learning better abstractions
with e-graphs and anti-unification

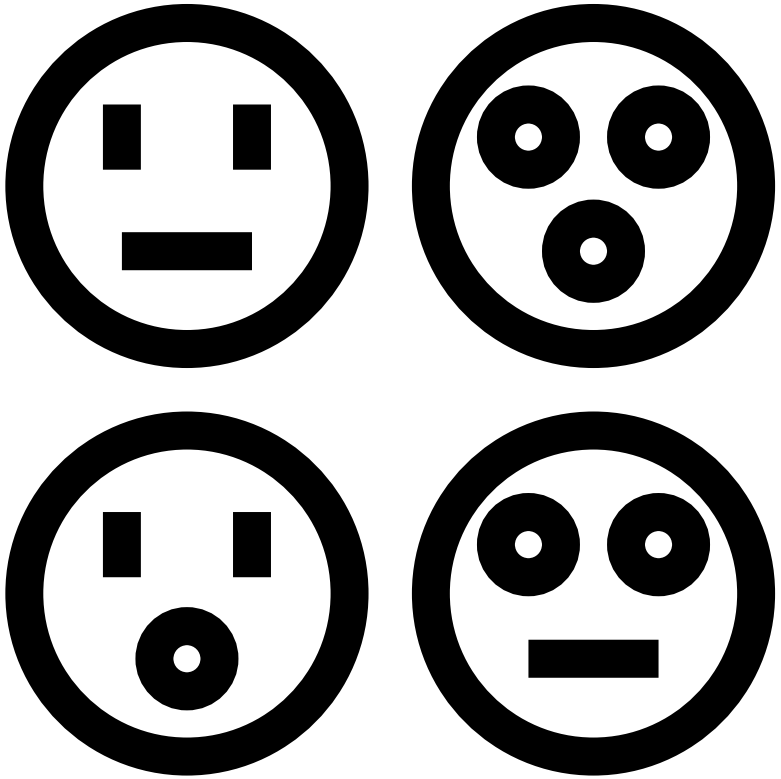
Nadia Polikarpova

WG2.8 2022

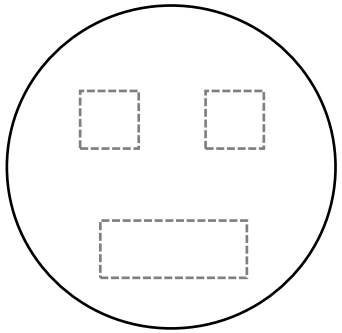
join work with Rose Kunkel, David Cao, Chandrakana Nandi, Max Willsey, Zach Tatlock

UC San Diego
COMPUTER SCIENCE & ENGINEERING

what do you see?

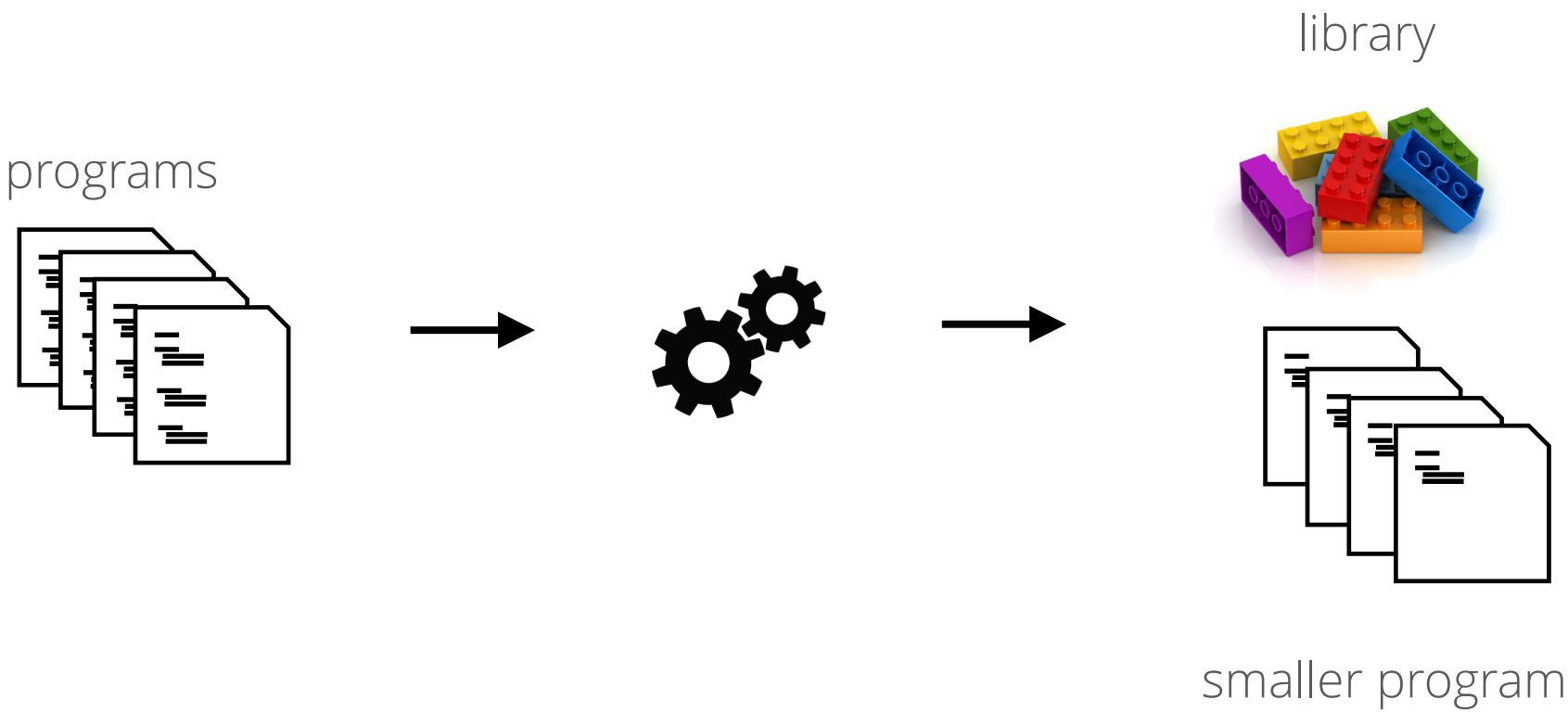


humans are good at abstraction

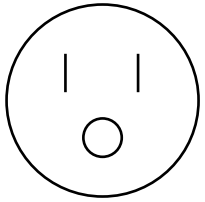


can machines do it too?

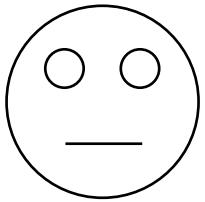
library learning



library learning



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```



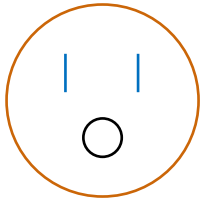
```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

library learning

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

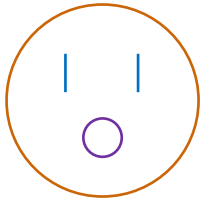
library learning



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

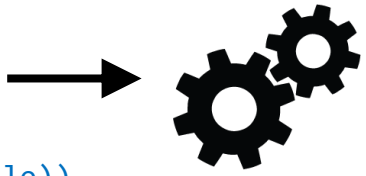
```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

library learning



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```



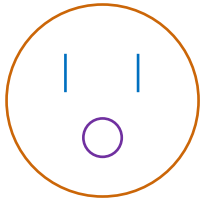
```
f = \eShape eScale mShape mScale ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale eScale eShape)),
     move  2 -1.5 (rotate 90 (scale eScale eShape)),
     move 0 2 (scale mScale mShape)
    ]
```

```
f line   2 circle 1

f circle 1 line   4
```

size: 50

size: 37

library learning



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

babble

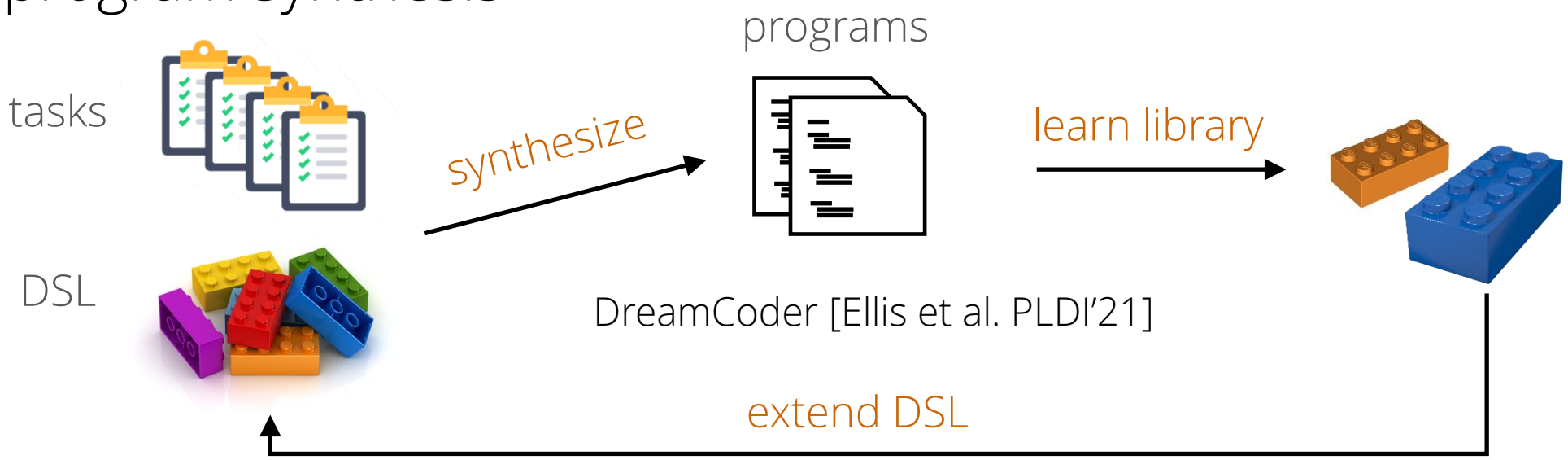
```
f = \eShape eScale mShape mScale ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale eScale eShape)),
     move  2 -1.5 (rotate 90 (scale eScale eShape)),
     move 0 2 (scale mScale mShape)
    ]

f line   2 circle 1

f circle 1 line   4
```

size: 50

size: 37

why do we care?

- modeling human visual perception
- compression
- automatic refactoring
- program synthesis



1. challenges
2. flexibility via *e*-graphs
3. scalability via anti-unification

1. challenges
2. flexibility via *e*-graphs
3. scalability via anti-unification

DreamCoder: beta inversion

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

DreamCoder: beta inversion

step 1: pick an arbitrary subterm

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

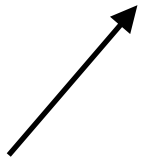
```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

DreamCoder: beta inversion

step 2: poke arbitrary holes

```
(rotate 90 (scale ?x line))
```



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

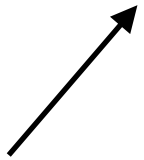
```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

DreamCoder: beta inversion

step 3: pattern-match

```
(rotate 90 (scale ?x line))
```



```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

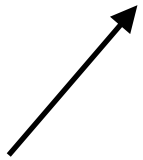
DreamCoder: beta inversion

step 3: rewrite program

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50



```
f = \x -> (rotate 90 (scale x line))
```

```
[scale 5 circle,
 move -2 -1.5 (f 2),
 move  2 -1.5 (f 2),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 51

DreamCoder: beta inversion

repeat

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

```
f = \x -> (rotate 90 (scale x line))
```

size: 51

```
f = \x y -> (rotate 90 (scale x y))
```

size: 45

...

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

size: 37

...

DreamCoder: limitations

repeat

```
[scale 5 circle,
 move -2 -1.
 move  2 -1.
 move 0 2 (s
[scale 5 cir
 move -2 -1.5 (rotate 90 (scale
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

size: 50

1. inefficient / incomplete

2. requires syntactic alignment

```
f = \x -> (rotate 90 (scale x line))
```

size: 51

```
(scale x y))
```

size: 45

...

```
[scale
 move -2 -1.5 (rotate 90 (scale y x)),
 move  2 -1.5 (rotate 90 (scale y x)),
 move 0 2 (scale b a)
]
```

size: 37

...

syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

DreamCoder

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

DreamCoder

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 circle]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 circle),
 move  2 -1.5 (rotate 90 circle),
 move 0 2 (scale 4 line)]
```

syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

DreamCoder

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 circle]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 circle),
 move  2 -1.5 (rotate 90 circle),
 move 0 2 (scale 4 line)]
```

syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

DreamCoder →

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 circle]
[scale 5 circle,
 move -2 -1.5 circle,
 move  2 -1.5 circle,
 move 0 2 (scale 4 line)]
```

syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

DreamCoder →

```
f = \x y a b ->
    [scale 5 circle,
     move -2 -1.5 (rotate 90 (scale y x)),
     move  2 -1.5 (rotate 90 (scale y x)),
     move 0 2 (scale b a)
    ]
```

babble →

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 circle]
```

```
[scale 5 circle,
 move -2 -1.5 circle,
 move  2 -1.5 circle,
 move 0 2 (scale 4 line)]
```

DreamCoder →

```
f = \x y ->
    [scale 5 circle,
     move -2 -1.5 x,
     move  2 -1.5 x,
     move 0 2 y
    ]
```

```
f (rotate 90 (scale 2 line)) circle

f circle                     (scale 4 line)
```

beyond syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 (scale 1 circle)]
```

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 1 circle)),
 move  2 -1.5 (rotate 90 (scale 1 circle)),
 move 0 2 (scale 4 line)]
```

idea: rewrite into this form
to expose syntactic alignment

```
[scale 5 circle,
 move -2 -1.5 (rotate 90 (scale 2 line)),
 move  2 -1.5 (rotate 90 (scale 2 line)),
 move 0 2 circle]
```

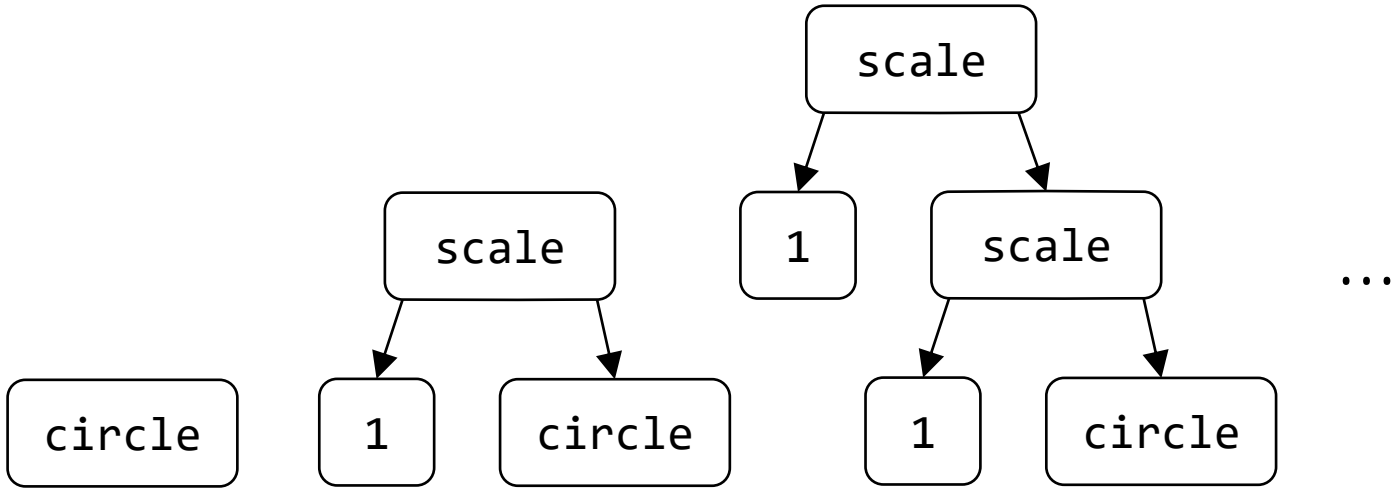
```
[scale 5 circle,
 move -2 -1.5 circle,
 move  2 -1.5 circle,
 move 0 2 (scale 4 line)]
```

challenge: find an equivalent program with the **best** syntactic alignment

1. challenges
2. **flexibility via *e*-graphs**
3. scalability via anti-unification

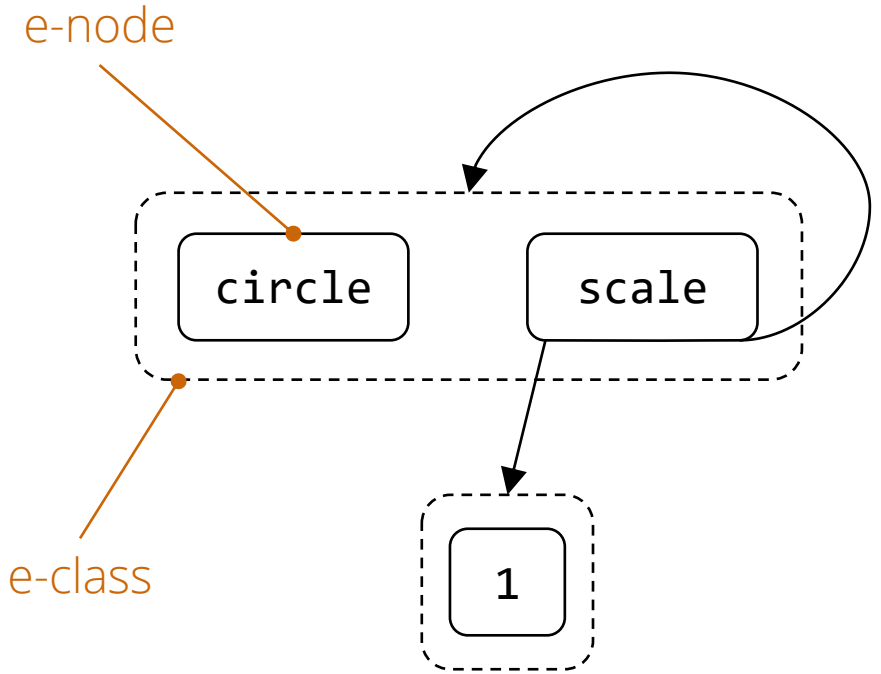
e-graphs

compactly represent
sets of equivalent terms



e-graphs

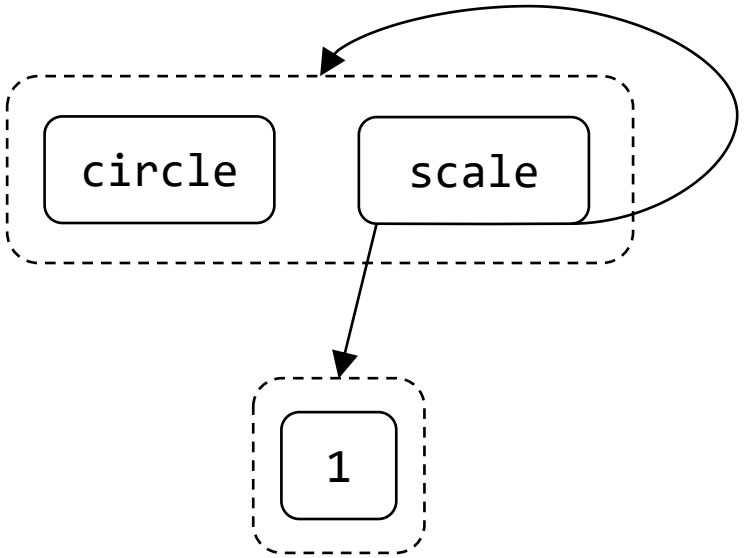
compactly represent
sets of equivalent terms



equality saturation

[Tate et al. POPL'09]

egg [Willsey et al. POPL'21]



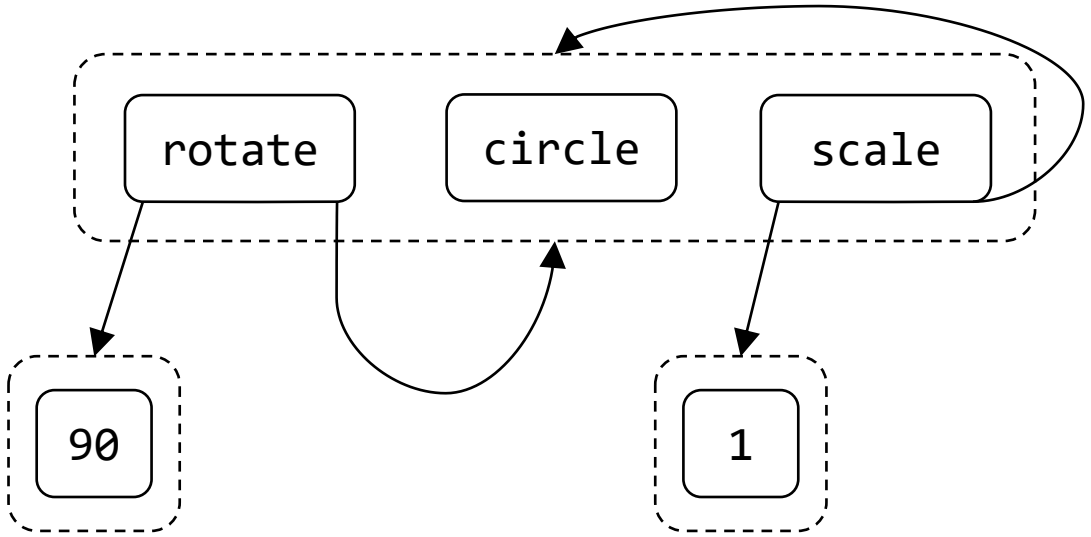
rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

equality saturation

[Tate et al. POPL'09]

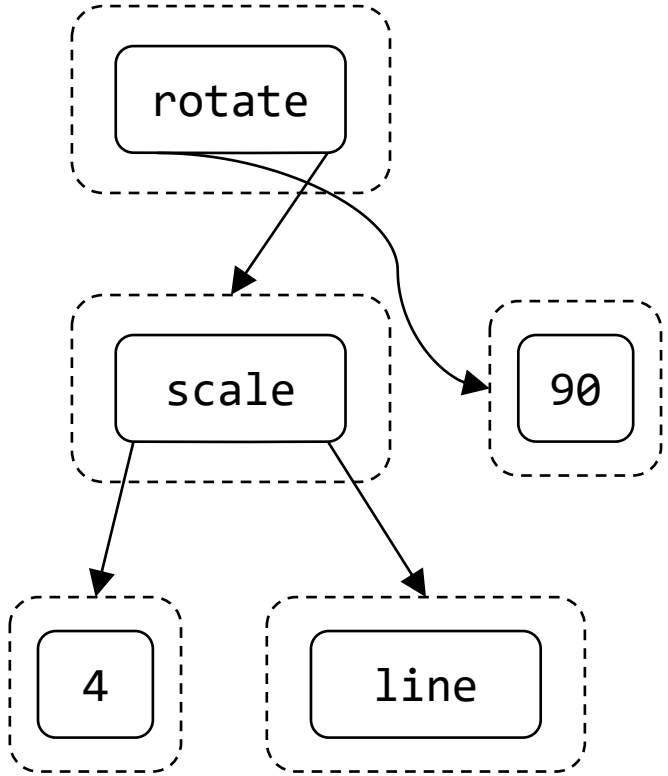
egg [Willsey et al. POPL'21]

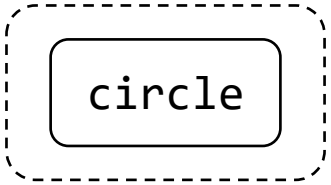


rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

beyond syntactic alignment





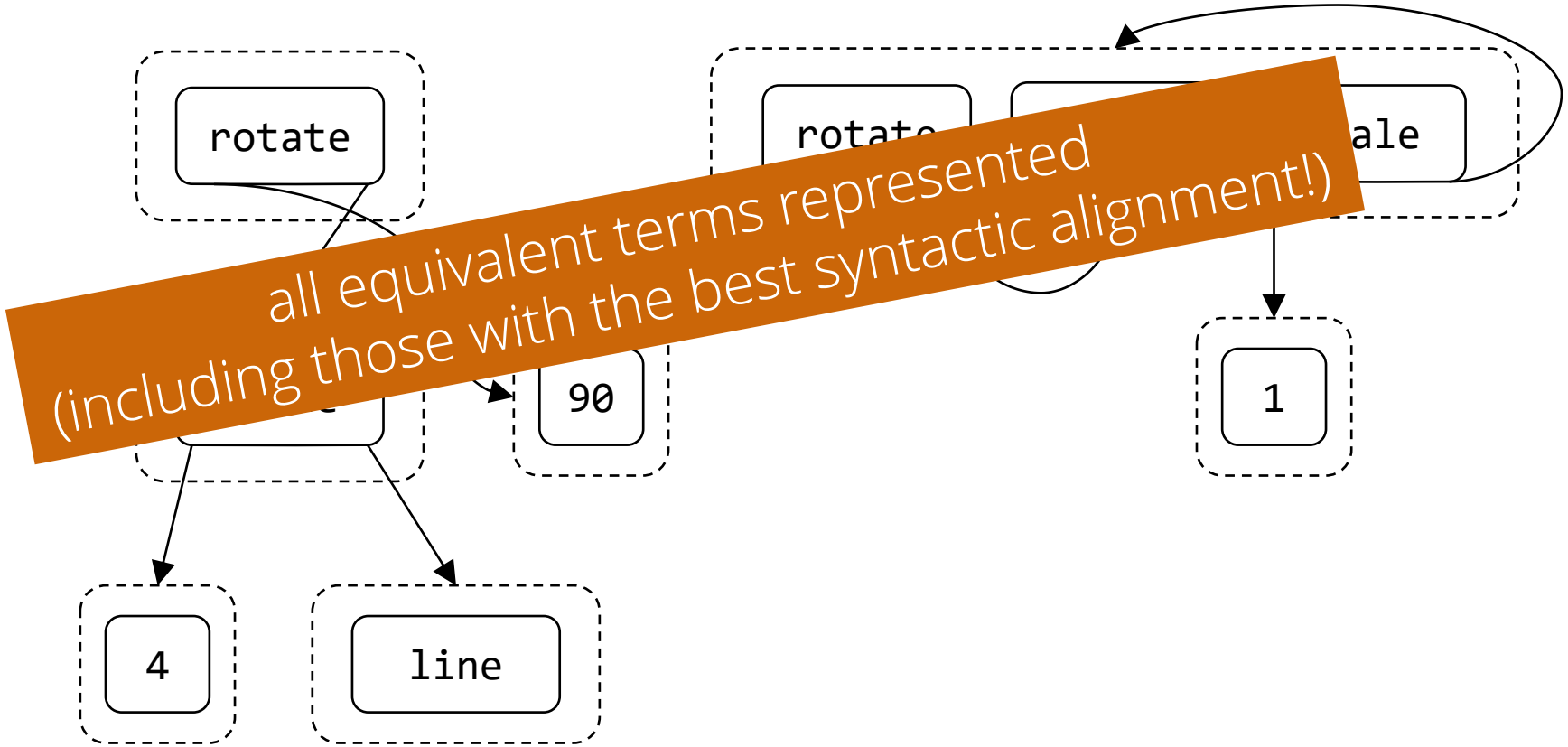
programs:

```
rotate 90 (scale 2 line)
circle
```

rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

beyond syntactic alignment



programs:

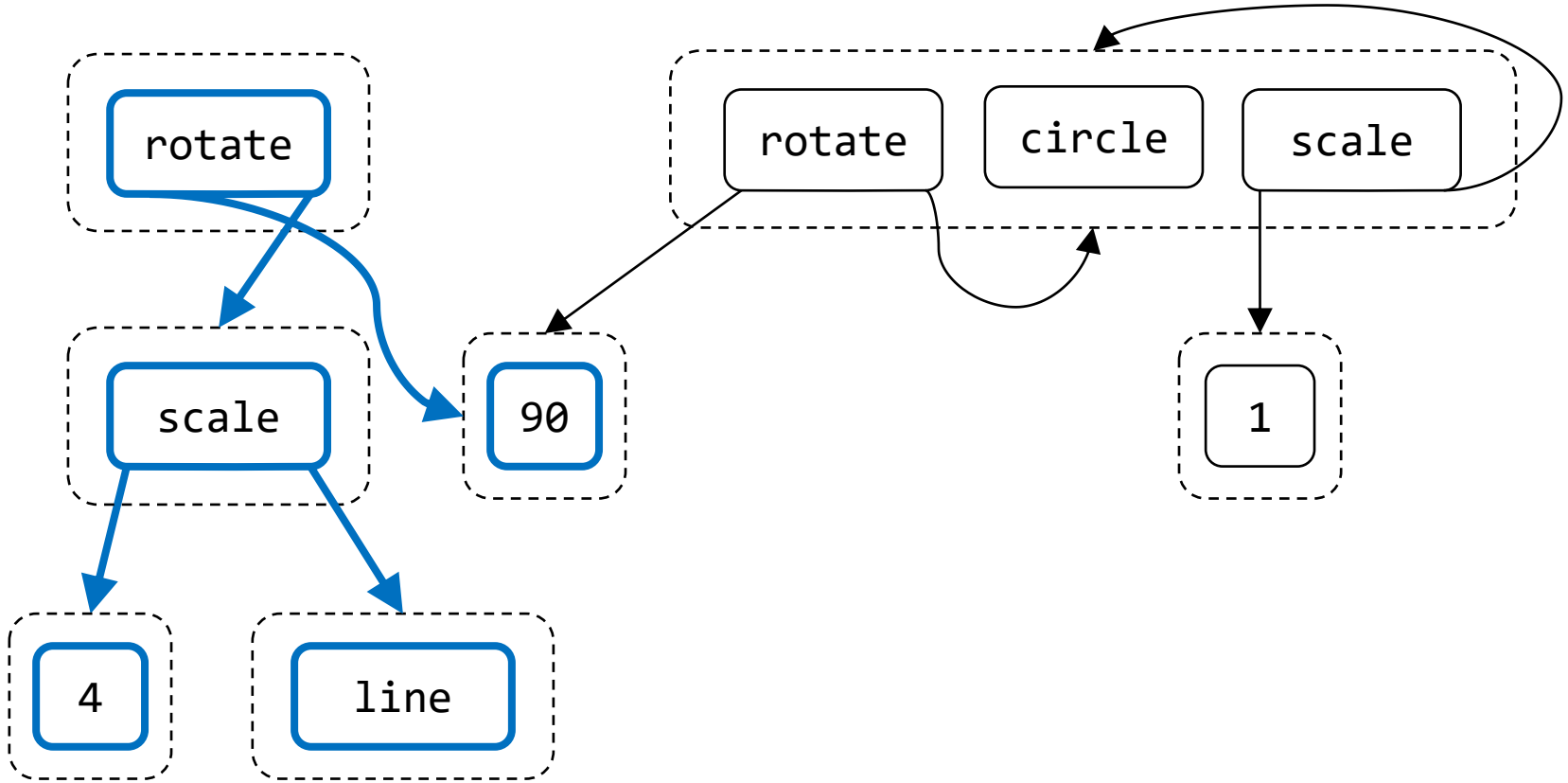
```
rotate 90 (scale 2 line)
circle
```

rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

beta inversion?

step 1: pick subterm



programs:

```
rotate 90 (scale 2 line)
circle
```

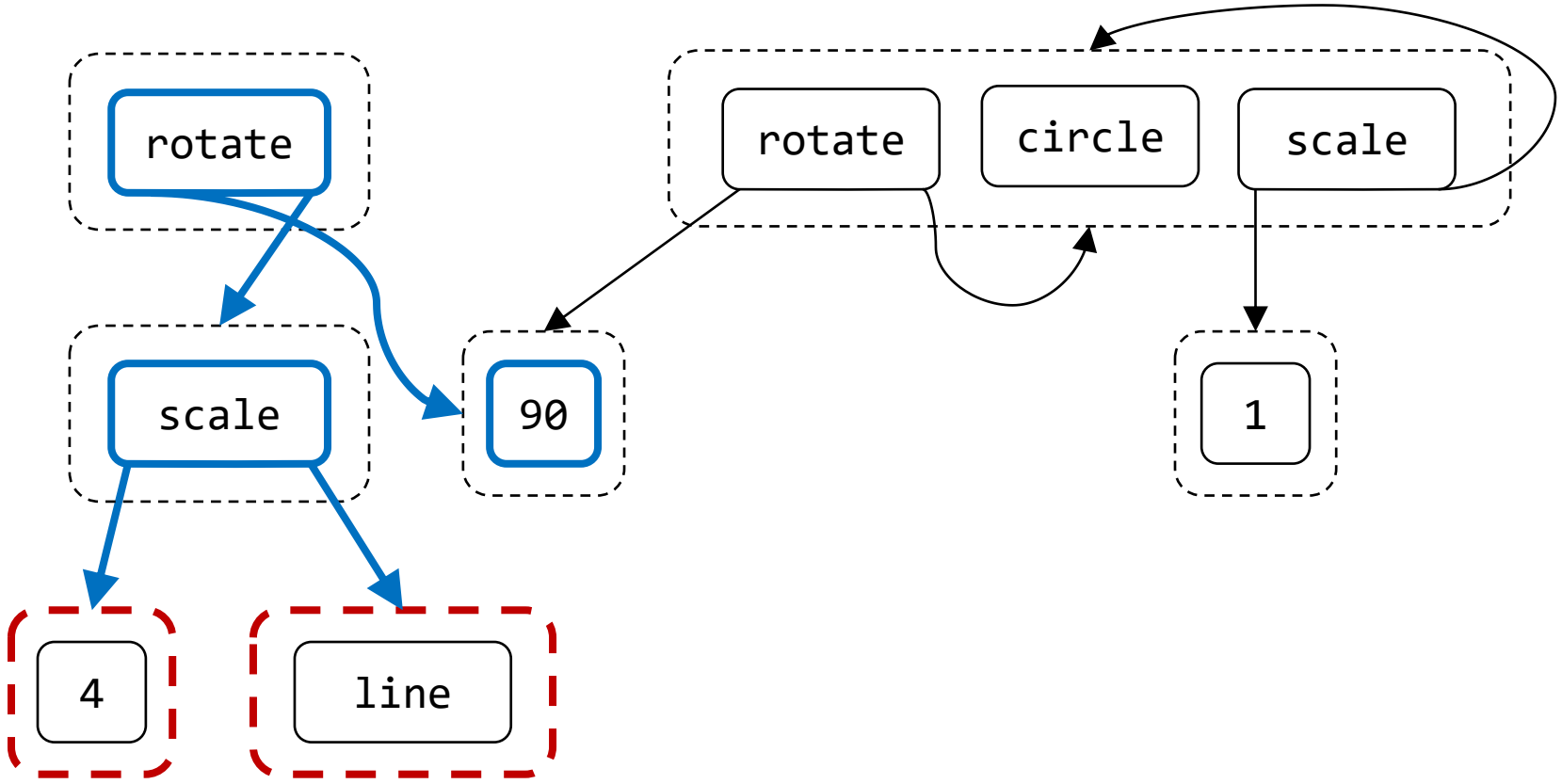
rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

beta inversion?

step 1: pick subterm

step 2: poke holes



programs:

```
rotate 90 (scale 2 line)
circle
```

rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

beta inversion?

step 1: pick subterm

step 2: poke holes

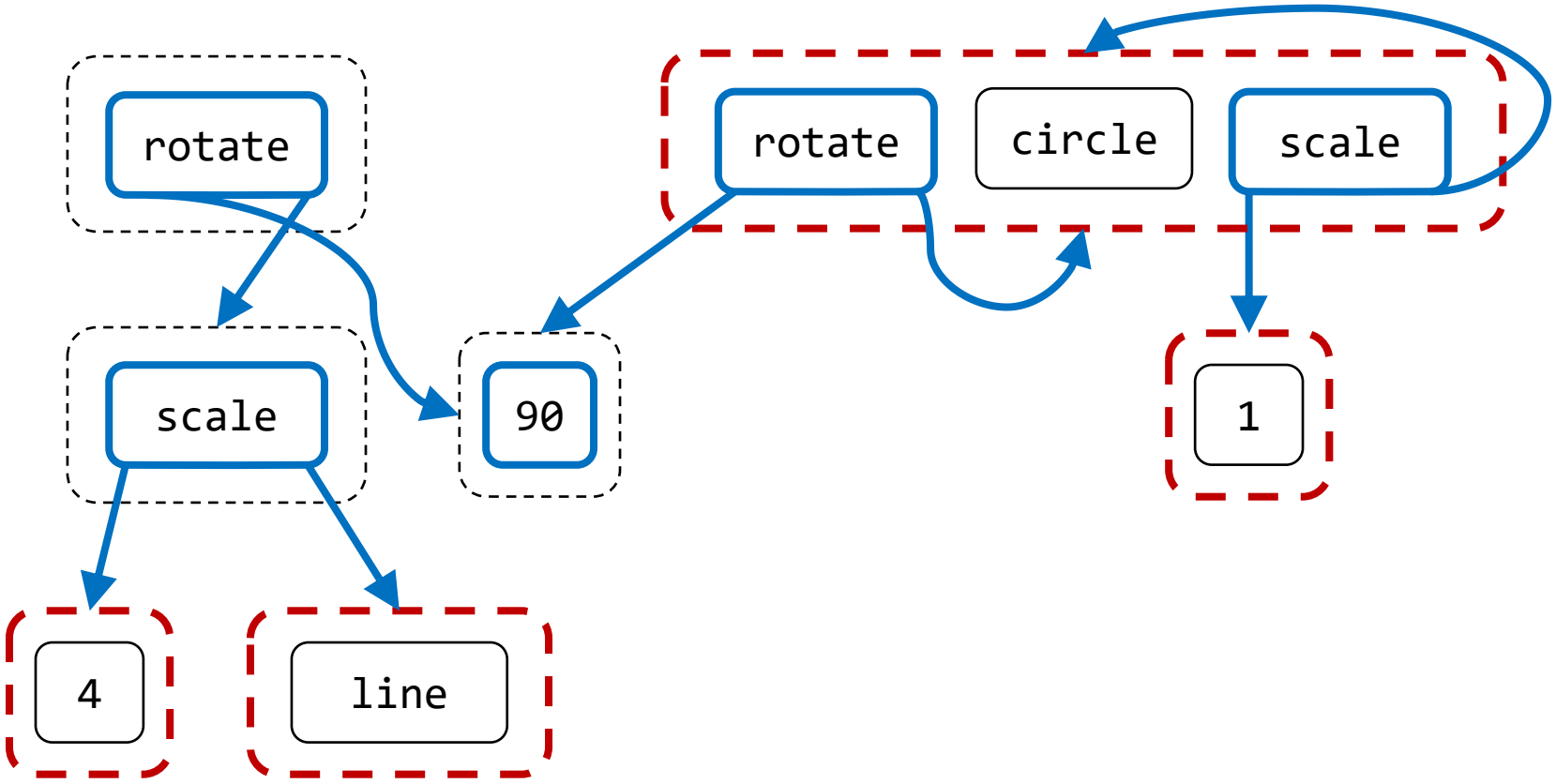
step 3: pattern-match

programs:

```
rotate 90 (scale 2 line)
circle
```

rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```



beta inversion?

step 1: pick subterm

step 2: poke holes

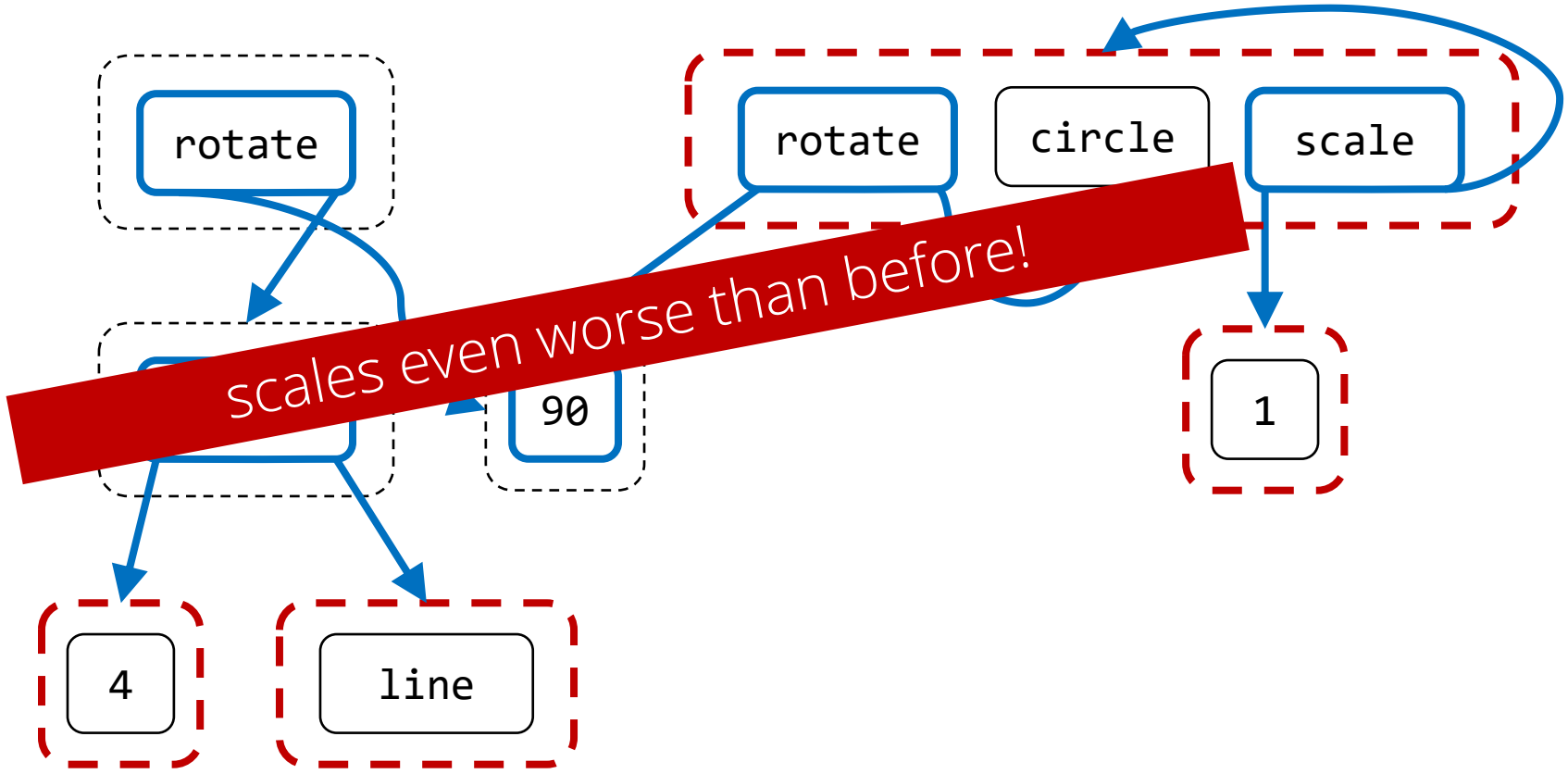
step 3: pattern-match

programs:

```
rotate 90 (scale 2 line)
circle
```

rewrite rules:

```
circle => (scale 1 circle)
circle => (rotate 90 circle)
```

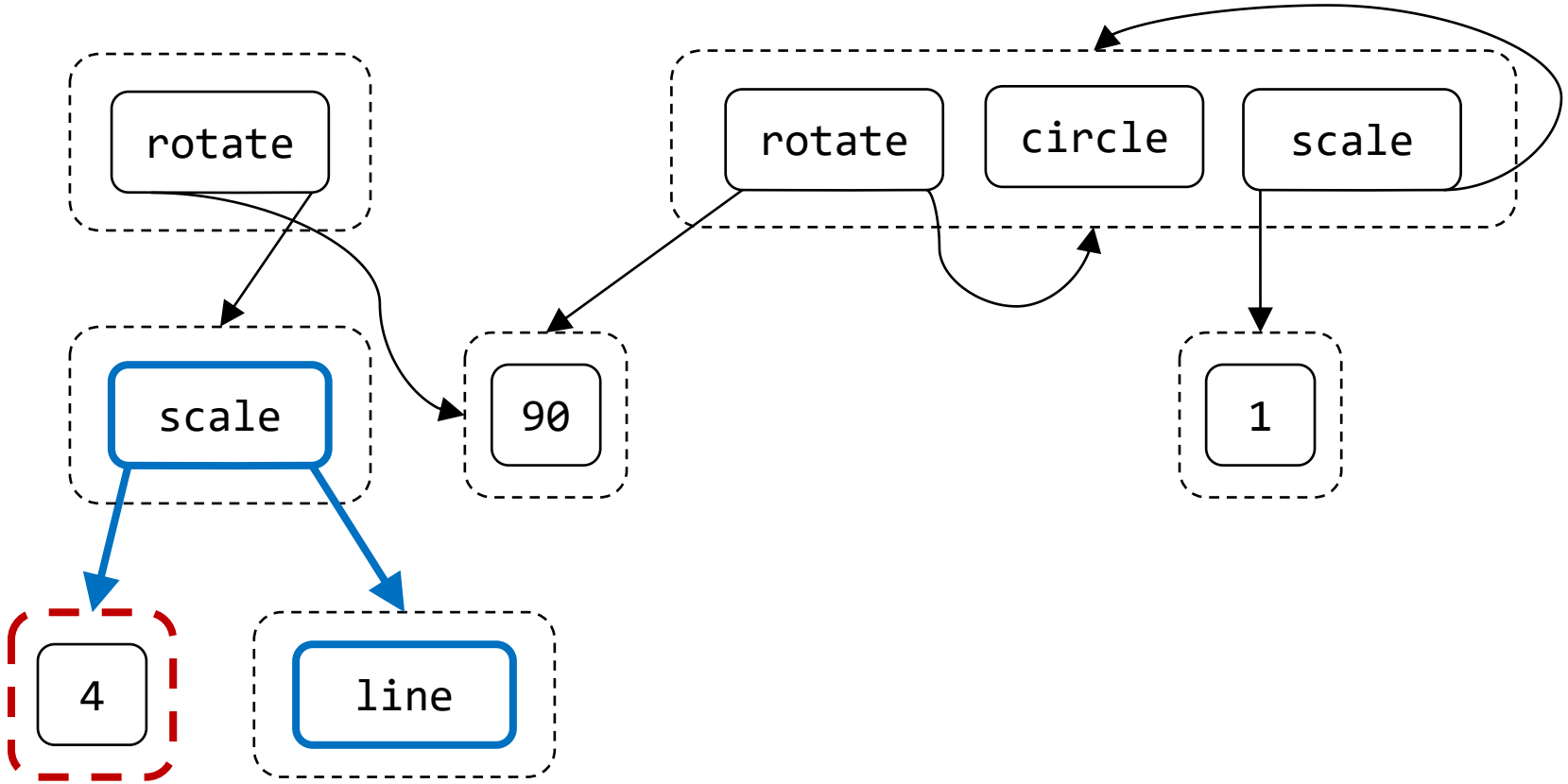


1. challenges
2. flexibility via *e*-graphs
3. **scalability via anti-unification**

better pattern guessing

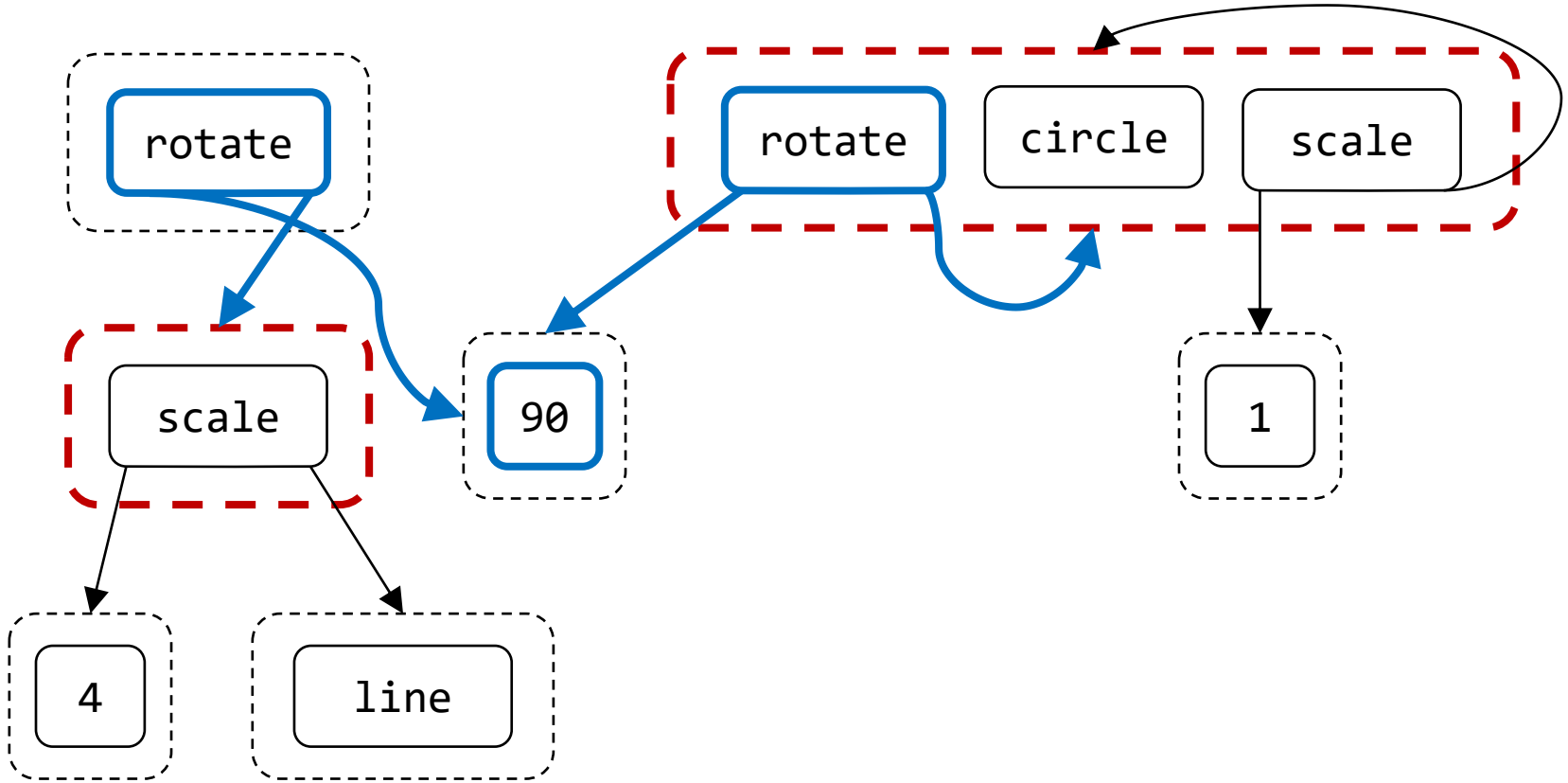
```
scale ?x line
```


observation 1:
pattern must occur
at least twice



better pattern guessing

```
rotate 90 ?x
```

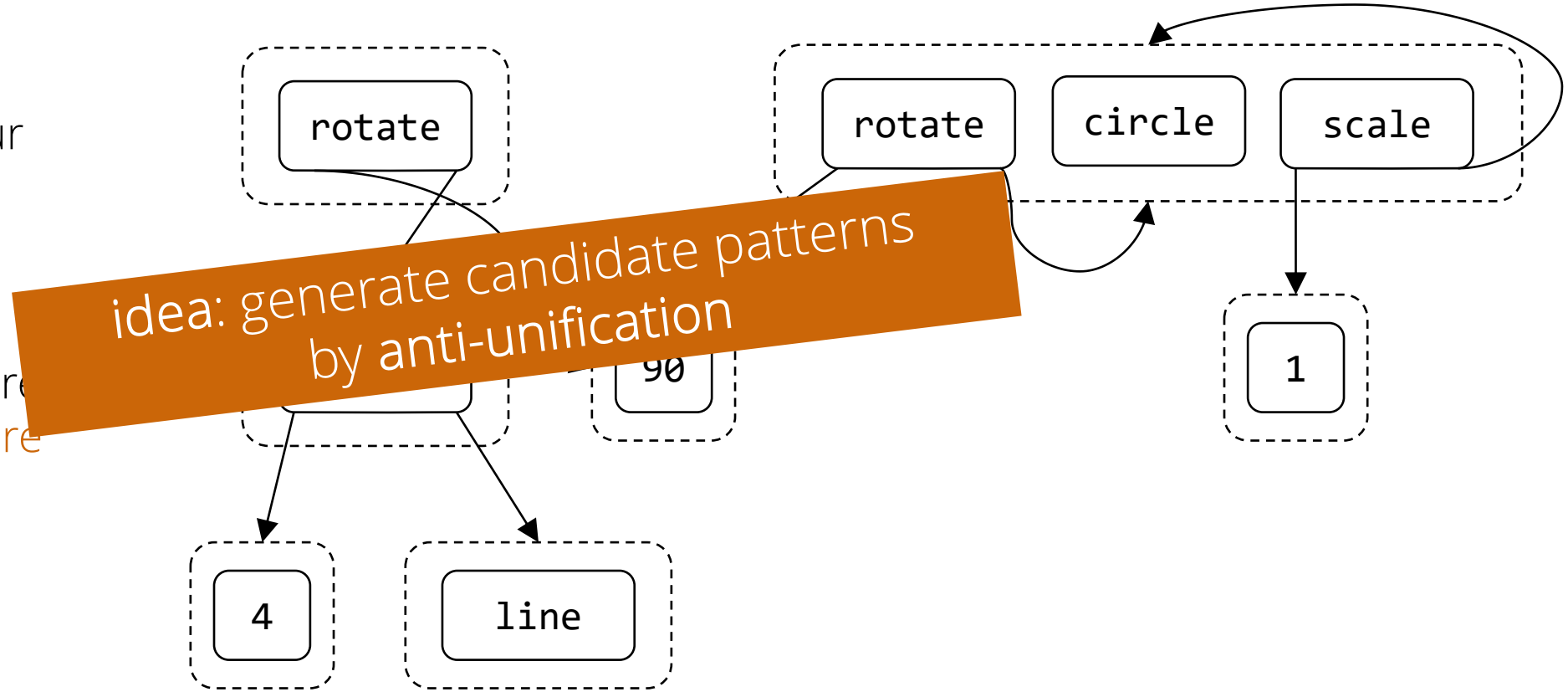


observation 2:
pattern must capture
all common structure

better pattern guessing

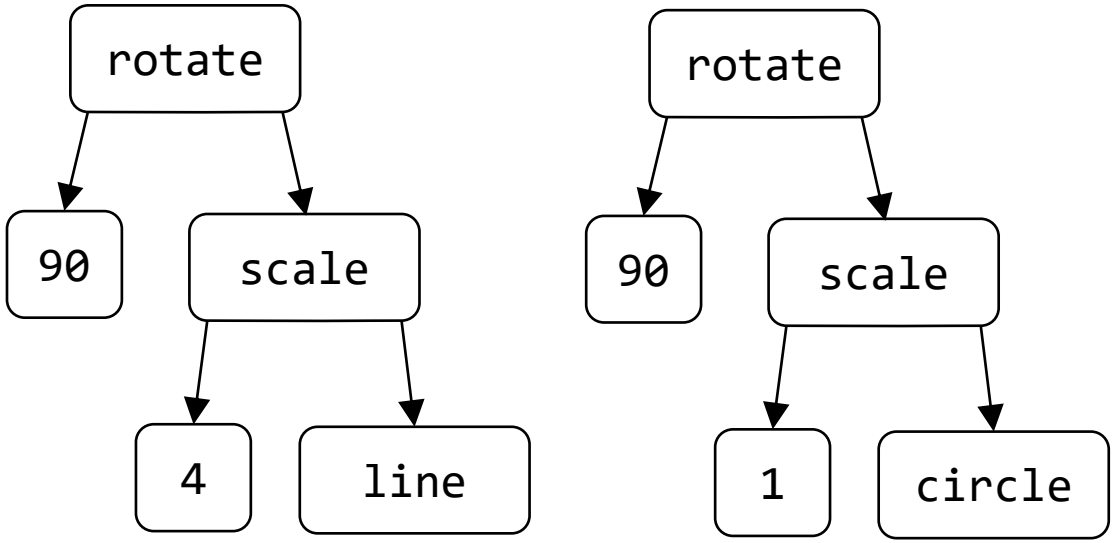
observation 1:
pattern must occur
at least twice

observation 2:
pattern must capture
all common structure



term anti-unification

two terms

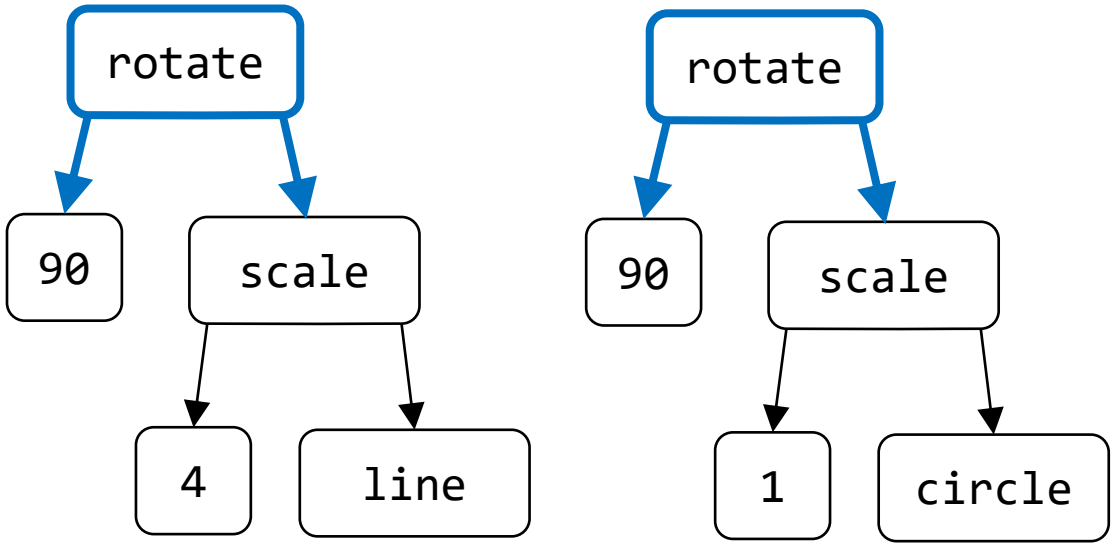


anti-unify

pattern

term anti-unification

two terms



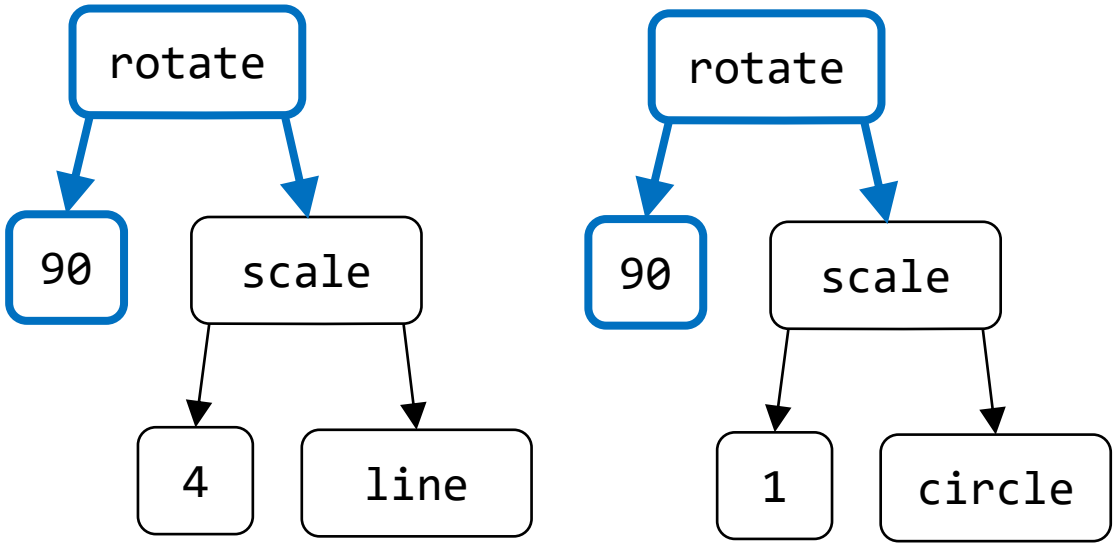
anti-unify

pattern

```
rotate
```

term anti-unification

two terms



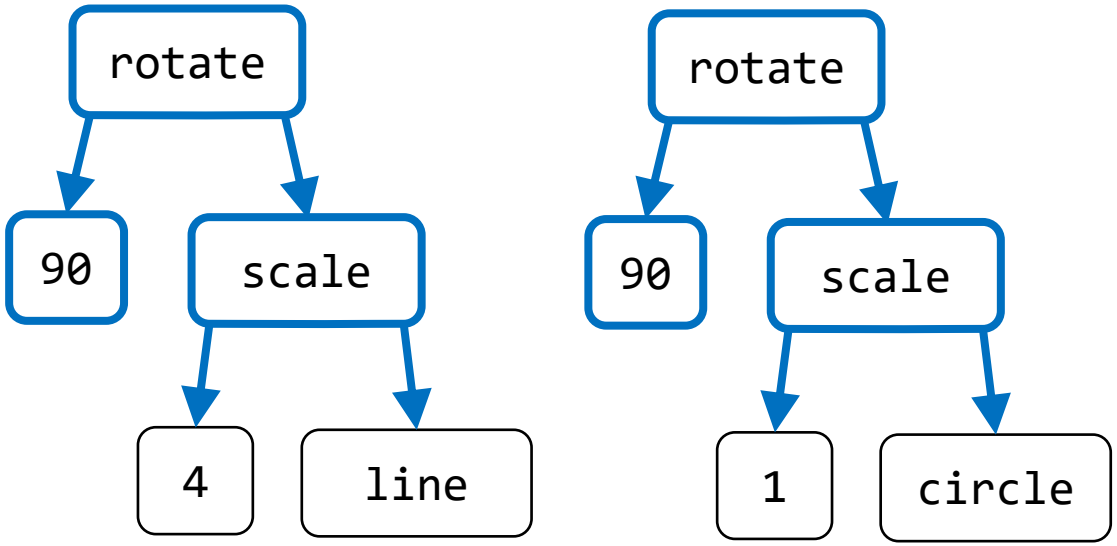
anti-unify

pattern

```
rotate 90
```

term anti-unification

two terms



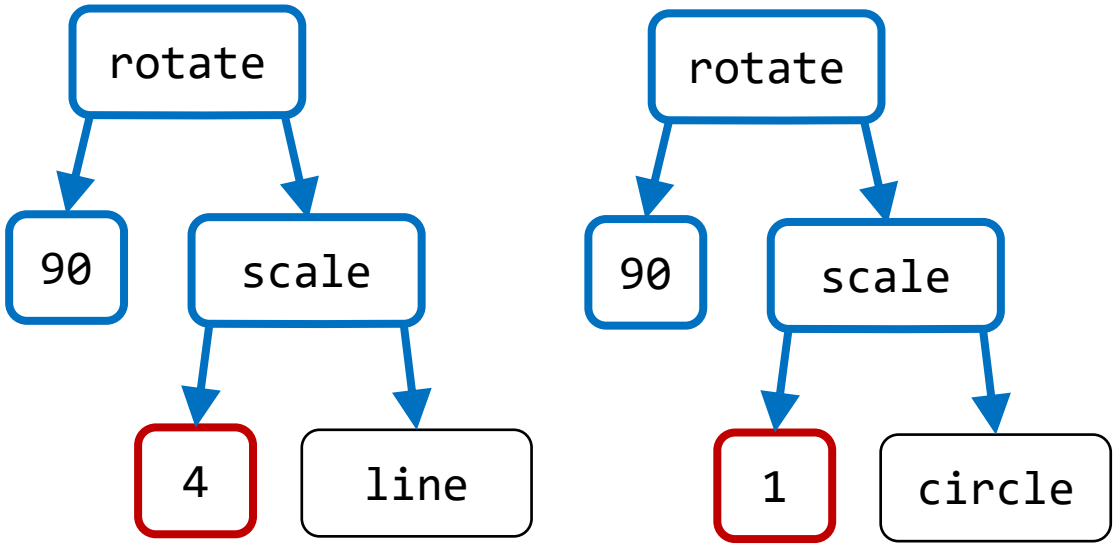
anti-unify

pattern

```
rotate 90 (scale      )
```

term anti-unification

two terms



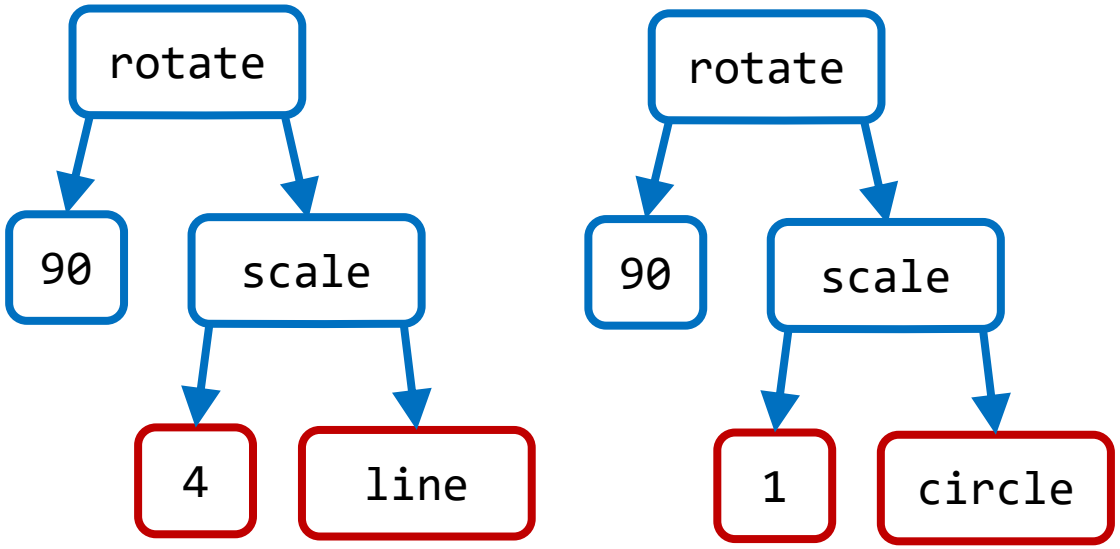
anti-unify

pattern

```
rotate 90 (scale ?x   )
```

term anti-unification

two terms



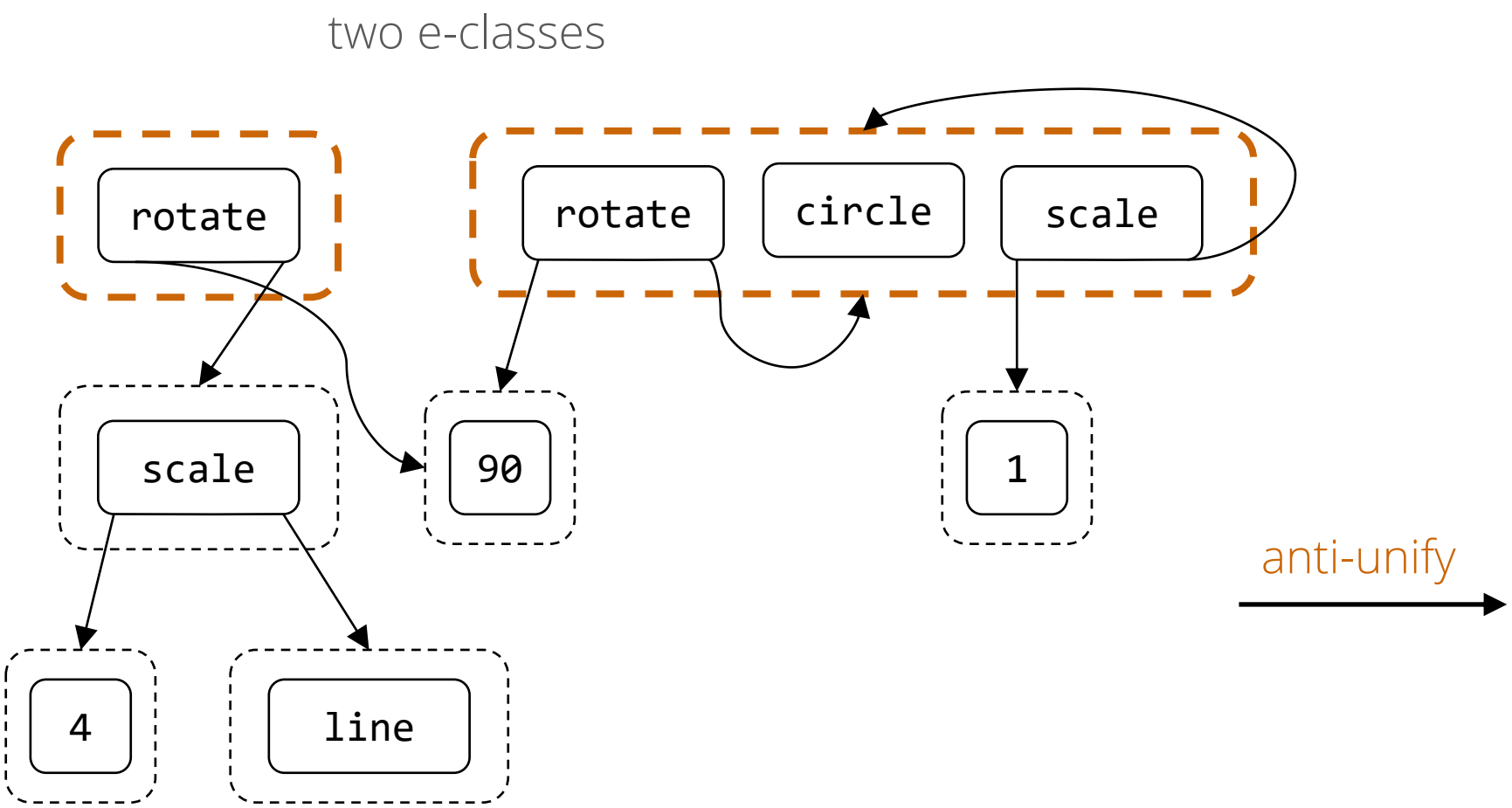
anti-unify

pattern

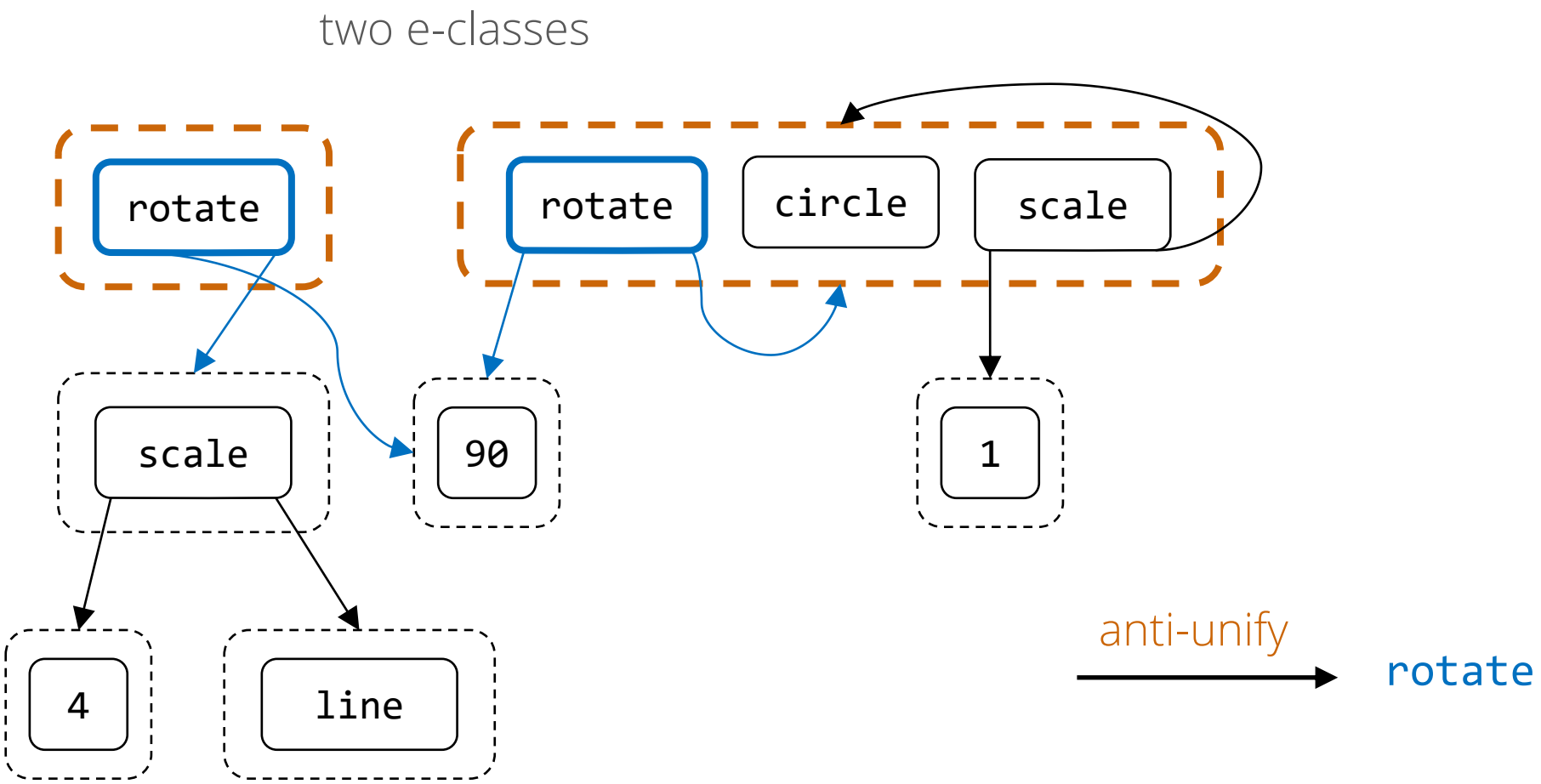
```
rotate 90 (scale ?x ?y)
```

challenge: how to extend this to e-graphs?

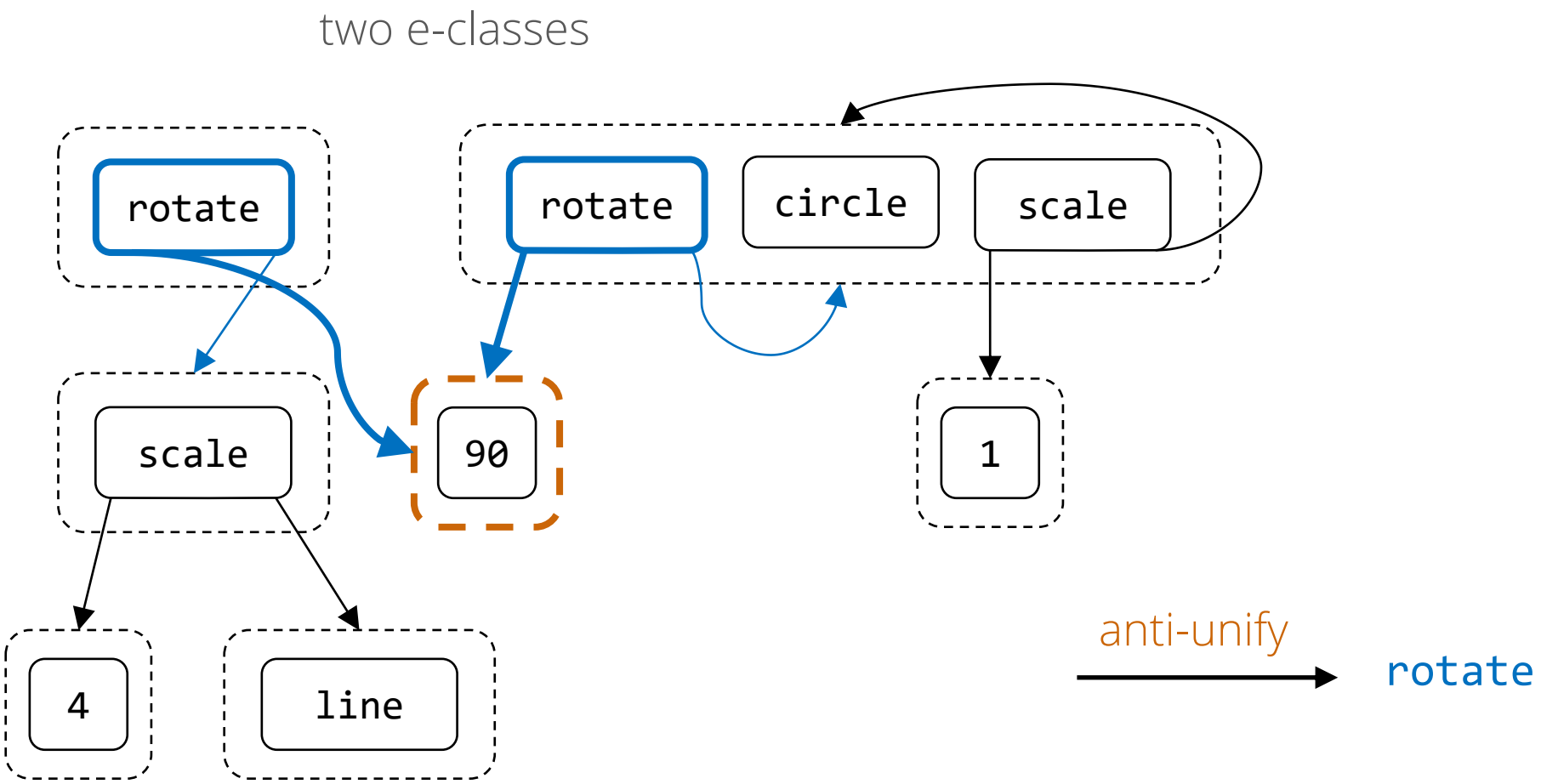
e-graph anti-unification



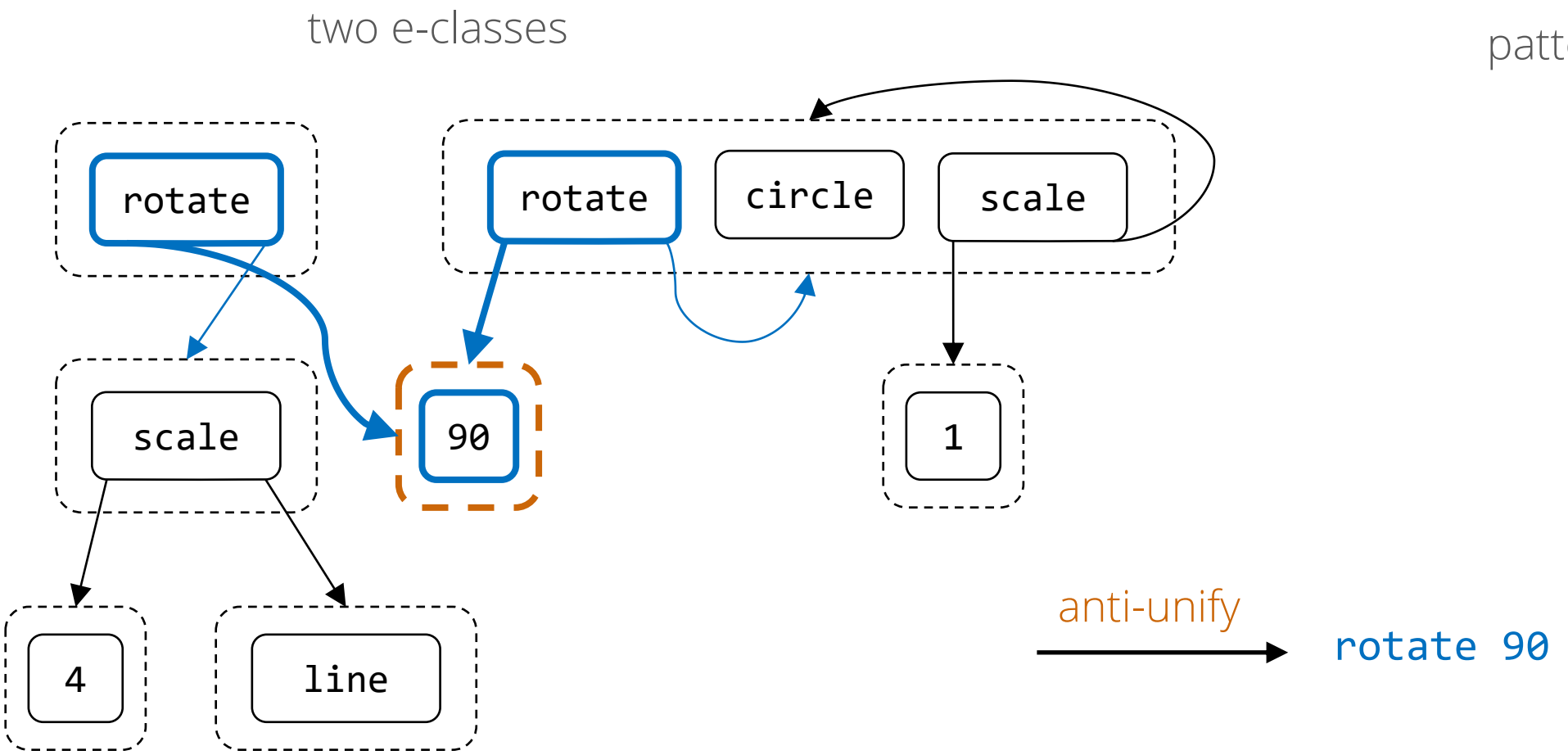
e-graph anti-unification



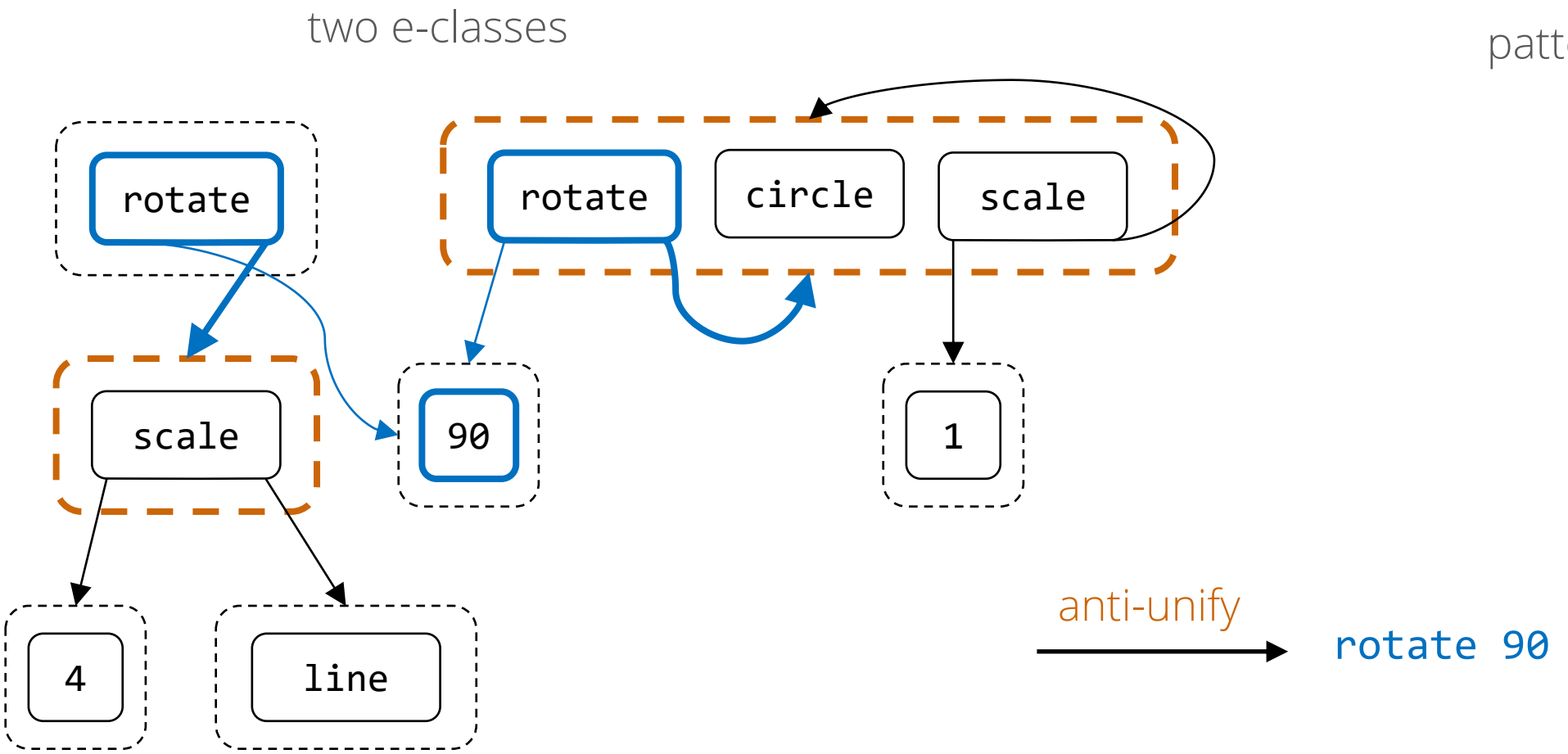
e-graph anti-unification



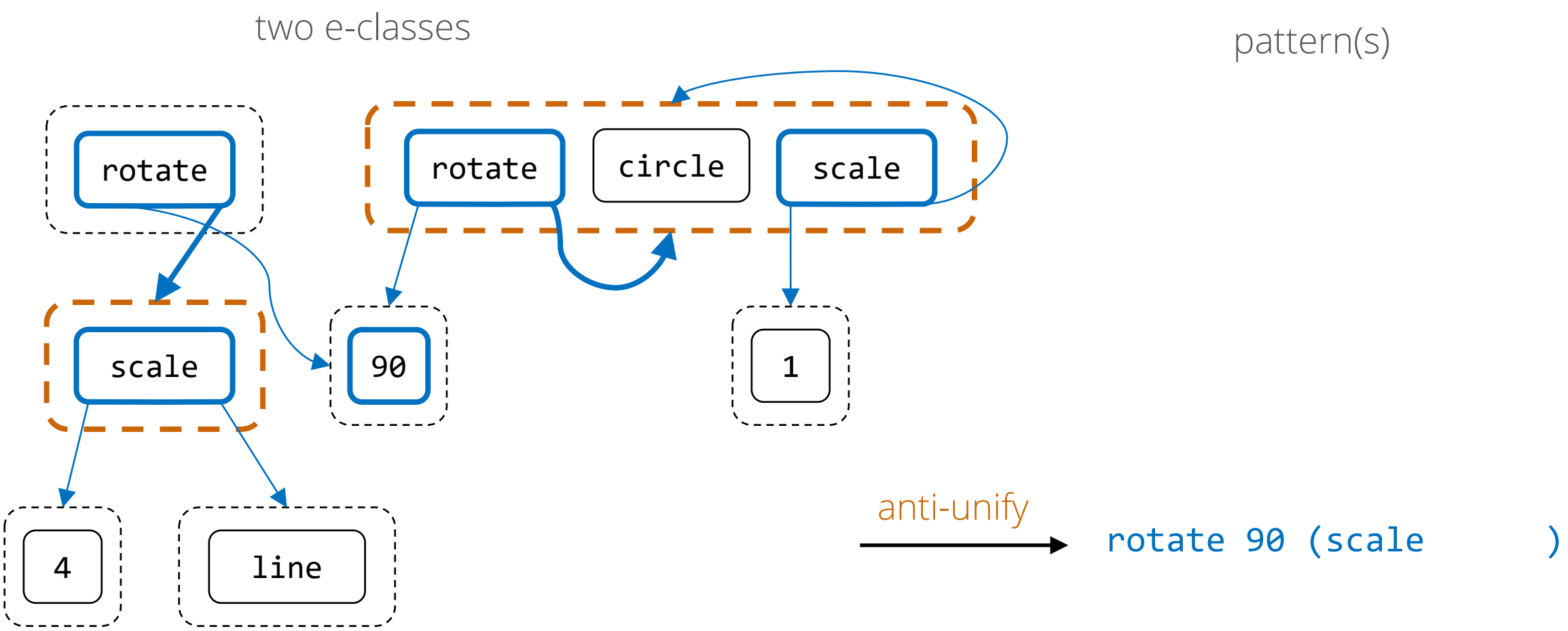
e-graph anti-unification



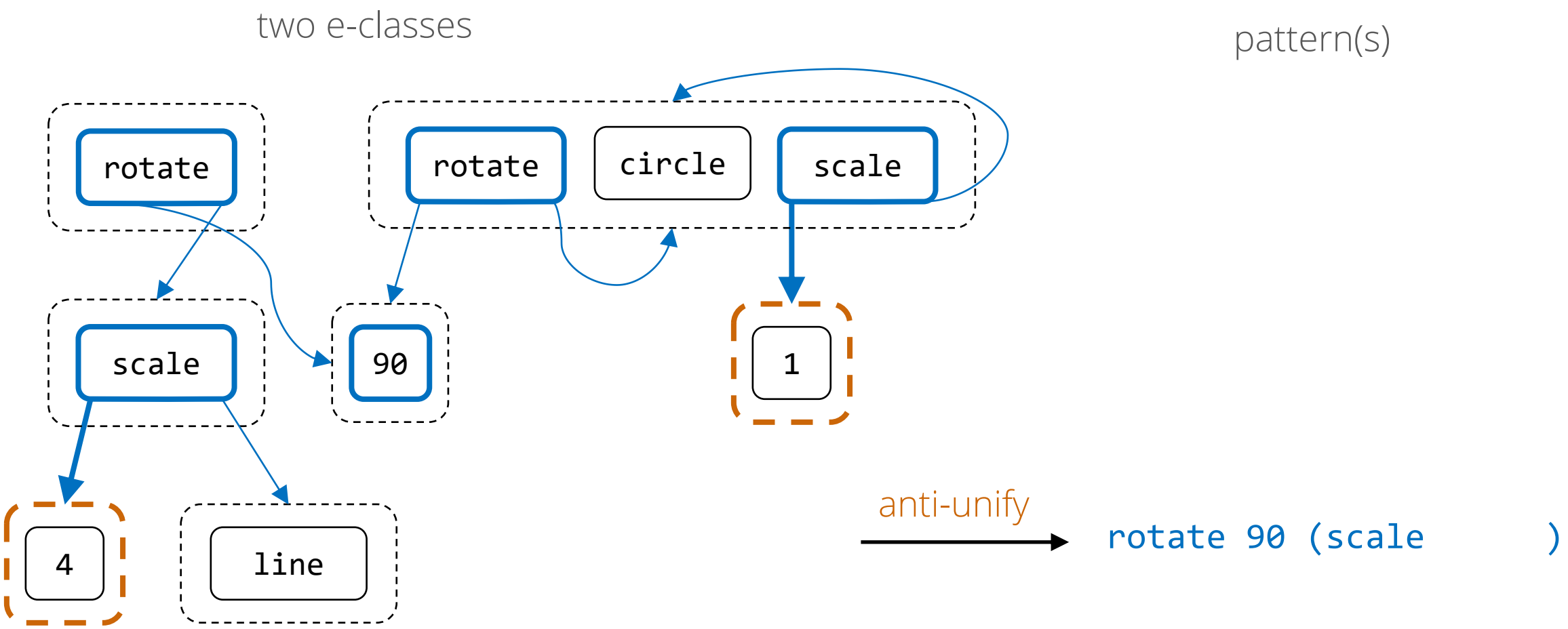
e-graph anti-unification



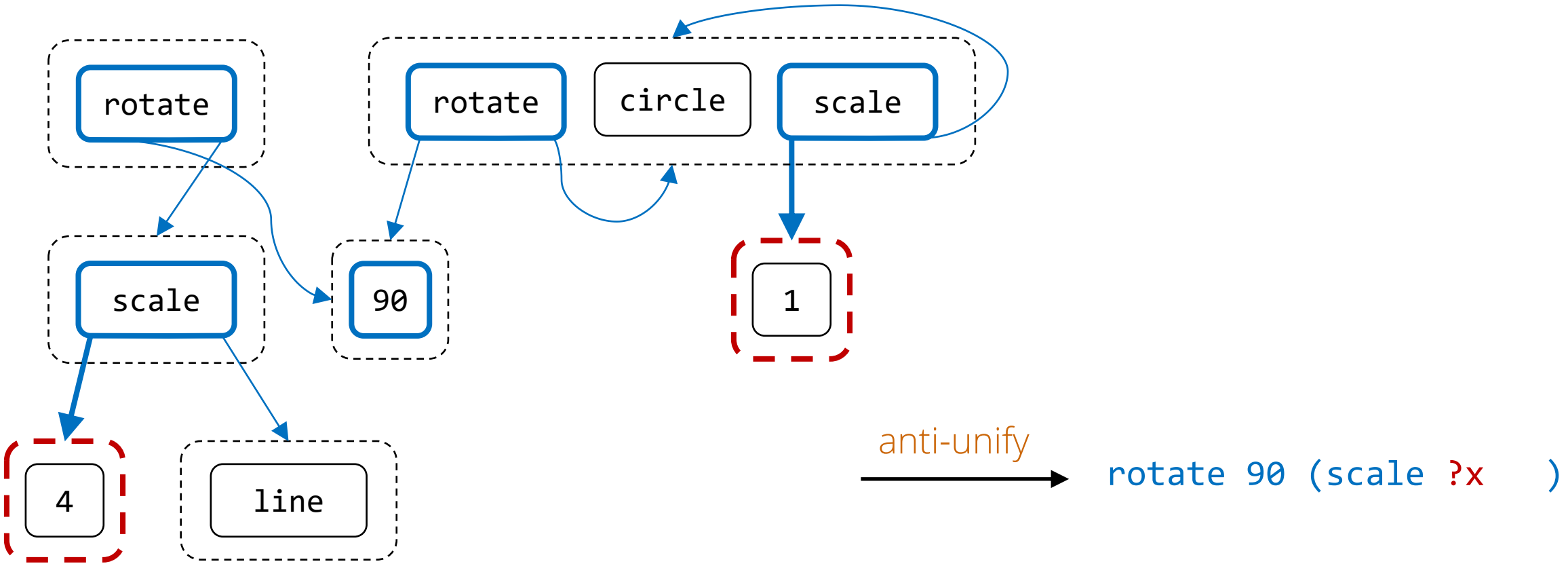
e-graph anti-unification



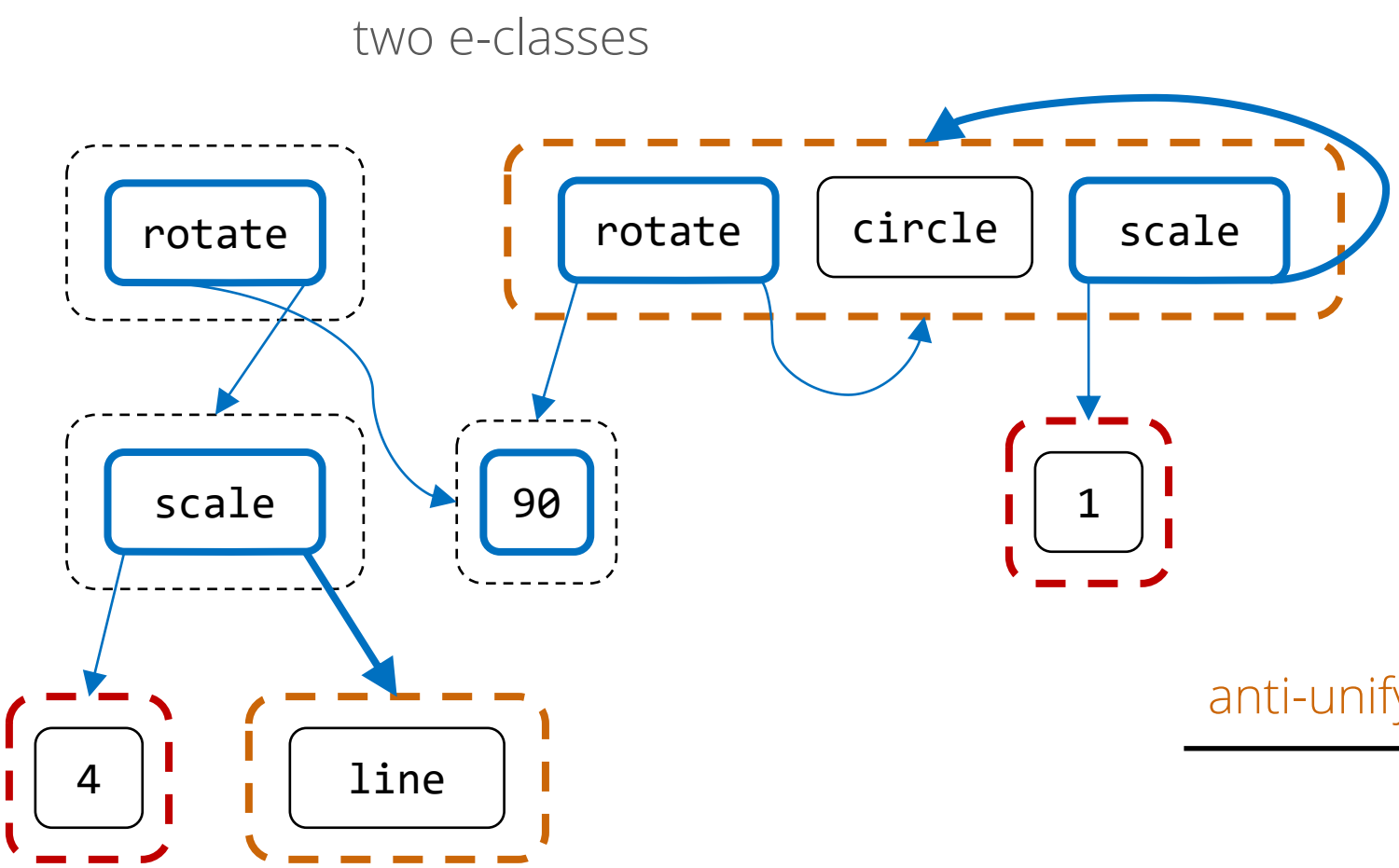
e-graph anti-unification



e-graph anti-unification



e-graph anti-unification

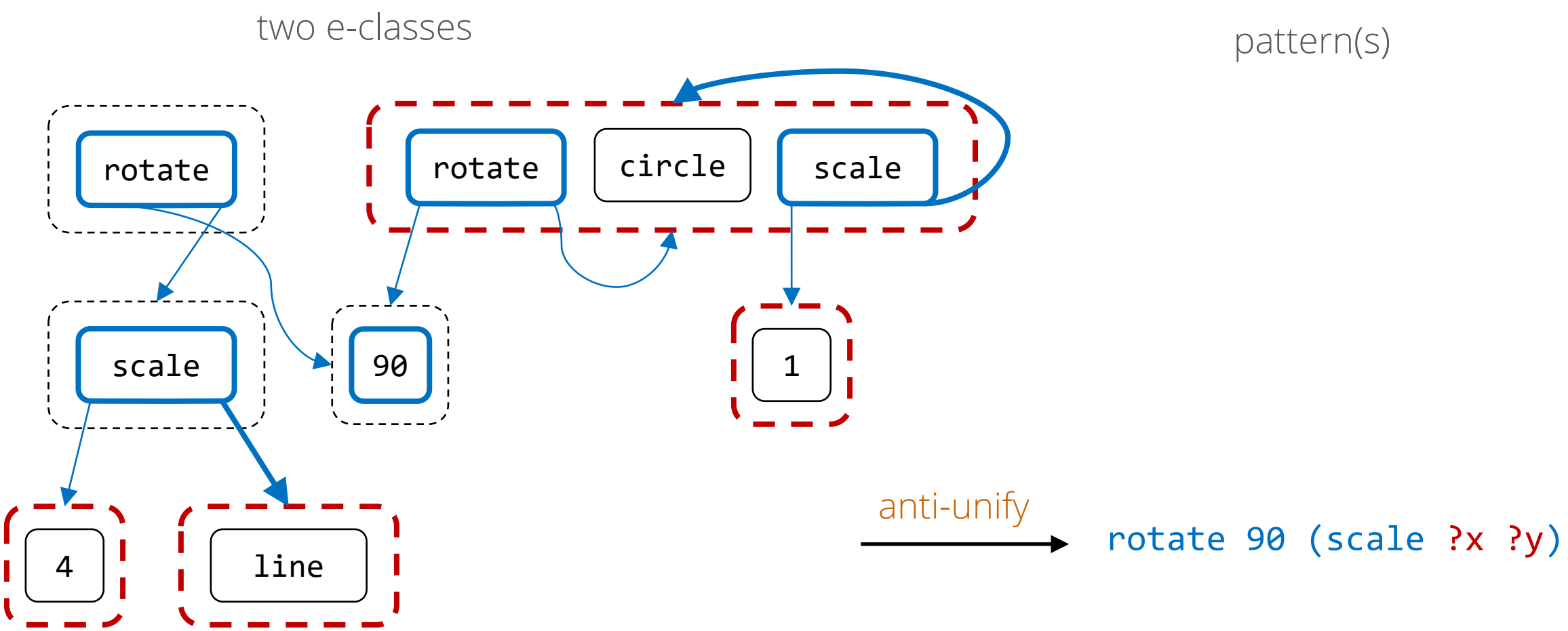


pattern(s)

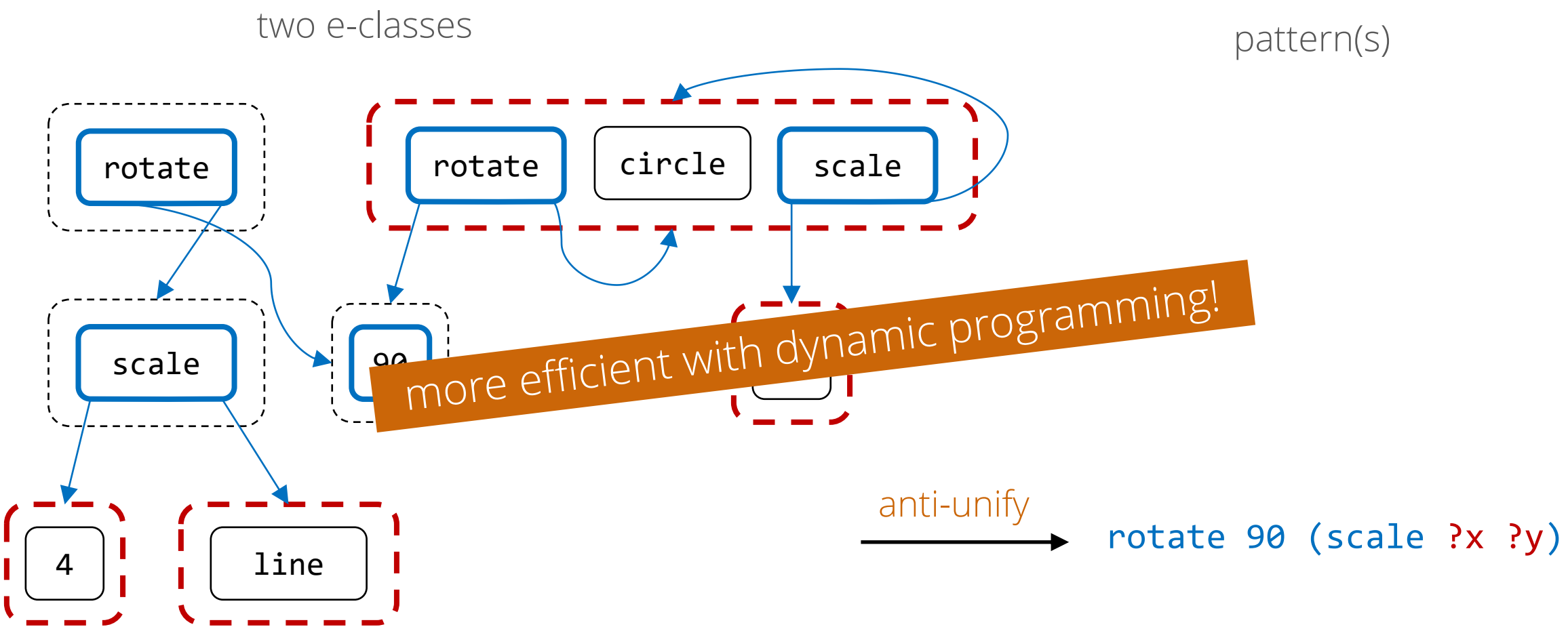
anti-unify

```
rotate 90 (scale ?x   )
```

e-graph anti-unification



e-graph anti-unification



learning better abstractions