Szalinski: A Tool for Synthesizing Structured CAD Models with Equality Saturation and Inverse Transformations

PLDI 2020

Chandrakana Nandi, Max Willsey, Adam Anderson, James R. Wilcox, Eva Darulova, Dan Grossman, Zachary Tatlock

PLSE
PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING
MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

Designing Physical Objects

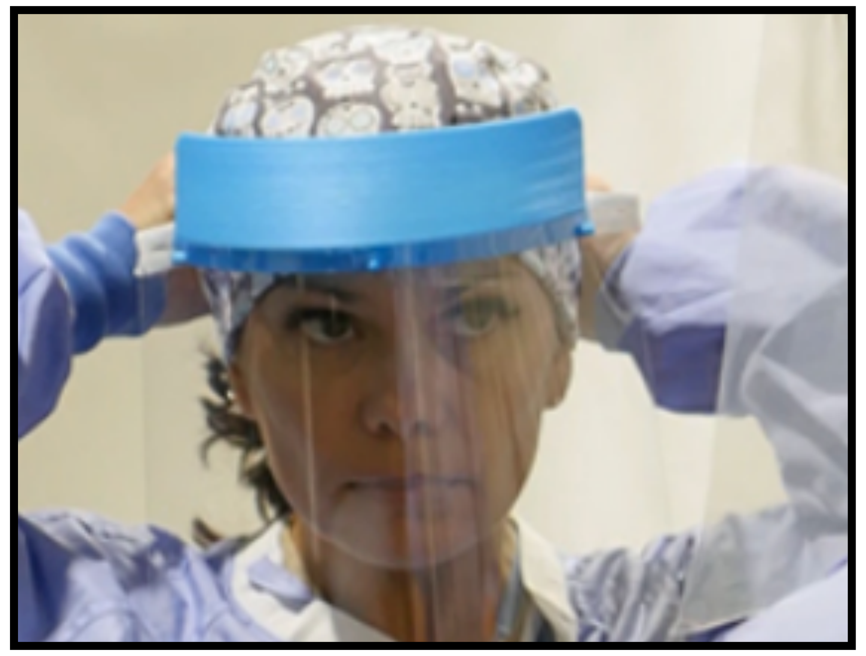
is

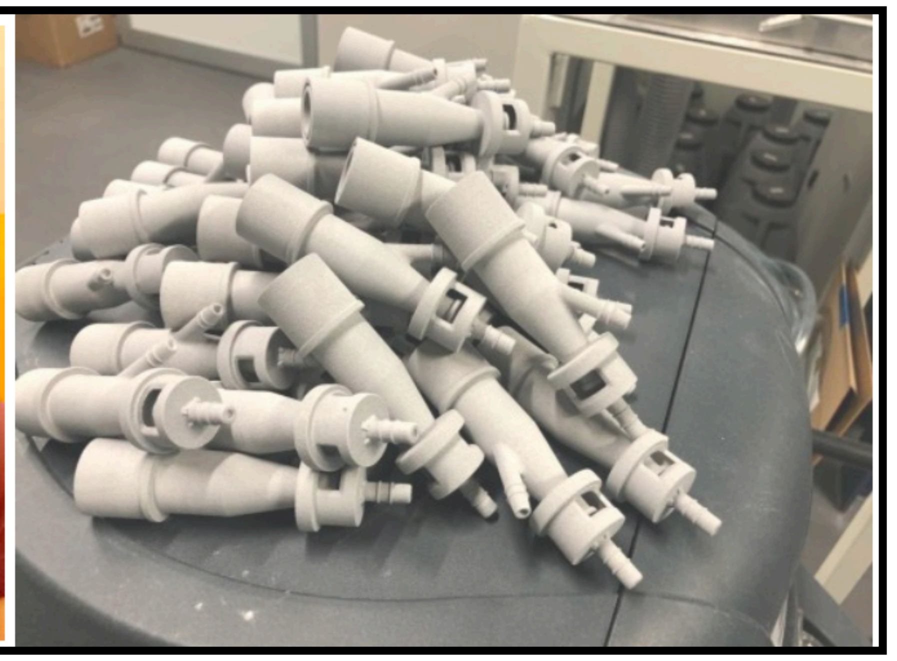
Programming!

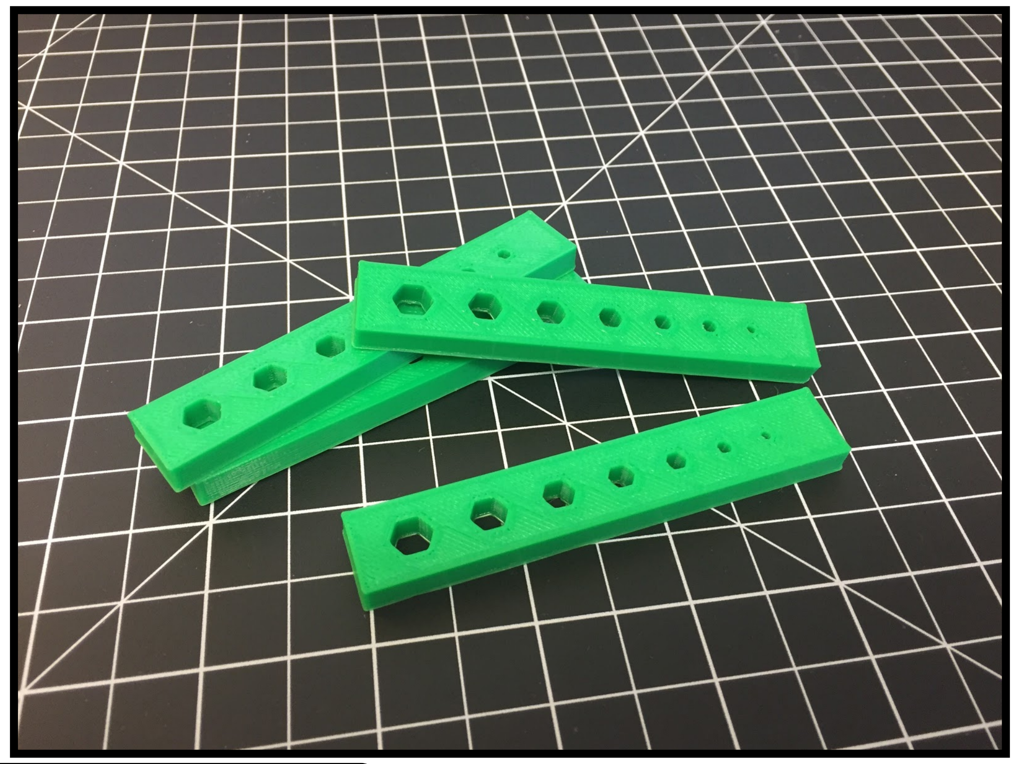
CAD and 3D Printing everywhere!

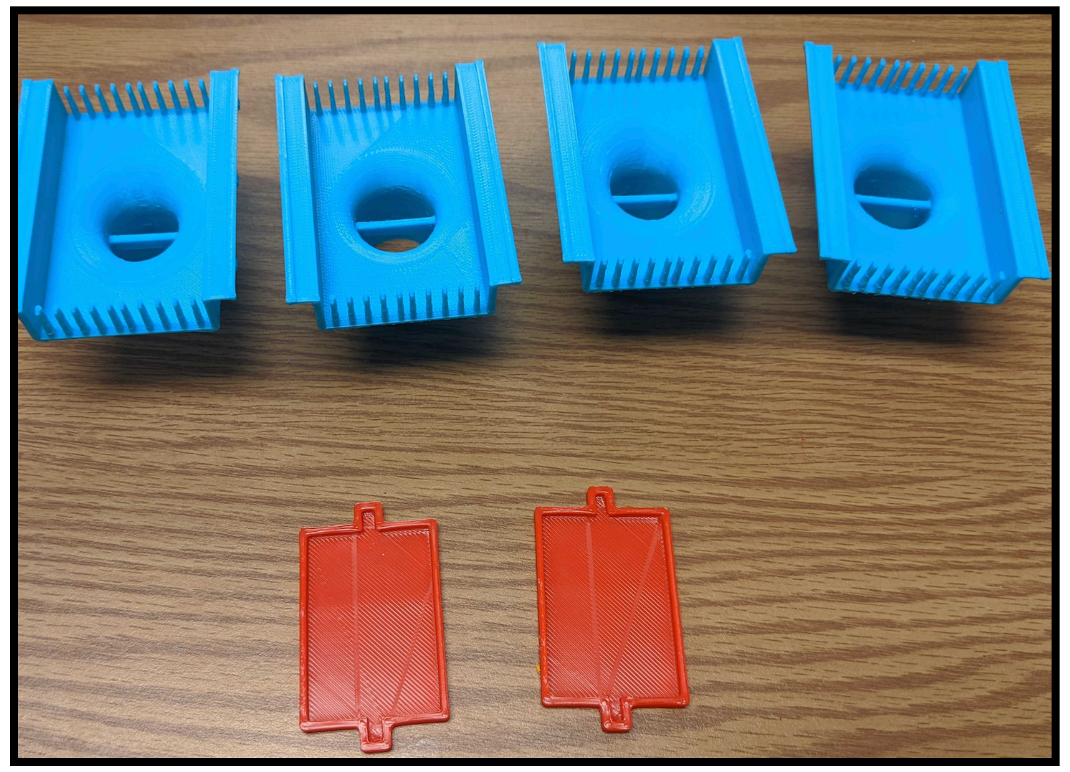
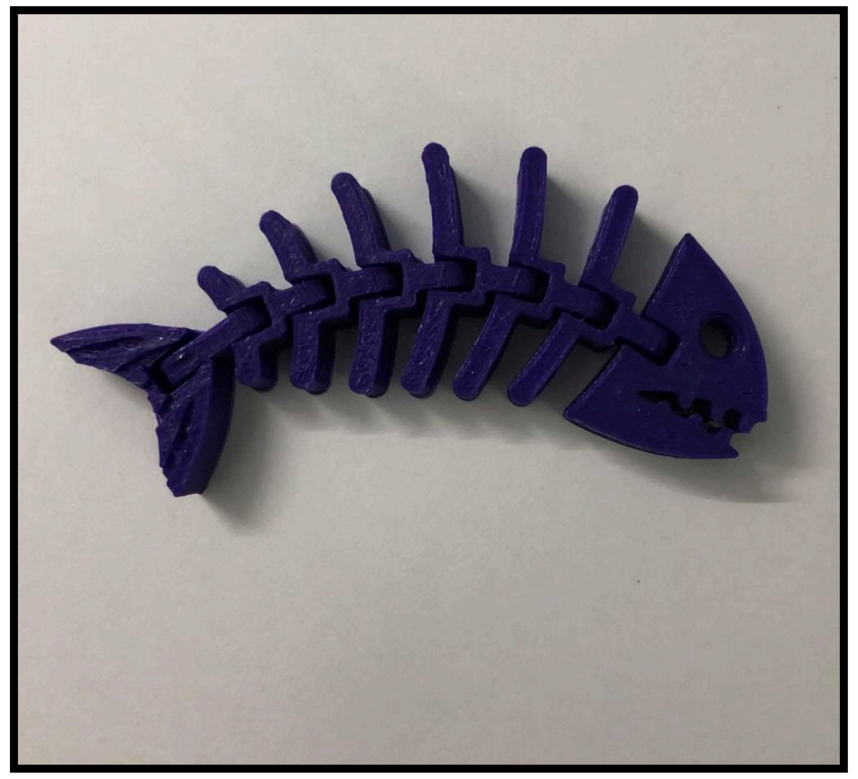
Make your own models



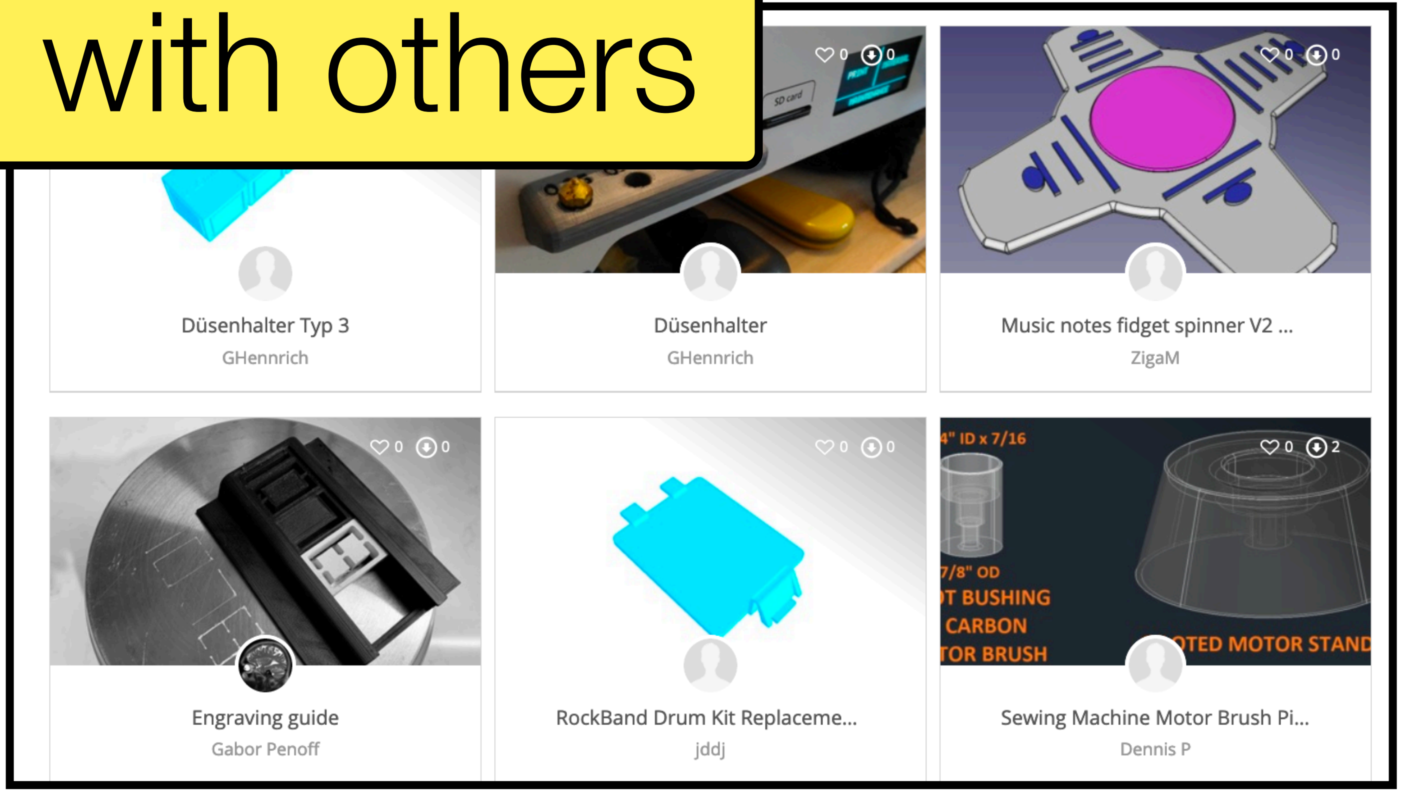


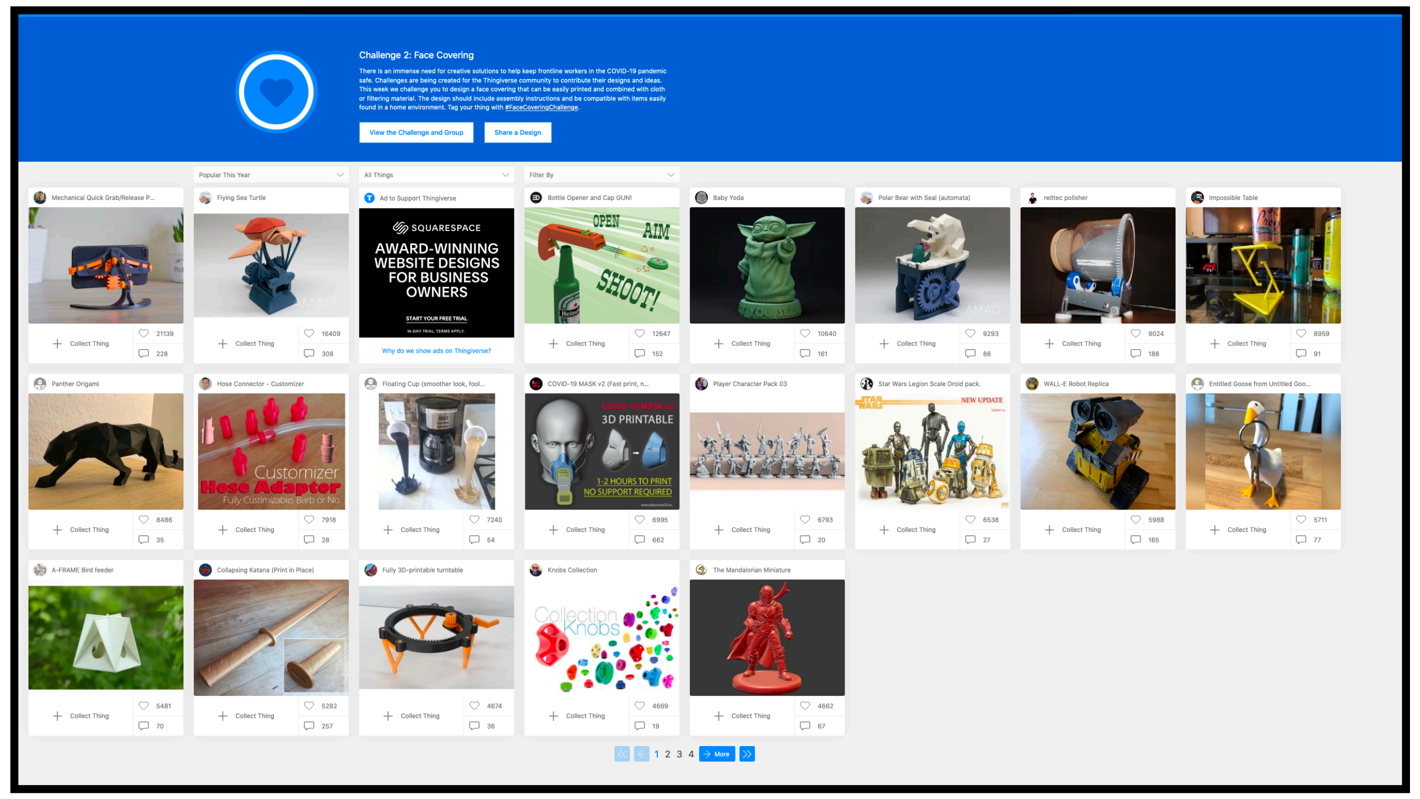
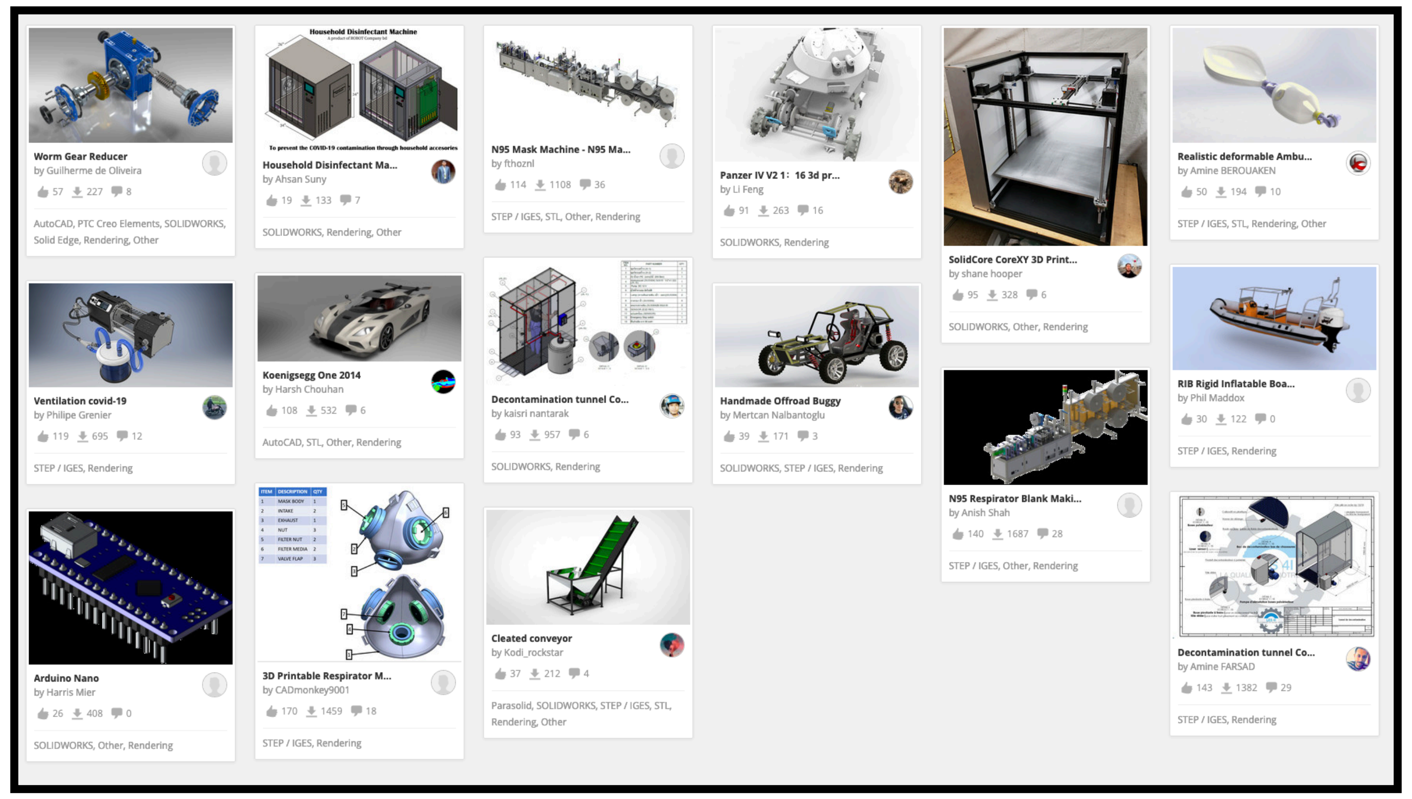


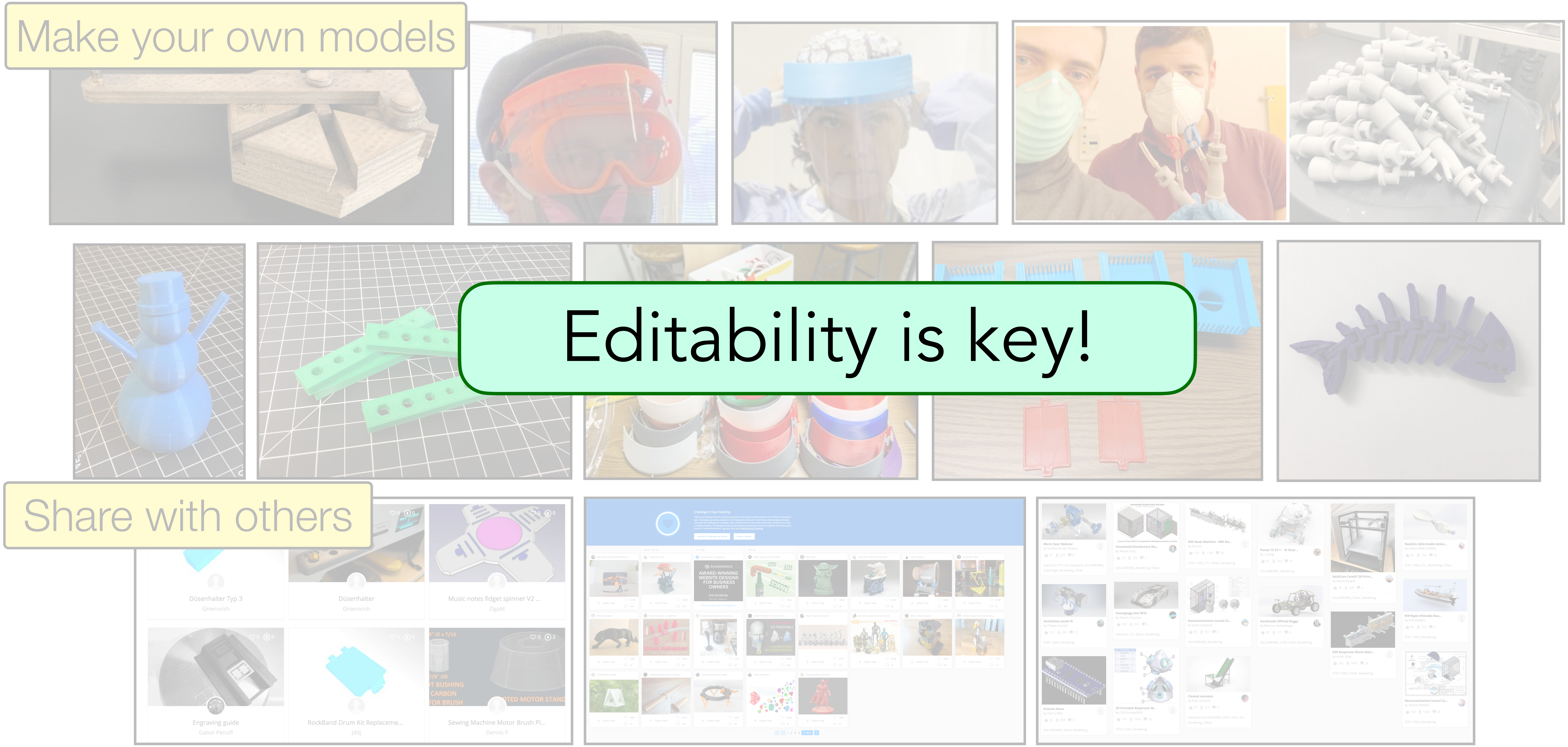



Share with others

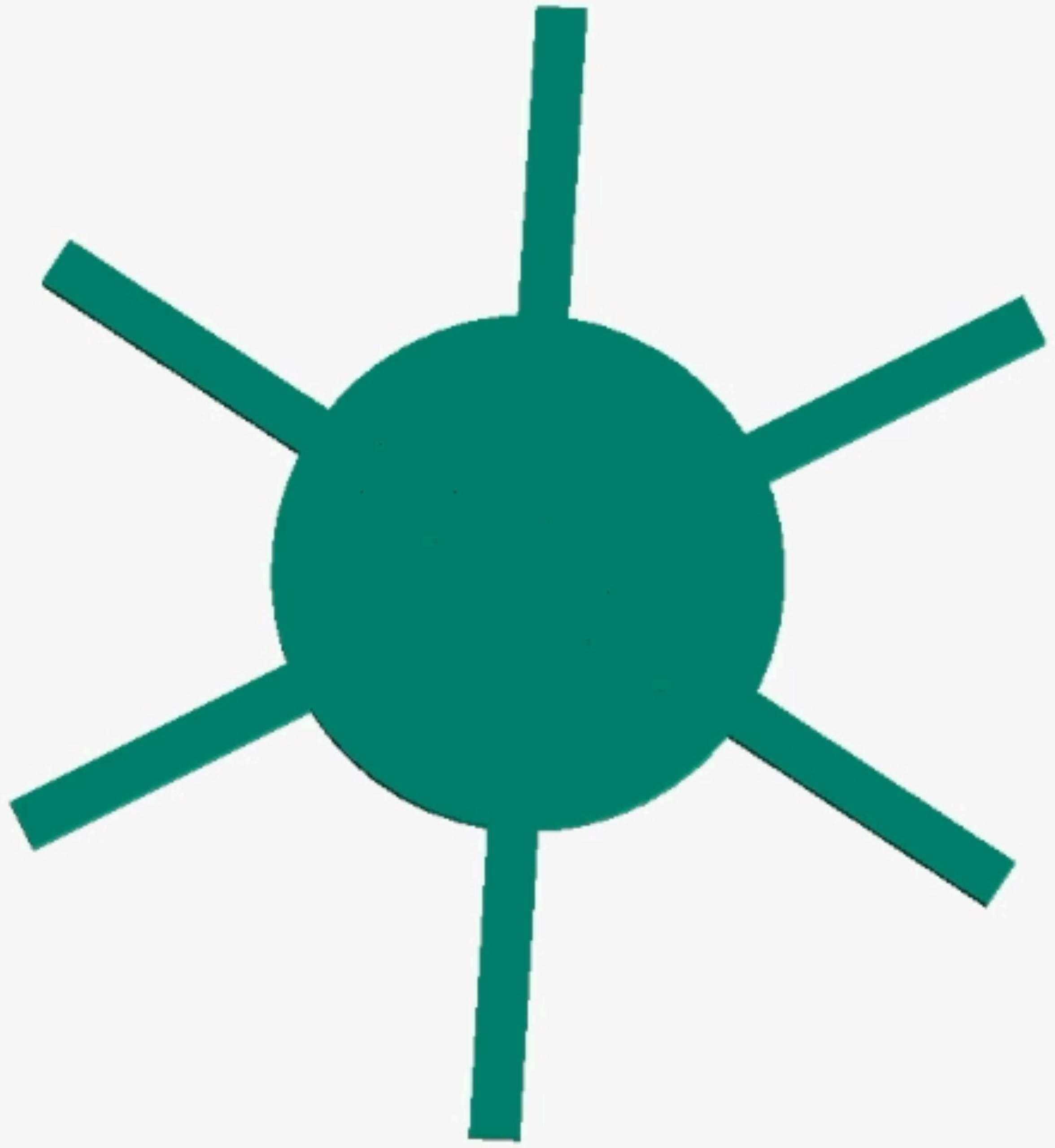


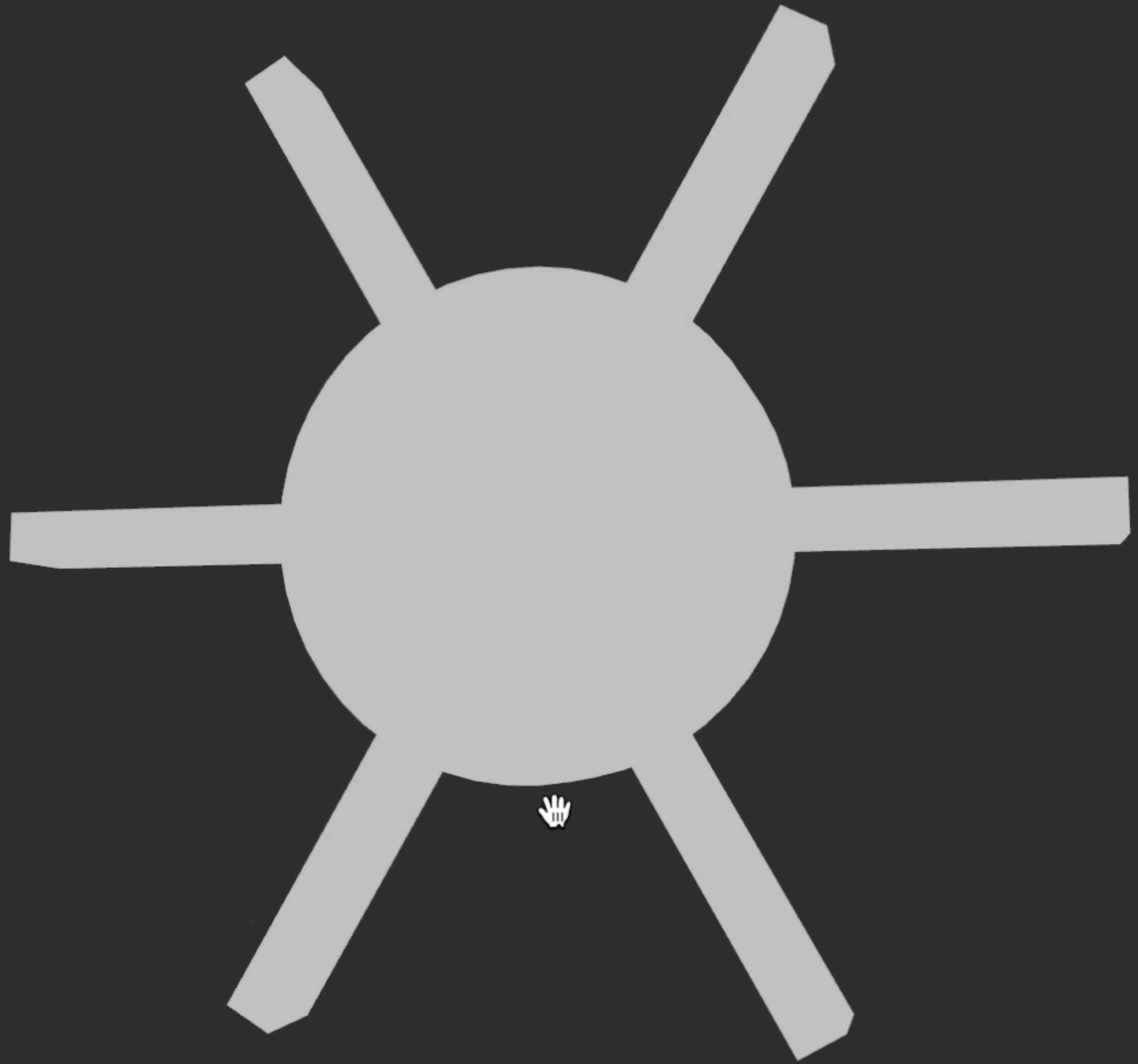



CAD and 3D Printing everywhere!

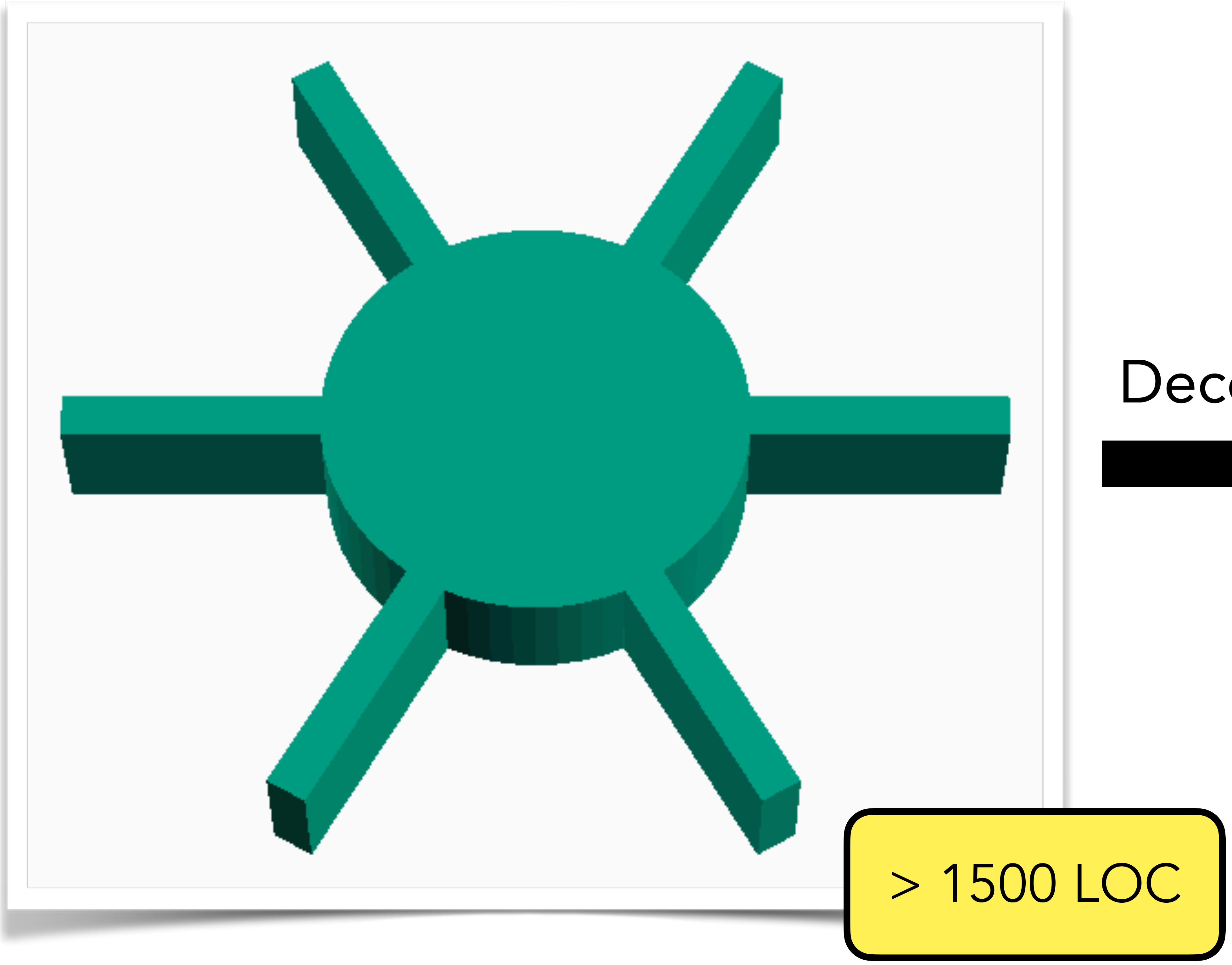


```
n = 6;

cylinder(h= 2, r=5, $fn=50);

for (i = [0:n-1]) {
    rotate([0, 0, i * 360 / n])
    translate([1, -0.5, 0])
    cube([10, 1, 2]);
}
```





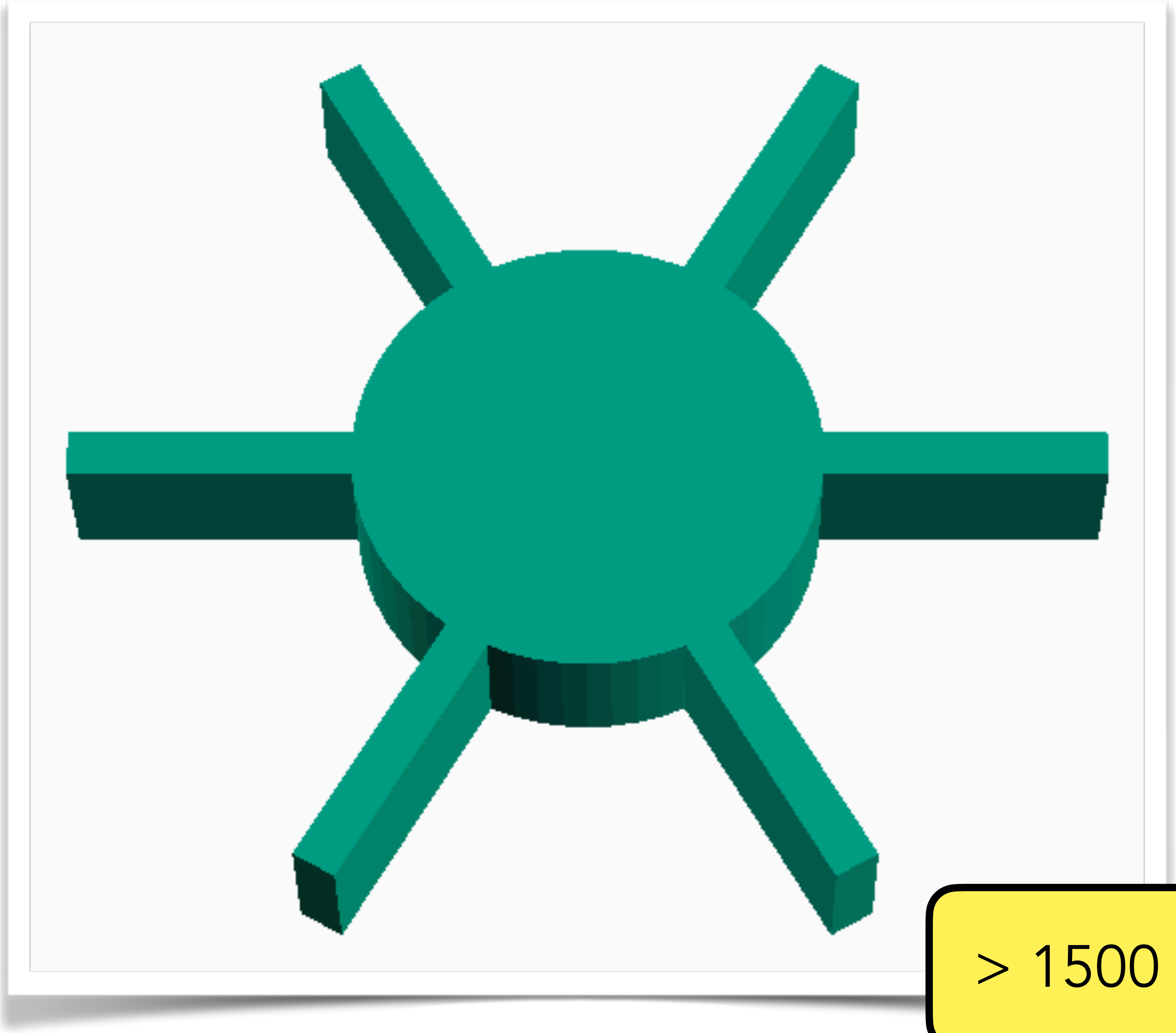
Mesh Decompilers Recover Flat Programs




```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [–1,0.5,0]
      (Scale [–1,–1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

* Reincarnate [ICFP 2018], InverseCSG [SIGGRAPH Asia 2018], Shape2Prog [ICLR 2019], CSGNet [CVPR 2018], …

Mesh Decompilers Recover Flat Programs



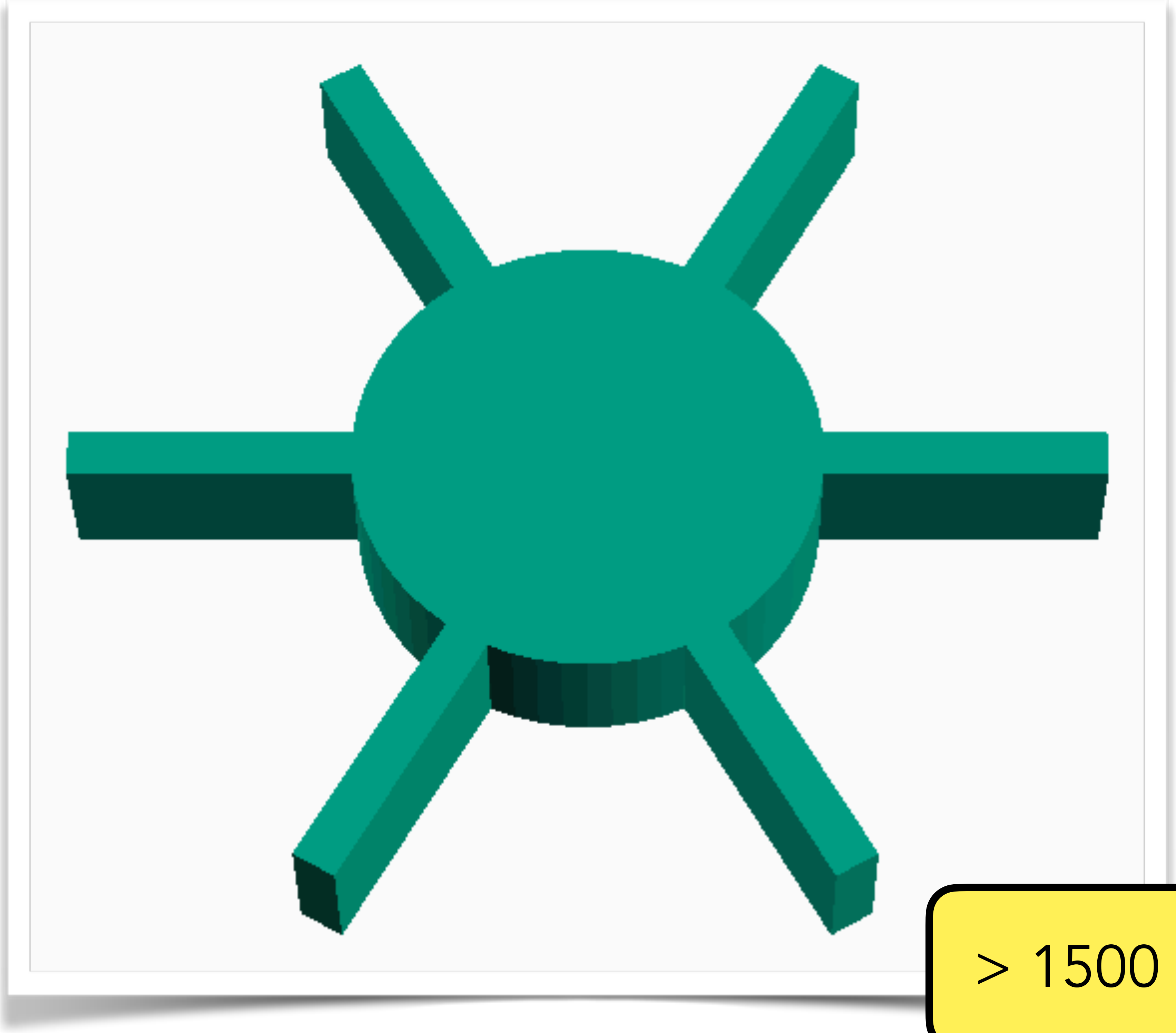
Mesh Decompilers *

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [–1,0.5,0]
      (Scale [–1,–1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

Primitives

* Reincarnate [ICFP 2018], InverseCSG [SIGGRAPH Asia 2018], Shape2Prog [ICLR 2019], CSGNet [CVPR 2018], …

Mesh Decompilers Recover Flat Programs



Mesh Decompilers *

Affine operators

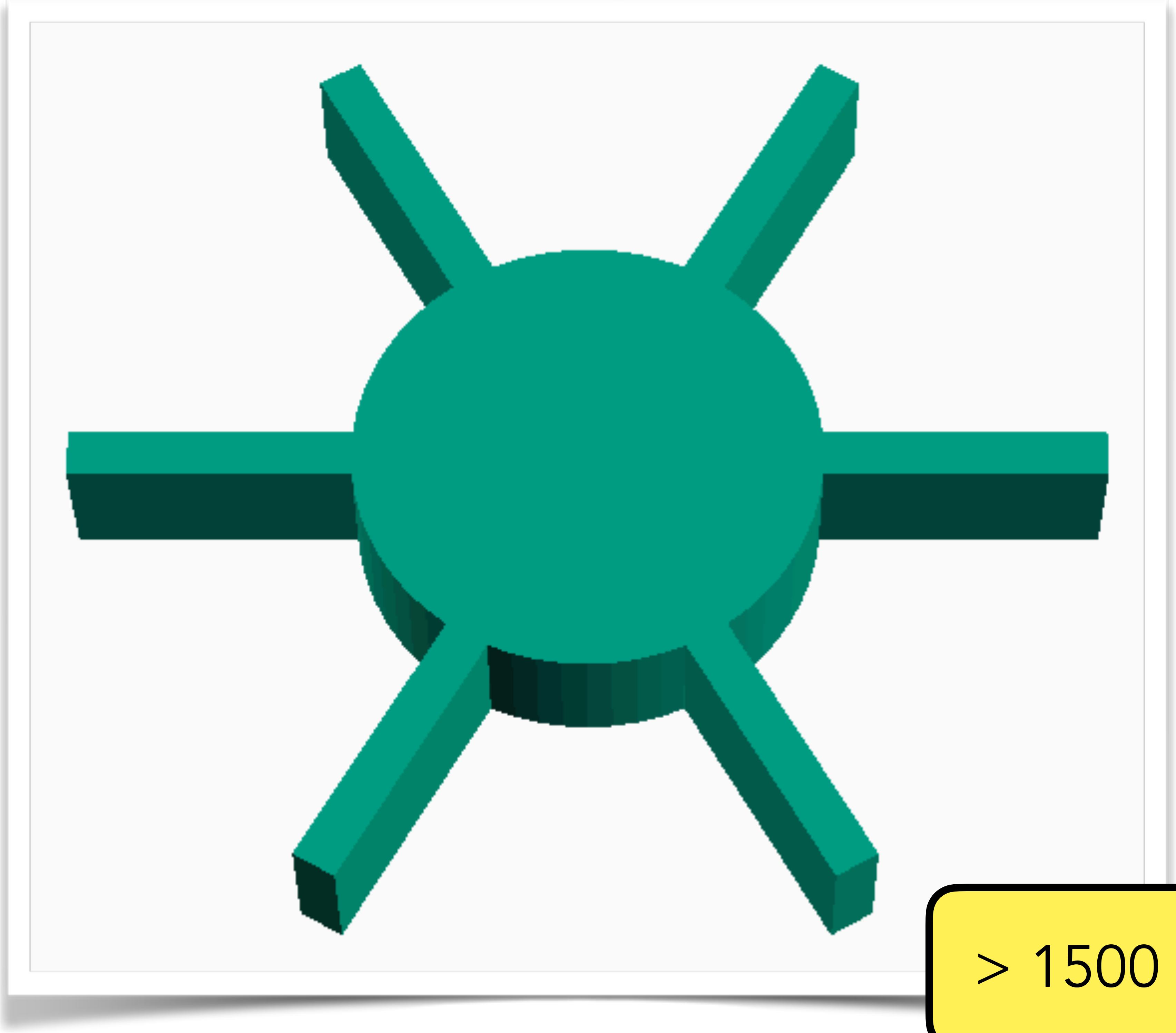
Primitives

> 1500 LOC

```
(Union
 (Scale [5,5,1] (Cylinder [1,1]))
 (Union
  (Rotate [0,0,120]
   (Translate [1,−0.5,0] (Cuboid [10,1,1])))
  (Scale [10,1,1]
   (Translate [0.1,−0.5,1] (Cuboid [1,1,1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Translate [−1,0.5,0]
   (Scale [−1,−1,1] Cuboid [10,1,1]))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

* Reincarnate [ICFP 2018], InverseCSG [SIGGRAPH Asia 2018], Shape2Prog [ICLR 2019], CSGNet [CVPR 2018], …

Mesh Decompilers Recover Flat Programs



> 1500 LOC

Mesh Decompilers *

Affine operators

Binary operators

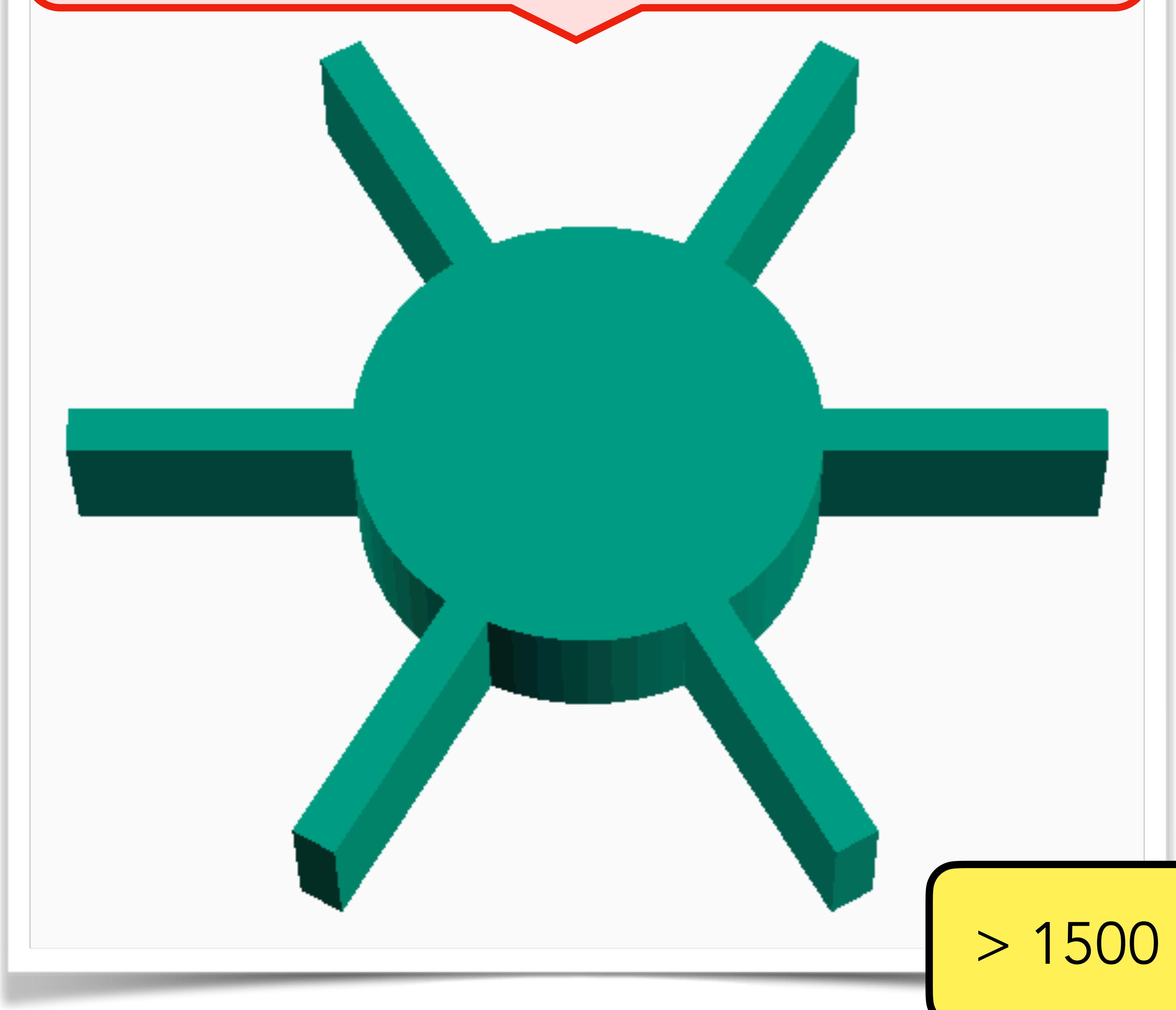
Primitives

```
(Union
 (Scale [5,5,1] (Cylinder [1,1]))
 (Union
  (Rotate [0,0,120]
   (Translate [1,–0.5,0] (Cuboid [10,1,1])))
  (Scale [10,1,1]
   (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
  (Rotate [0, 0, 300]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Translate [–1,0.5,0]
   (Scale [–1,–1,1] Cuboid [10,1,1]))
  (Rotate [0, 0, 240]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

* Reincarnate [ICFP 2018], InverseCSG [SIGGRAPH Asia 2018], Shape2Prog [ICLR 2019], CSGNet [CVPR 2018], …

Mesh Decompilers Recover Flat Programs

Repetition of spokes is not captured by flat program



> 1500 LOC

Mesh Decompilers *

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [–1,0.5,0]
      (Scale [–1,–1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

* Reincarnate [ICFP 2018], InverseCSG [SIGGRAPH Asia 2018], Shape2Prog [ICLR 2019], CSGNet [CVPR 2018], …

Szalinski: flat CAD → parametrized CAD

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,−0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,−0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [−1,0.5,0]
      (Scale [−1,−1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

Szalinski →

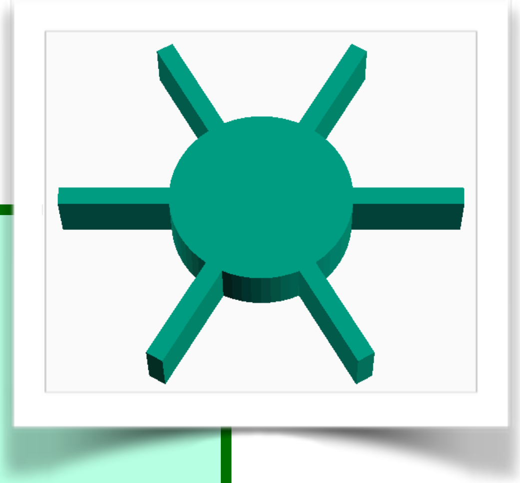
This talk

Szalinski: flat CAD → parametrized CAD

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [−1,0.5,0]
      (Scale [−1,−1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))))
```

Szalinski →

This talk



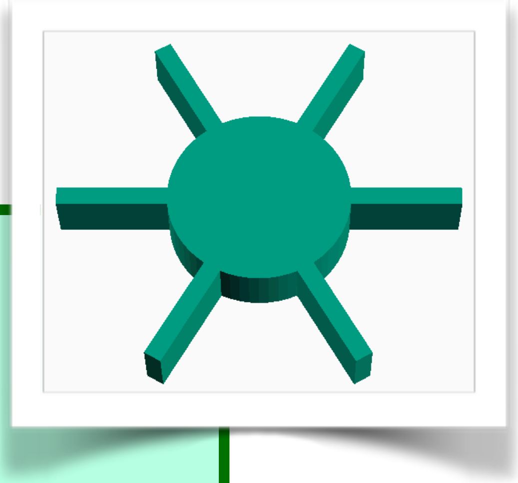
```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Szalinski: flat CAD → parametrized CAD

```
(Union
 (Scale [5,5,1] (Cylinder [1,1]))
 (Union
  (Rotate [0,0,120]
   (Translate [1,–0.5,0] (Cuboid [10,1,1])))
  (Scale [10,1,1]
   (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
  (Rotate [0, 0, 300]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Translate [−1,0.5,0]
   (Scale [−1,−1,1] Cuboid [10,1,1]))
  (Rotate [0, 0, 240]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))))
```

Szalinski

This talk



```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

A language, called *Caddy* that supports CAD features & functional programming features like Fold, Tabulate, Map

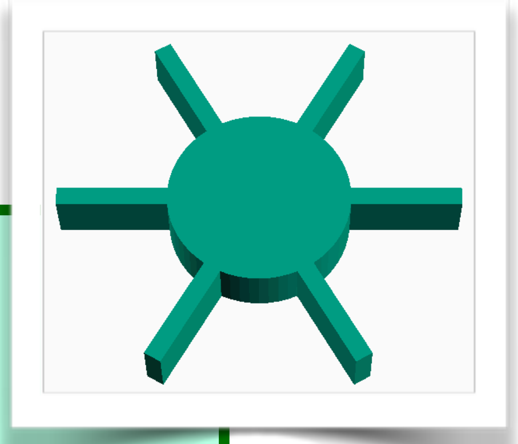
Fold and Tabulate represent loops

Szalinski: flat CAD → parametrized CAD

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [–1,0.5,0]
      (Scale [–1,–1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

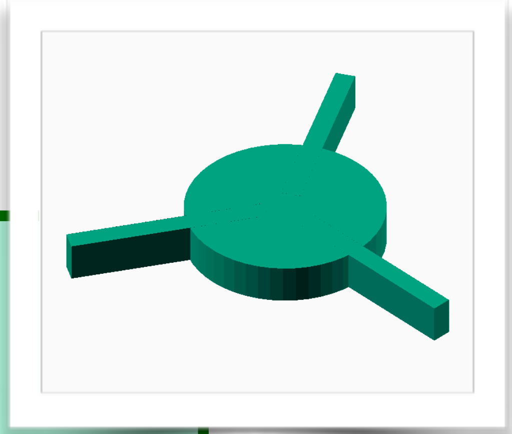
Szalinski

This talk



```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

edits

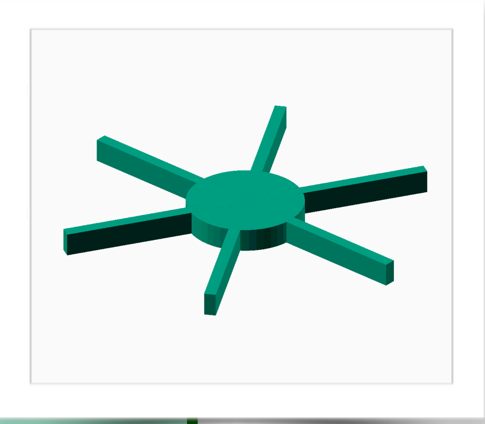


```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 3)
   (Rotate [0, 0, 120i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

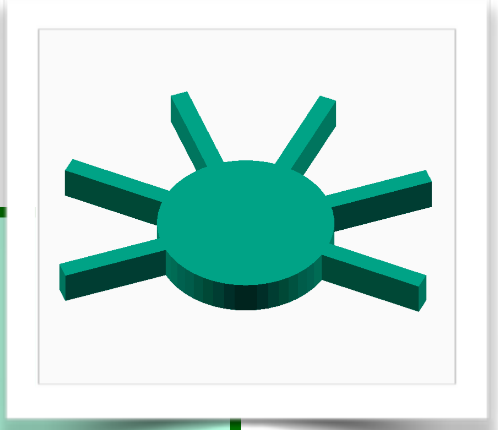
edits

Fold and Tabulate represent loops

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 8)
   (Rotate [0, 0, 45i]
    (Translate [1,–0.5,0]
     (Cuboid [15, 1, 1]))))))
```



```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [15, 1, 1]))))))
```



```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 45i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

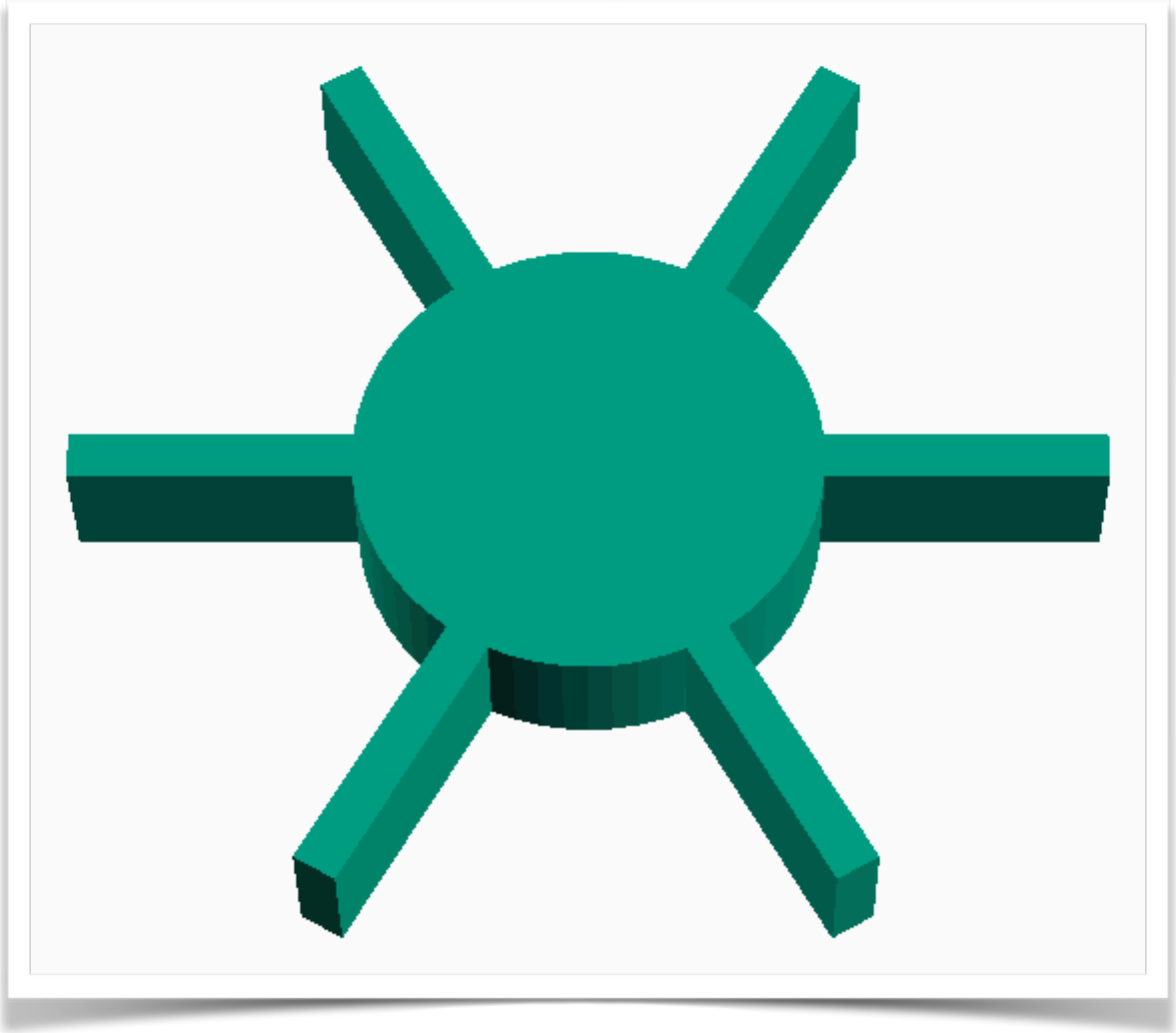
Szalinski: flat CAD ➔ parametrized CAD

Automatically infer loops from straight line programs in the form of **Folds**, **Maps**, and **Tabulates**

Hypothesis: Parametrized programs are easier to read/customize than flat programs

Ideal Input to Szalinski

```
(Union
 (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```



Term Rewriting

```
(Union (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

Term Rewriting

```
(Union (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

Fold Union Rewrite

```
(Union (Cylinder [1, 5, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
...
```

Term Rewriting

```
(Union (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

Fold Union Rewrite

```
(Union (Cylinder [1, 5, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
...
```

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] ... [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) ...
```

Term Rewriting

(Union (Cylinder [1, 5])
(Union
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 120]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 180]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))

→

Fold Union Rewrite

(Union (Cylinder [1, 5, 5])
(Fold Union (List
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
. . .

→

Structure Finder

(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
(List
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …

↓ *

Structure Finder

(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
(Map2 Translate
(**Repeat** 6 [1, −0.5, 0]
(**Repeat** 6 (Cuboid [10, 1, 1]))))))


Term Rewriting
(Union (Cylinder [1, 5])
(Union
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 120]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 180]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
Fold Union Rewrite
(Union (Cylinder [1, 5, 5])
(Fold Union (List
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
...
Structure Finder
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] ... [0, 0, 300])
(List
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) ...
*
Structure Finder
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] ... [0, 0, 300])
(Map2 Translate
(Repeat 6 [1, −0.5, 0]
(Repeat 6 (Cuboid [10, 1, 1]))))))
*
Custom Solver
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(Tabulate (i 6) (0, 0, 60i))
(Map2 Translate
(Repeat 6 [1, −0.5, 0]
(Repeat 6 (Cuboid [10, 1, 1]))))))

Term Rewriting


Term Rewriting
(Union (Cylinder [1, 5])
(Union
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 120]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 180]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
Fold Union Rewrite
(Union (Cylinder [1, 5, 5])
(Fold Union (List
(Rotate [0, 0, 0]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
. . .
Structure Finder
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
(List
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
*
Structure Finder
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
(Map2 Translate
(Repeat 6 [1, −0.5, 0]
(Repeat 6 (Cuboid [10, 1, 1]))))))
*
Custom Solver
(Union (Cylinder [1, 5, 5])
(Fold Union
(Map2 Rotate
(Tabulate (i 6) (0, 0, 60i))
(Map2 Translate
(Repeat 6 [1, −0.5, 0]
(Repeat 6 (Cuboid [10, 1, 1]))))))
*
Lift Tabulate Rewrite
(Union (Cylinder [1, 5, 5])
(Fold Union
(Tabulate (i 6)
(Rotate [0, 0, 60i]
(Translate [1,−0.5,0]
(Cuboid [10, 1, 1]))))))

Structure Finder

```
List
  (Op [param 1] (arg 1))
  (Op [param 2] (arg 2))
  (Op [param 3] (arg 3)) …
```

→

```
Map2 Op
  (List [param 1] [param 2] [param 3])
  (List (arg 1) (arg 2) (arg 3))
```

Structure Finder

```
List
  (Op [param 1] (arg 1))
  (Op [param 2] (arg 2))
  (Op [param 3] (arg 3)) …
```

→

```
Map2 Op
  (List [param 1] [param 2] [param 3])
  (List (arg 1) (arg 2) (arg 3))
```

Fold Union Rewrite

```
(Union (Cylinder [1, 5])
 (Fold Union (List
 (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

→

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
 (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] [0, 0, 120] [0, 0, 180] [0, 0, 240] [0, 0, 300])
   (List
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```

List 1

List 2

Structure Finder

```
List
  (Op [param 1] (arg 1))
  (Op [param 2] (arg 2))
  (Op [param 3] (arg 3)) …
```

→

```
Map2 Op
  (List [param 1] [param 2] [param 3])
  (List (arg 1) (arg 2) (arg 3))
```

Fold Union Rewrite

```
(Union (Cylinder [1, 5])
 (Fold Union (List
 (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

→

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
 (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] [0, 0, 120] [0, 0, 180] [0, 0, 240] [0, 0, 300])
   (List
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
     (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```

Map2 applies the operator to the i^{th} element of the first list and i^{th} element of the second list

List 1

List 2

Structure Finder

```
List
  (Op [param 1] (arg 1))
  (Op [param 2] (arg 2))
  (Op [param 3] (arg 3)) …
```

→

```
Map2 Op
  (List [param 1] [param 2] [param 3])
  (List (arg 1) (arg 2) (arg 3))
```

Fold Union Rewrite

```
(Union (Cylinder [1, 5])
 (Fold Union (List
 (Rotate [0, 0, 0]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 60]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 120]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 180]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 240]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
 (Rotate [0, 0, 300]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

→

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
 (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] [0, 0, 120] [0, 0, 180] [0, 0, 240] [0, 0, 300])
   (List
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])) …
```

Map2 applies the operator to the i^th element of the first list and i^th element of the second list

List 1

List 2

Also applies to this list

Custom Solvers

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (Map2 Translate
    (Repeat 6 [1, −0.5, 0]
    (Repeat 6 (Cuboid [10, 1, 1]))))))
```

Custom solver

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (Tabulate (i 6) (0, 0, 60i))
   (Map2 Translate
    (Repeat 6 [1, −0.5, 0]
    (Repeat 6 (Cuboid [10, 1, 1]))))))
```

The concrete list of vectors is passed to a custom solver that finds a closed form arithmetic expression

Term Rewriting

Term Rewriting

Inputs to Szalinski are rarely ideal!

Ideal Input vs Actual Input

```
(Union
 (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,−0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,−0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [−1,0.5,0]
      (Scale [−1,−1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))
```

Ideal Input vs Actual Input

```
(Union
 (Cylinder [1, 5])
 (Union
  (Rotate [0, 0, 0]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Union
    (Rotate [0,0,120]
      (Translate [1,–0.5,0] (Cuboid [10,1,1])))
    (Scale [10,1,1]
      (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
    (Rotate [0, 0, 300]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Translate [–1,0.5,0]
      (Scale [–1,–1,1] Cuboid [10,1,1]))
    (Rotate [0, 0, 240]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
    (Rotate [0, 0, 60]
      (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

Ideal Input vs Actual Input

Previous rewriting strategy no longer works!

```
(Rotate [0, 0, 0]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 120]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 180]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

```
(Rotate [0,0,120]
  (Translate [1,–0.5,0] (Cuboid [10,1,1])))
(Scale [10,1,1]
  (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
(Rotate [0, 0, 300]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Translate [–1,0.5,0]
  (Scale [–1,–1,1] Cuboid [10,1,1]))
(Rotate [0, 0, 240]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

Ideal Input vs Actual Input

Must <u>interleave</u> rewriting strategy with CAD identities to line up subexpressions

```
(Rotate [0, 0, 60]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 120]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 180]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
  (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

```
  (Scale [10,1,1]
    (Translate [0.1,–0.5,1] (Cuboid [1,1,1])))
  (Rotate [0, 0, 300]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Translate [–1,0.5,0]
    (Scale [–1,–1,1] Cuboid [10,1,1]))
  (Rotate [0, 0, 240]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))))
```

Ideal Input vs Actual Input

Must <u>interleave</u> rewriting strategy with CAD identities to line up subexpressions

Phase ordering problem: order of rewriting matters!

(Rotate [0, 0, 60]

(Rotate [0, 0, 180]
  (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 240]
  (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 300]
  (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))

(Scale [10,1,1]

(Translate [−1,0.5,0]
  (Scale [−1,−1,1] Cuboid [10,1,1]))
(Rotate [0, 0, 240]
  (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
(Rotate [0, 0, 60]
  (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))

Ideal Input vs Actual Input

Must <u>interleave</u> rewriting strategy with CAD identities to line up subexpressions

(Rotate [0, 0, 60]

(Scale [10,1,1]

Phase ordering problem: order of rewriting matters!

(Rotate [0, 0, 180]

(Translate [−1,0.5,0]

E-graphs* can solve phase ordering

(Rotate [0, 0, 300]

(Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))

(Rotate [0, 0, 60]

(Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))

* Equality Saturation: A New Approach to Optimization. Tate, Stepp, Tatlock, Lerner. POPL'09

Semantically Equivalent, Syntactically Different

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Translate [1, −0.5, 0] (Cube [10, 1, 1]))
  (Rotate [0,0,60]
   (Translate [1,−0.5,0] (Cube[10,1,1])))
  (Rotate [0,0,120]
   (Translate [1,−0.5,0] (Cube[10,1,1])))
  (Scale  [−1,−1,1]
   (Translate [1,−0.5,0] (Cube[10,1,1])))
  (Rotate [0,0,240]
   (Translate [1,−0.5,0] (Cube[10,1,1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cube [10, 1, 1]))))))
```

Rotate [0, 0, 180] is replaced by equivalent Scale [-1, -1, 1]

Semantically Equivalent, Syntactically Different

Rotate [0, 0, 180]
  (Translate [1,–0.5,0] (Cube[10,1,1]))

=

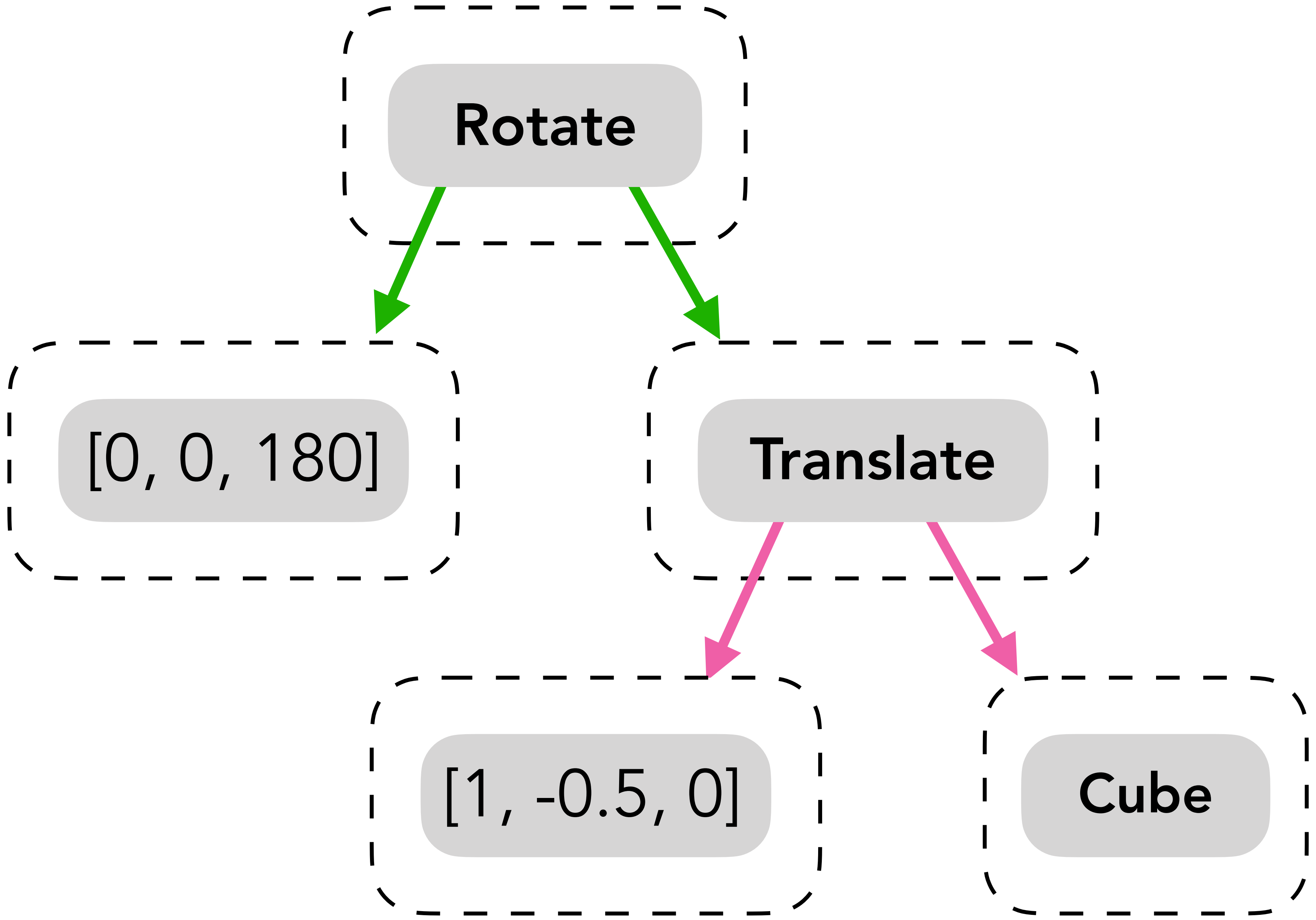
Scale [-1, -1, 1]
  (Translate [1,–0.5,0] (Cube[10,1,1]))

Syntactic rewrite

$$Rotate\ (0,\ 0,\ 180,\ c)) \leftrightsquigarrow Scale\ (-1,\ -1,\ 1,\ c))$$

Store Expressions in an E-graph

Rotate [0, 0, 180]
(Translate [1,–0.5,0] (Cube[10,1,1])) = Scale [-1, -1, 1]
(Translate [1,–0.5,0] (Cube[10,1,1]))

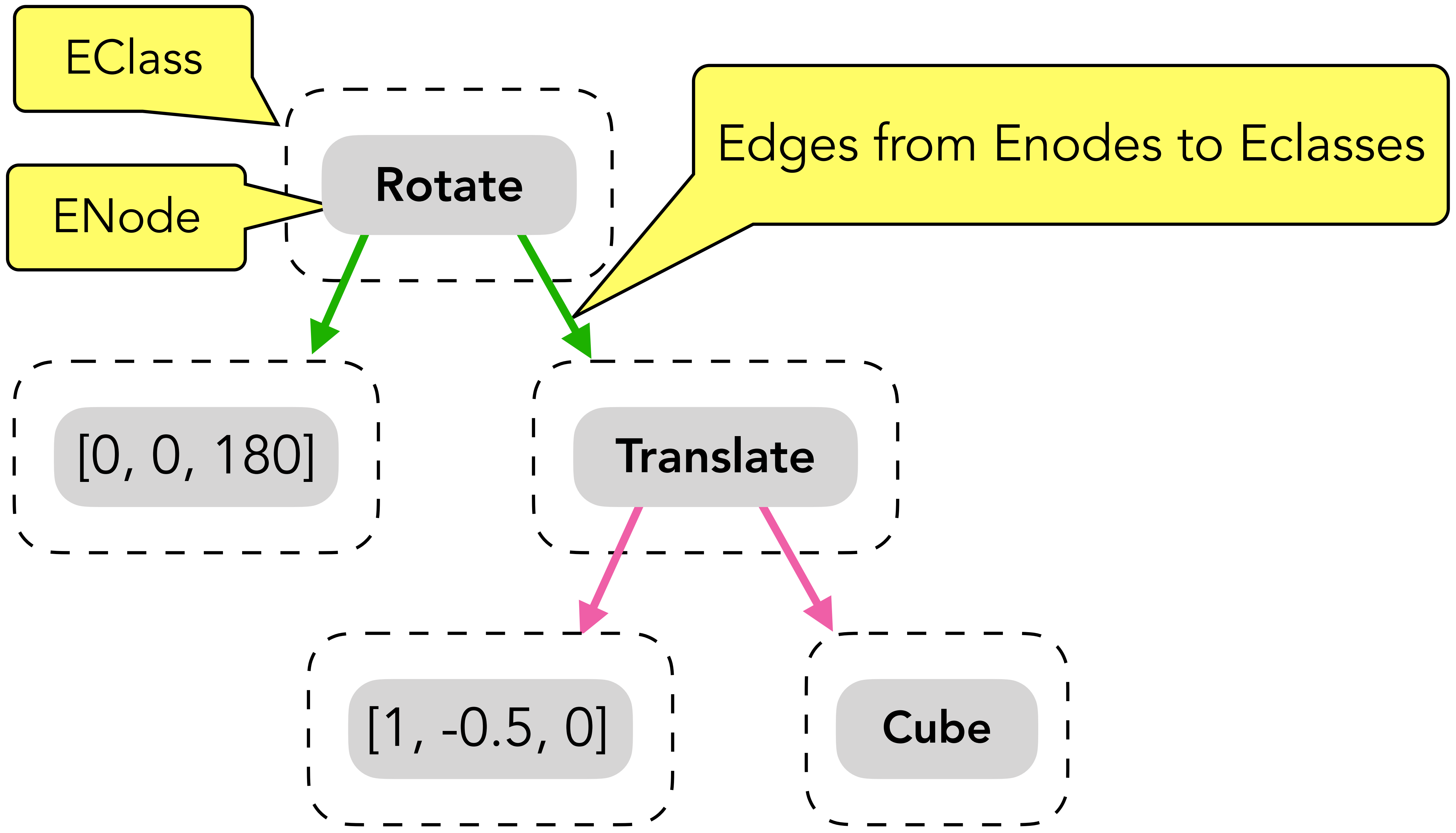


Syntactic rewrite

$$\mathit{Rotate}\ (0,\ 0,\ 180,\ c)) \rightsquigarrow \mathit{Scale}\ (-1,\ -1,\ 1,\ c))$$

Store Expressions in an E-graph

Rotate [0, 0, 180] (Translate [1,–0.5,0] (Cube[10,1,1])) = Scale [-1, -1, 1] (Translate [1,–0.5,0] (Cube[10,1,1]))

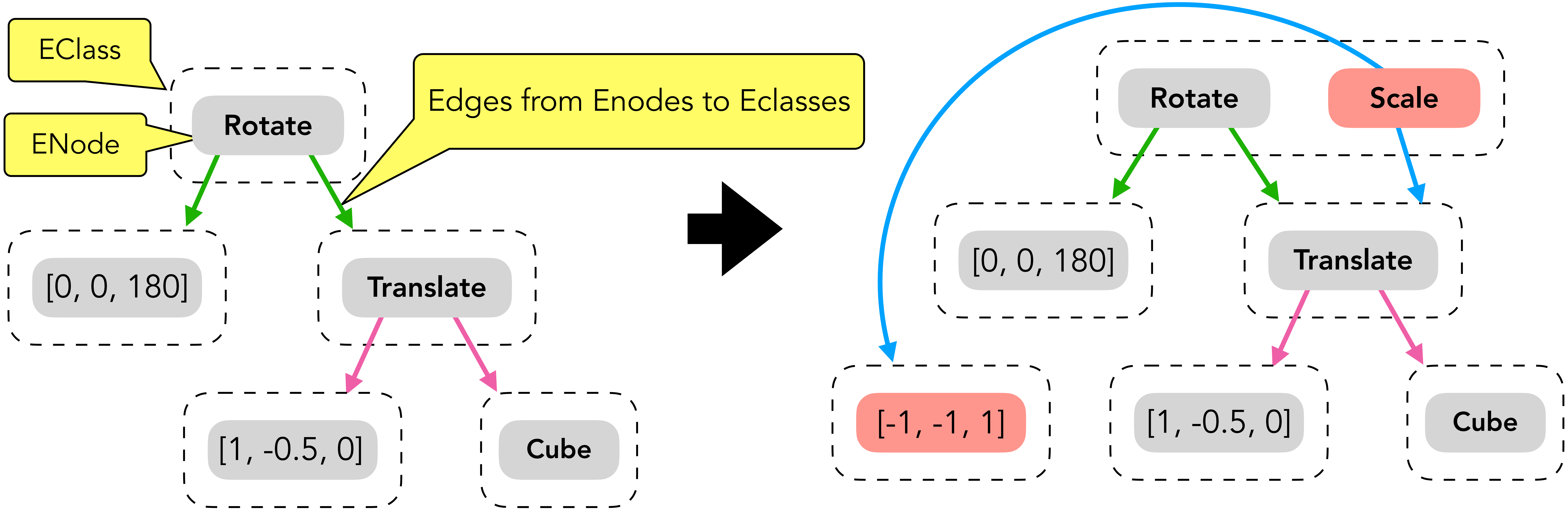


Syntactic rewrite

$$\mathit{Rotate}\ (0,\ 0,\ 180,\ c)) \leftrightsquigarrow \mathit{Scale}\ (-1,\ -1,\ 1,\ c))$$

Store Expressions in an E-graph

Rotate [0, 0, 180]
(Translate [1,–0.5,0] (Cube[10,1,1])) = Scale [-1, -1, 1]
(Translate [1,–0.5,0] (Cube[10,1,1]))



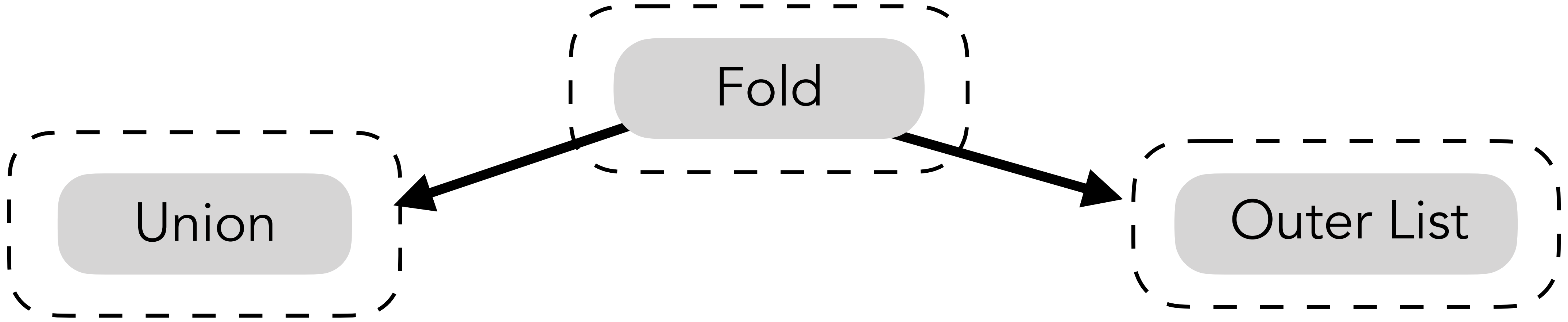
Syntactic rewrite

$$Rotate\ (0,\ 0,\ 180,\ c)) \leftrightsquigarrow Scale\ (-1,\ -1,\ 1,\ c))$$

Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Outer list



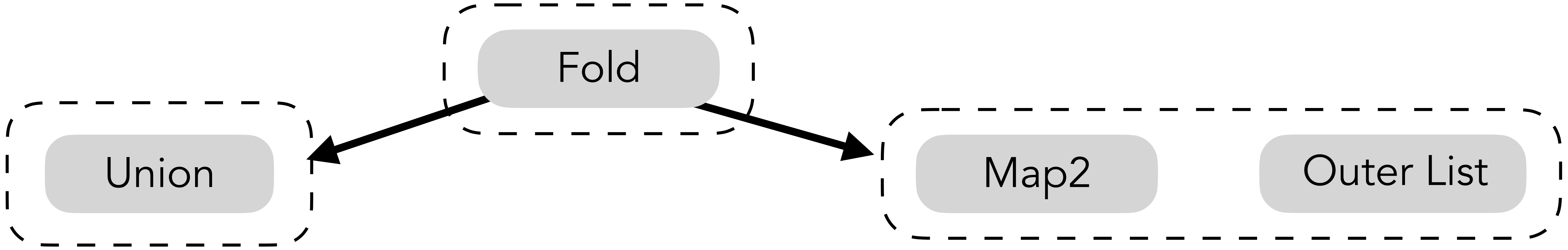
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



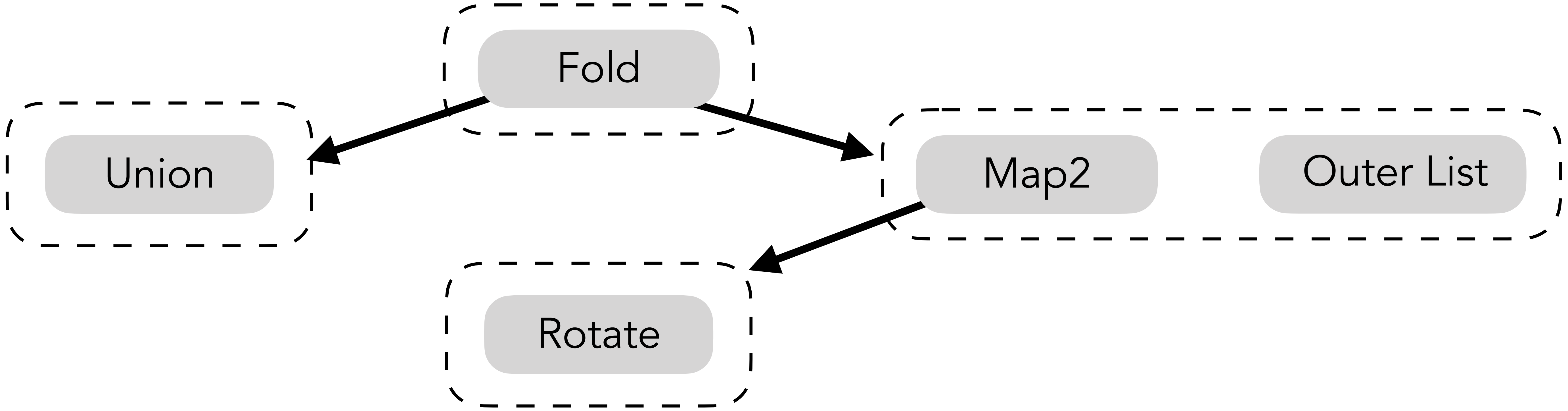
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



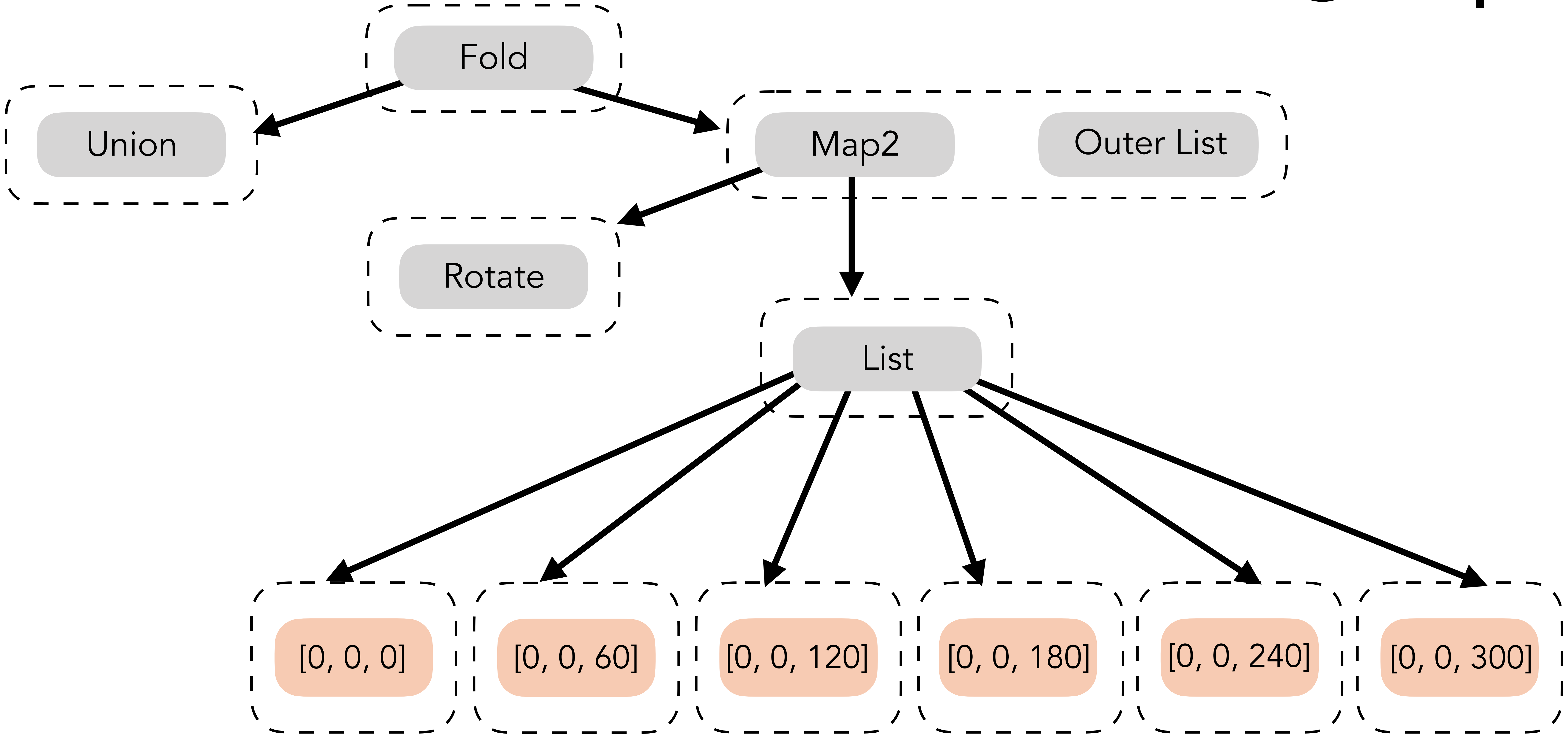
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



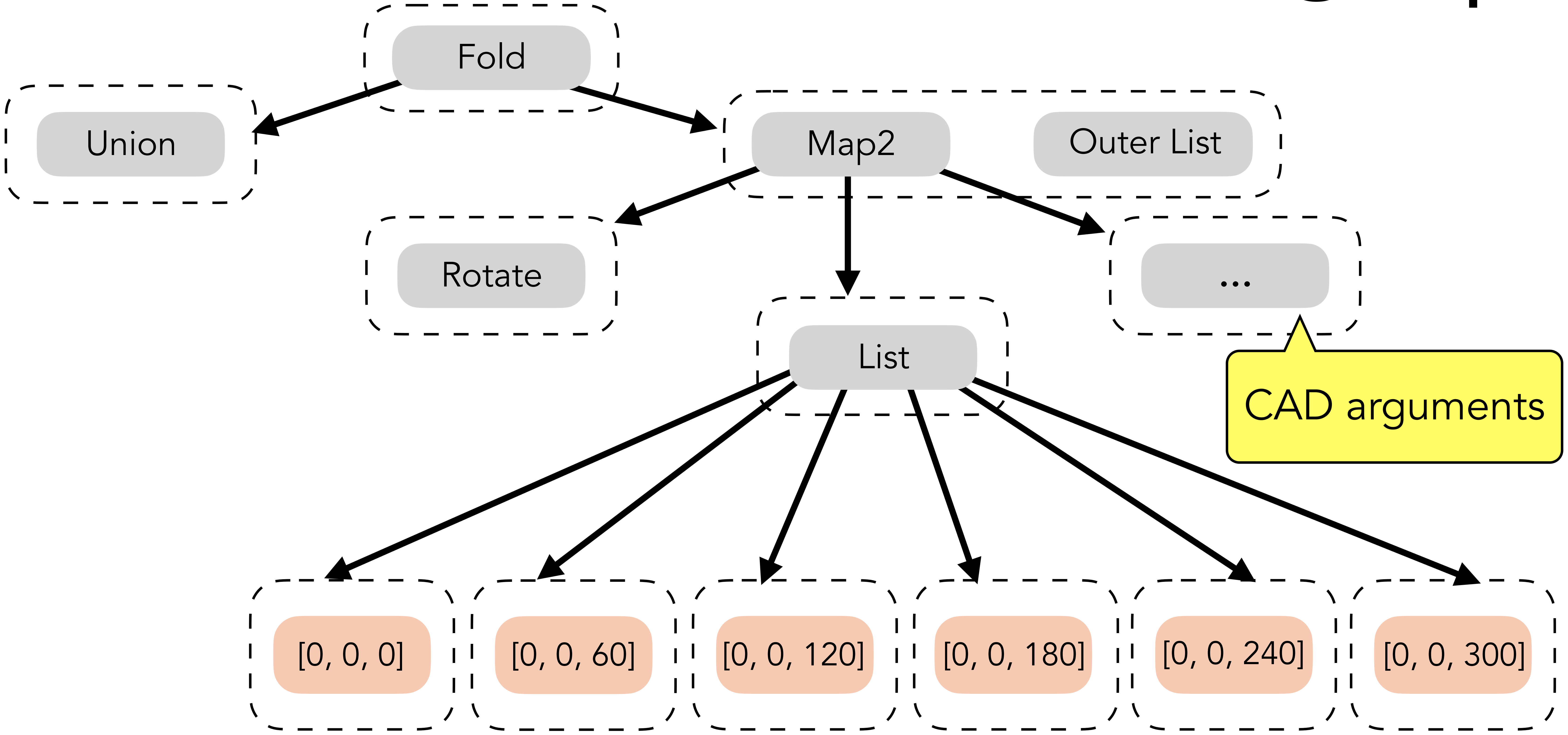
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



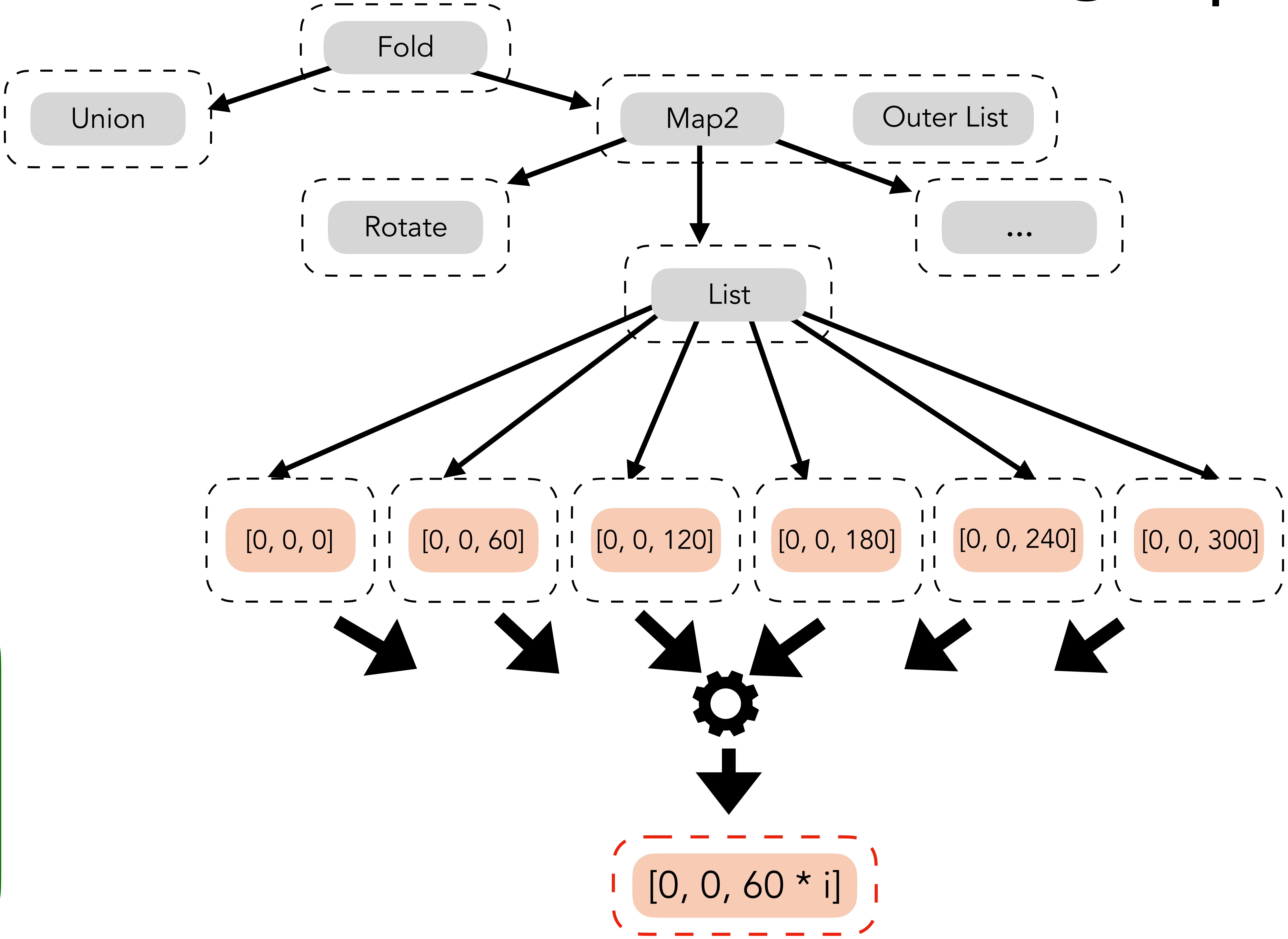
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

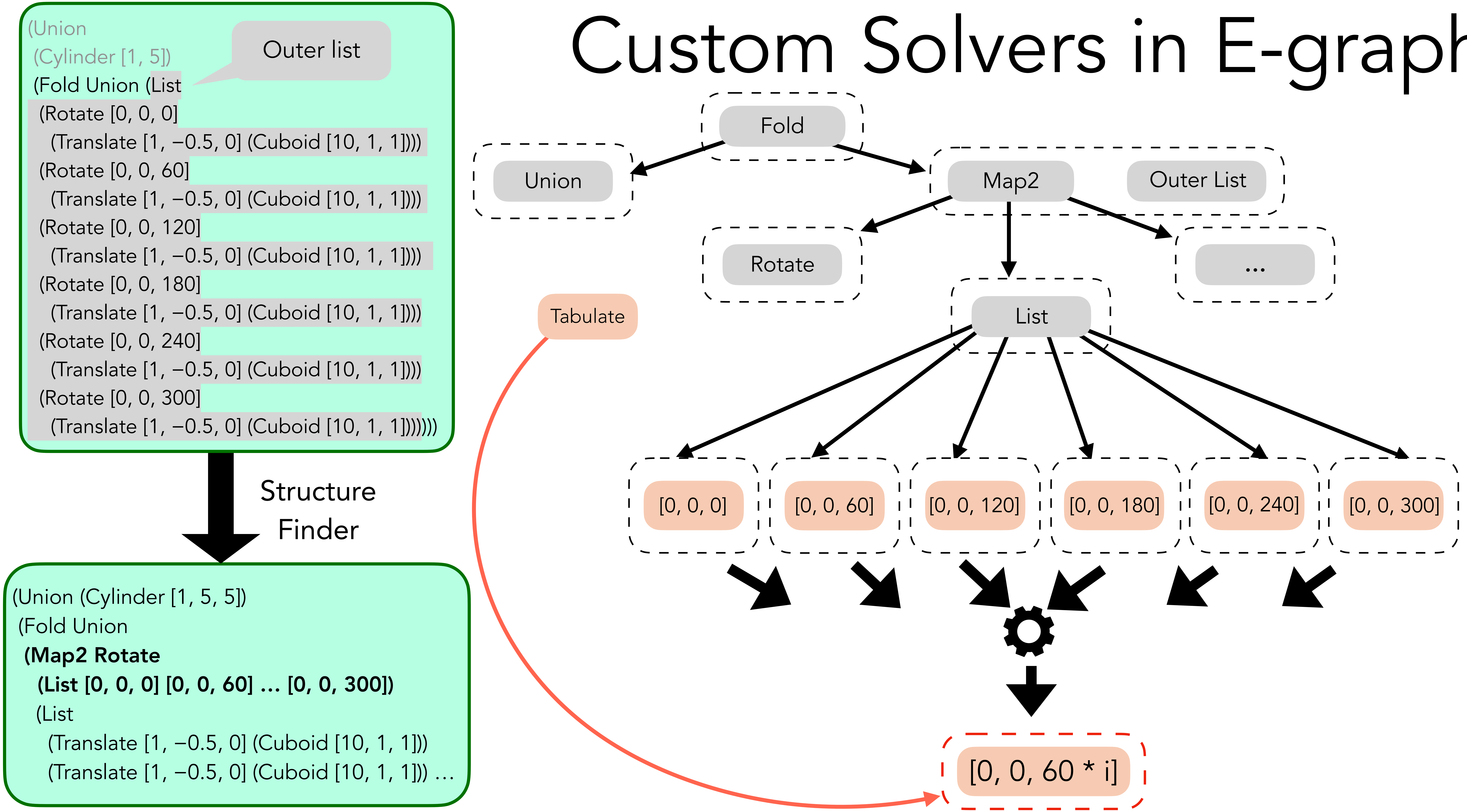
Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



Custom Solvers in E-graph



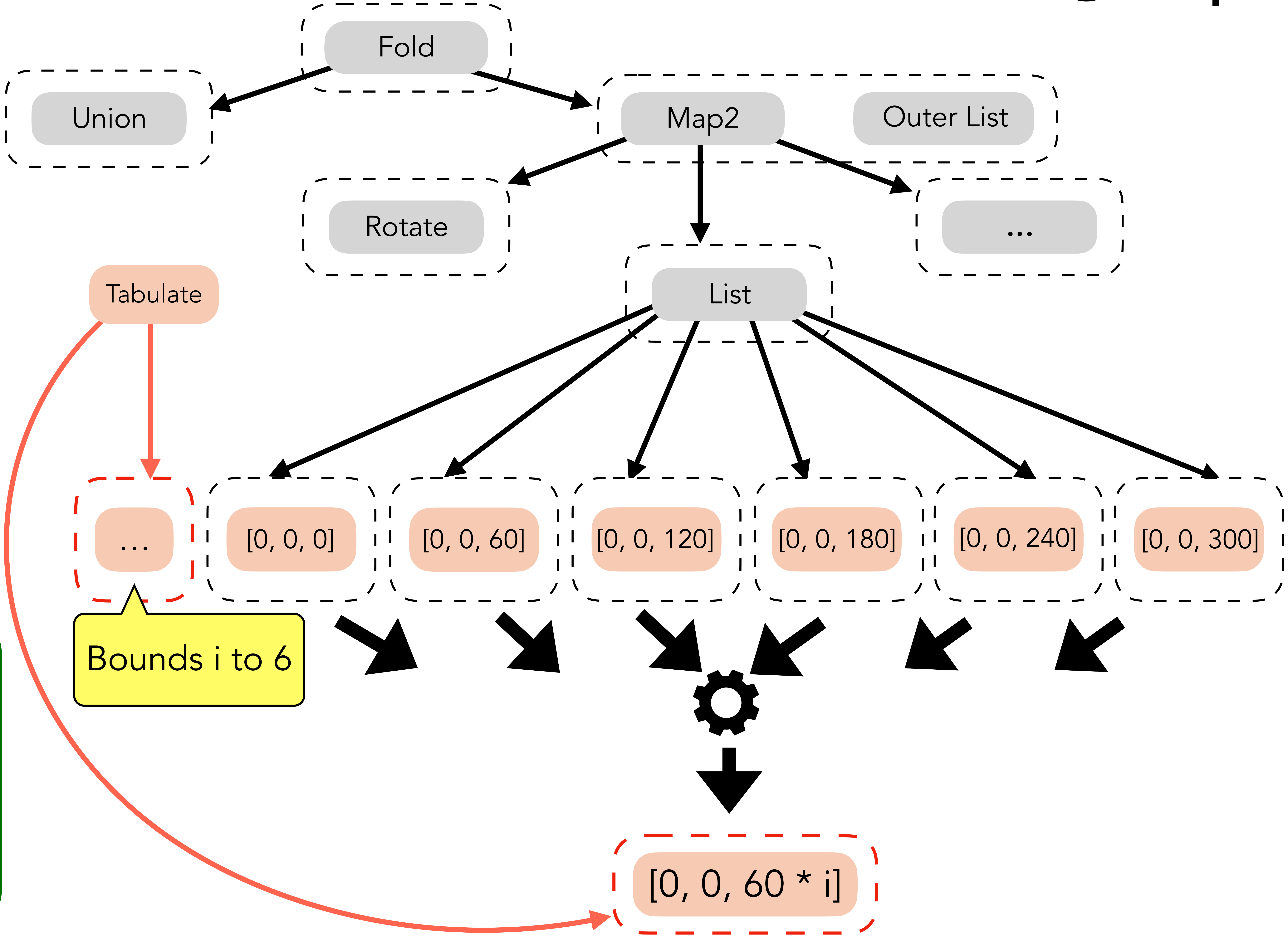
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



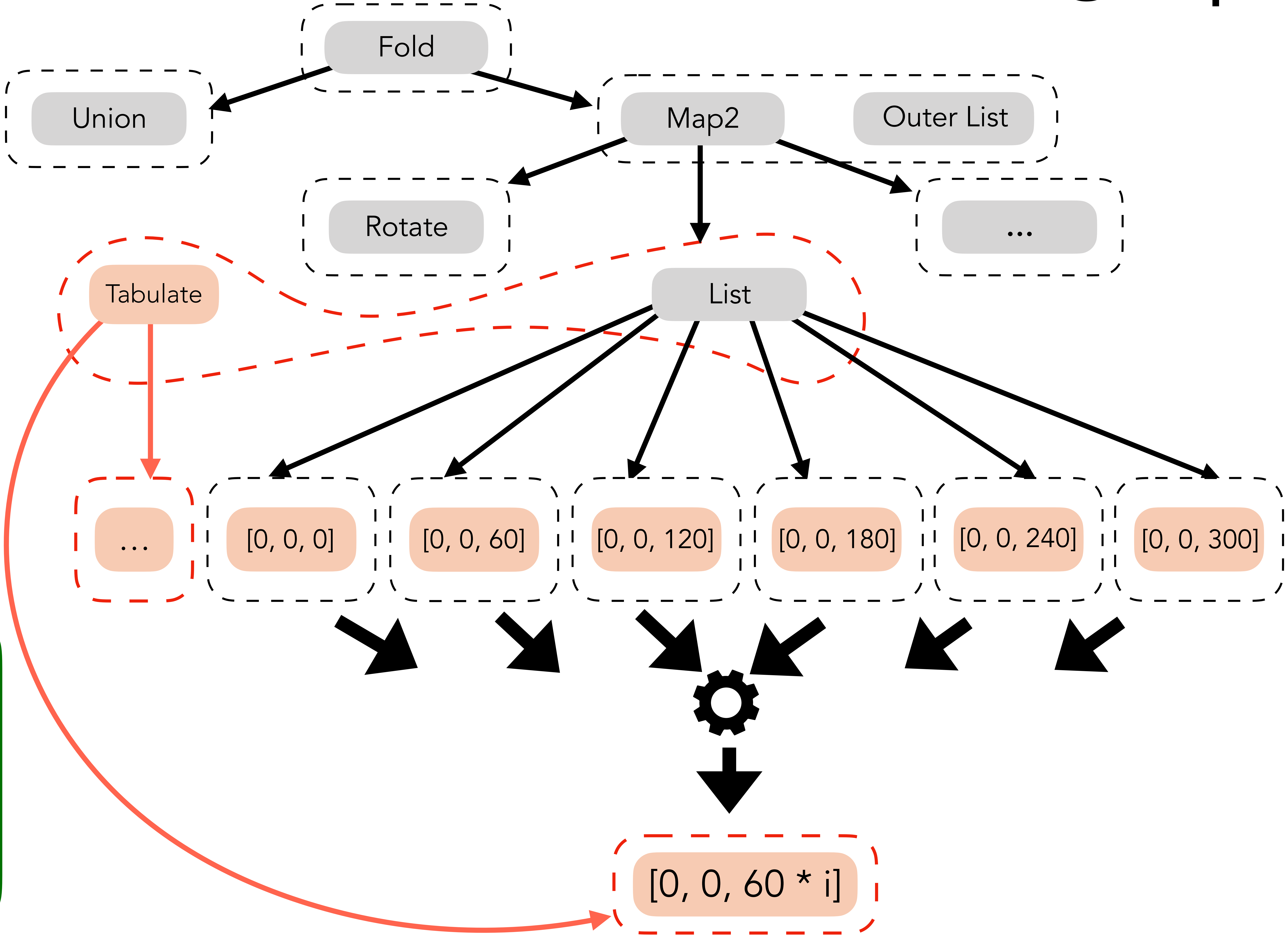
Custom Solvers in E-graph

```
(Union
 (Cylinder [1, 5])
 (Fold Union (List
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Outer list

Structure Finder

```
(Union (Cylinder [1, 5, 5])
 (Fold Union
  (Map2 Rotate
   (List [0, 0, 0] [0, 0, 60] … [0, 0, 300])
   (List
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])) …
```



Custom Solvers for Non-Ideal Inputs

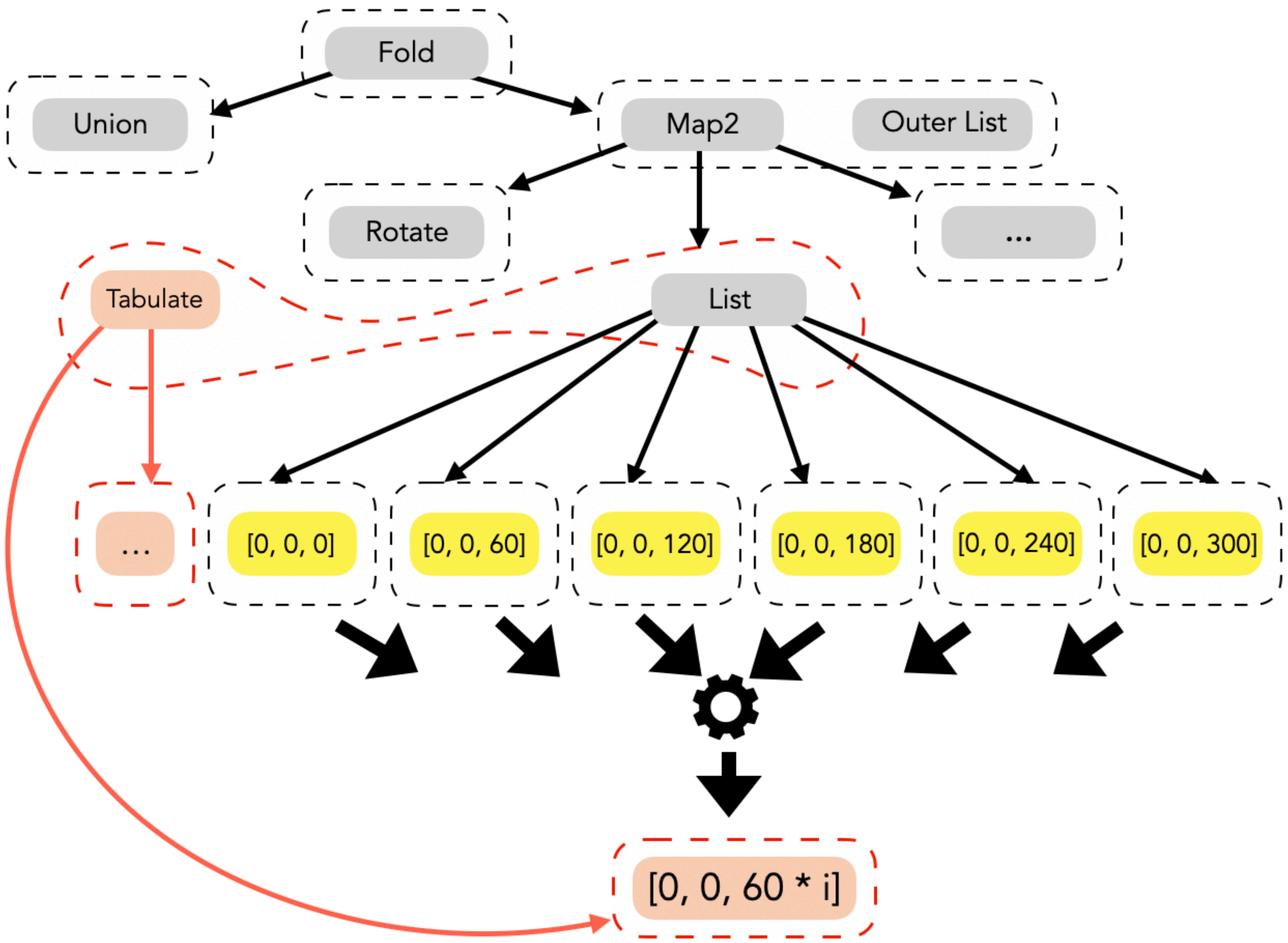
```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Expressions are arbitrarily ordered

Parameters of Rotate are not sorted

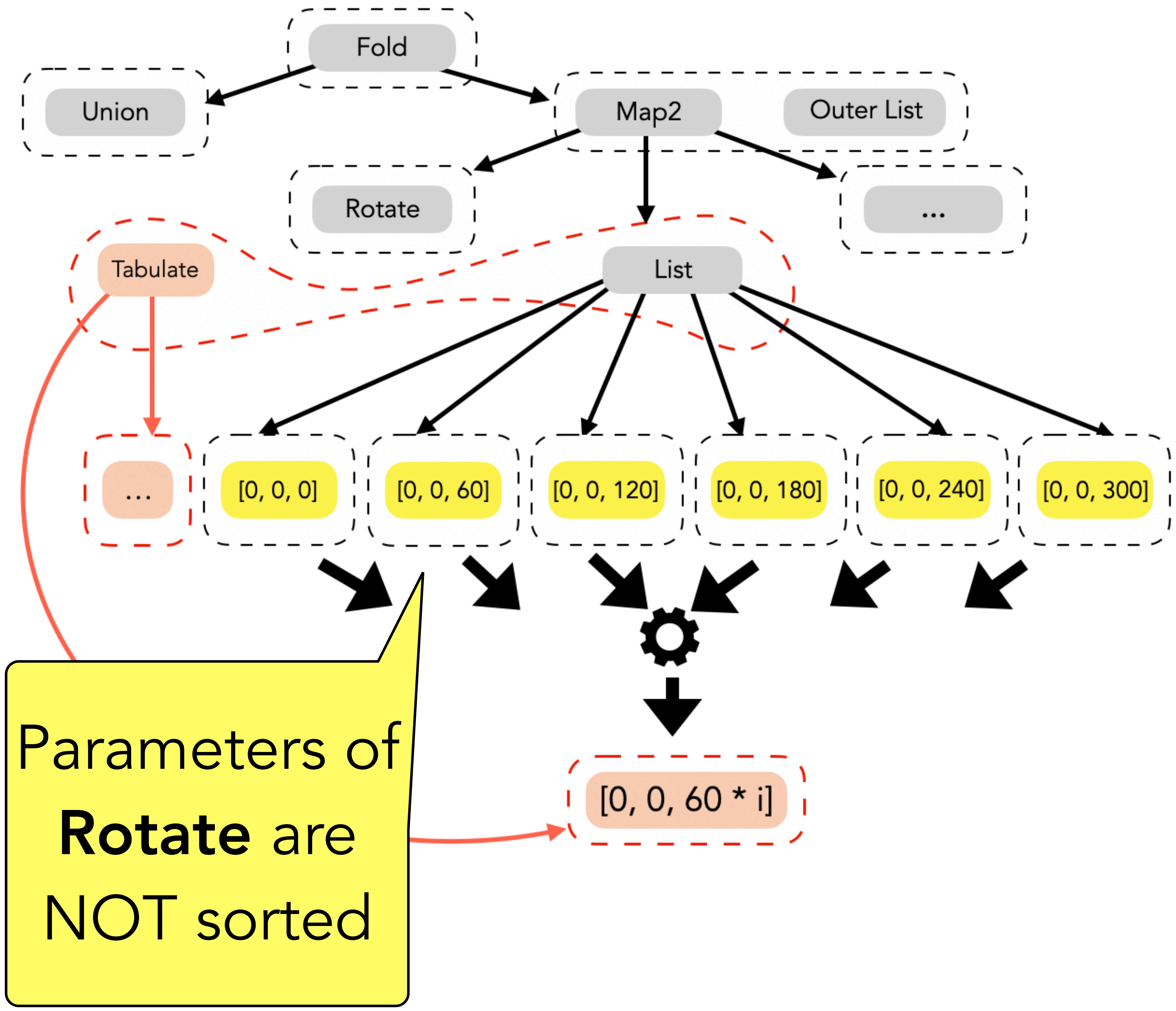
Custom Solvers for Non-Ideal Inputs

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```



Custom Solvers for Non-Ideal Inputs

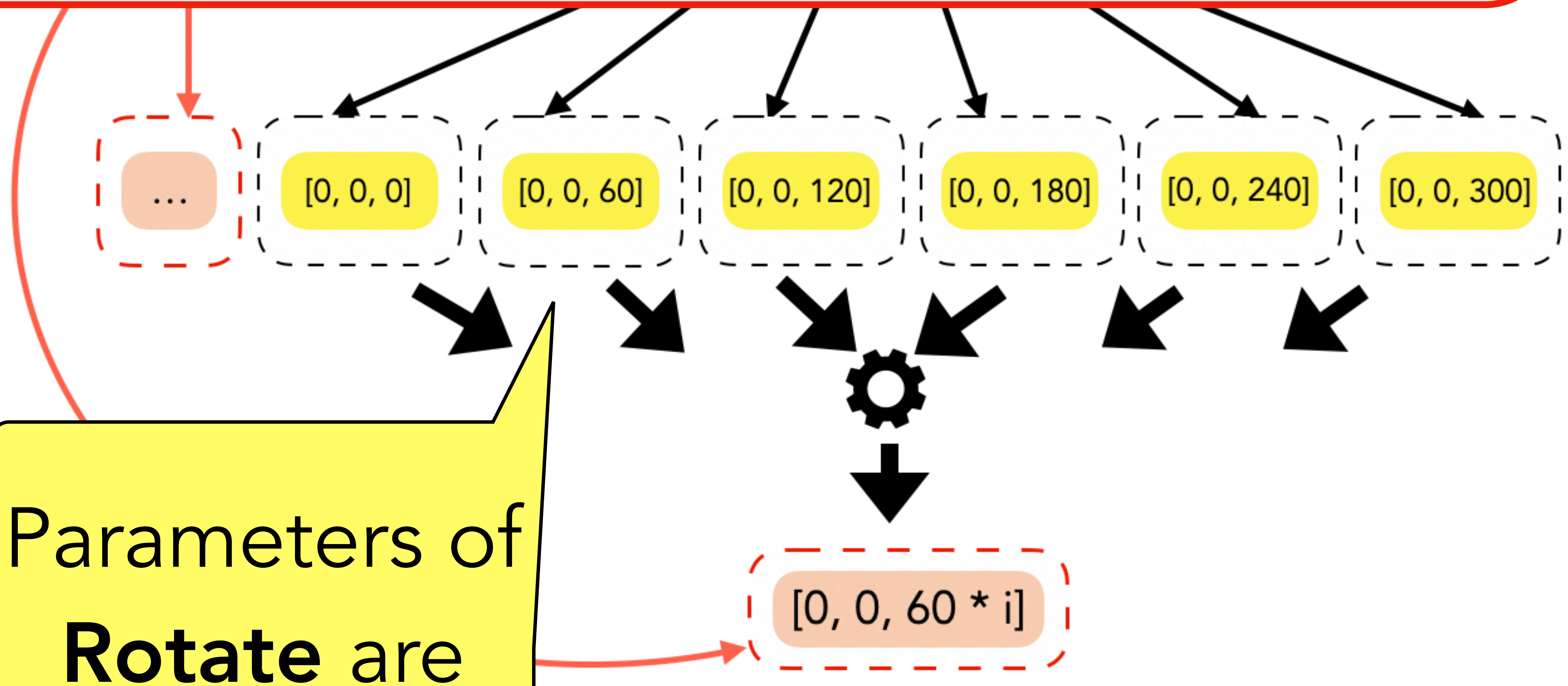
```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```



Custom Solvers for Non-Ideal Inputs

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

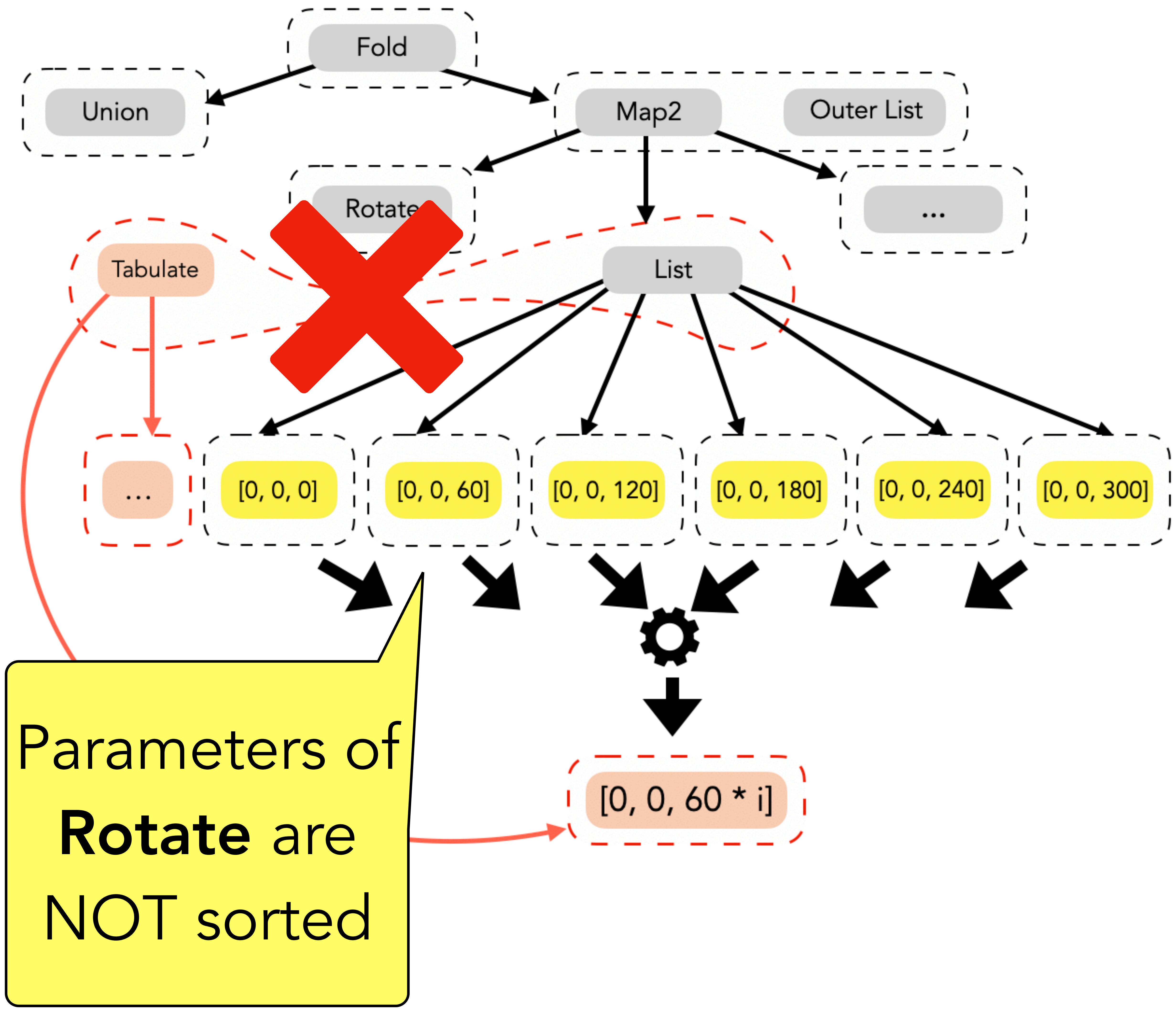
List of vectors must be sorted for the solver to be able to find the closed form and unify the Tabulate with the concrete list



Parameters of **Rotate** are NOT sorted

Custom Solvers for Non-Ideal Inputs

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```



Naive Solution for Finding Closed Form

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))))))
```

Add all permutations of the list elements in the E-graph

Naive Solution Causes the AC-Matching Problem

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Add all permutations of the list elements in the E-graph

Exponentially many choices in an E-graph due to associative-commutative operations like permuting lists, called AC-matching in the SMT community

Inverse Transformations

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))))))
```

Key insight: allows solvers to speculatively transform their inputs to enable more profitable rewriting

Inverse Transformations

```
(Union
 (Scale [5,5,1] (Cylinder [1,1]))
 (Fold Union (List
  (Rotate [0, 0, 120]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 0]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, –0.5, 0] (Cuboid [10, 1, 1]))))))
```

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

→ Structure Finder

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Goal

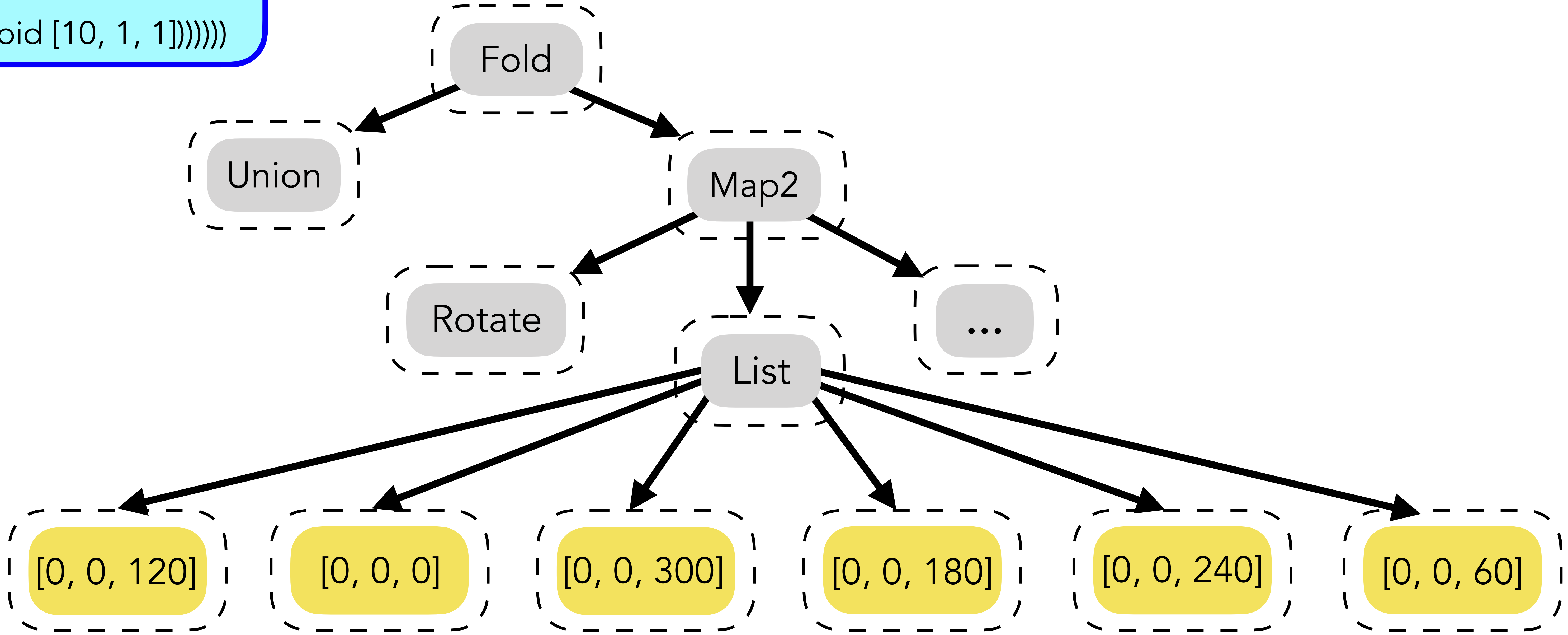
```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Union
 (Scale [5,5,1] (Cylinder [1,1]))
 (Fold Union (List
  (Rotate [0, 0, 120]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 0]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 300]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 180]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 240]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
  (Rotate [0, 0, 60]
   (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Structure Finder

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```



Goal

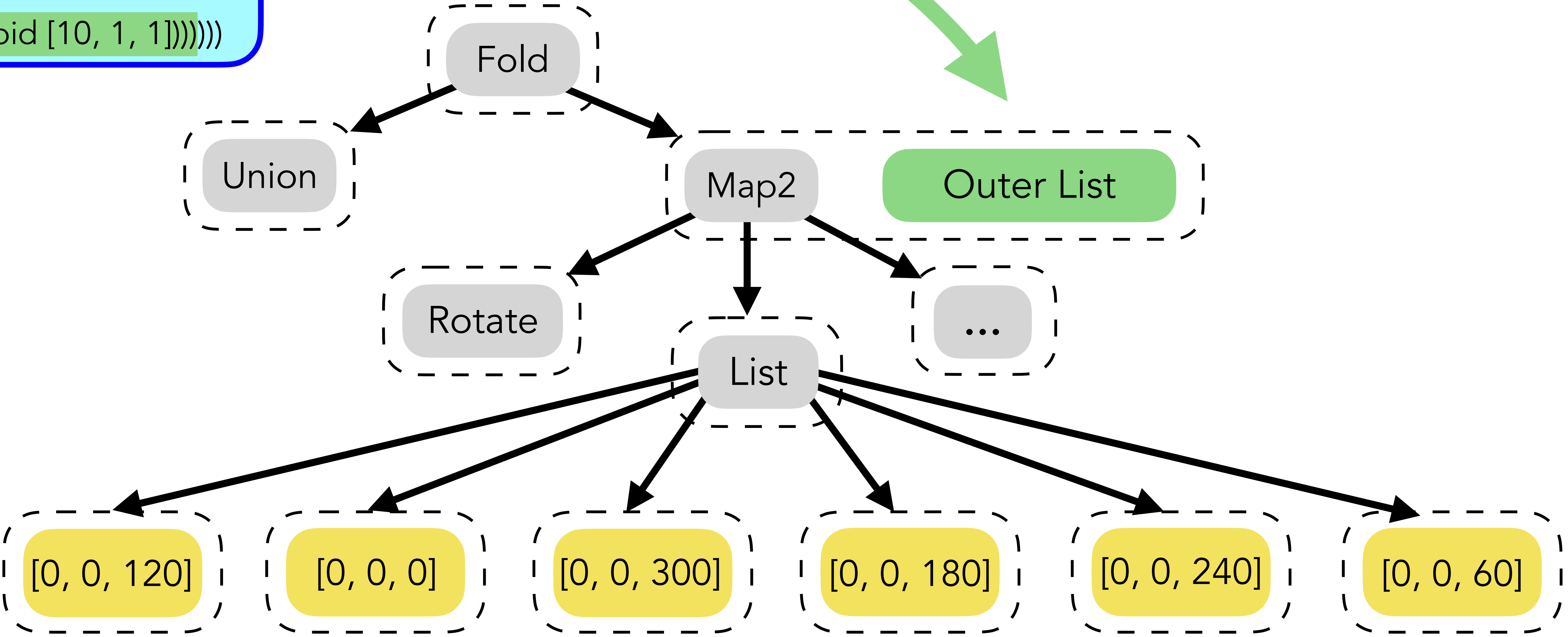
```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Union
  (Scale [5,5,1] (Cylinder [1,1]))
  (Fold Union (List
   (Rotate [0, 0, 120]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 0]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 300]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 180]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 240]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1])))
   (Rotate [0, 0, 60]
    (Translate [1, −0.5, 0] (Cuboid [10, 1, 1]))))))
```

Structure Finder

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

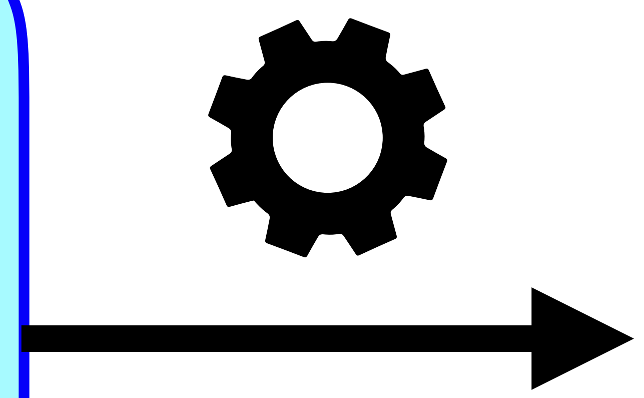


Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

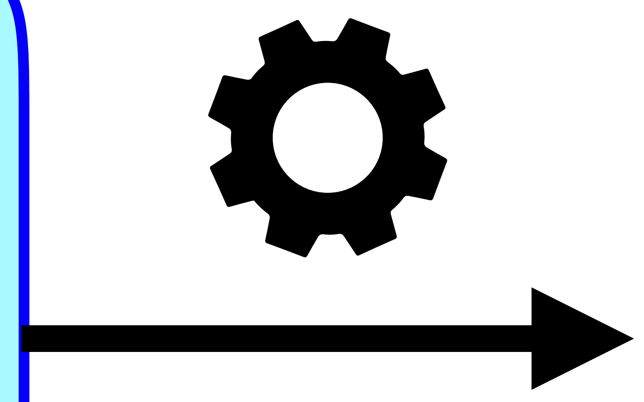


Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```



Solver permutes the list to find closed form!

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
   (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
   (Repeat 6
     (Translate [1, −0.5, 0]
       (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
   (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
   (Repeat 6
     (Translate [1, −0.5, 0]
       (Cuboid [10, 1, 1])))))
```

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
   (Tabulate (i 6)
     (Rotate [0, 0, 60i]
       (Translate [1,−0.5,0]
         (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver annotates the expression with the profitable permutation

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver annotates the expression with the profitable permutation

If a solver cannot simplify A, but it can simplify f(A) to B, then f^{-1}(B) can be unified with A

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
   (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
   (Repeat 6
     (Translate [1, −0.5, 0]
       (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
   (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
   (Repeat 6
     (Translate [1, −0.5, 0]
       (Cuboid [10, 1, 1])))))
```

Solver annotates the expression with the profitable permutation

Flexibly combines solvers with an egraph-driven rewrite system

Solvers allowed to transform their input however they want

BUT they must 'undo' the transformation to restore equivalence

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
   (Tabulate (i 6)
     (Rotate [0, 0, 60i]
       (Translate [1,−0.5,0]
         (Cuboid [10, 1, 1]))))))
```

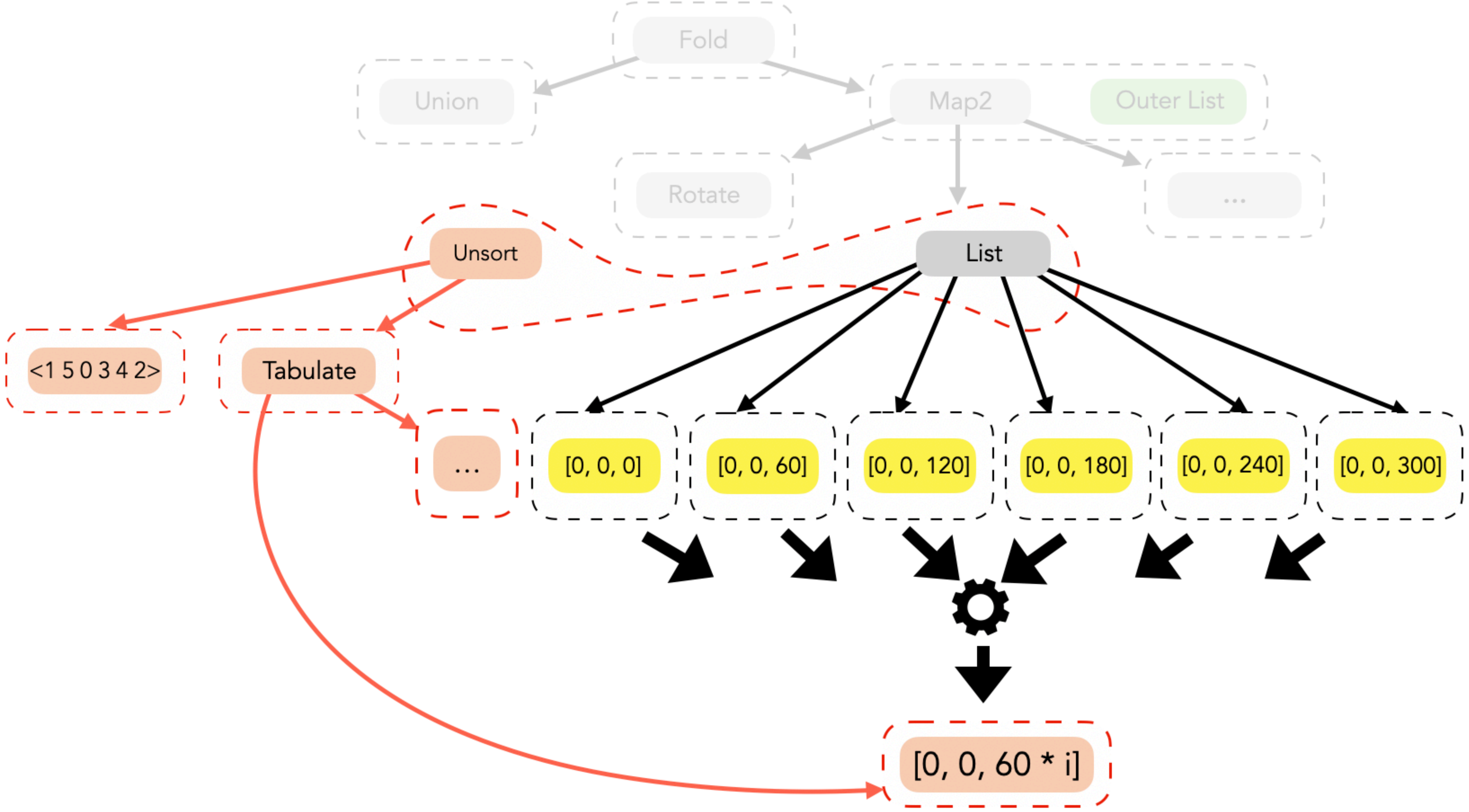
Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver annotates the expression with the profitable permutation



If a solver cannot simplify A, but it can simplify f(A) to B, then f^{-1}(B) can be unified with A

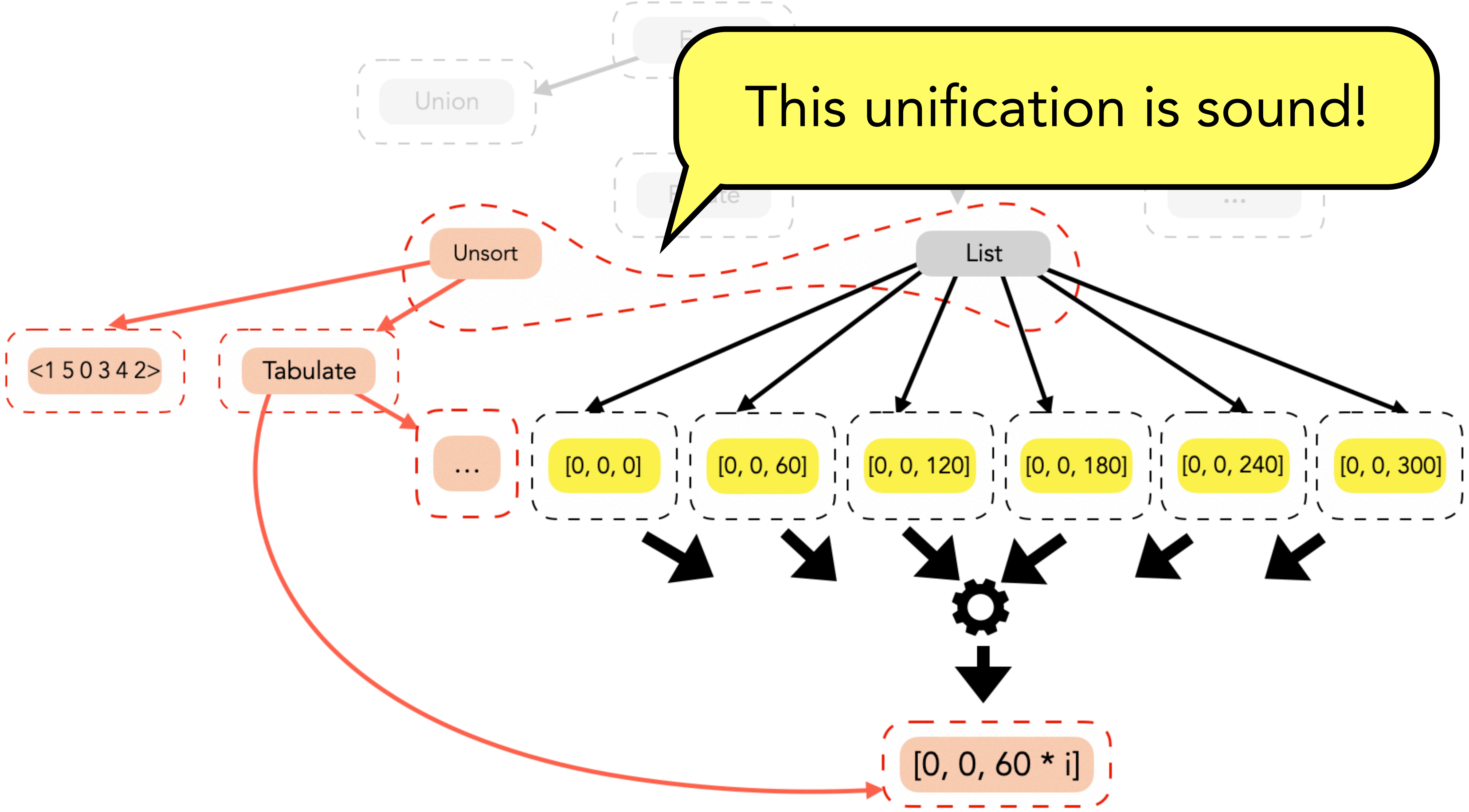
Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (List [0, 0, 120] [0, 0, 0] [0, 0, 300] [0, 0, 180] [0, 0, 240] [0, 0, 60])
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver permutes the list to find closed form!

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Solver annotates the expression with the profitable permutation



If a solver cannot simplify A, but it can simplify f(A) to B, then f^{-1}(B) can be unified with A

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2>  bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

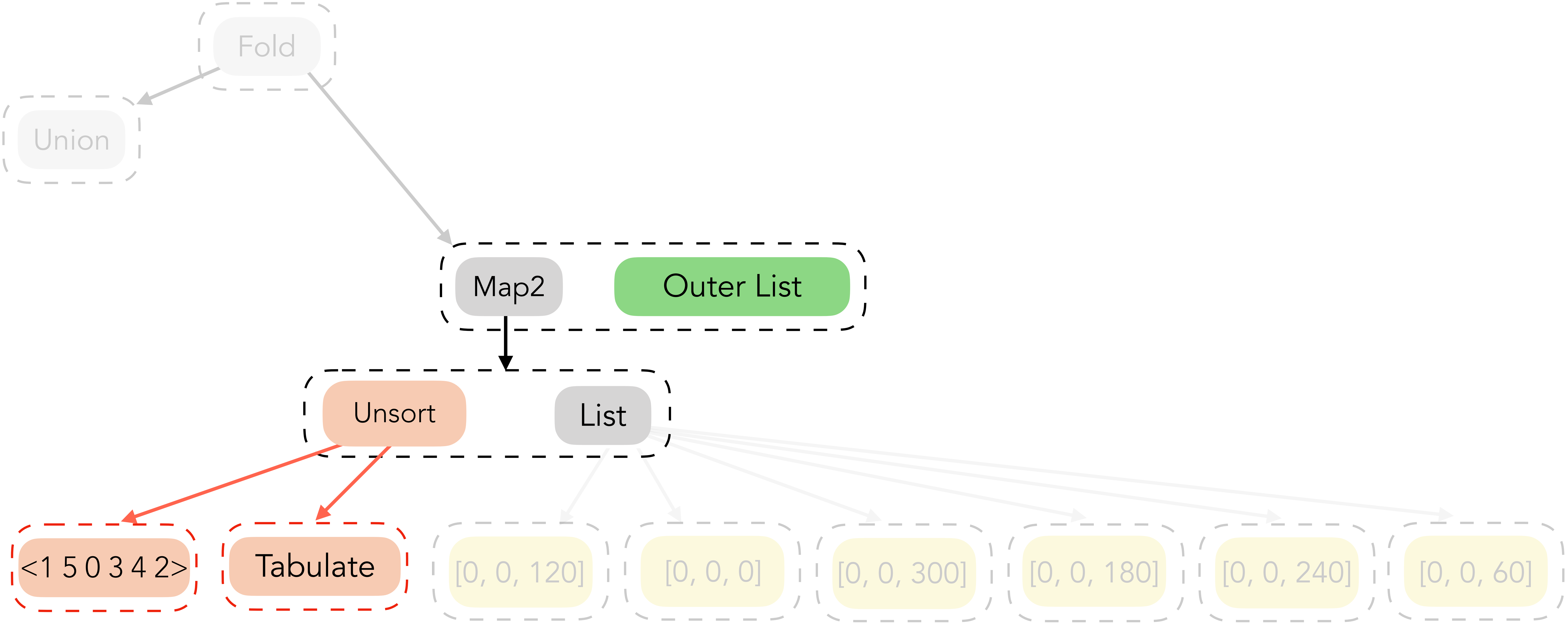
```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

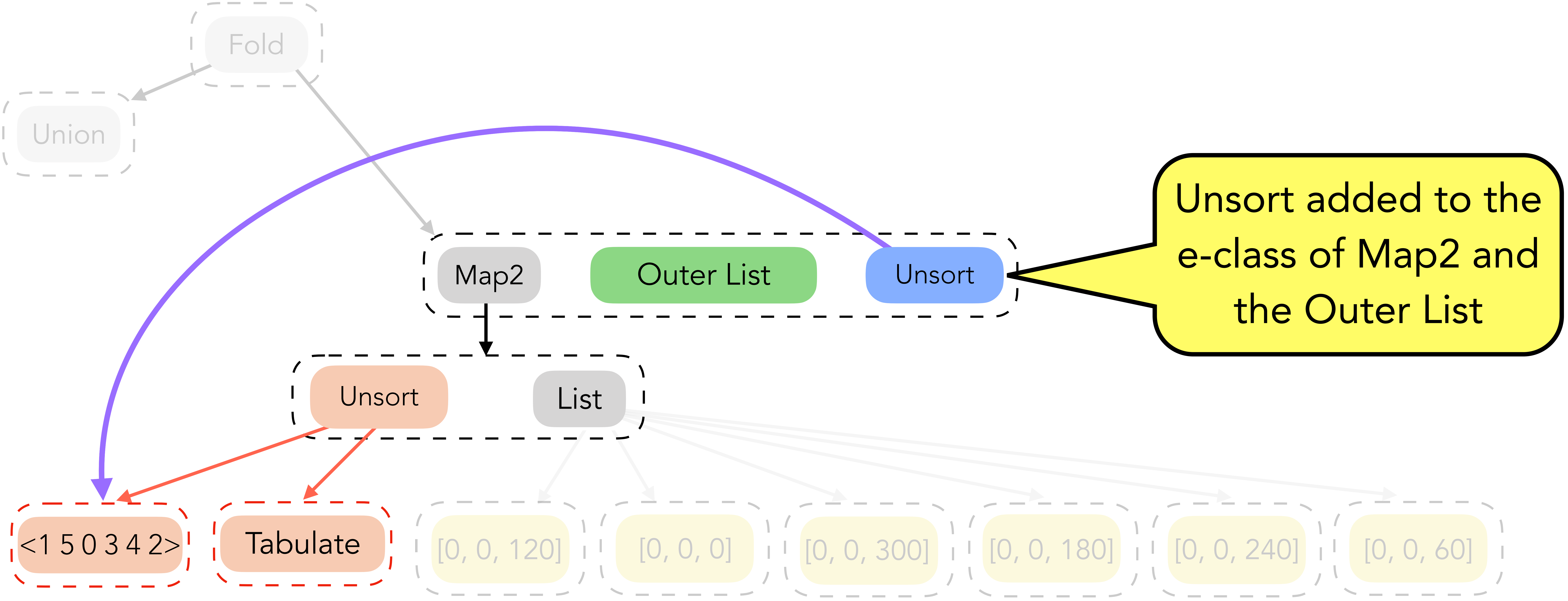
Propagate and Eliminate

→

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Ta… bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

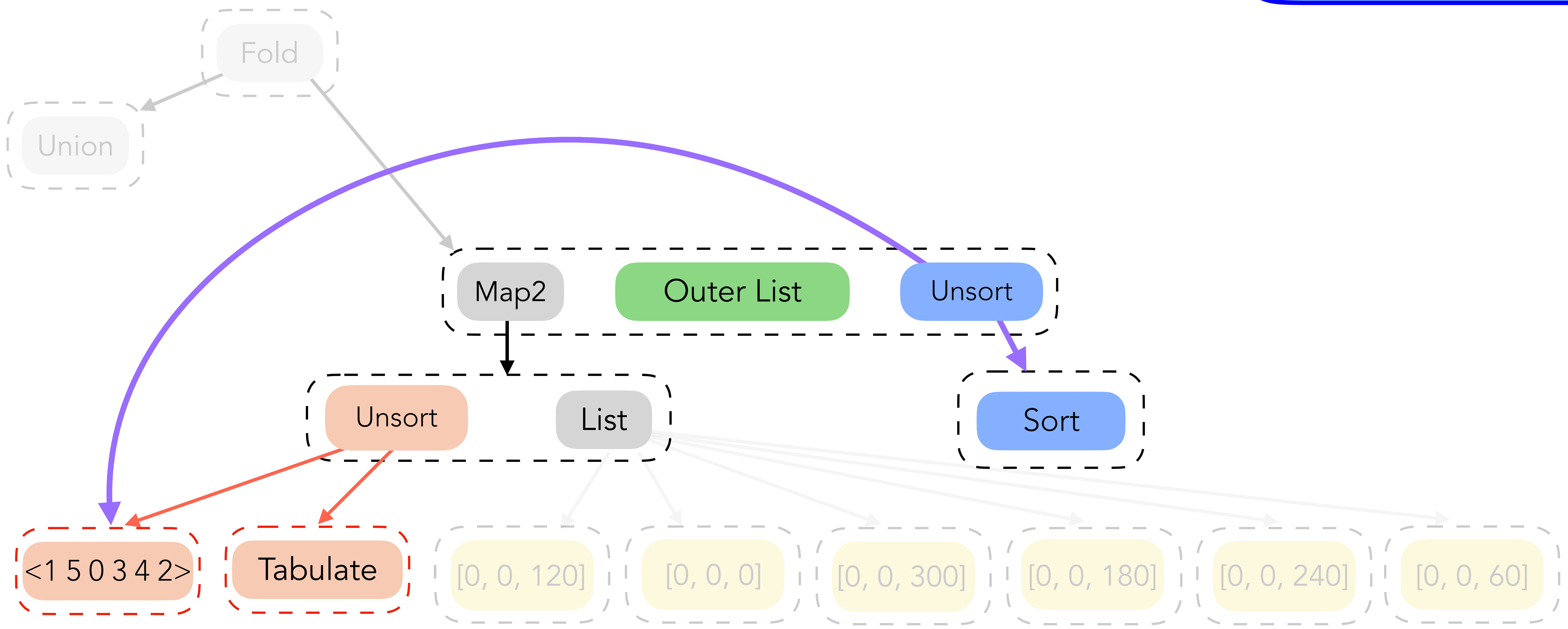
```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

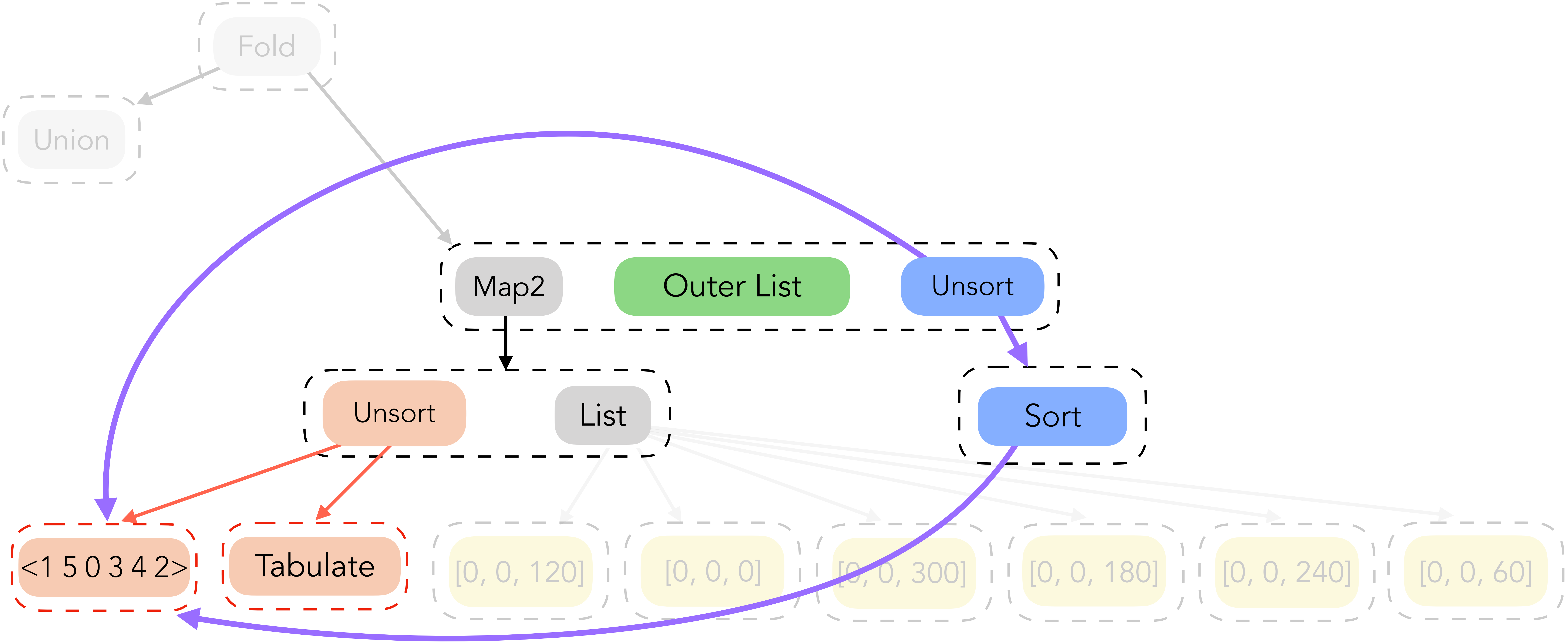
```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

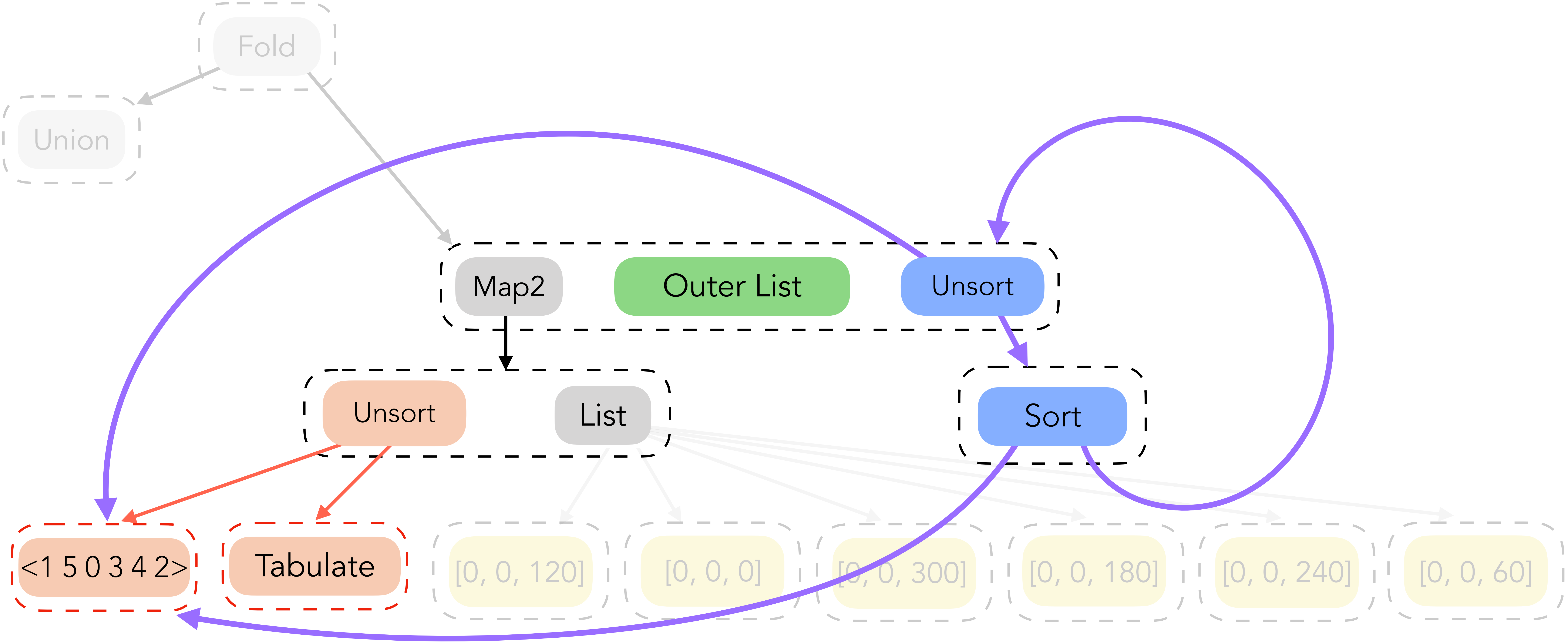
```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))
```

Propagate and Eliminate

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

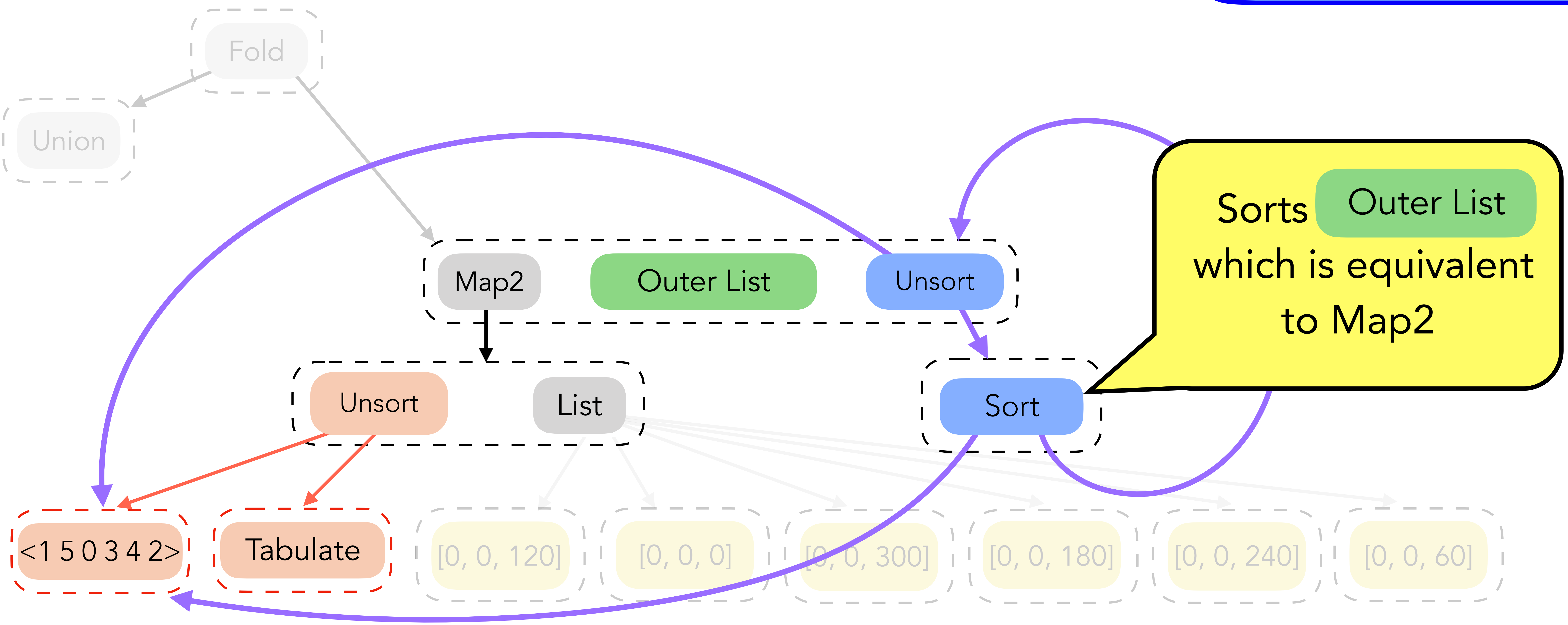
(Fold Union
 (Map2 Rotate
  (**Unsort <1 5 0 3 4 2>** (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))

Propagate and Eliminate

Syntactic rewrites

(Fold Union
 (**Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>**
 (Map2 Rotate
  (**Unsort <1 5 0 3 4 2>** (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))))

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Goal

(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))

Inverse Transformations

```
(Fold Union
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))
```

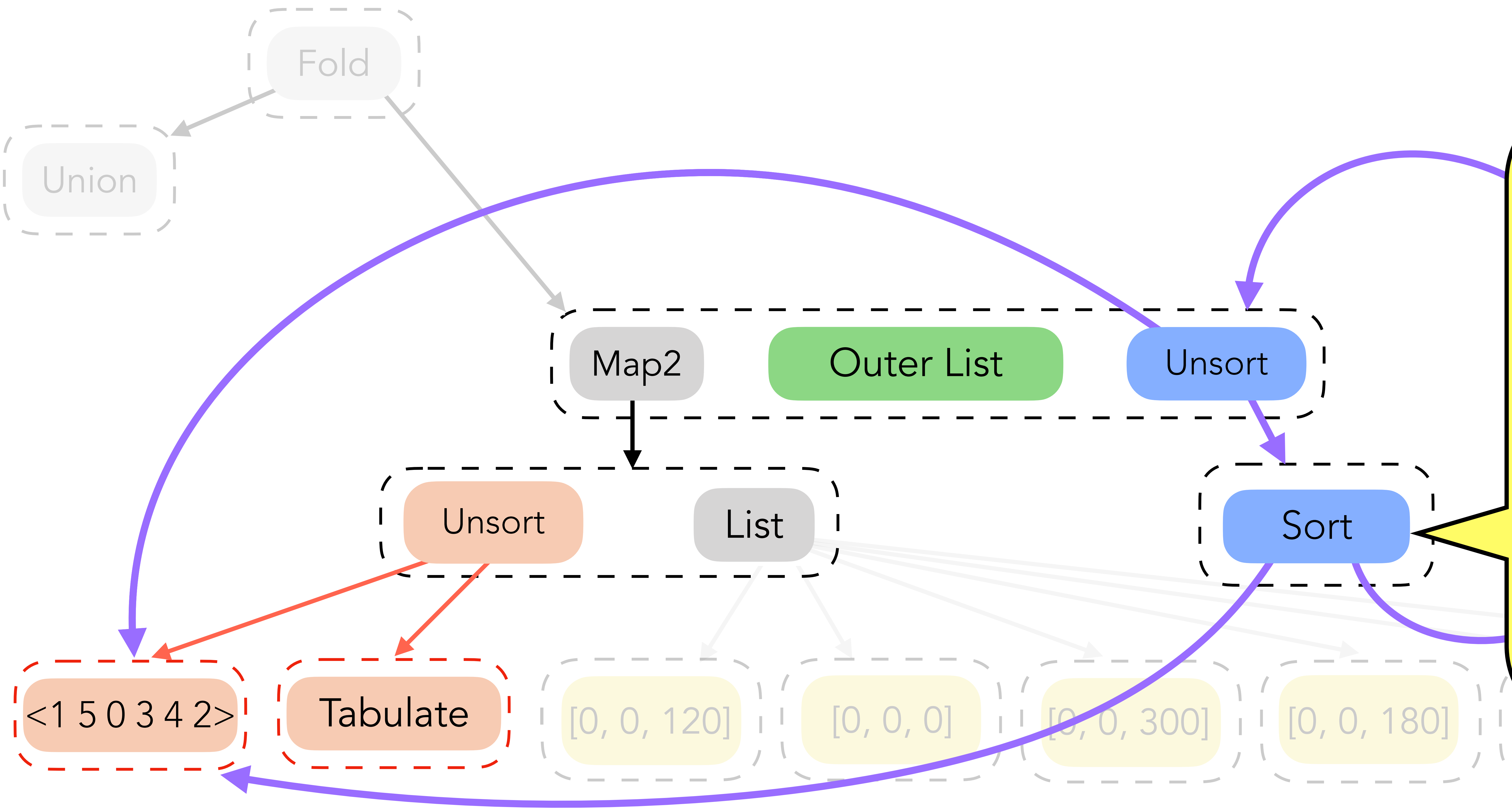
Propagate and Eliminate

→

Syntactic rewrites

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Ta bulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, –0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Effectively a no-op, but allows sorting the concrete list equivalent to **Map2**



Structure Finder and Custom Solvers apply on this sorted list

Goal

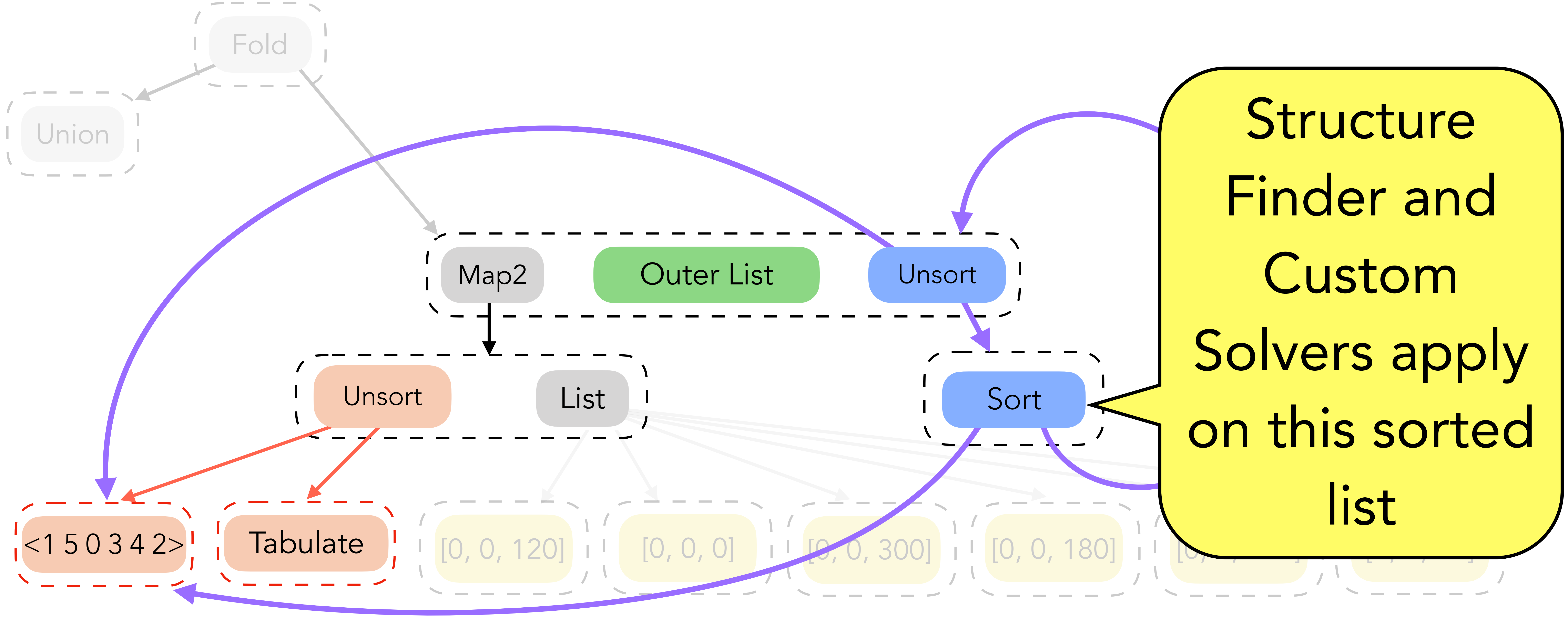
```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,–0.5,0]
     (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
  (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
  (Map2 Rotate
    (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
    (Repeat 6
      (Translate [1, −0.5, 0]
        (Cuboid [10, 1, 1])))))))
```

Custom solvers on the sorted outer list

```
(Fold Union
  (Unsort <1 5 0 3 4 2>
    (Tabulate (i 6)
      (Rotate [0, 0, 60 * i])
        (Translate [1, −0.5, 0]
          (Cuboid [10, 1, 1]))))))
```



Goal

```
(Union
  (Cylinder [1, 5, 5])
  (Fold Union
    (Tabulate (i 6)
      (Rotate [0, 0, 60i]
        (Translate [1,–0.5,0]
          (Cuboid [10, 1, 1]))))))
```

Inverse Transformations

```
(Fold Union
 (Unsort <1 5 0 3 4 2> (Sort <1 5 0 3 4 2>
 (Map2 Rotate
  (Unsort <1 5 0 3 4 2> (Tabulate (i 6) [0, 0, 60 * i]))
  (Repeat 6
   (Translate [1, −0.5, 0]
    (Cuboid [10, 1, 1])))))))
```

Custom solvers on the sorted outer list

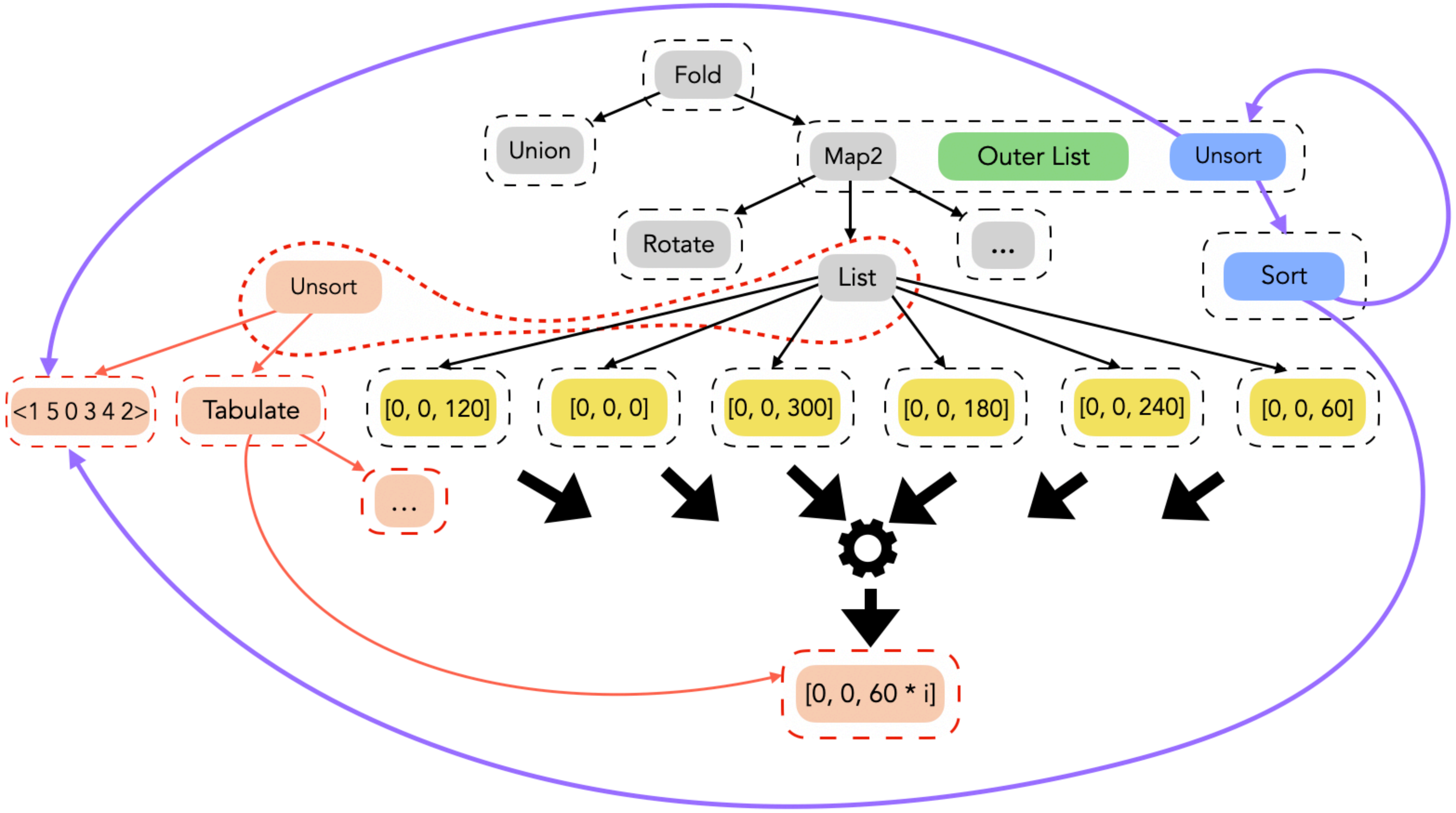
```
(Fold Union
 (Unsort <1 5 0 3 4 2>
  (Tabulate (i 6)
   (Rotate [0, 0, 60 * i])
    (Translate [1, −0.5, 0]
     (Cuboid [10, 1, 1]))))))
```

Fold Union is invariant to list order

Syntactic rewrite to eliminate Unsort

Goal

```
(Union
 (Cylinder [1, 5, 5])
 (Fold Union
  (Tabulate (i 6)
   (Rotate [0, 0, 60i]
    (Translate [1,−0.5,0]
     (Cuboid [10, 1, 1]))))))
```



Inverse Transformations

Example transformations: sorting, partitioning, cartesian-to-spherical

Rewrites applied until saturation (or timeout) and a cost function (AST size) used to extract best program

Implementation

~ 2000 LOC in Rust

65 rewrites

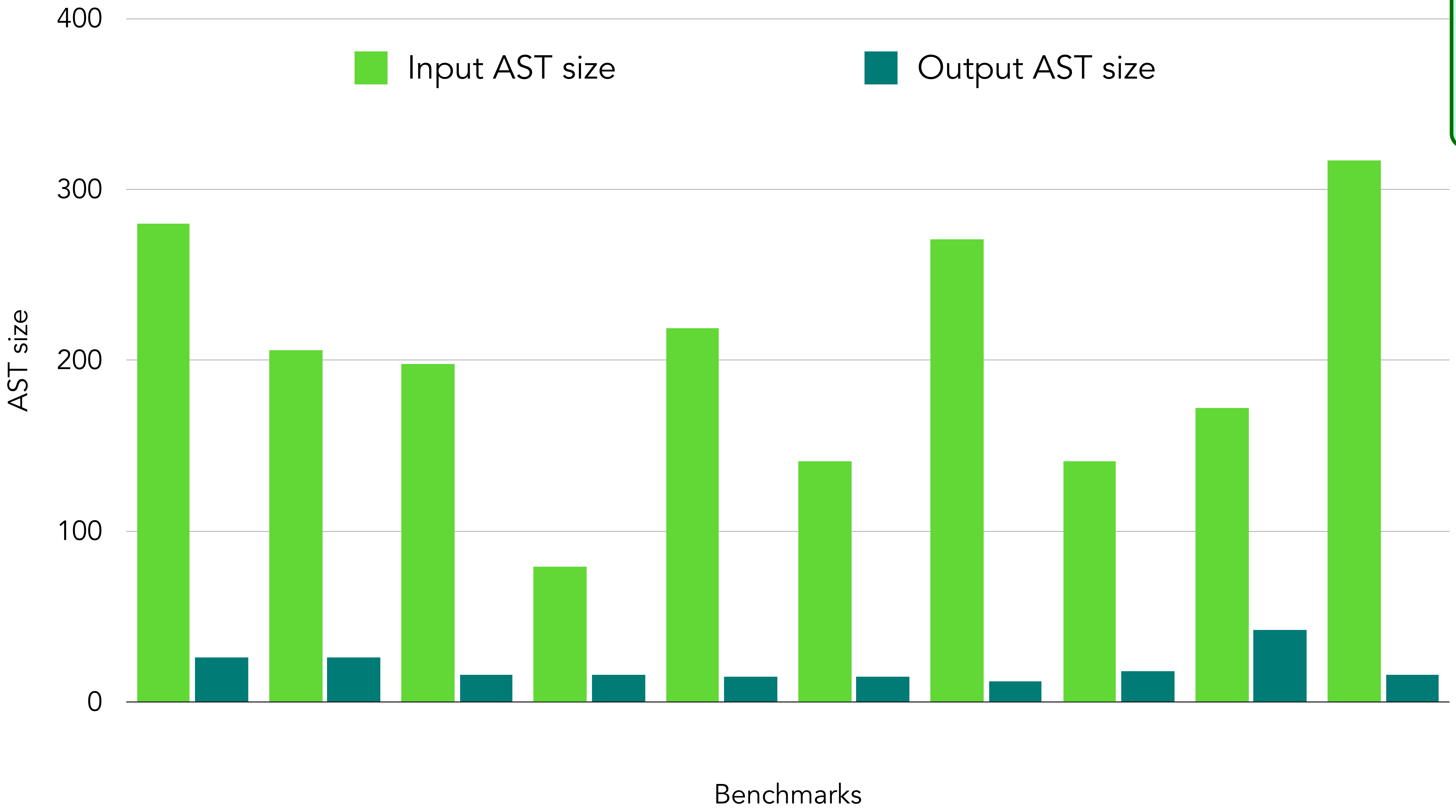
https://github.com/uwplse/szalinski

Uses the Egg E-graph library: https://github.com/mwillsey/egg

Talk to Max about Egg!

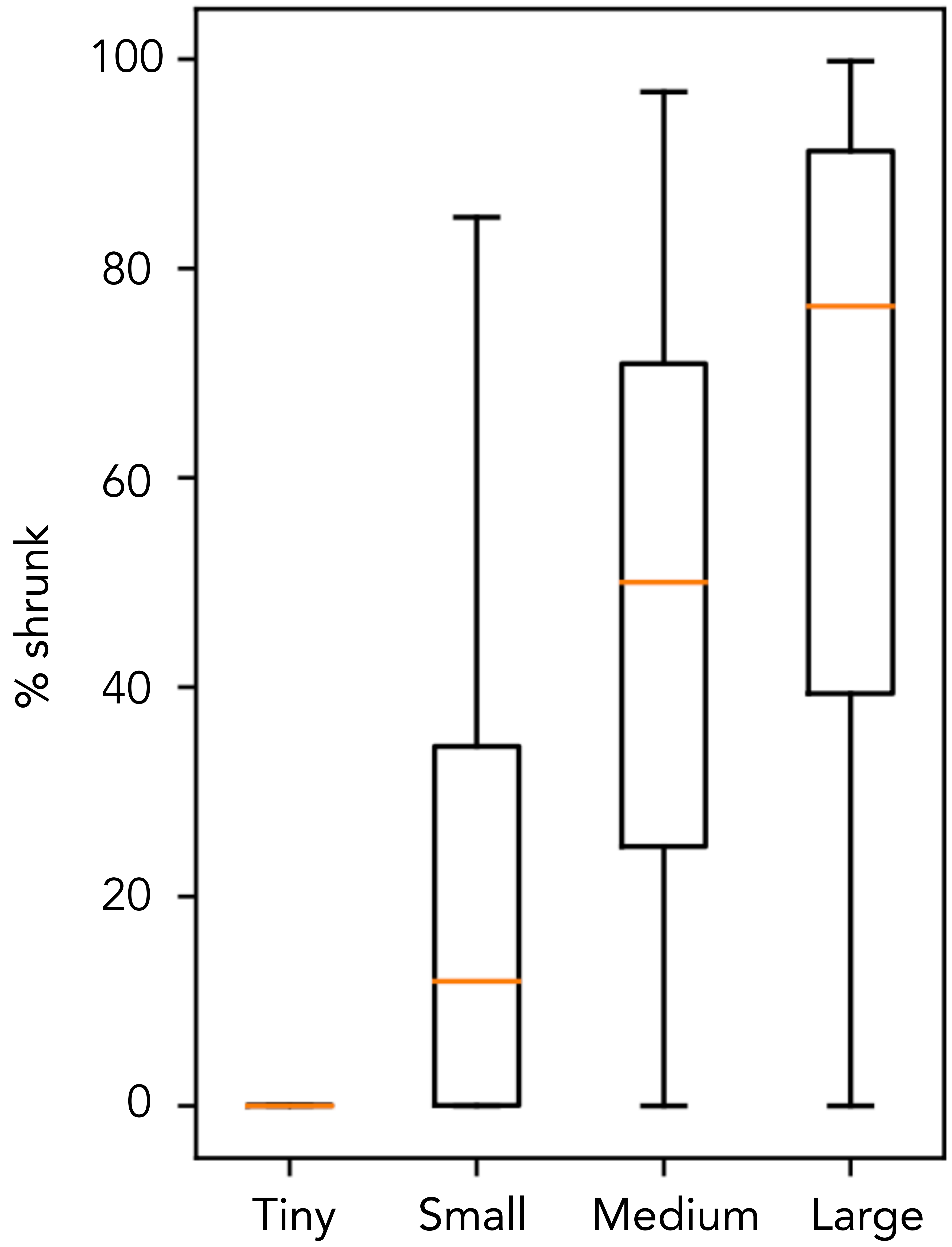
End-to-End Evaluation

Results of running Szalinski on outputs of Reincarnate*



* [ICFP 2018]

Scalability



2127 programs from Thingiverse

Tiny: AST size < 30

Small: 30 < AST size < 100

Medium: 100 < AST size < 300

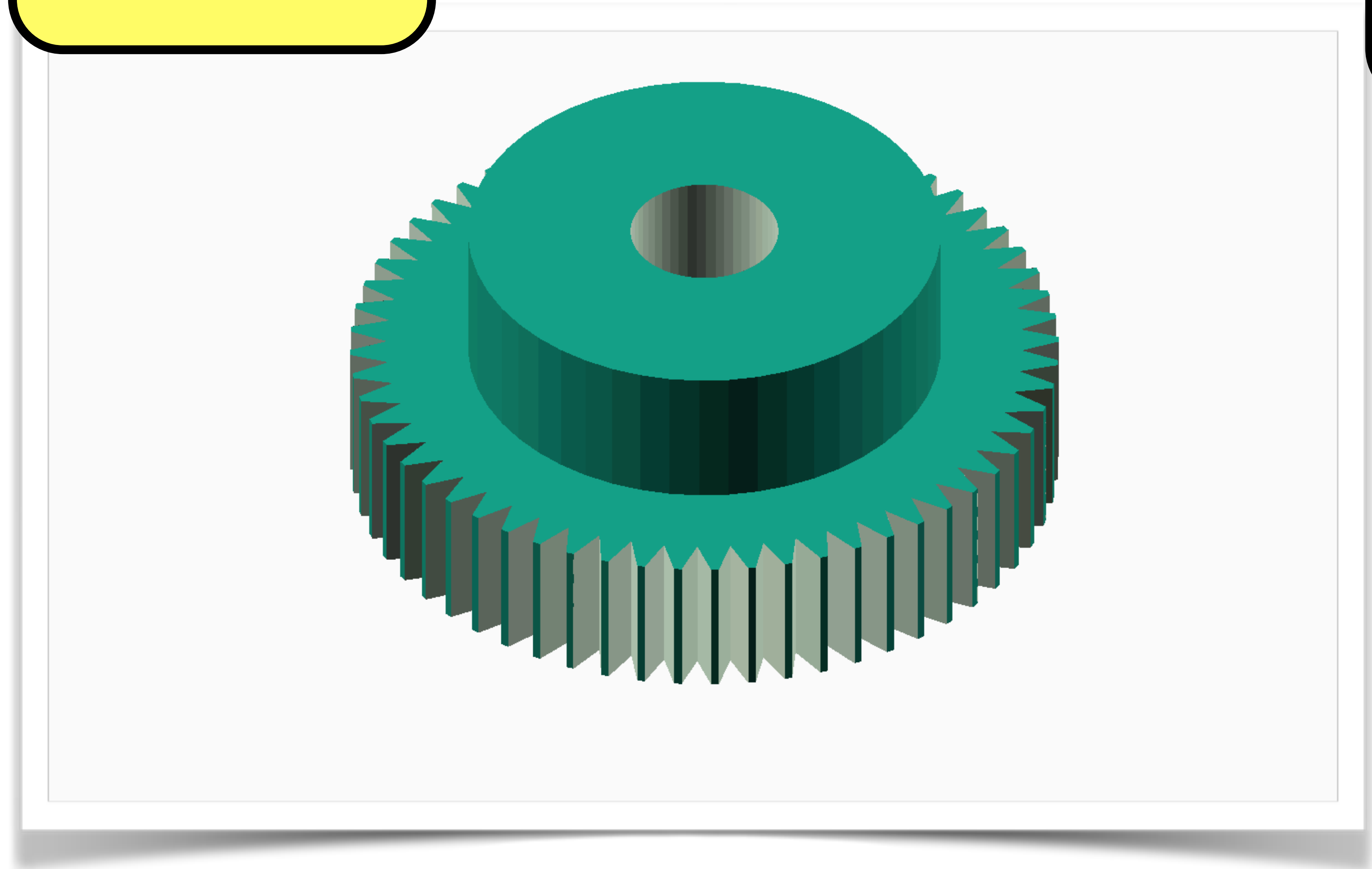
Large: AST size > 300

Larger programs shrink more

< 1 second

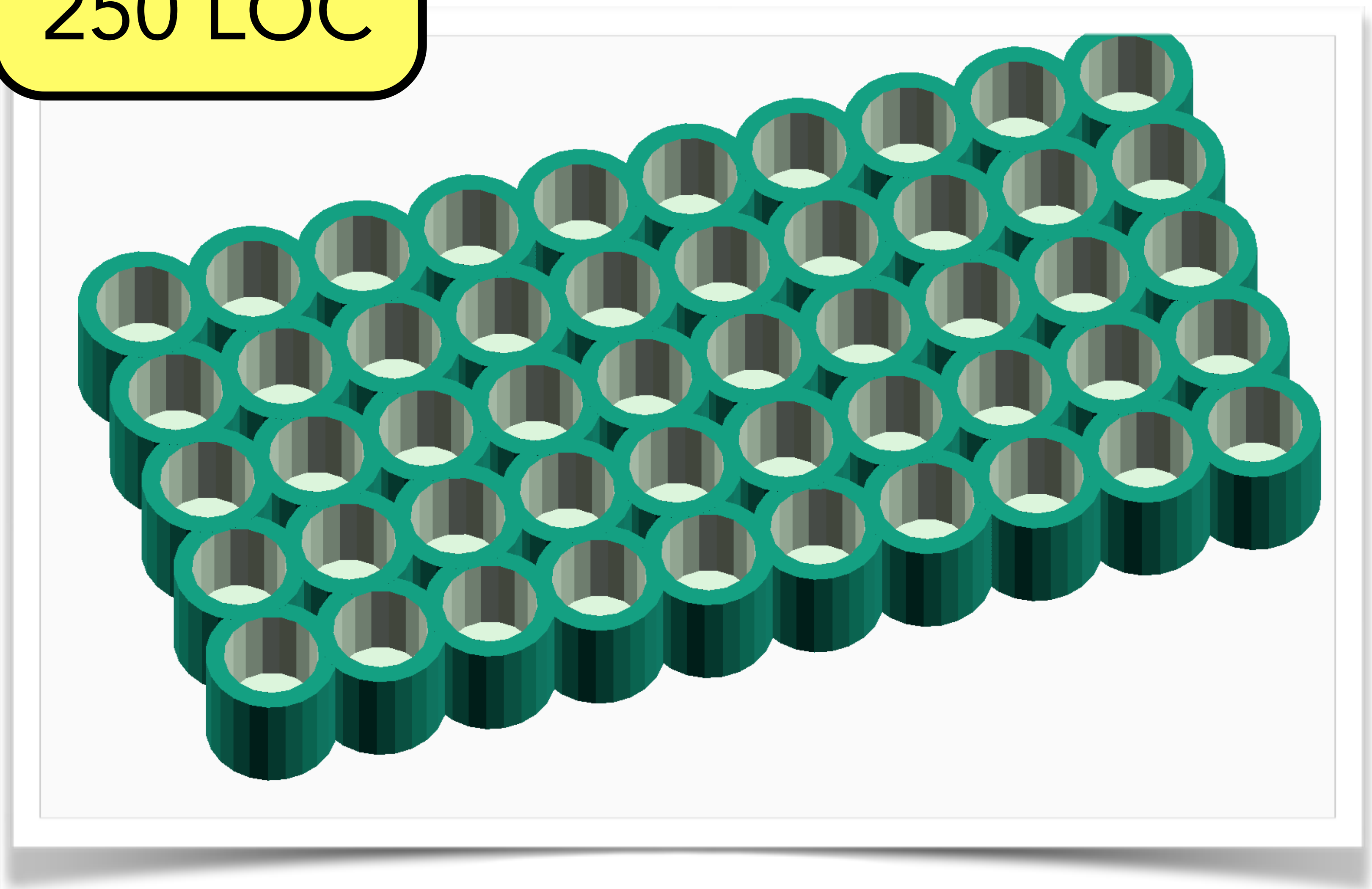
Examples

350 LOC



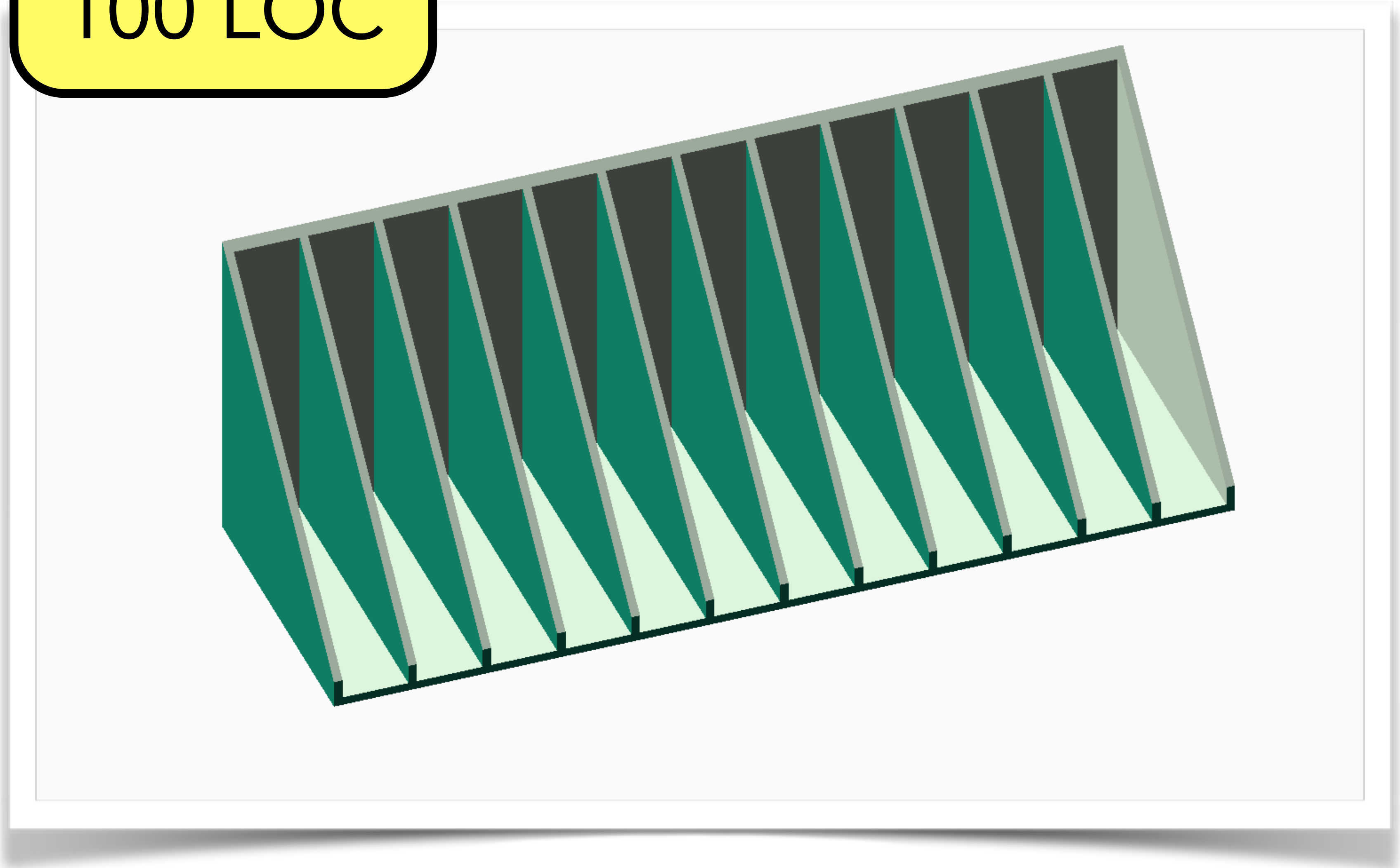
```
(Fold Difference
 (List (Union
    (Cylinder [100, 80, 80])
    (Cylinder [50, 120, 120]))
  (Translate [0, 0, -1] (Cylinder [102, 25, 25]))
  (Fold Union (Tabulate (i 60)
    (Rotate [0, 0, 6 * i]
     (Translate [125, 0, 0]
      (Scale [2.5, 1, 1]
       (Rotate [0, 0, 45]
        (Translate [0, 0, 25]
          (Cuboid [10, 10, 52]))))))))))
```

250 LOC



```
(Fold Union
 (Tabulate (i 10) (j 5)
  (Translate
    [12.2 * i + 12.2, 12.2 * j + 12.2, 0]
   (Difference
    (Cylinder [13, 7.1, 7.1])
    (Translate [0, 0, 3]
      (Cylinder [11, 5.1, 5.1]))))))
```

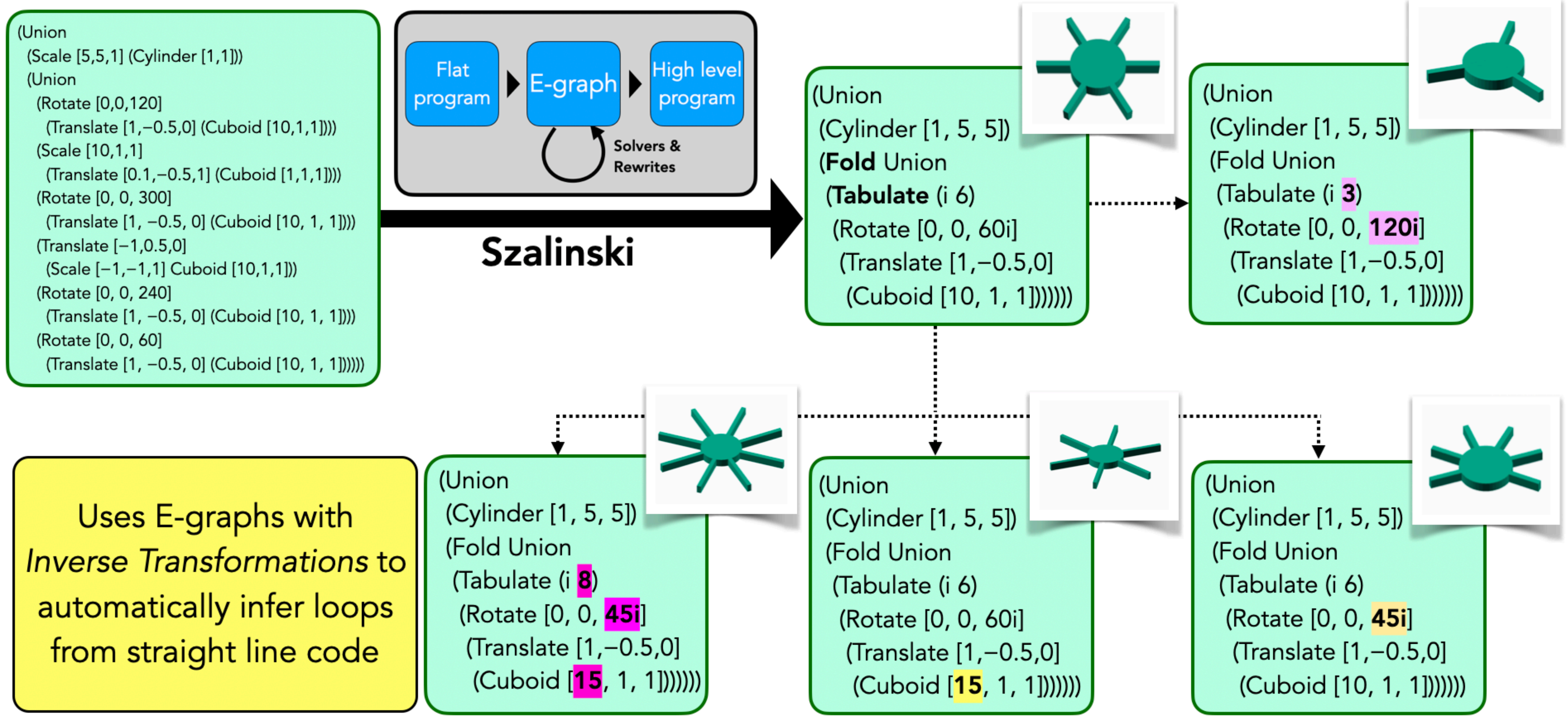
100 LOC



```
(Fold Union
 (Tabulate (i 12)
  (Translate [0, 13* i, 0]
   (Fold Difference
    (List
     (Cuboid [53.1 14.5 58])
     (Translate [1.5, 1.5, 1.5]
       (Cuboid [51.6, 11.5, 56.6]))
     (Translate [0 0 58]
      (Rotate [0, 45, 0]
       (Cuboid [101.5, 14.5, 100]))))))))
```

Szalinski: A Tool for Synthesizing Structured CAD Models with Equality Saturation and Inverse Transformations

https://github.com/uwplse/szalinski



Inverse Transformations with E-graphs to find concise, structured programs in < 1 second

Chandrakana Nandi, Max Willsey, Adam Anderson, James R. Wilcox, Eva Darulova, Dan Grossman, Zachary Tatlock