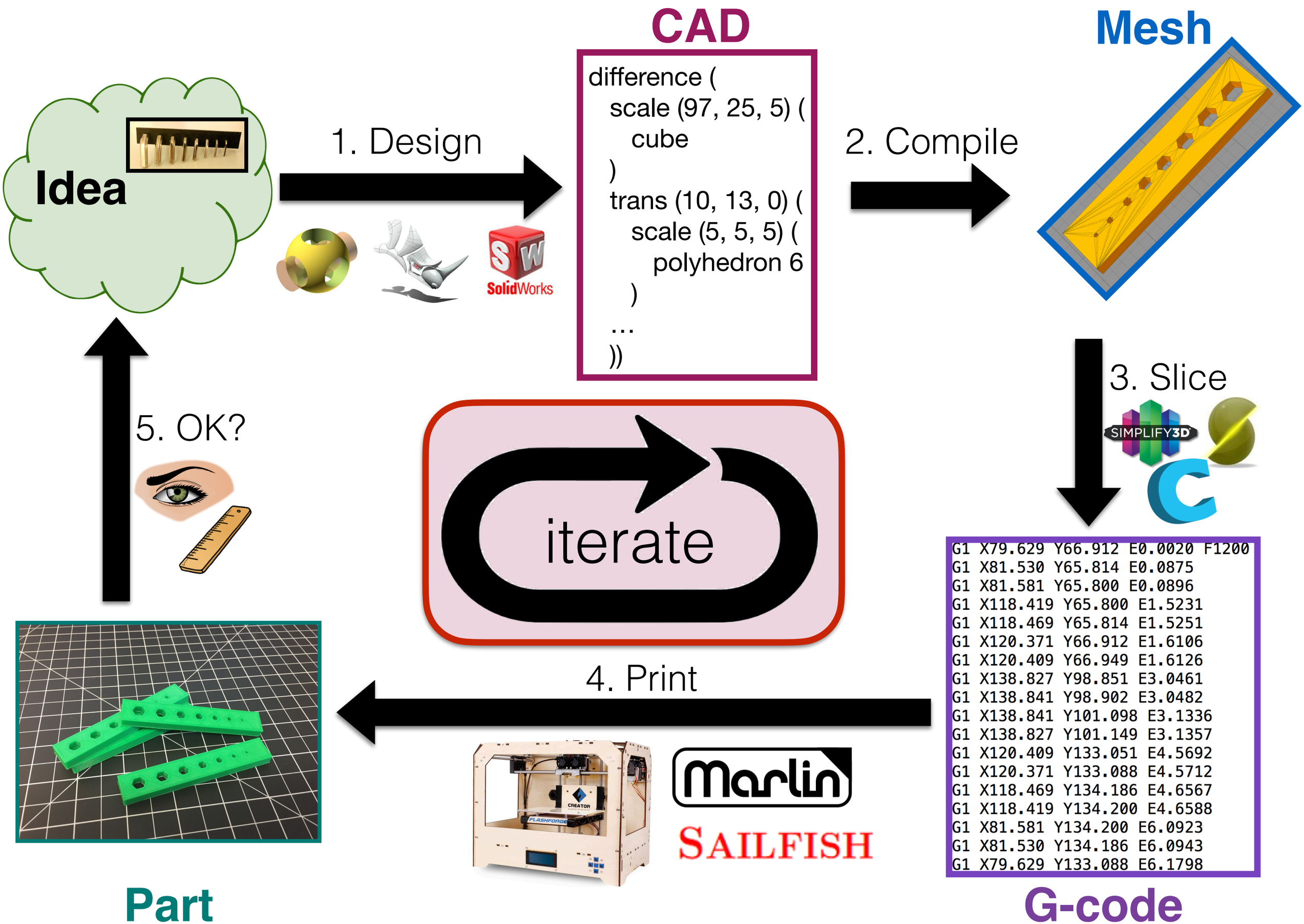
```
difference (
   scale (97, 25, 5) (
      cube
   )
   trans (10, 13, 0) (
      scale (5, 5, 5) (
         polyhedron 6
      )
   ...
   ))
```

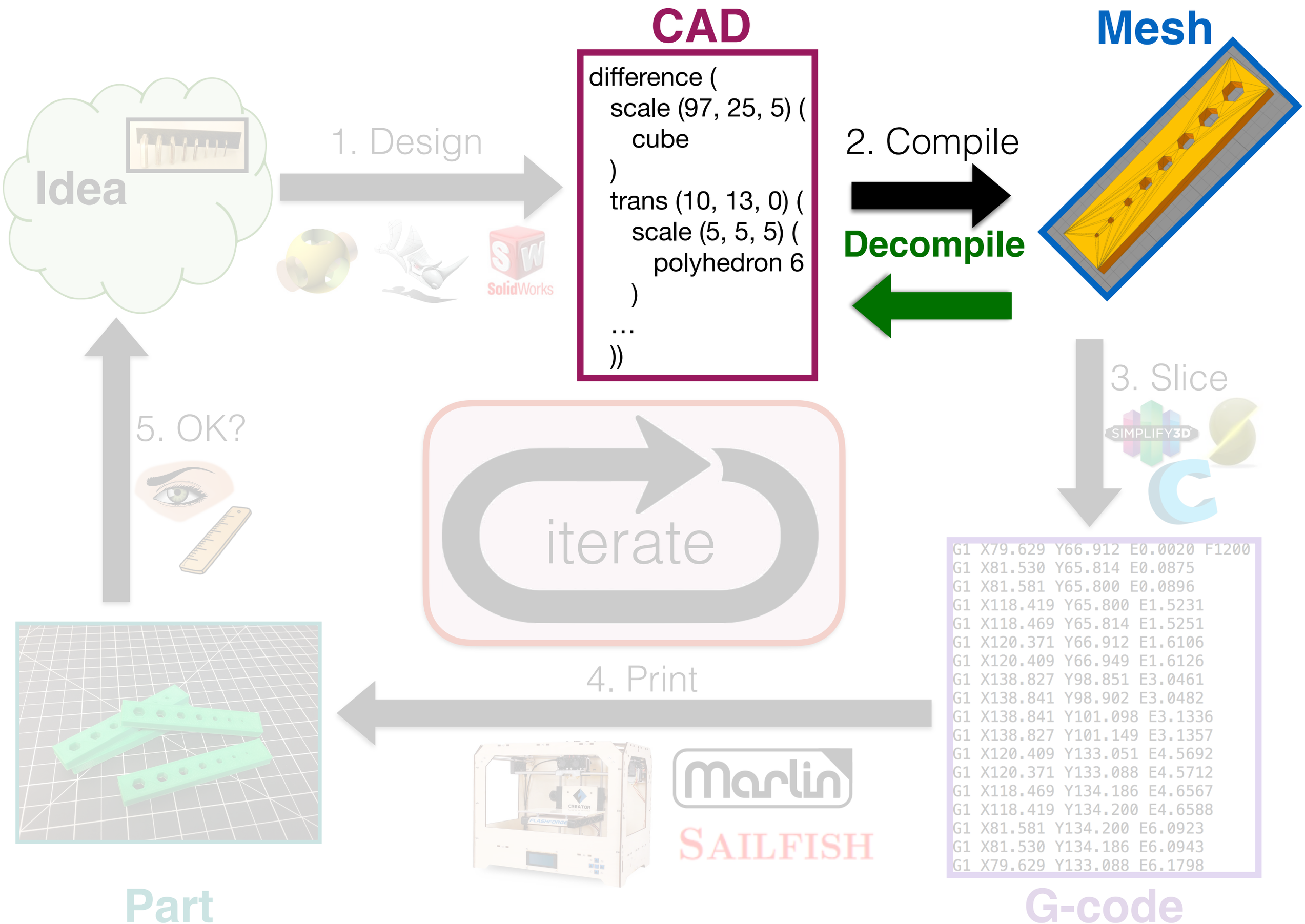

Idea
1. Design
SolidWorks
CAD
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
2. Compile
Mesh


Idea
1. Design
SolidWorks
CAD
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
2. Compile
Mesh
3. Slice
SIMPLIFY3D
G1 X79.629 Y66.912 E0.0020 F1200
G1 X81.530 Y65.814 E0.0875
G1 X81.581 Y65.800 E0.0896
G1 X118.419 Y65.800 E1.5231
G1 X118.469 Y65.814 E1.5251
G1 X120.371 Y66.912 E1.6106
G1 X120.409 Y66.949 E1.6126
G1 X138.827 Y98.851 E3.0461
G1 X138.841 Y98.902 E3.0482
G1 X138.841 Y101.098 E3.1336
G1 X138.827 Y101.149 E3.1357
G1 X120.409 Y133.051 E4.5692
G1 X120.371 Y133.088 E4.5712
G1 X118.469 Y134.186 E4.6567
G1 X118.419 Y134.200 E4.6588
G1 X81.581 Y134.200 E6.0923
G1 X81.530 Y134.186 E6.0943
G1 X79.629 Y133.088 E6.1798
G-code


Idea
1. Design
SolidWorks
CAD
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
2. Compile
Mesh
3. Slice
SIMPLIFY3D
G1 X79.629 Y66.912 E0.0020 F1200
G1 X81.530 Y65.814 E0.0875
G1 X81.581 Y65.800 E0.0896
G1 X118.419 Y65.800 E1.5231
G1 X118.469 Y65.814 E1.5251
G1 X120.371 Y66.912 E1.6106
G1 X120.409 Y66.949 E1.6126
G1 X138.827 Y98.851 E3.0461
G1 X138.841 Y98.902 E3.0482
G1 X138.841 Y101.098 E3.1336
G1 X138.827 Y101.149 E3.1357
G1 X120.409 Y133.051 E4.5692
G1 X120.371 Y133.088 E4.5712
G1 X118.469 Y134.186 E4.6567
G1 X118.419 Y134.200 E4.6588
G1 X81.581 Y134.200 E6.0923
G1 X81.530 Y134.186 E6.0943
G1 X79.629 Y133.088 E6.1798
G-code
4. Print
Marlin
SAILFISH
Part


Idea
1. Design
SolidWorks
CAD
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
2. Compile
Mesh
3. Slice
SIMPLIFY3D
G1 X79.629 Y66.912 E0.0020 F1200
G1 X81.530 Y65.814 E0.0875
G1 X81.581 Y65.800 E0.0896
G1 X118.419 Y65.800 E1.5231
G1 X118.469 Y65.814 E1.5251
G1 X120.371 Y66.912 E1.6106
G1 X120.409 Y66.949 E1.6126
G1 X138.827 Y98.851 E3.0461
G1 X138.841 Y98.902 E3.0482
G1 X138.841 Y101.098 E3.1336
G1 X138.827 Y101.149 E3.1357
G1 X120.409 Y133.051 E4.5692
G1 X120.371 Y133.088 E4.5712
G1 X118.469 Y134.186 E4.6567
G1 X118.419 Y134.200 E4.6588
G1 X81.581 Y134.200 E6.0923
G1 X81.530 Y134.186 E6.0943
G1 X79.629 Y133.088 E6.1798
G-code
4. Print
Marlin
SAILFISH
5. OK?
Part


Idea
1. Design
SolidWorks
CAD
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
2. Compile
Mesh
3. Slice
SIMPLIFY3D
G1 X79.629 Y66.912 E0.0020 F1200
G1 X81.530 Y65.814 E0.0875
G1 X81.581 Y65.800 E0.0896
G1 X118.419 Y65.800 E1.5231
G1 X118.469 Y65.814 E1.5251
G1 X120.371 Y66.912 E1.6106
G1 X120.409 Y66.949 E1.6126
G1 X138.827 Y98.851 E3.0461
G1 X138.841 Y98.902 E3.0482
G1 X138.841 Y101.098 E3.1336
G1 X138.827 Y101.149 E3.1357
G1 X120.409 Y133.051 E4.5692
G1 X120.371 Y133.088 E4.5712
G1 X118.469 Y134.186 E4.6567
G1 X118.419 Y134.200 E4.6588
G1 X81.581 Y134.200 E6.0923
G1 X81.530 Y134.186 E6.0943
G1 X79.629 Y133.088 E6.1798
G-code
4. Print
Marlin
SAILFISH
CREATOR
FLASHFORGE
Part
5. OK?
iterate


CAD
Mesh
difference (
scale (97, 25, 5) (
cube
)
trans (10, 13, 0) (
scale (5, 5, 5) (
polyhedron 6
)
...
))
Idea
1. Design
2. Compile
Decompile
3. Slice
5. OK?
iterate
4. Print
G1 X79.629 Y66.912 E0.0020 F1200
G1 X81.530 Y65.814 E0.0875
G1 X81.581 Y65.800 E0.0896
G1 X118.419 Y65.800 E1.5231
G1 X118.469 Y65.814 E1.5251
G1 X120.371 Y66.912 E1.6106
G1 X120.409 Y66.949 E1.6126
G1 X138.827 Y98.851 E3.0461
G1 X138.841 Y98.902 E3.0482
G1 X138.841 Y101.098 E3.1336
G1 X138.827 Y101.149 E3.1357
G1 X120.409 Y133.051 E4.5692
G1 X120.371 Y133.088 E4.5712
G1 X118.469 Y134.186 E4.6567
G1 X118.419 Y134.200 E4.6588
G1 X81.581 Y134.200 E6.0923
G1 X81.530 Y134.186 E6.0943
G1 X79.629 Y133.088 E6.1798
SolidWorks
SIMPLIFY3D
Marlin
SAILFISH
Part
G-code

Talk Outline

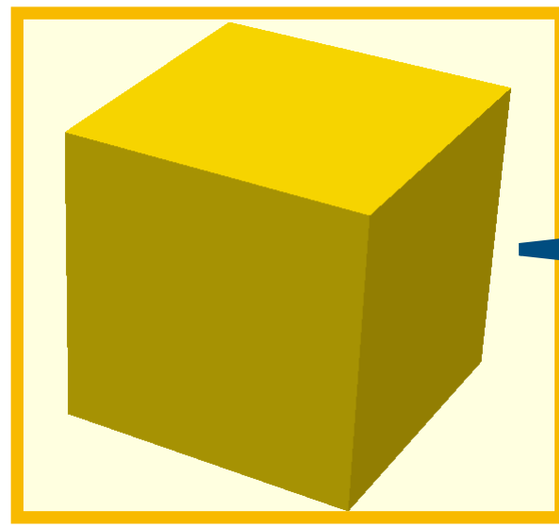
- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
- *Usefulness*: the first decompiler from mesh to CAD

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh
  - Inductive compiler definition
  - Proof of correctness for compiler
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

CAD

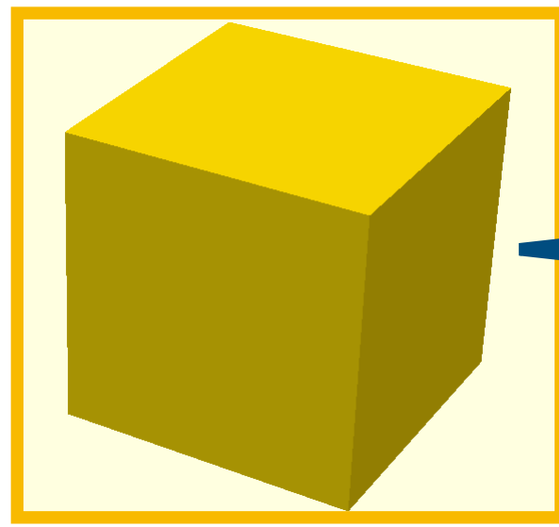
cube



3D primitive representing a cube of unit length

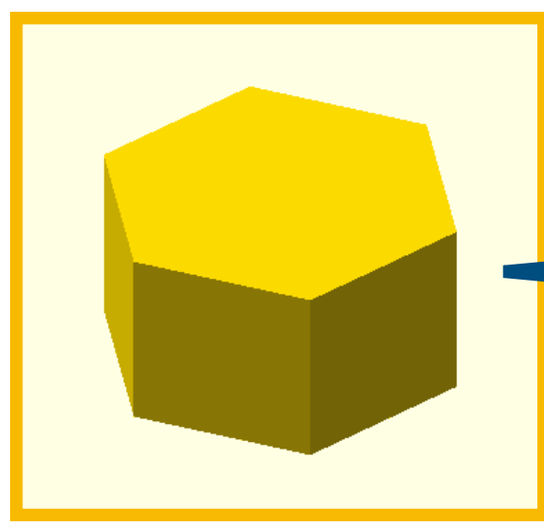
CAD

cube



3D primitive representing a cube of unit length

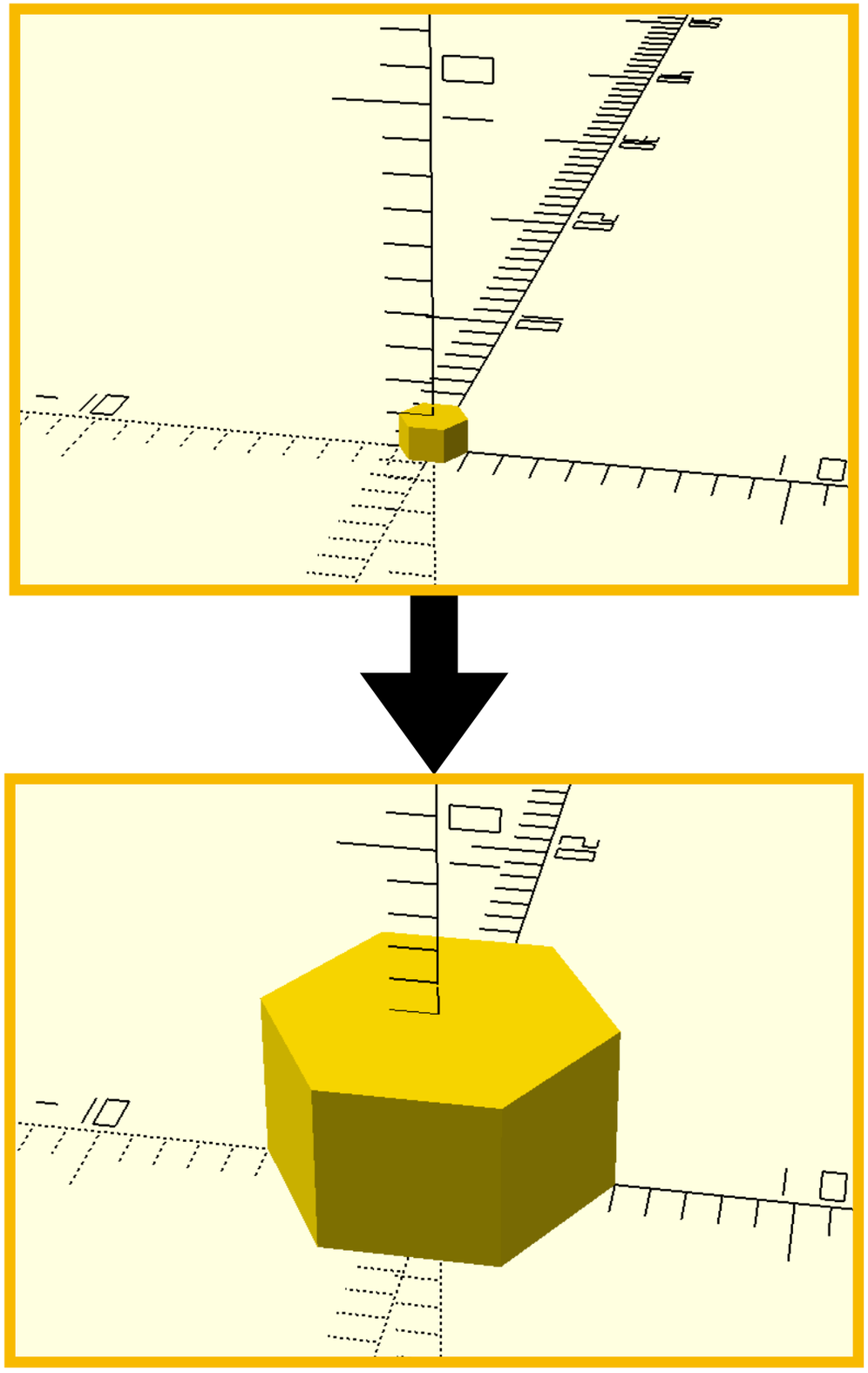
polyhedron 6



3D primitive representing a hexagonal prism

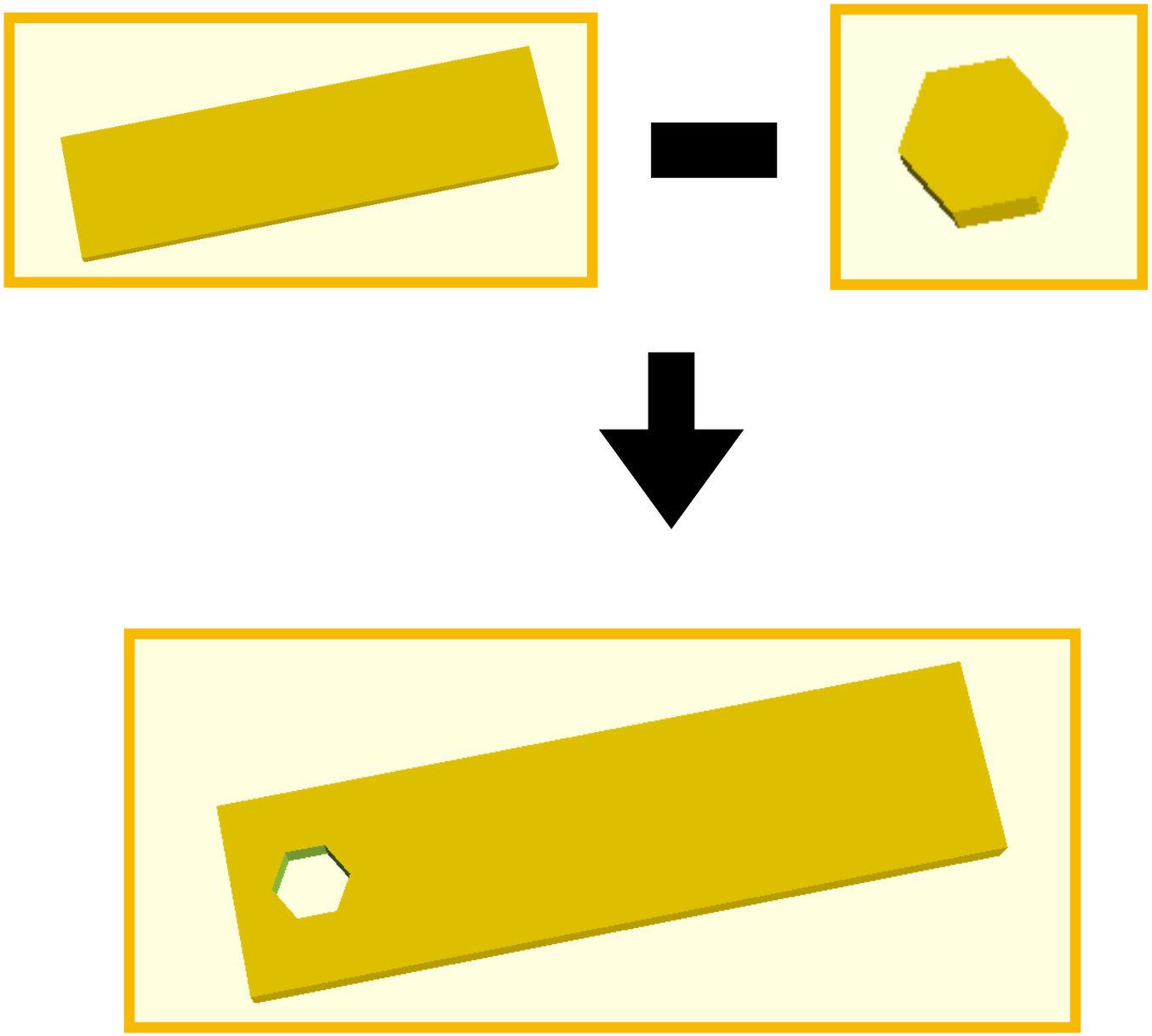
CAD

scale (5, 5, 5) polyhedron **6**



CAD

```
difference (
   scale (97, 25, 5) (
      cube
   )
   trans (10, 13, 0) (
      scale (5, 5, 5) (
         polyhedron 6
      )
)
```



CAD

rotateX
rotateY
rotateZ
translation
scale

$$
\begin{array}{lcl}
c & ::= & \mathit{Empty} \\
 & | & \mathit{Cube} \\
 & | & \mathit{Polyhedron}\ \mathbb{N} \\
 & | & \mathit{Affine}\ \mathbb{R}^{3\times 3}\ \mathbb{R}^{3}\ c \\
 & | & \mathit{Binop}\ \mathit{op}\ c\ c
\end{array}
$$

Linear transformation matrix: $\mathbb{R}^{3\times 3}$

Vector: $\mathbb{R}^{3}$

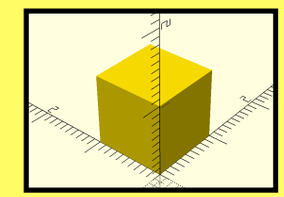
$$
\begin{array}{lcl}
\mathit{op} & ::= & \mathit{Union} \\
 & | & \mathit{Difference} \\
 & | & \mathit{Intersection}
\end{array}
$$

CAD: Denotational semantics

$$[\![c]\!]_{cad} : \{all\ points\ inside\ c\}$$

$$[\![Empty]\!]_{cad} = \{\}$$

$$[\![Cube]\!]_{cad} = \{(0,1)^3\}$$

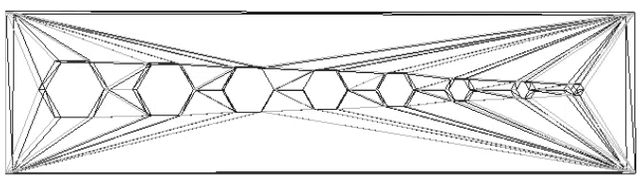


$$[\![Affine\ p\ q\ c]\!]_{cad} = \{pv + q | v \in [\![c]\!]_{cad}\}$$

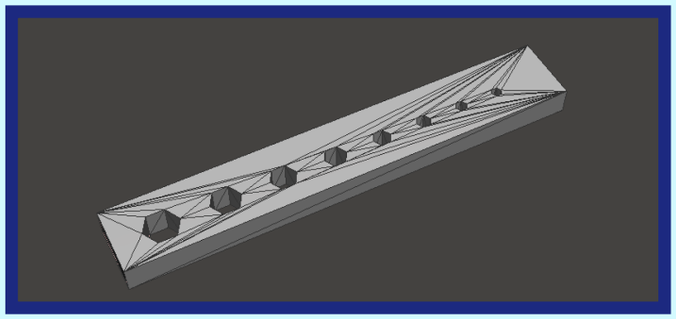
$$[\![Binop\ Union\ c_1\ c_2]\!]_{cad} = [\![c_1]\!]_{cad} \cup [\![c_2]\!]_{cad}$$

Regular opens

Mesh



List of faces



A face is a triangular plane

Convenient for geometric operations

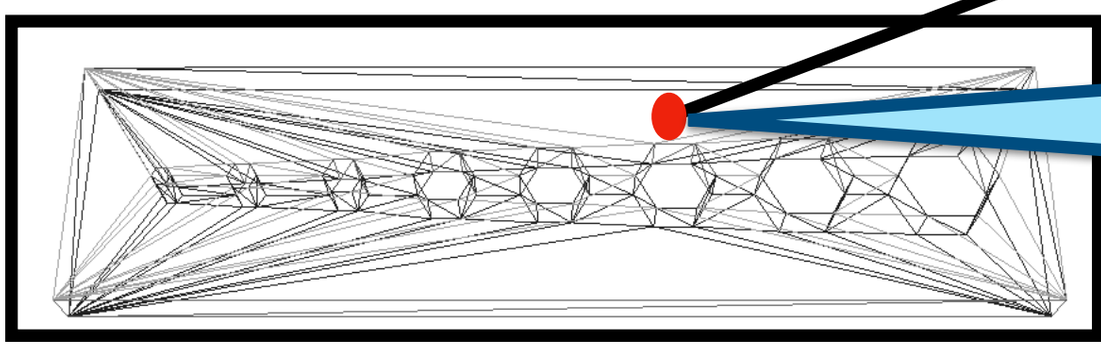
CAD tool agnostic

Standard format for 3D models

Widely shared in online repositories

Mesh: Denotational semantics

$$[\![m]\!]_{mesh} : \{all\ points\ inside\ m\}$$



pt is inside m iff a ray starting at pt crosses an ***odd*** number of faces of m

Key insight: view the computational fabrication pipeline as a compiler

CAD is based on *solid* geometry, mesh is based on *surface* geometry

We need tools that relate these two different representations

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition
  - Proof of correctness for compiler
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

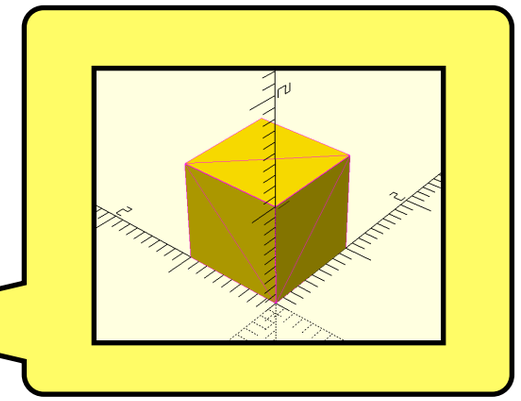
Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition
  - Proof of correctness for compiler
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Compiling CAD to mesh

$$compile : c \rightarrow m$$

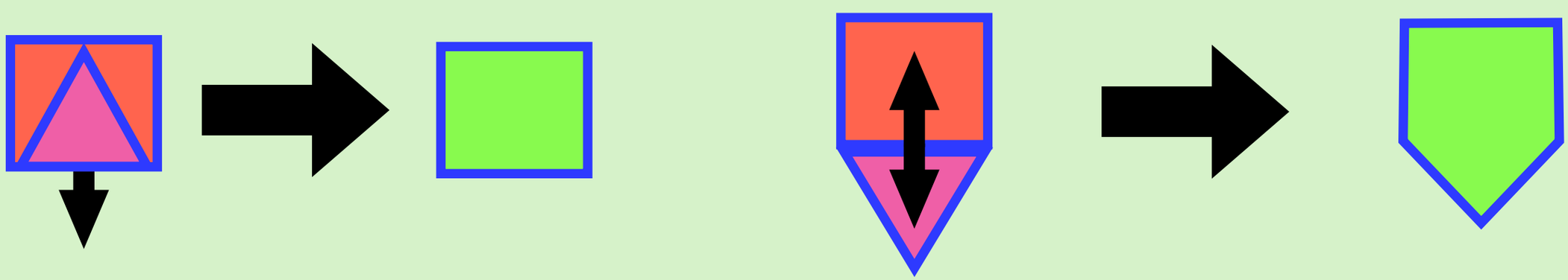
$$compile\ cube = m_{cube}$$



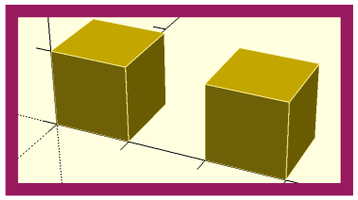
$$compile\ (Affine\ p\ q\ c) = map_{vertex}(\lambda v\,.\,pv + q)\ (compile\ c)$$

$$compile\ (Binop\ \cup\ c_1\ c_2) = meshBinop(\cup)\ (compile(c_1), compile(c_2))$$

Regular opens at mesh level too: check normals!

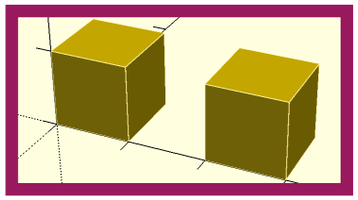


Compiler on an example



union
  cube
  trans (2, 0, 0) cube

Compiler on an example

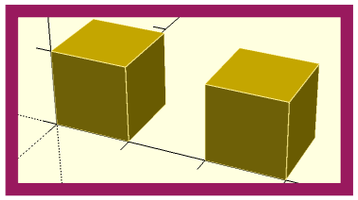


union
  cube
  trans (2, 0, 0) cube

$\rightarrow c\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$

$[\text{cube}]$

Compiler on an example



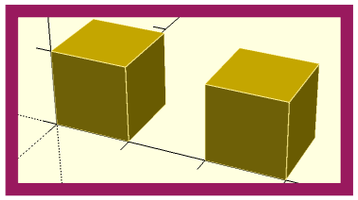
union
  cube
  trans (2, 0, 0) cube

$\rightarrow c\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$

$[\text{cube}]$

Evaluation context

Compiler on an example

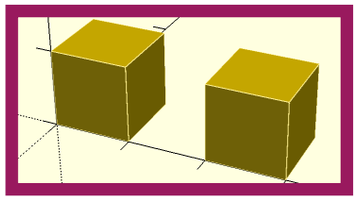


union
cube
trans (2, 0, 0) cube

$\rightarrow c\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]\left[\text{cube}\right] \rightarrow c\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]\left[\text{Mesh } [m_{cube}]\right]$

Evaluation context

Compiler on an example



union
cube
trans (2, 0, 0) cube

$\rightarrow c$

union
[.]
trans (2, 0, 0) cube

[cube]

$\rightarrow c$

union
[.]
trans (2, 0, 0) cube

[Mesh [m_{cube}]]

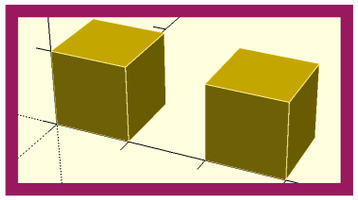
Evaluation context

$\downarrow c$
$*$

union
Mesh [m_{cube}]
[.]

[trans (2, 0, 0) cube]

Compiler on an example



union
cube
trans (2, 0, 0) cube

$\rightarrow c$

$$\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$$

$$[\text{cube}]$$

Evaluation context

$\rightarrow c$

$$\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$$

$$\left[\ \text{Mesh}\ [m_{\text{cube}}]\ \right]$$

$\downarrow c$
$*$

$$\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh}\ [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$$

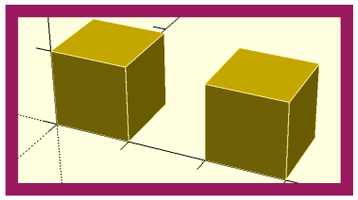
$$\left[\ \text{Mesh}\ [m_{\text{trans (2, 0, 0) cube}}]\ \right]$$

$\leftarrow c$

$$\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh}\ [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$$

$$\left[\ \text{trans (2, 0, 0) cube}\ \right]$$

Compiler on an example



union
cube
trans (2, 0, 0) cube

$\xrightarrow{c}$ $\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$ $[\text{cube}]$

$\xrightarrow{c}$ $\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$ $[\text{Mesh } [m_{\text{cube}}]]$

Evaluation context

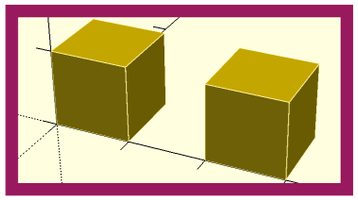
$\downarrow c^{*}$

$\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh } [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$ $[\text{trans (2, 0, 0) cube}]$

$\xleftarrow{c}$ $\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh } [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$ $[\text{Mesh } [m_{\text{trans (2, 0, 0) cube}}]]$

$\xleftarrow{c}$ **[.]** union
Mesh $[m_{\text{cube}}]$
Mesh $[m_{\text{trans (2, 0, 0) cube}}]$

Compiler on an example



union
cube
trans (2, 0, 0) cube

$\rightarrow_c$

$$\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$$

$$\left[\text{cube}\right]$$

Evaluation context

$\rightarrow_c$

$$\left[\begin{array}{l}\text{union}\\ \quad [.]\\ \quad \text{trans (2, 0, 0) cube}\end{array}\right]$$

$$\left[\text{Mesh } [m_{\text{cube}}]\right]$$

$\downarrow_{c}^{*}$

$$\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh } [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$$

$$\left[\text{Mesh } [m_{\text{trans (2, 0, 0) cube}}]\right]$$

$\leftarrow_c$

$$\left[\begin{array}{l}\text{union}\\ \quad \text{Mesh } [m_{\text{cube}}]\\ \quad [.]\end{array}\right]$$

$$\left[\text{trans (2, 0, 0) cube}\right]$$

$\leftarrow_c$

[.] union
Mesh $[m_{\text{cube}}]$
Mesh $[m_{\text{trans (2, 0, 0) cube}}]$

$\downarrow_c$

Mesh [...]

Compiler on an example

union
cube
trans (2, 0, 0) cube

$\rightarrow_c$

Evaluation contex

[.] union
Mesh [m_{cube}]
Mesh [$m_{trans\ (2,\ 0,\ 0)\ cube}$]

$\downarrow_c$

Mesh [...]

TRICKY when faces overlap!

Need to split meshes w.r.t one another to resolve overlaps

Need to determine which faces from $compile(c_1)$ and $compile(c_2)$ to keep in the final mesh

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition ✓
  - Proof of correctness for compiler
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition ✓
  - Proof of correctness for compiler
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Compiler Proof

$$Thm : \forall e, [\![compile\ e]\!]_{mesh} = [\![e]\!]_{cad}$$

Proof: By induction on e

- Case Primitives: …
- Case Affine Transformations: …
- Case Union:

Compiler Proof

$$Thm : \forall e, [\![compile\ e]\!]_{mesh} = [\![e]\!]_{cad}$$

Proof: By induction on e

- Case Primitives: …
- Case Affine Transformations: …
- Case Union:

Union lemma

$$[\![meshBinop\ (Union)\ (m_1, m_2)]\!]_{mesh} = [\![m_1]\!]_{mesh} \cup [\![m_2]\!]_{mesh}$$

Compiler Proof

$$Thm : \forall e, [\![compile\ e]\!]_{mesh} = [\![e]\!]_{cad}$$

Union lemma

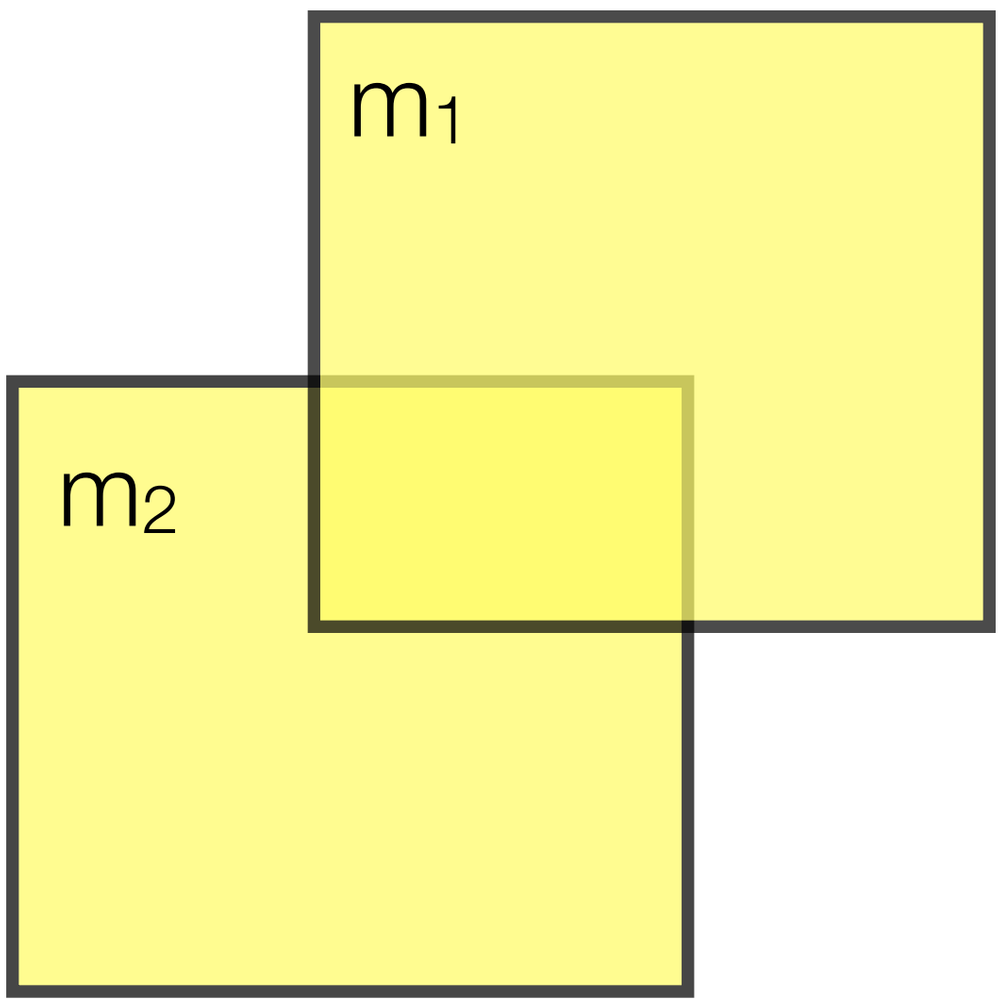
$$[\![meshBinop\ (Union)\ (m_1, m_2)]\!]_{mesh} = [\![m_1]\!]_{mesh} \cup [\![m_2]\!]_{mesh}$$

$$let\ m_3 = mBop\ (Union)\ (m_1, m_2)$$

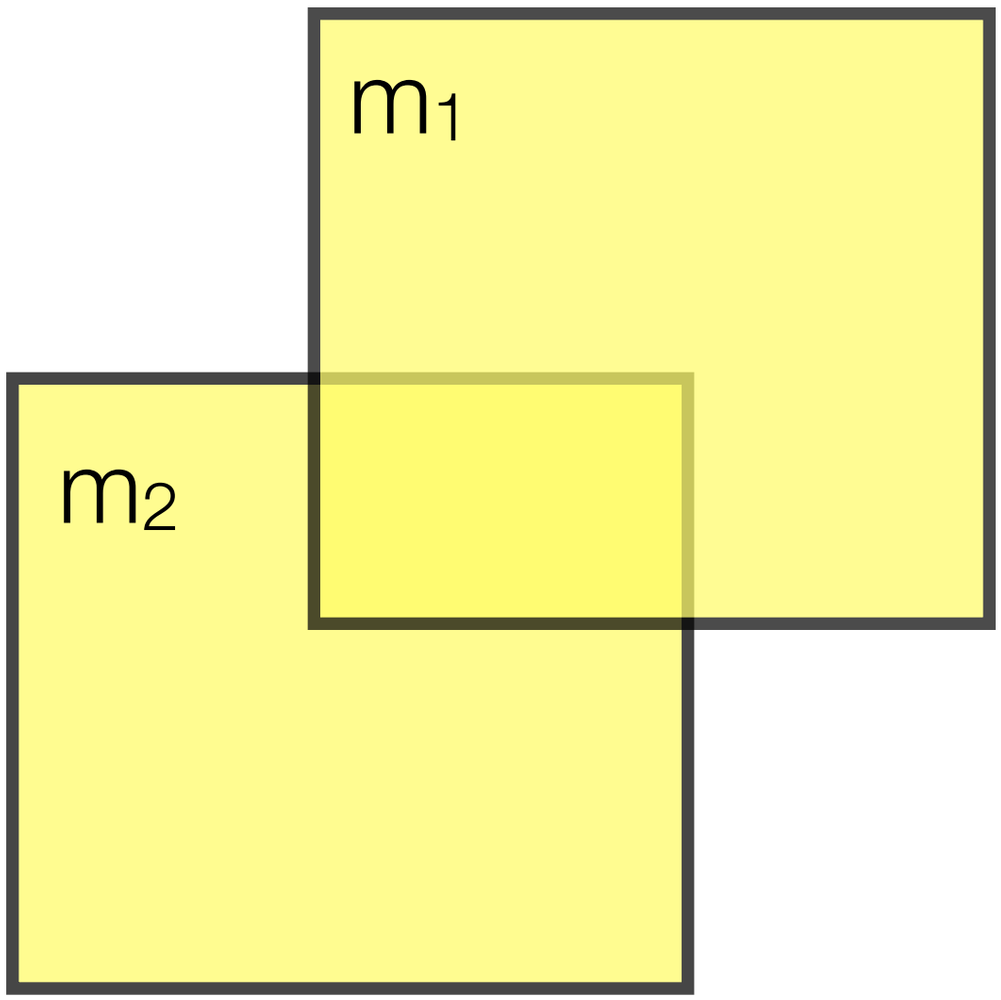
$$pt \in [\![m_3]\!]_{mesh} \iff pt \in [\![m_1]\!]_{mesh} \cup [\![m_2]\!]_{mesh}$$

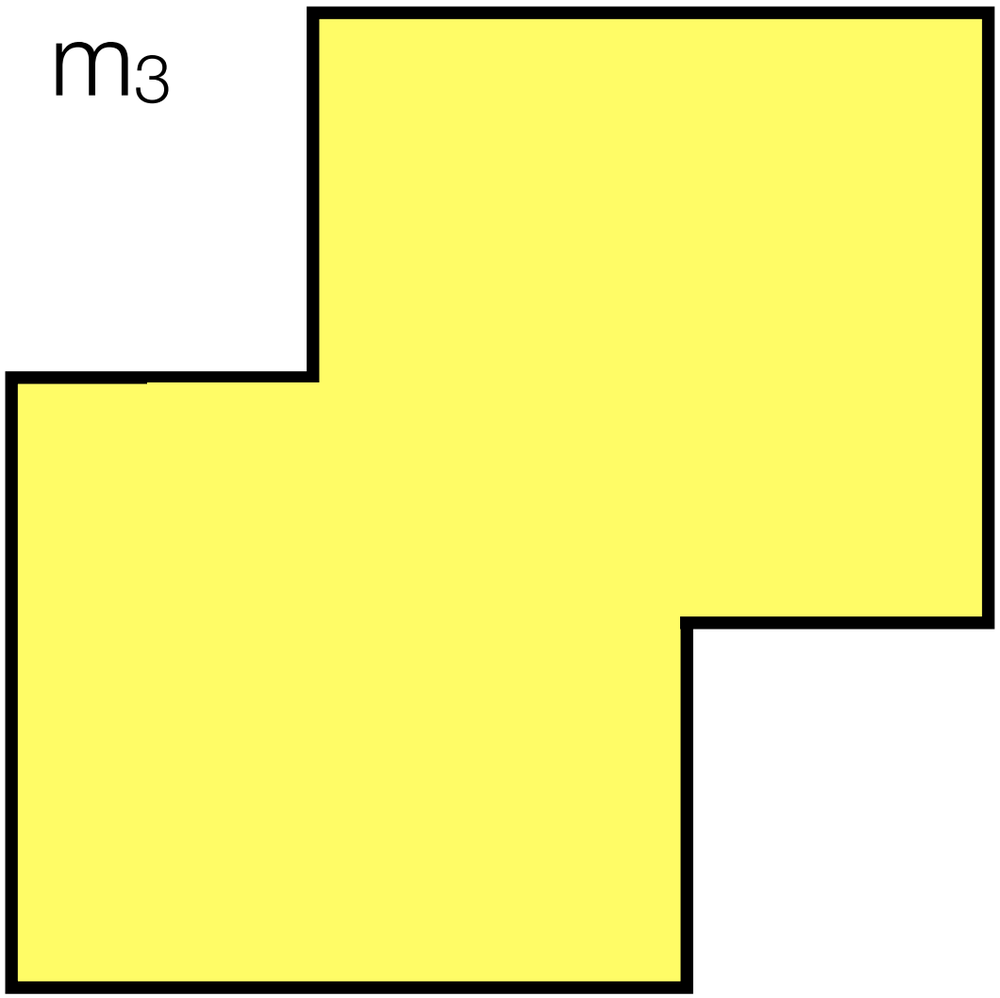
Use Ray casting

Case: overlapping meshes

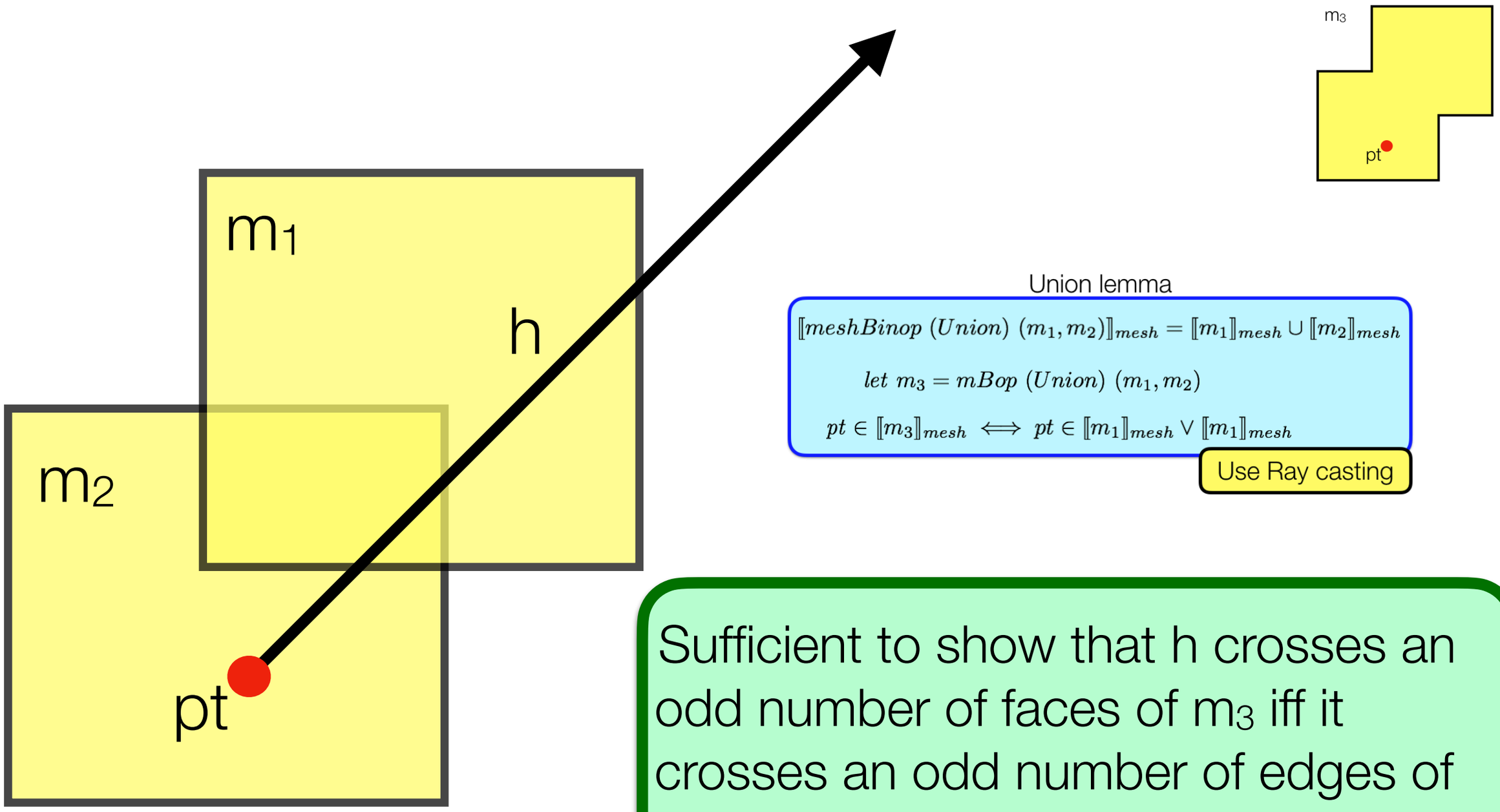


Case: overlapping meshes





Case: overlapping meshes



Union lemma

$$[\![meshBinop\ (Union)\ (m_1, m_2)]\!]_{mesh} = [\![m_1]\!]_{mesh} \cup [\![m_2]\!]_{mesh}$$

$$let\ m_3 = mBop\ (Union)\ (m_1, m_2)$$

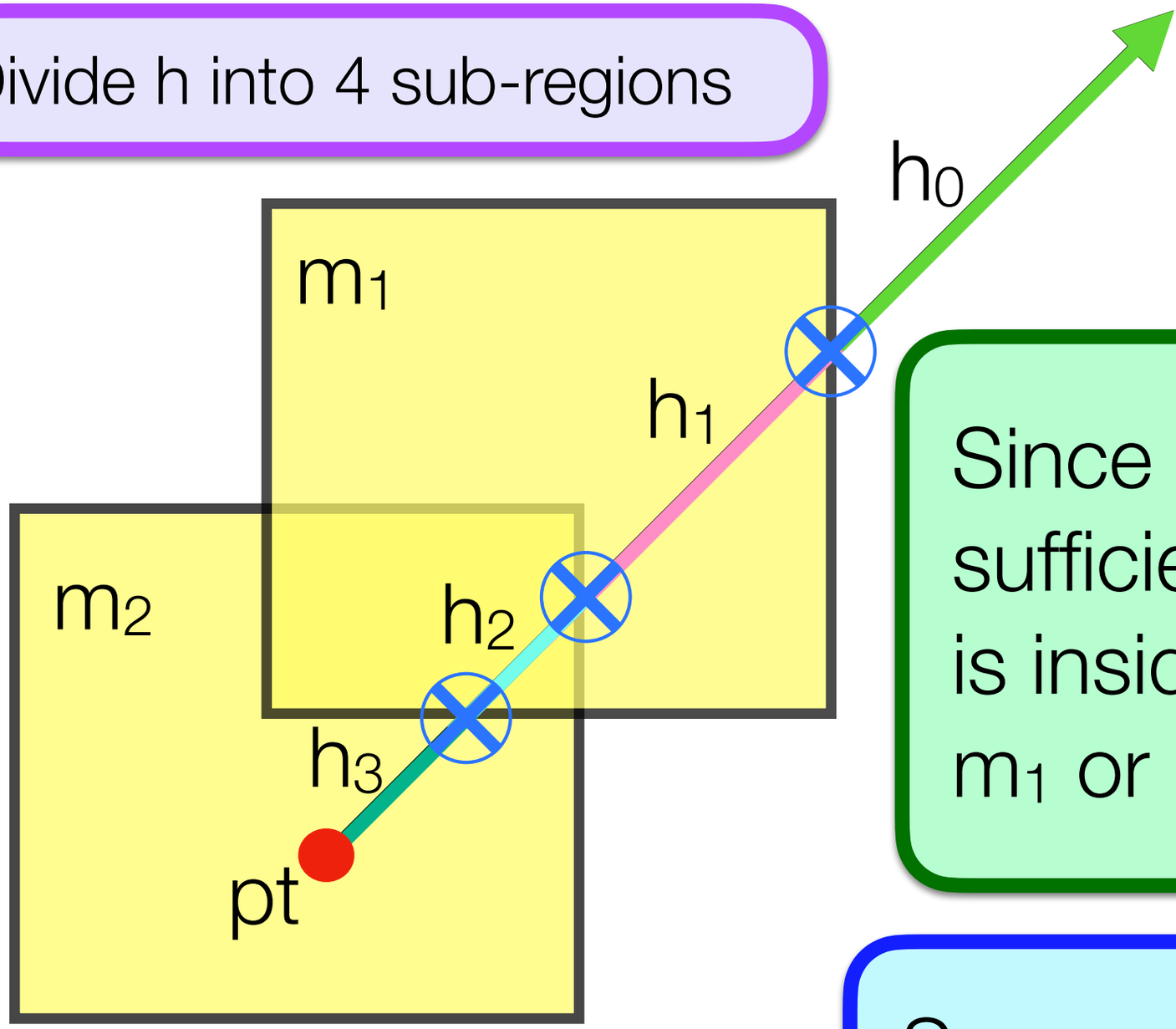
$$pt \in [\![m_3]\!]_{mesh} \iff pt \in [\![m_1]\!]_{mesh} \vee [\![m_1]\!]_{mesh}$$

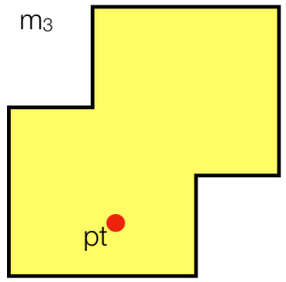
Use Ray casting

Sufficient to show that h crosses an odd number of faces of m_3 iff it crosses an odd number of edges of m_1 or m_2

Case: overlapping meshes

Divide h into 4 sub-regions





Since meshes are split, sufficient to show that h_i is inside m_3 iff it is inside m_1 or m_2

See paper for more details

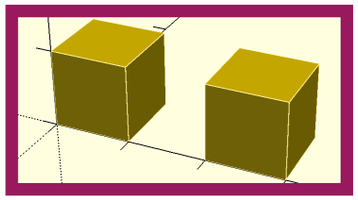
Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition ✓
  - Proof of correctness for compiler ✓
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Talk Outline

- 3D Printing Workflow
- *Clarity* achieved by applying FP to fabrication
  - Denotational semantics for CAD and mesh ✓
  - Inductive compiler definition ✓
  - Proof of correctness for compiler ✓
- *Usefulness*: the <u>first</u> decompiler from mesh to CAD

Synthesis: flip the arrows!



union
cube
trans (2, 0, 0) cube

$\xrightarrow{c}$

union
[.]
trans (2, 0, 0) cube

[cube]

$\xrightarrow{c}$

union
[.]
trans (2, 0, 0) cube

[Mesh [m_{cube}]]

$\xrightarrow[*]{c}$

union
Mesh [m_{cube}]
[.]

[Mesh [$m_{trans\ (2,\ 0,\ 0)\ cube}$]]

$\xleftarrow{c}$

union
Mesh [m_{cube}]
[.]

[trans (2, 0, 0) cube]

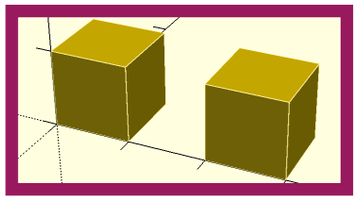
$\xleftarrow{c}$

[.] union
Mesh [m_{cube}]
Mesh [$m_{trans\ (2,\ 0,\ 0)\ cube}$]

$\xrightarrow{c}$

Mesh […]

Synthesis: flip the arrows!



union
cube
trans (2, 0, 0) cube

$\xleftarrow{s}$ $\xrightarrow{c}$

[union
[.]
trans (2, 0, 0) cube]

[cube]

$\xleftarrow{s}$ $\xrightarrow{c}$

[union
[.]
trans (2, 0, 0) cube]

[Mesh [m_{cube}]]

$\downarrow_{c}$ $\boxed{\uparrow_{s}}$

[.] union
Mesh [m_{cube}]
Mesh [$m_{trans\ (2,\ 0,\ 0)\ cube}$]

$\boxed{\rightarrow_{s}}$ $\xleftarrow{c}$

[union
Mesh [m_{cube}]
[.]]

[Mesh [$m_{trans\ (2,\ 0,\ 0)\ cube}$]]

$\boxed{\rightarrow_{s}}$ $\xleftarrow{c}$

[union
Mesh [m_{cube}]
[.]]

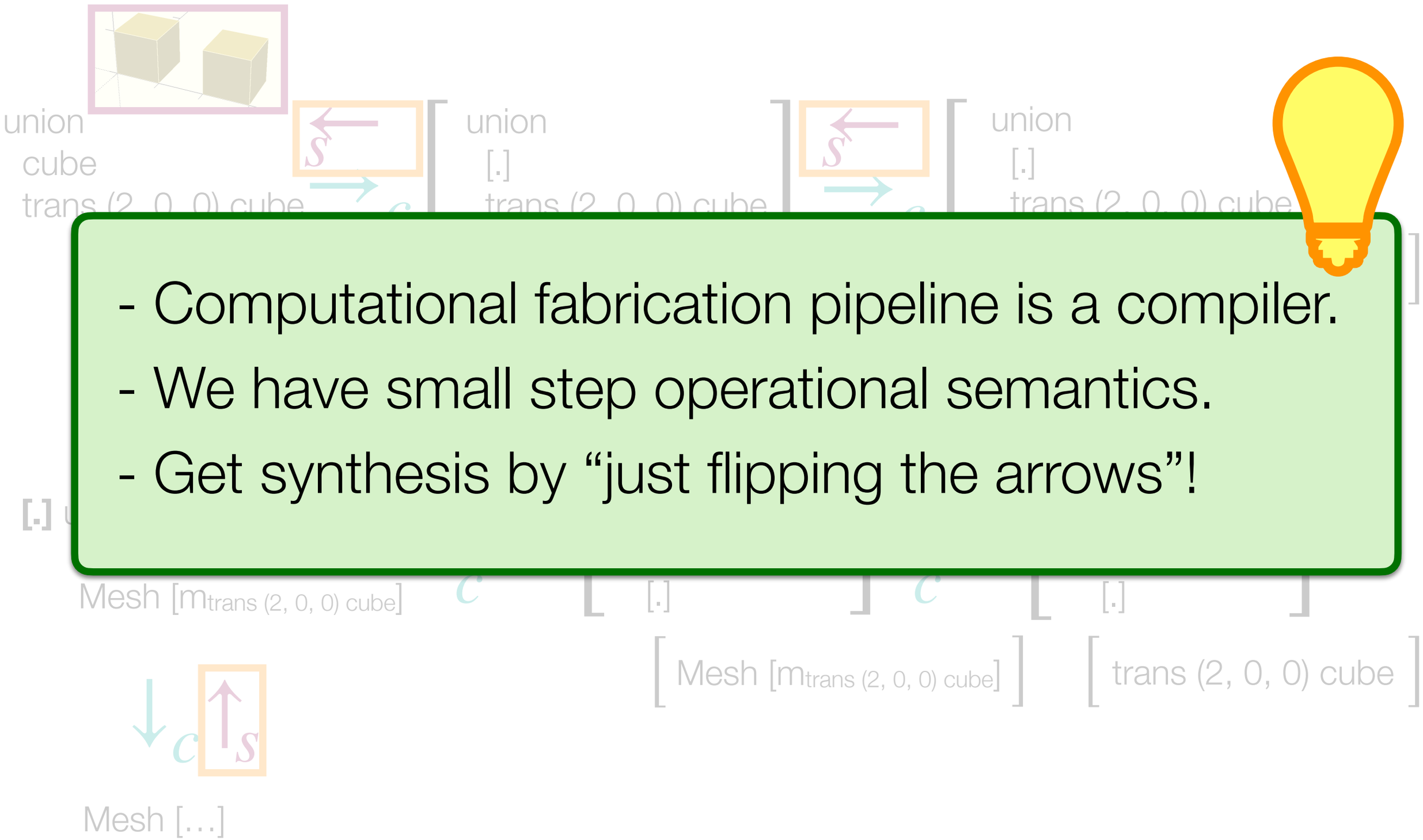
[trans (2, 0, 0) cube]

$\downarrow_{c}$ $\boxed{\uparrow_{s}}$

Mesh [...]

$\downarrow_{*}$ c $\boxed{\uparrow_{s}}$

Synthesis: flip the arrows!



- Computational fabrication pipeline is a compiler.
- We have small step operational semantics.
- Get synthesis by “just flipping the arrows”!

Synthesis: flip the arrows!

union
cube
trans (2, 0

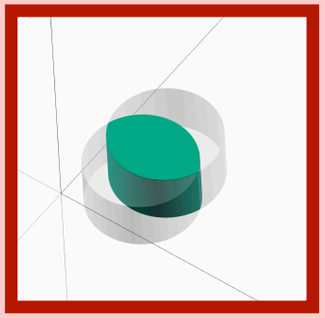
union

union

[.] union
Mesh
Mesh

Mesh [...]

- Loss of information



- Geometric challenges
- Non-deterministic

Synthesis: flip the arrows!

union
cube
trans (2, 0

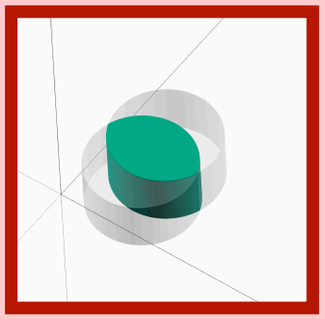
union

[.] union
Mesh
Mesh

Mesh [...]

Evaluation context

- Loss of information



- Geometric challenges
- Non-deterministic

Synthesis: flip the arrows!

union
cube
trans (2, 0

union

ube

esh [m_{cube}]

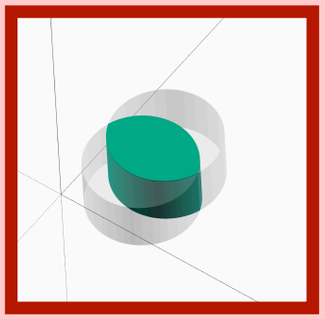
[.] union
Mesh
Mesh

0, 0) cube

Mesh [...]

Evaluation context

- Loss of information



- Geometric challenges

Geometric oracles

- Non-deterministic

Synthesis: flip the arrows!

union
cube
trans (2, 0

union

ube

esh [m_{cube}]

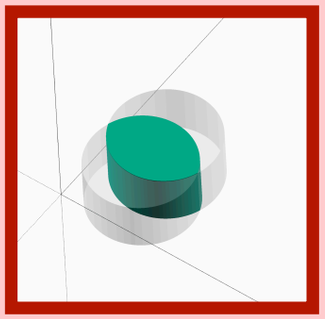
[.] union
Mesh
Mesh

0, 0) cube

Mesh [...]

Evaluation context

- Loss of information



- Geometric challenges

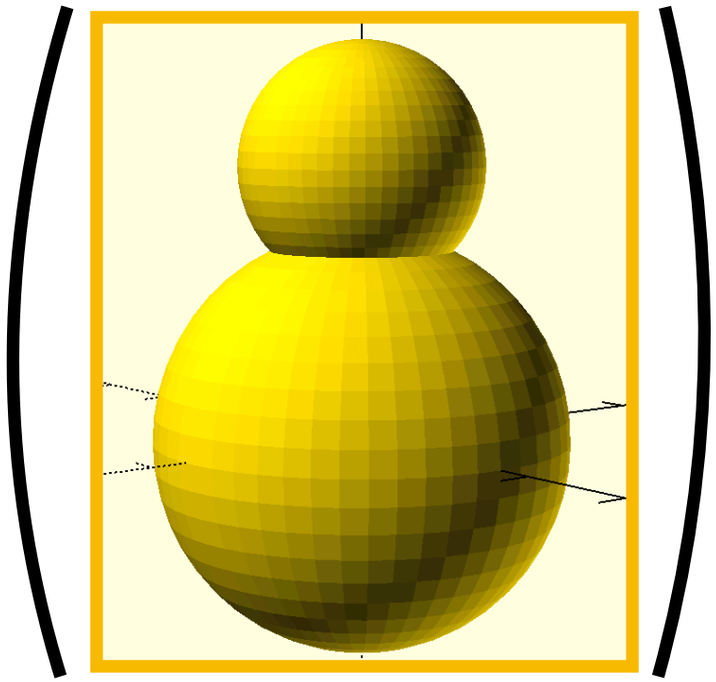
Geometric oracles

- Non-deterministic

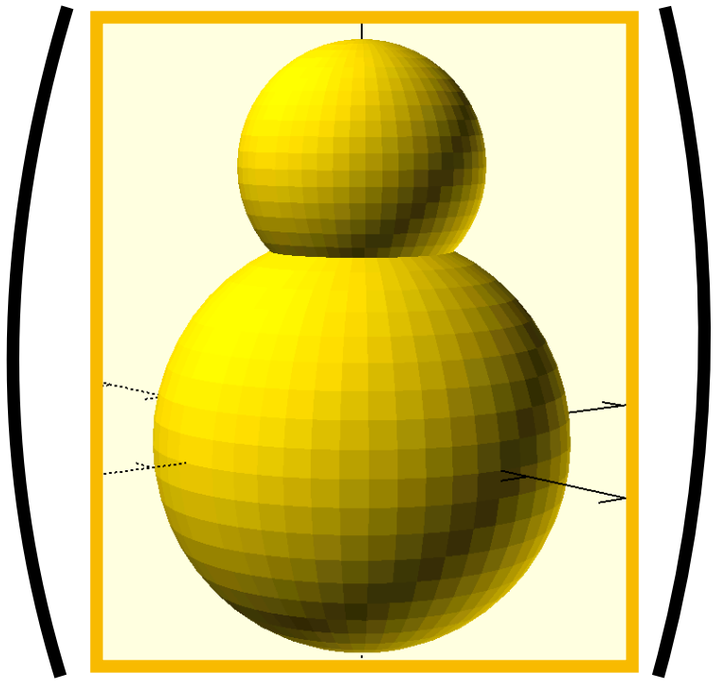
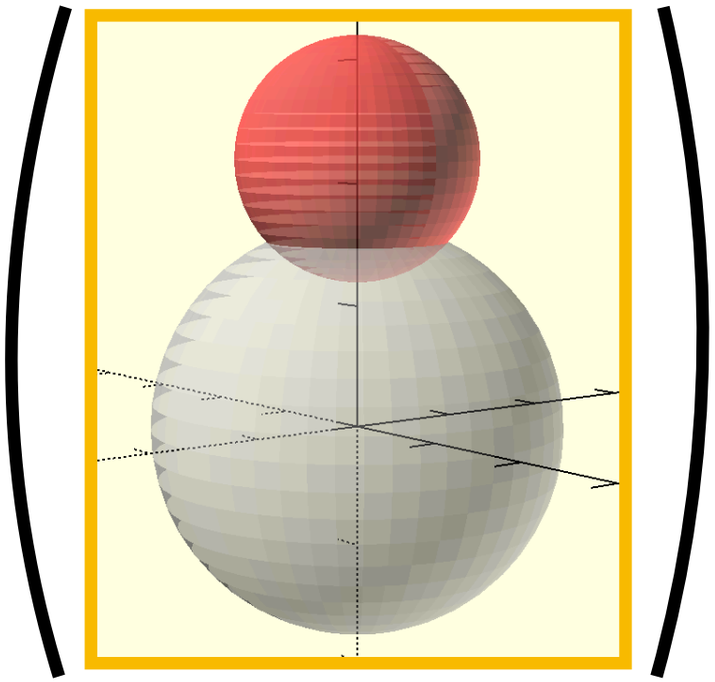
search

Eval context is geometric context!

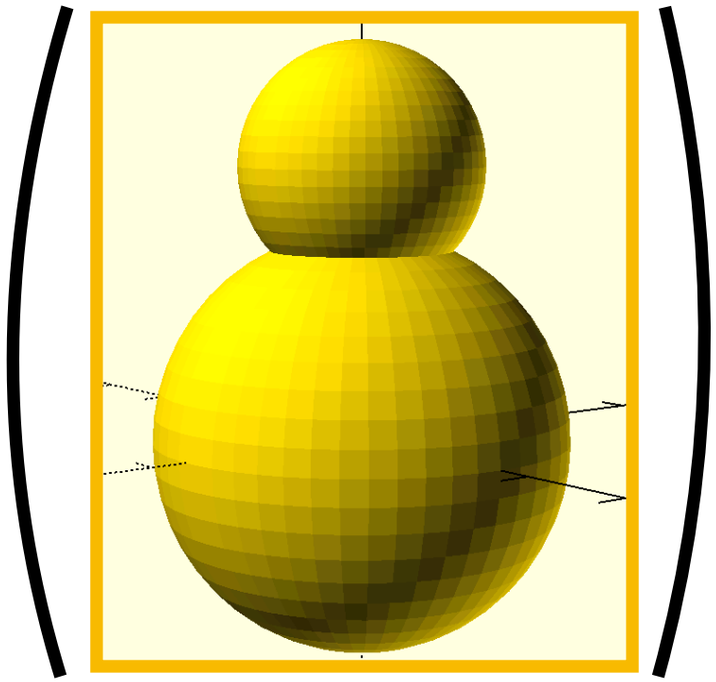
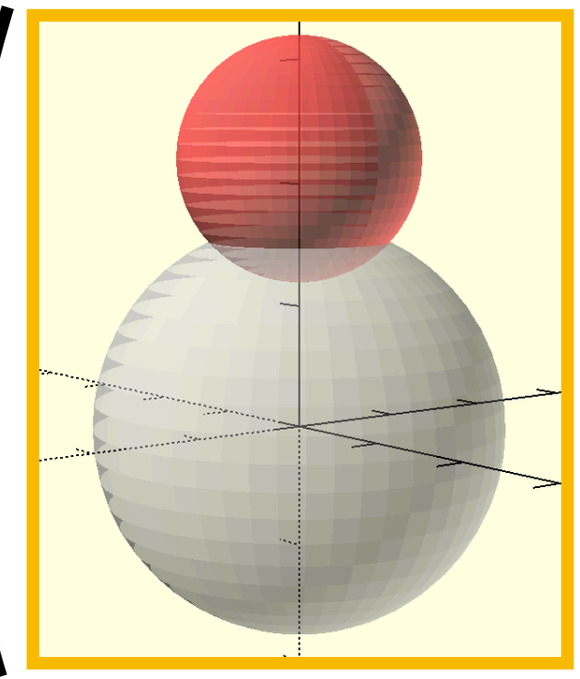
synthesize



Eval context is geometric context!

synthesize  = synthesize 

Eval context is geometric context!

synthesize  = synthesize 

```
union
  sphere 5
  trans (0, 0, 5) sphere 3
```

In the **context** of **union**, the top can be replaced by a sphere, even though it does not match a sphere primitive

Synthesis: flip the arrows!

union
cube
trans (2, 0

union

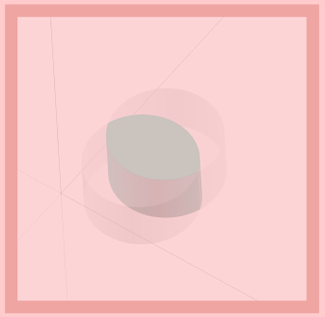
union

[.] union
Mesh
Mesh

Mesh […]

Evaluation context

- Loss of information



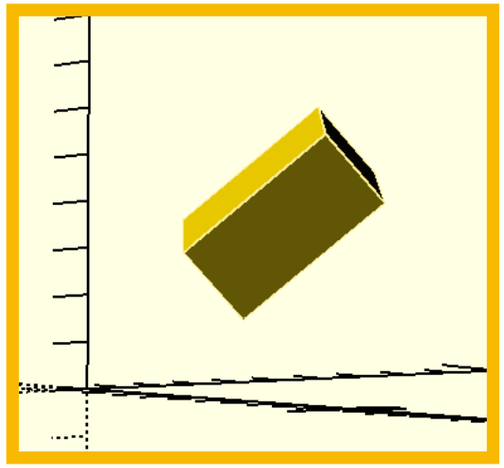
- Geometric challenges

Geometric oracles

- Non-deterministic

search

Oracles : Primitive Detection

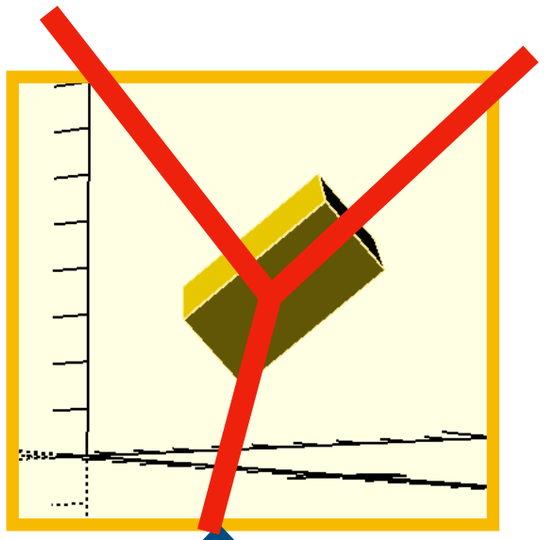


Ω_{prim}

```
cube
  |> rotateX (30)
  |> rotateY (45)
  |> rotateZ (60)
  |> scale (2, 3, 4)
  |> traslate (1, 2, 3)
```

$$\frac{p \in \Omega_{prim}(m)}{Mesh\ m \rightarrow_{\Omega} p}$$

Oracles : Primitive Detection



Ω_{prim}

```
cube
  |> rotateX (30)
  |> rotateY (45)
  |> rotateZ (60)
  |> scale (2, 3, 4)
  |> traslate (1, 2, 3)
```

Rotation:

Find an object coordinate system

Align it with the world's coordinates

Scale:

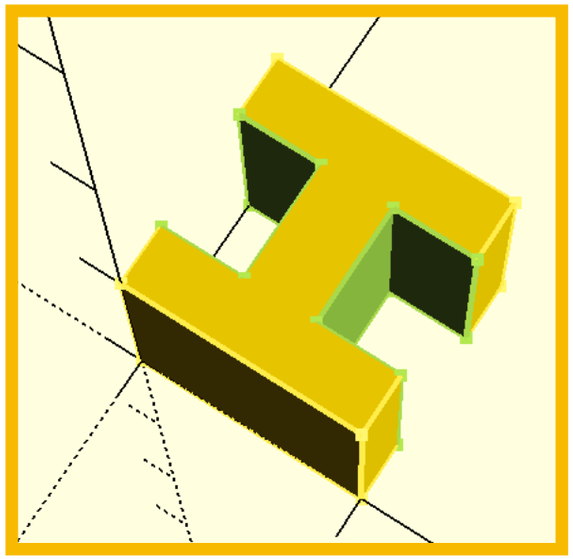
Scale the object down to unit dimensions

Translate:

Move the object back to origin

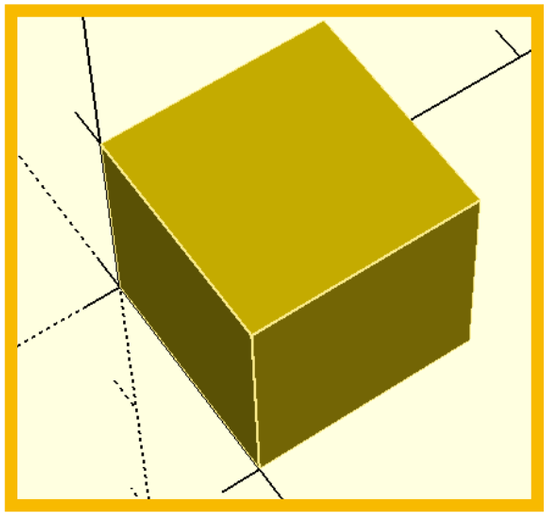
Apply affine transformations to primitive in reverse order

Oracles : Subtractive



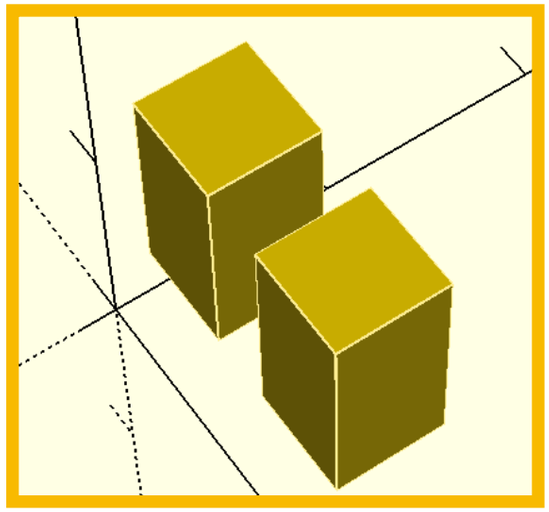
m

Ω_{sub}



Bounding primitive

−

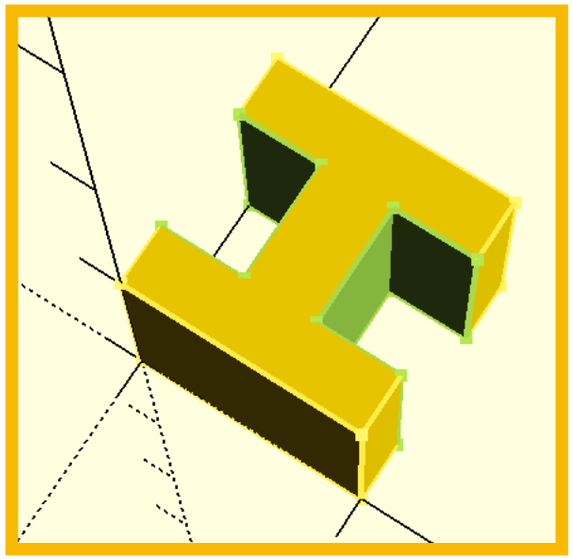


Remaining mesh

$$(m_1, m_2) \in \Omega_{sub}(m)$$

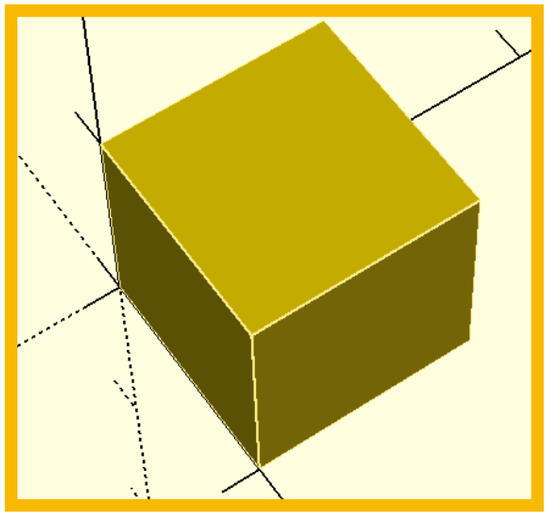
$$Mesh\ m \rightarrow_{\Omega} Binop\ Diff\ (Mesh\ m_1)\ (Mesh\ m_2)$$

Oracles : Subtractive



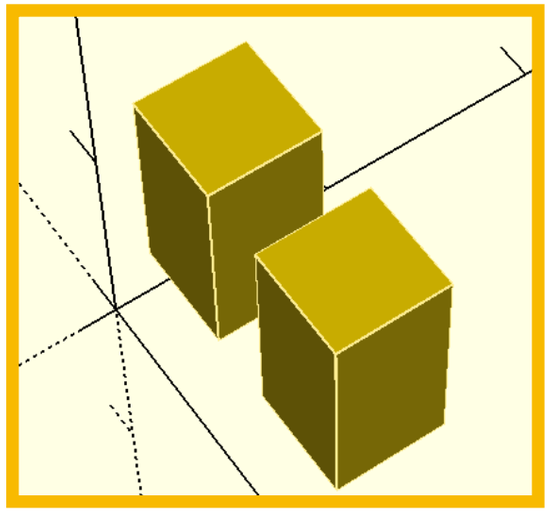
m

Ω_{sub}



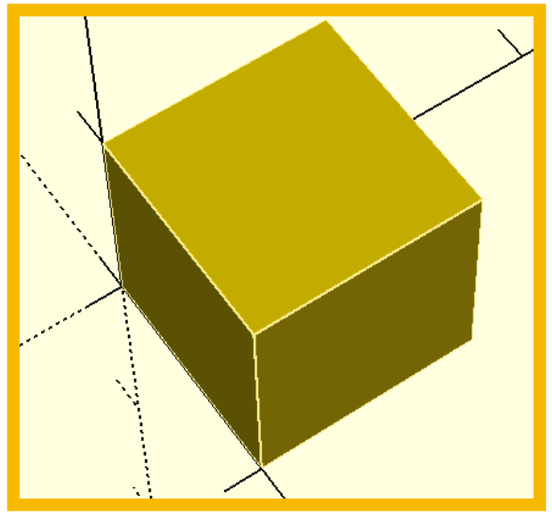
Bounding primitive

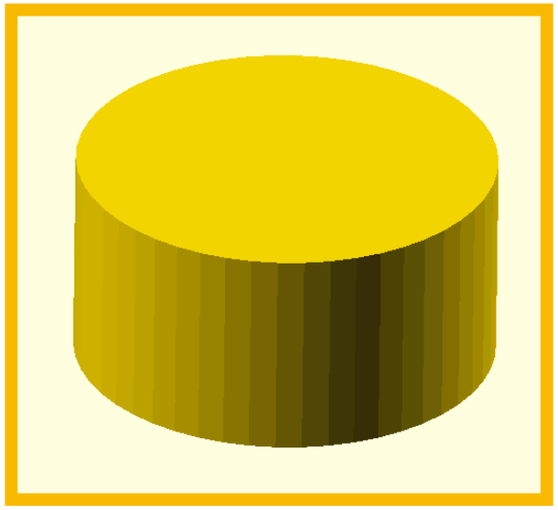
−

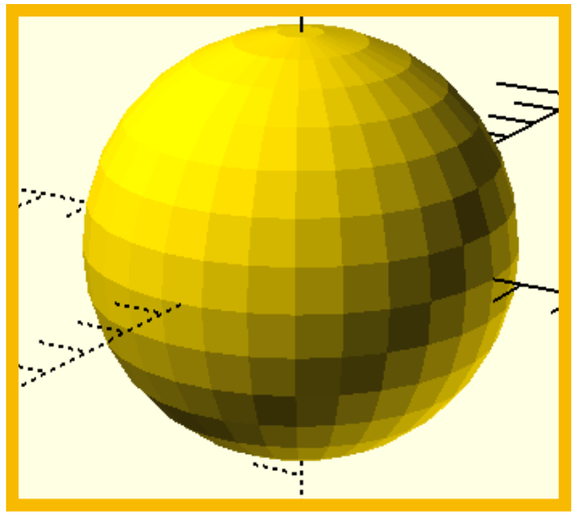


Remaining mesh

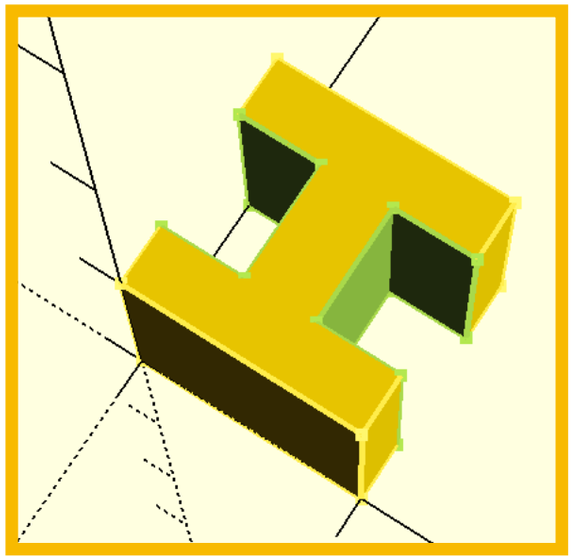
Many possible bounding primitives





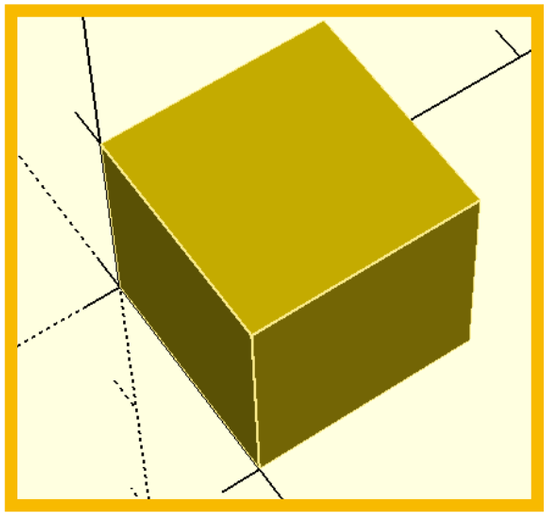


Oracles : Subtractive



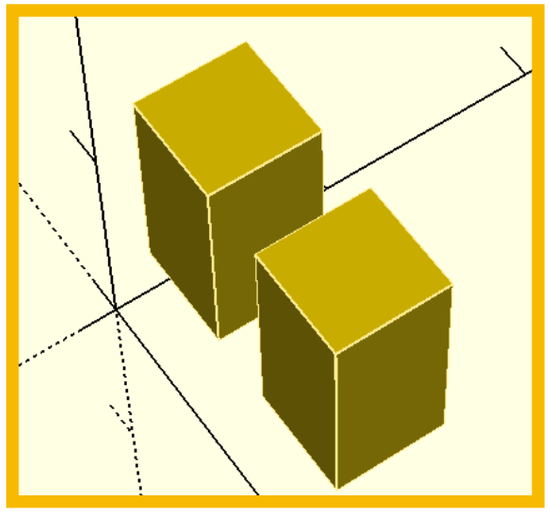
m

Ω_{sub}



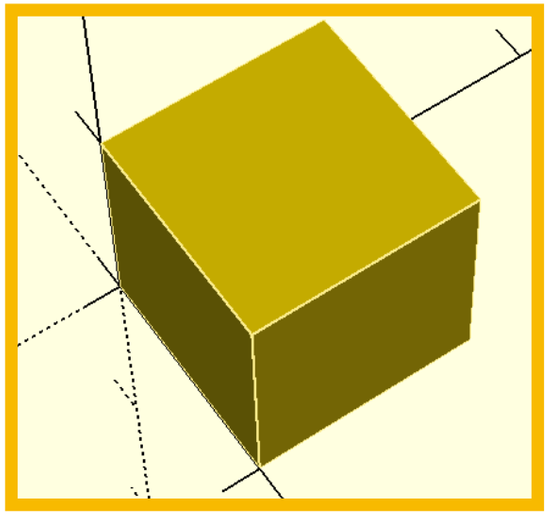
Bounding primitive

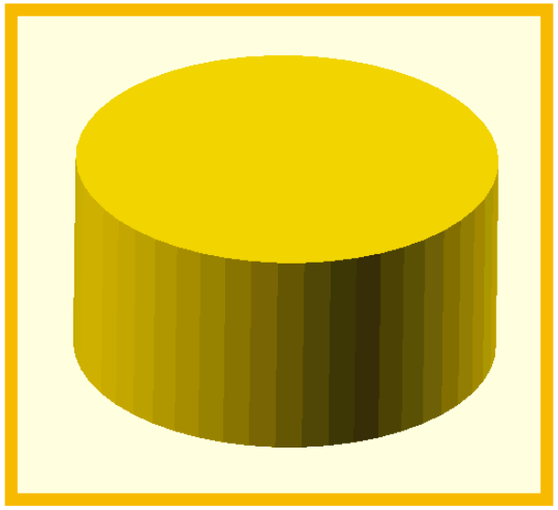
−

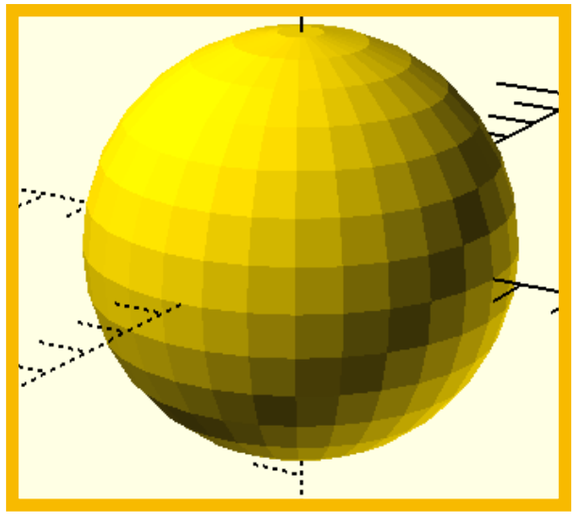


Remaining mesh

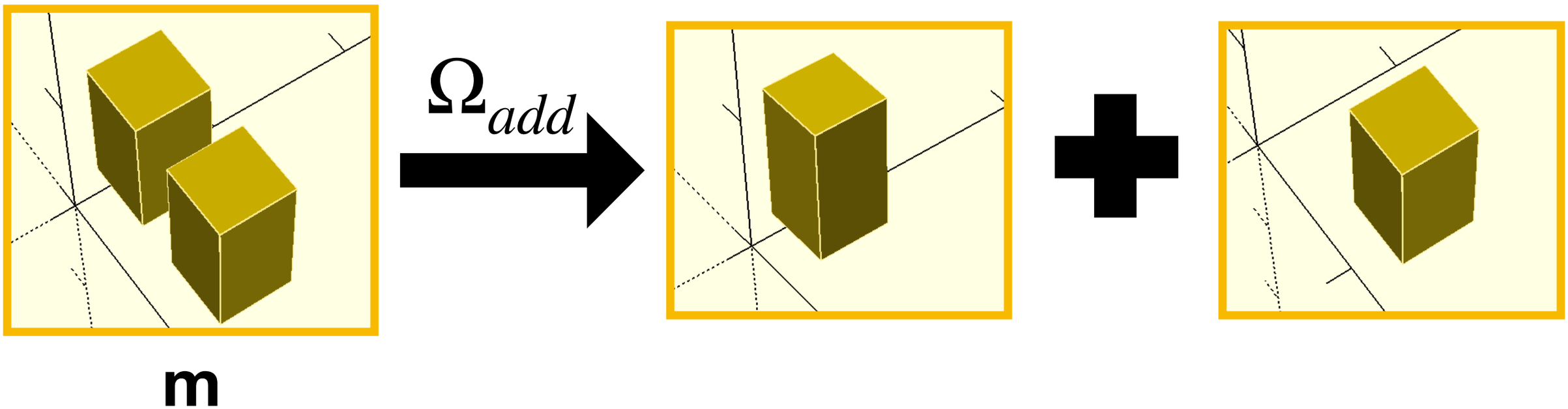
p_{best} = argmin_p volume of difference(p, m)





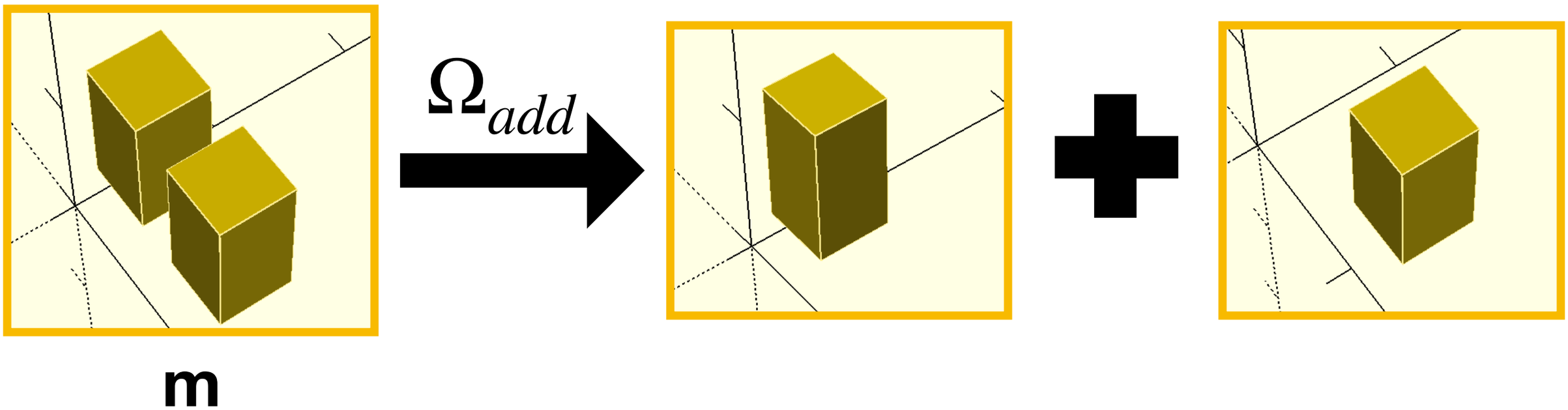


Oracles : Additive



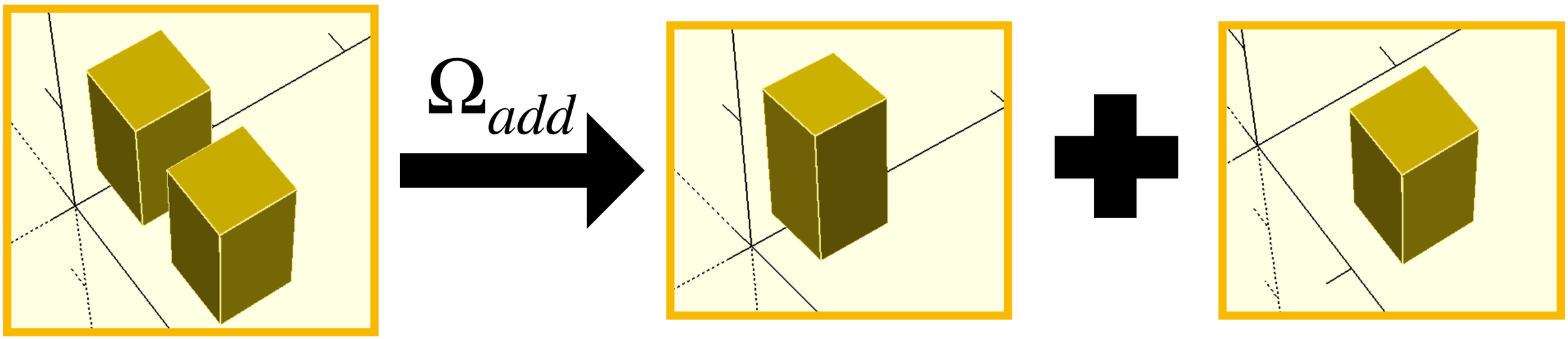
$$\frac{(m_1, m_2) \in \Omega_{add}(m)}{Mesh\ m \rightarrow_{\Omega} Binop\ Union\ (Mesh\ m_1)\ (Mesh\ m_2)}$$

Oracles : Additive



Infinite ways to split a mesh!

Oracles : Additive



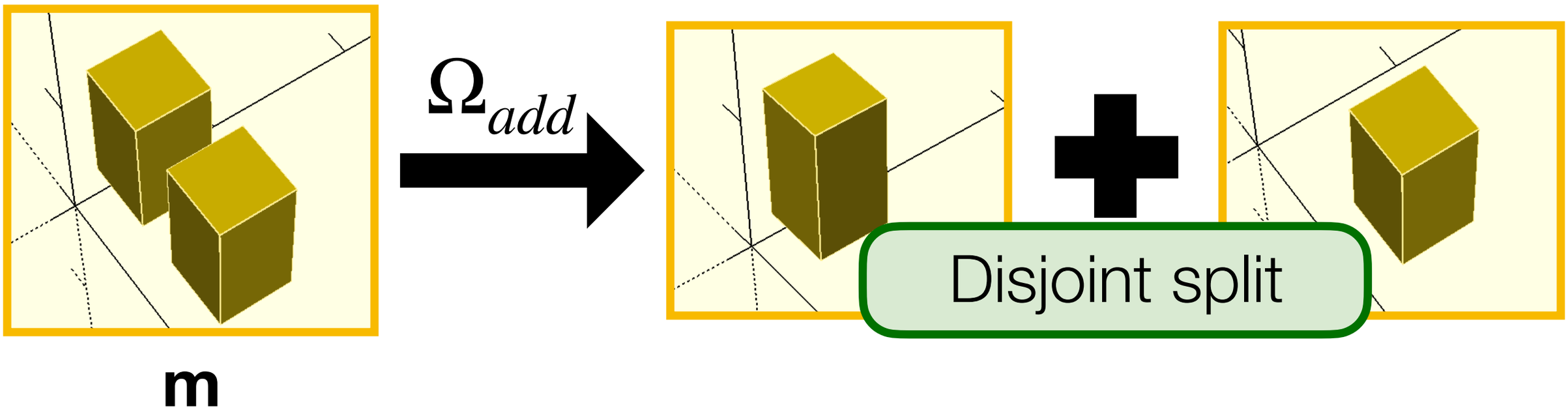
m

Infinite ways to split a mesh!

Disjoint splits

Convex splits

Oracles : Additive

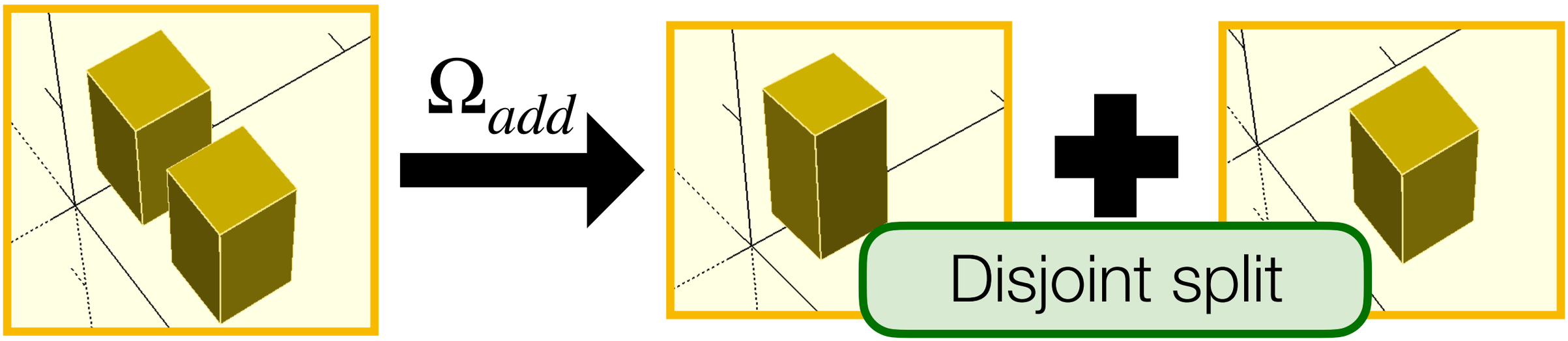


Infinite ways to split a mesh!

Disjoint splits

Convex splits

Oracles : Additive

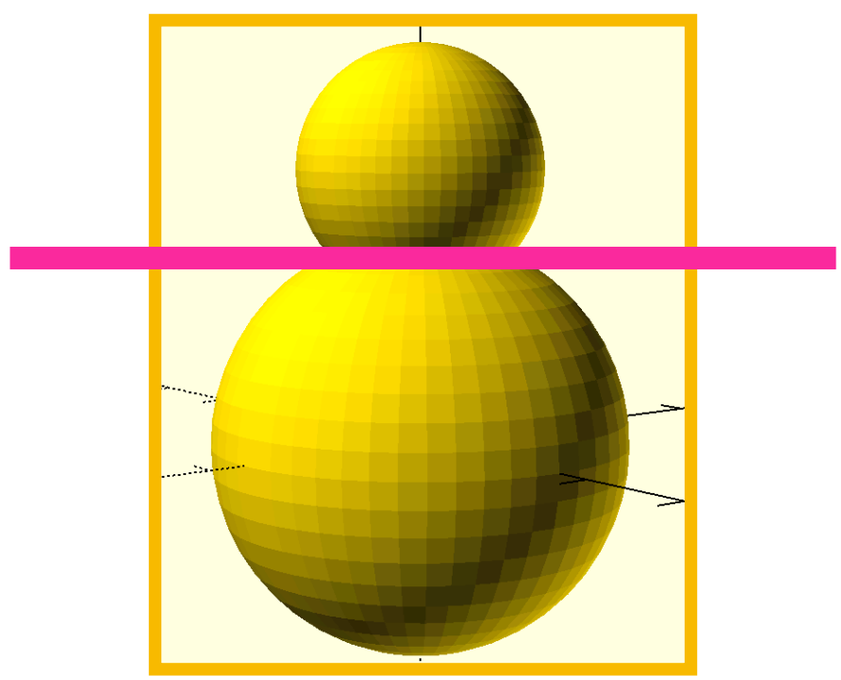


m

Infinite ways to split a mesh!

Disjoint splits

Convex splits



Convex split: Split mesh along plane where convexity changes

Synthesis: flip the arrows!

union
cube
trans (2, 0

union

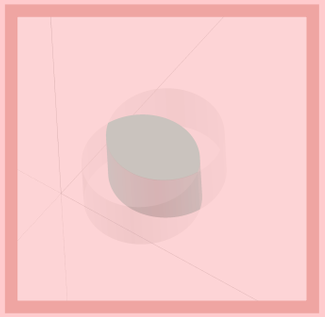
union

[.] union
Mesh
Mesh

Mesh [...]

Evaluation context

- Loss of information
- Geometric challenges
- Non-deterministic

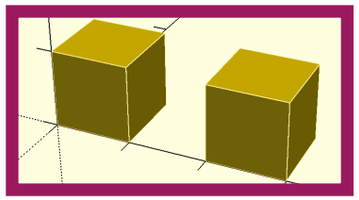


Geometric oracles

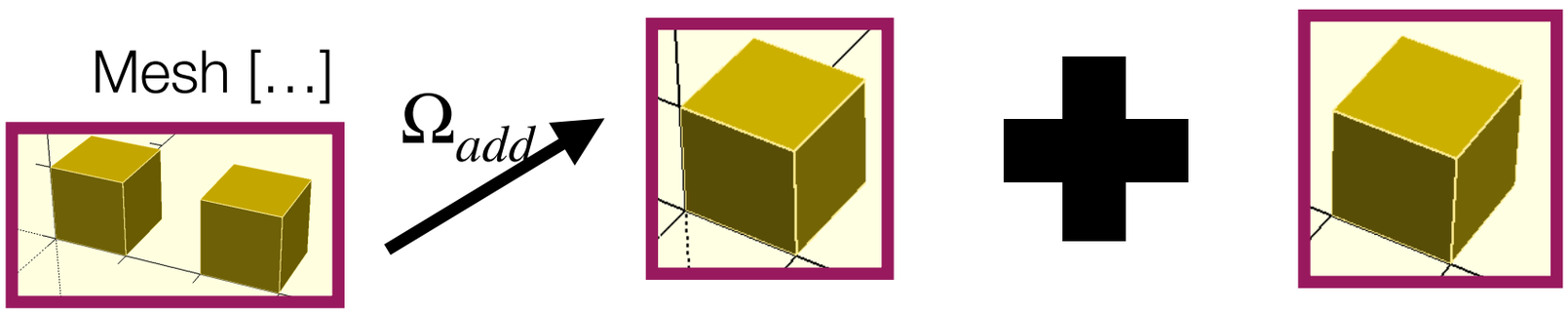
search

Synthesis: try all three steps

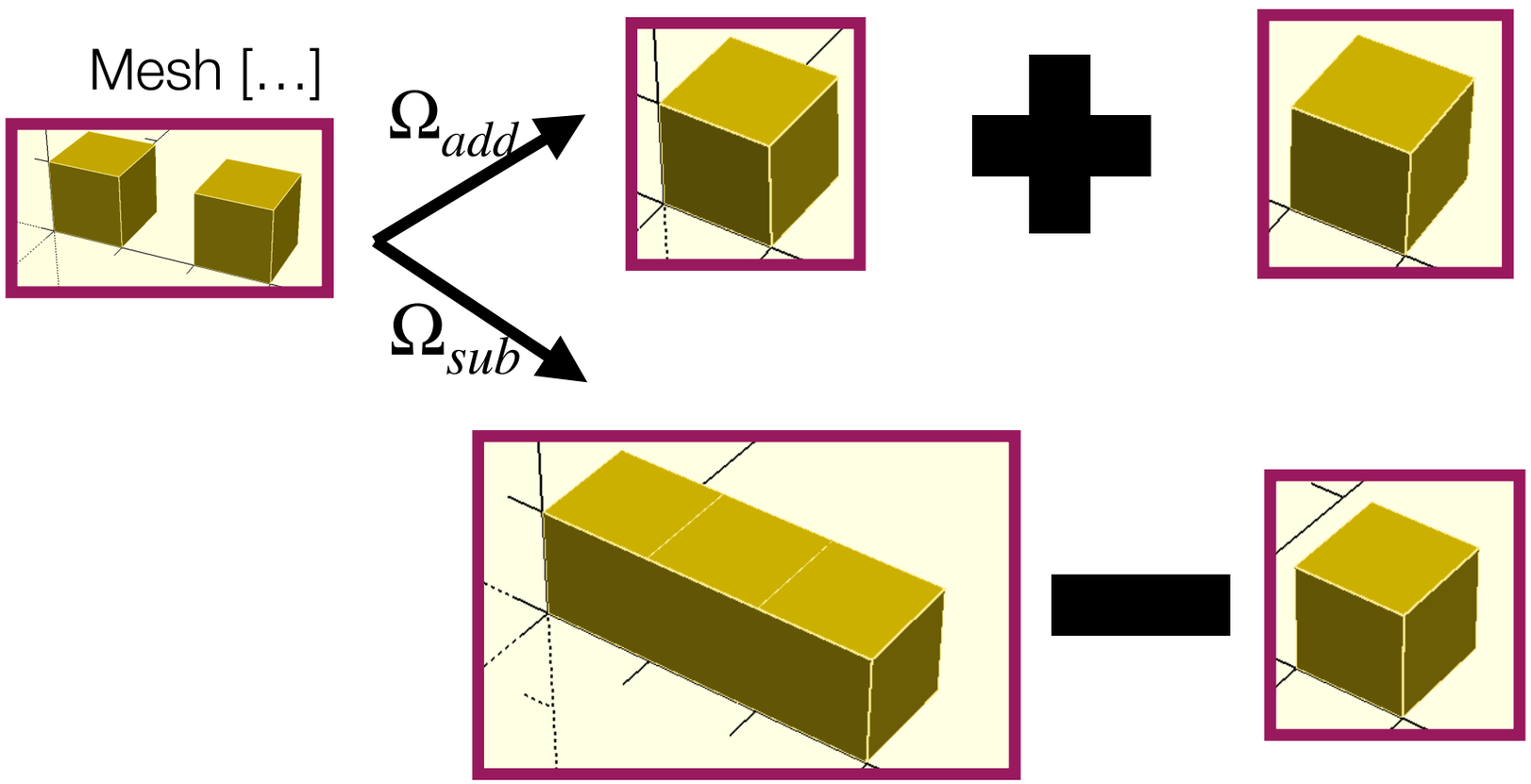
Mesh […]



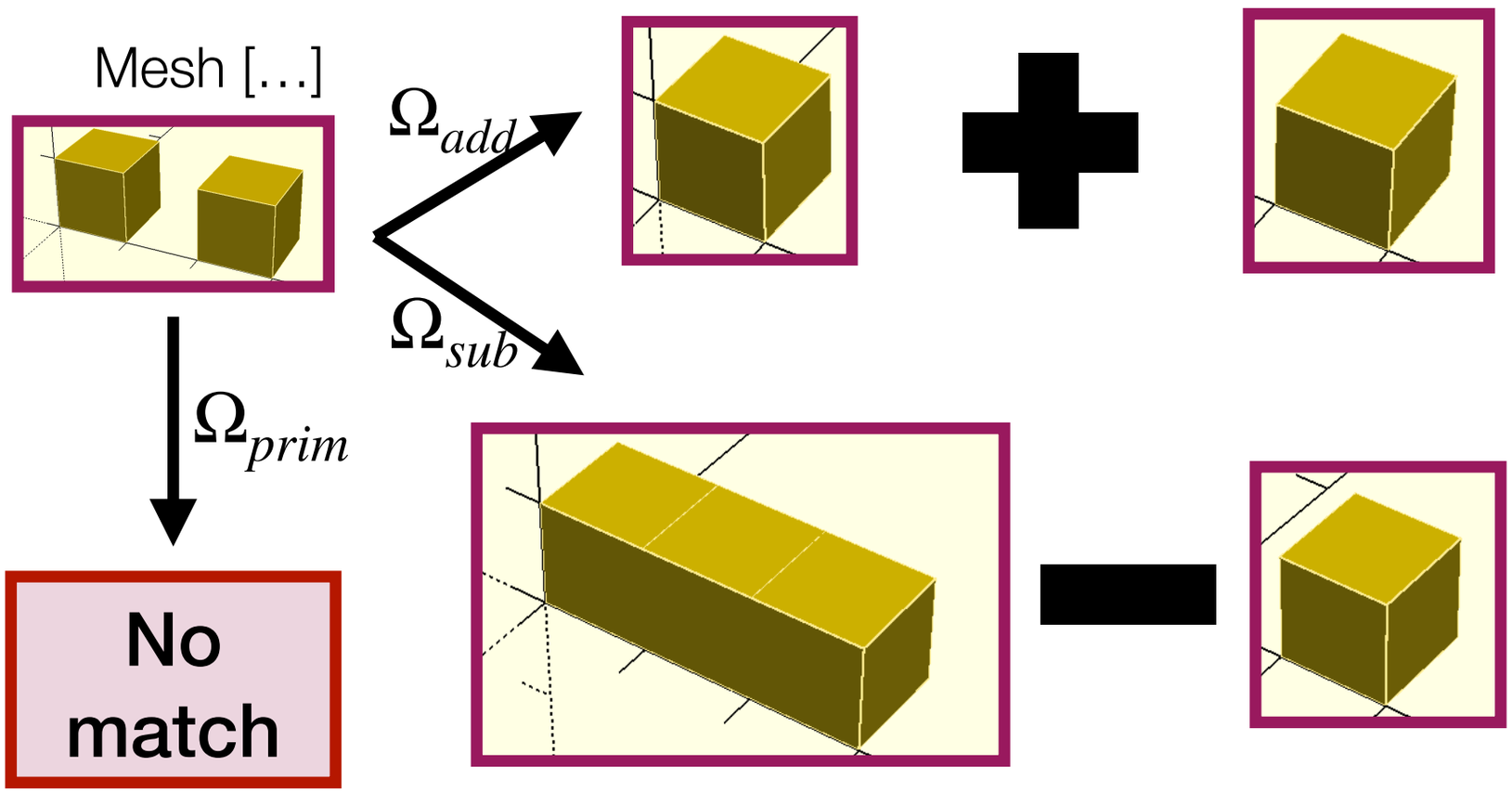
Synthesis: try all three steps



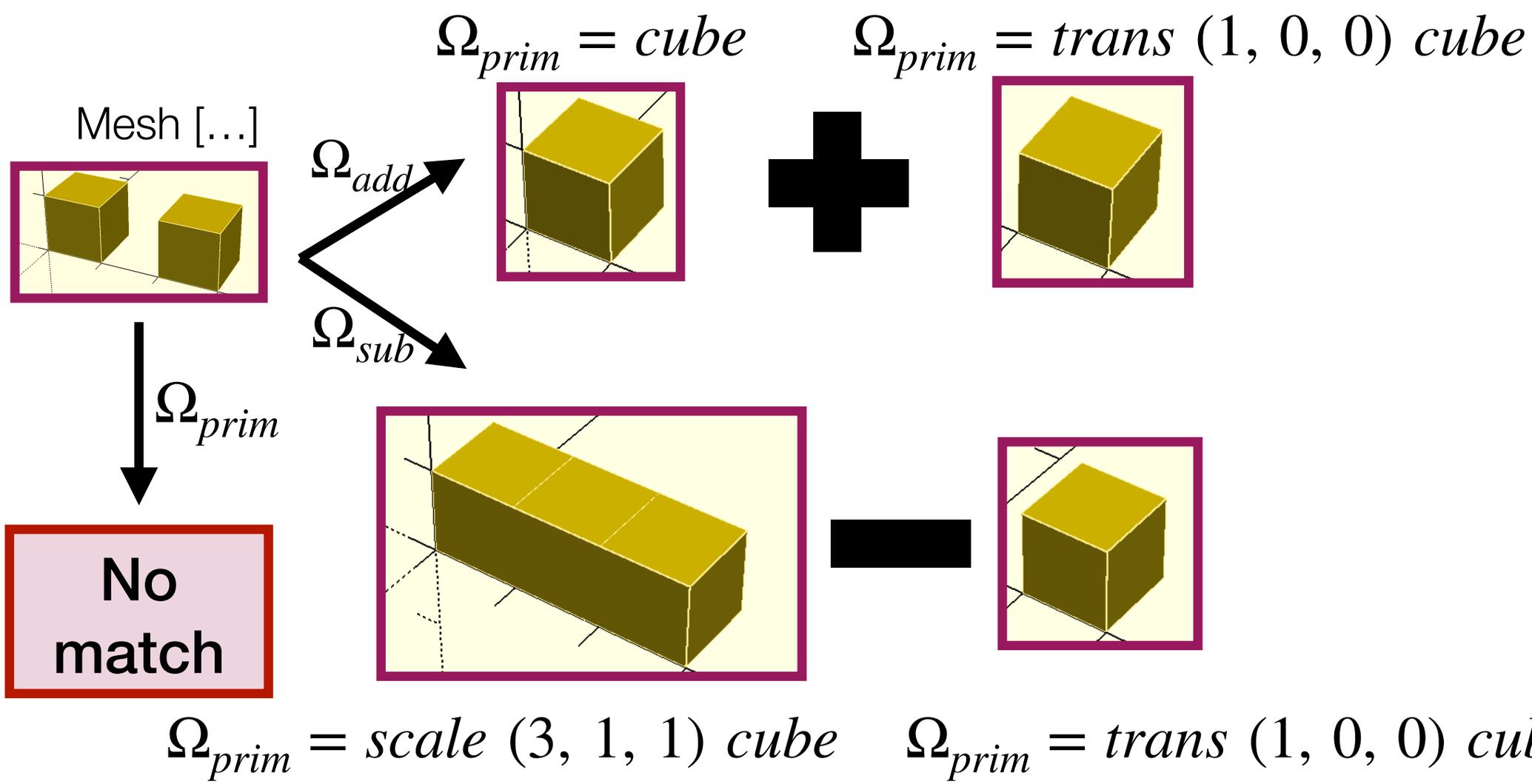
Synthesis: try all three steps



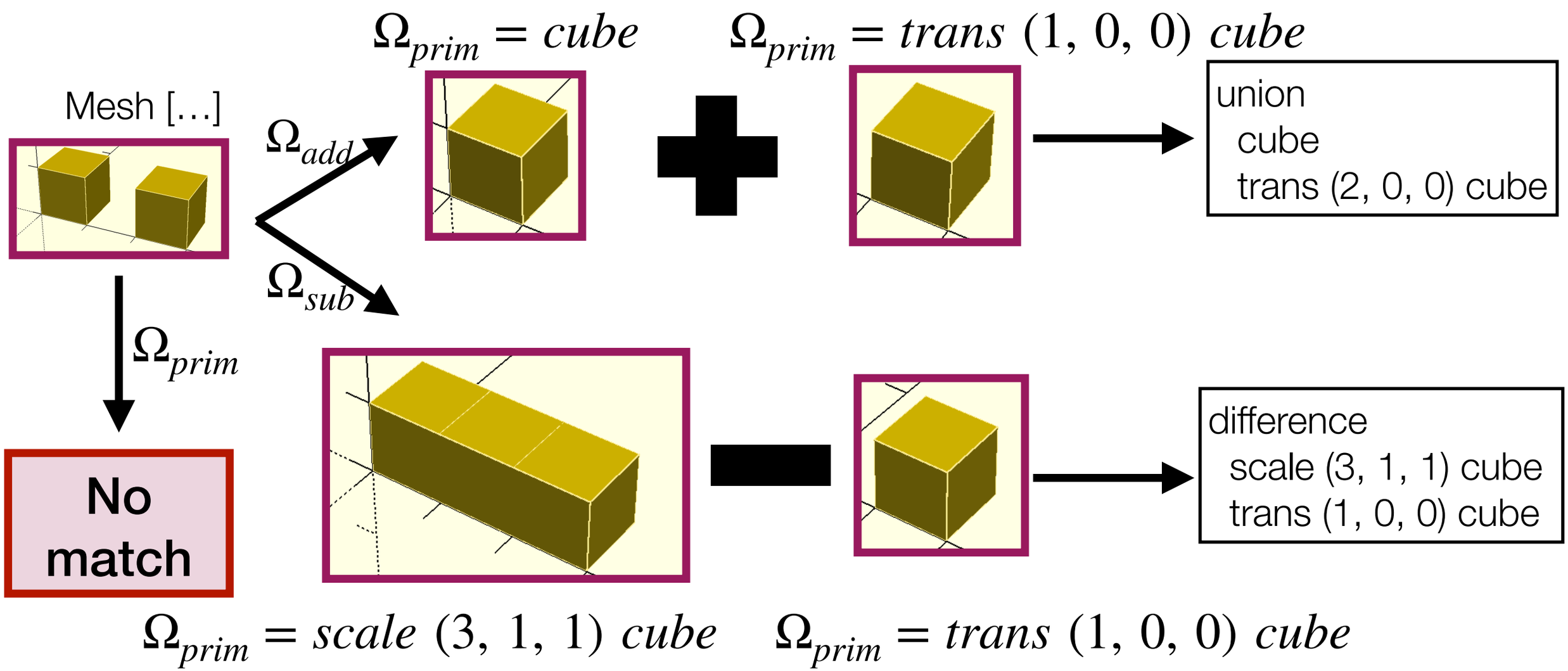
Synthesis: try all three steps



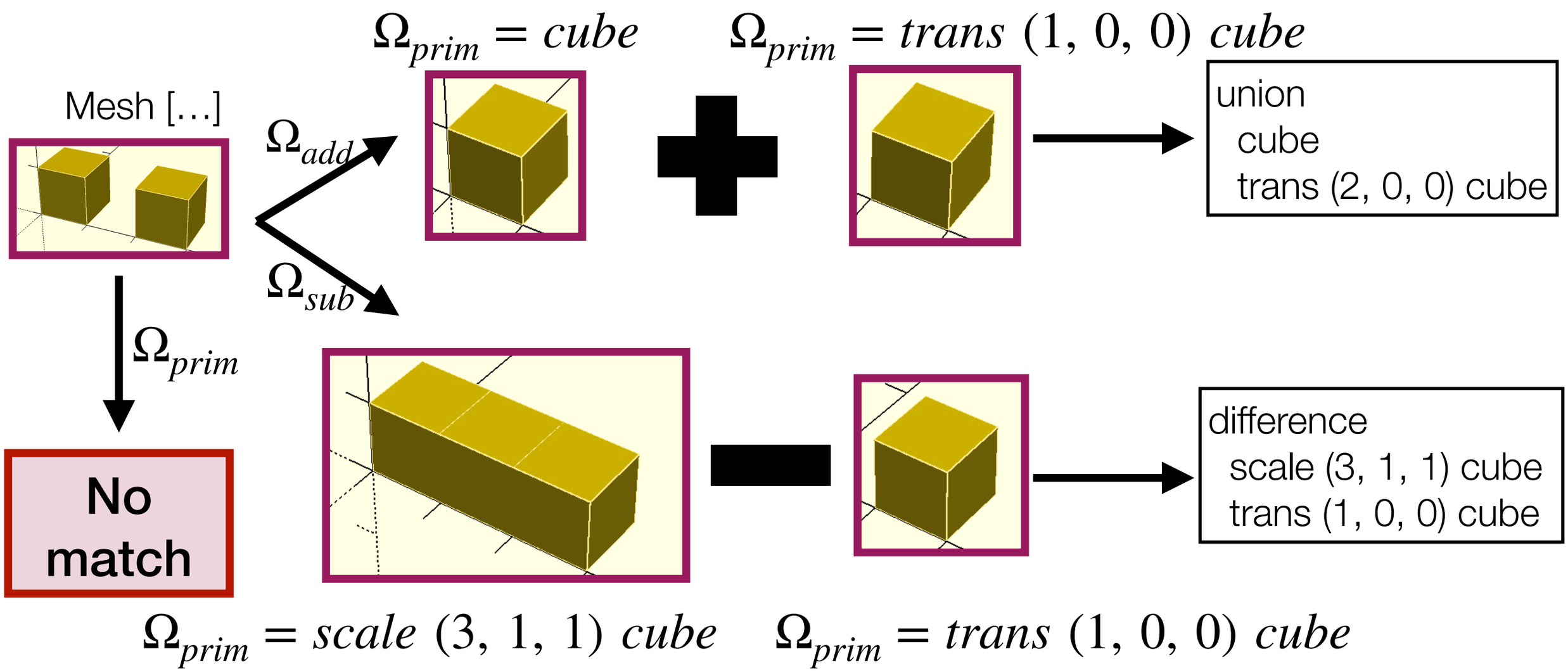
Synthesis: try all three steps



Synthesis: try all three steps



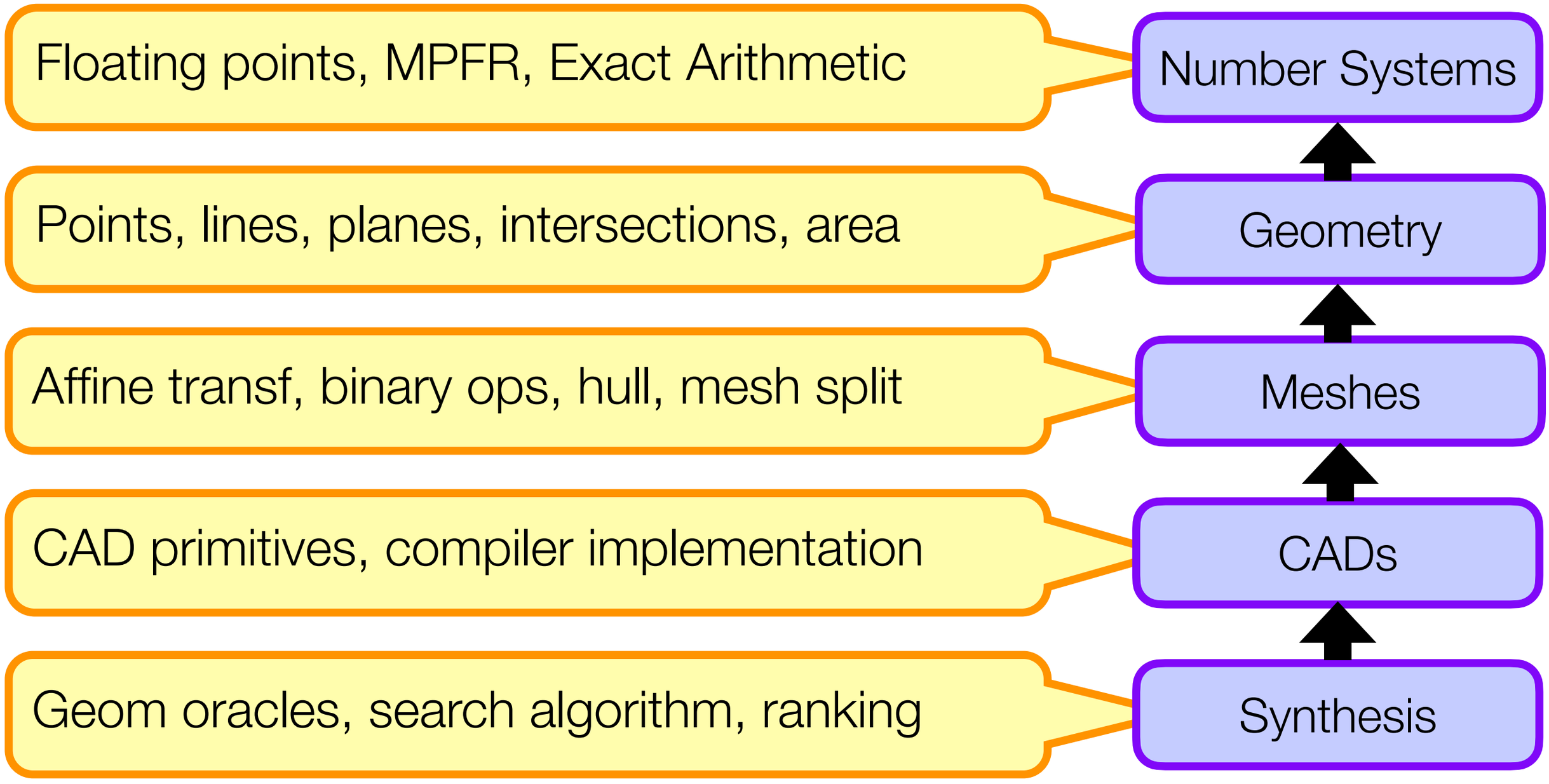
Synthesis: try all three steps



Pick best based on ranking function

Implementations in OCaml

25,000 LOC: Supports 1D, 2D, 3D CAD & Mesh



https://github.com/uwplse/reincarnate-aec

Implementations in OCaml

25,000 LOC: Supports 1D, 2D, 3D CAD & Mesh

Floating points, MPFR, Exact Arithmetic

Points, line area

Affine transf, binary ops, null, mesh split

CAD primitives, compiler implementation

Geom oracles, search algorithm, ranking

Pavel Panchekha
NPFL, Thursday

Number Systems

Geometry

Meshes

CADs

Synthesis

https://github.com/uwplse/reincarnate-aec

Results

ICFP (our design)

Thingiverse

Hexagonal Candle Holder

Ultimate 22 Hex-Wrench Holder

40mm Cube Test Object

25mm Calibration with Empty Top

Measuring Cylinder

Basic Box with Lid

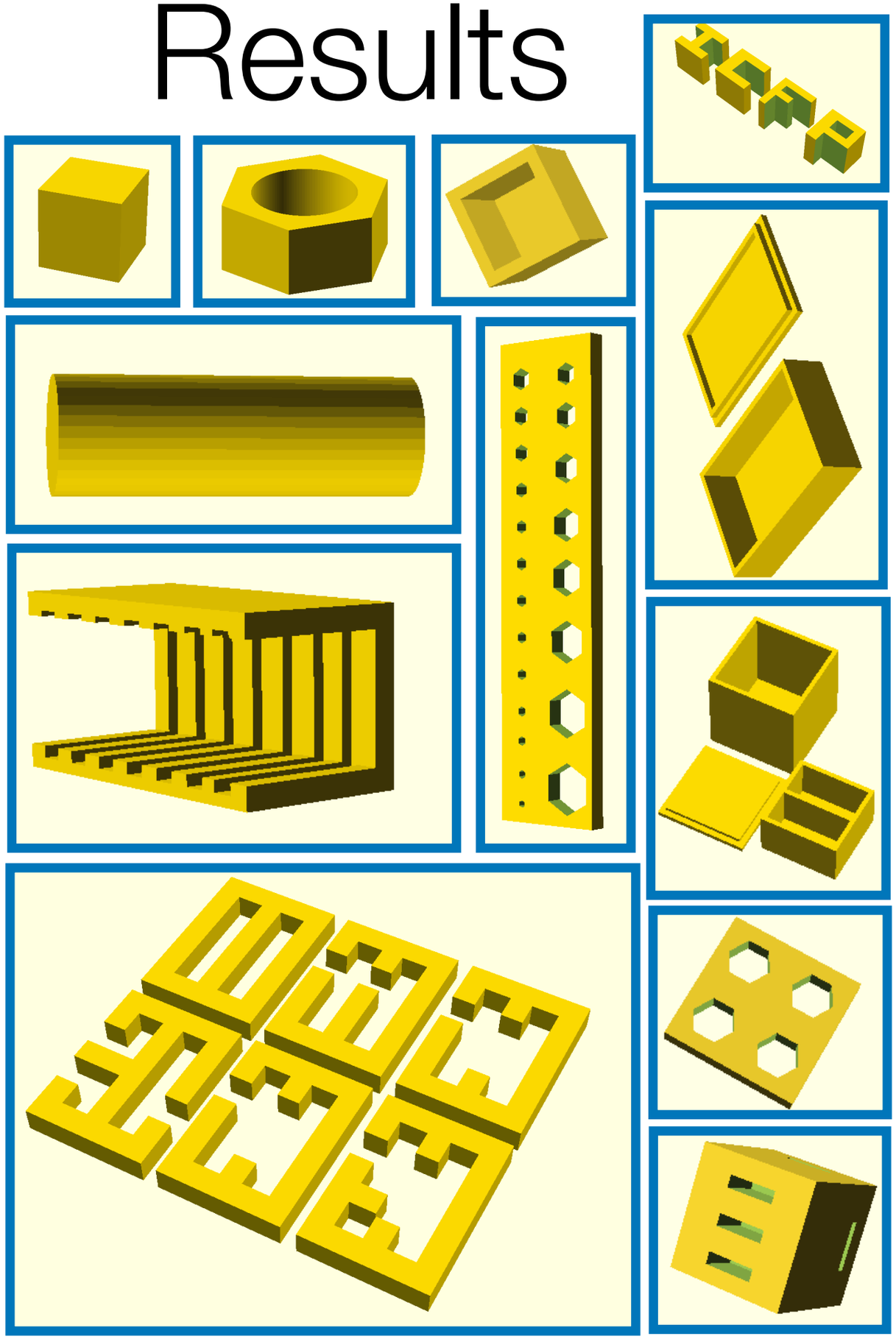
Modular Memory Holder (USB)

Circle Cell Block Generator

Jewelry Box with Inlay

SD Card Rack

Gordian knot 3D Puzzle



Conclusions

Functional PL for fabrication (3D printing)

***Clarity**:* semantics and compiler correctness

Usefulness: the first decompiler from mesh to CAD

Functional PL for fabrication (3D printing)

***Clarity**:* semantics and compiler correctness

Usefulness: the **first** decompiler from mesh to CAD






Check out our web IDE! — Adam Anderson

http://reincarnate.uwplse.org/

https://github.com/uwplse/reincarnate-aec

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

PLSE