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
what do you see?

humans are good at abstraction

can machines do it too?
library learning

programs → library

smaller programs
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

[\texttt{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)]

[\texttt{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

\[ f = \langle e\text{Shape } e\text{Scale } m\text{Shape } m\text{Scale} \rightarrow \]
\[ \langle \text{scale 5 circle}, \]
\[ \text{move } -2\ -1.5 \ (\text{rotate } 90 \ (\text{scale eScale eShape})), \]
\[ \text{move } 2\ -1.5 \ (\text{rotate } 90 \ (\text{scale eScale eShape})), \]
\[ \text{move } 0\ 2 \ (\text{scale mScale mShape}) \rangle \]

\[ f\ \text{line } 2\ \text{circle } 1 \]
\[ f\ \text{circle } 1\ \text{line } 4 \]

size: 50

size: 37
library learning

\[
\begin{align*}
\text{f} & = \{\text{eShape eScale mShape mScale} \rightarrow \\
&\quad \text{[scale 5 circle,} \\
&\quad \text{move -}2 -1.5 \text{ (rotate 90 (scale eScale eShape)),} \\
&\quad \text{move 2 -}1.5 \text{ (rotate 90 (scale eScale eShape)),} \\
&\quad \text{move 0 2 (scale mScale mShape)}\}
\end{align*}
\]

\[
\begin{align*}
\text{size: 50} & \quad \text{size: 37}
\end{align*}
\]
why do we care?

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

DreamCoder [Ellis et al. PLDI’21]
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

[size: 50

[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)]
DreamCoder: beta inversion

step 3: pattern-match

(size: 50)

[ 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

\[ f = \{ x \rightarrow (\text{rotate 90 (scale x line)}) \} \]

\[
[\text{scale 5 circle,}
\text{move } -2 \text{ -1.5 (rotate 90 (scale 2 line))},
\text{move } 2 \text{ -1.5 (rotate 90 (scale 2 line))},
\text{move } 0 \text{ 2 (scale 1 circle)}]\

\[
[\text{scale 5 circle,}
\text{move } -2 \text{ -1.5 (rotate 90 (scale 1 circle))},
\text{move } 2 \text{ -1.5 (rotate 90 (scale 1 circle))},
\text{move } 0 \text{ 2 (scale 4 line)}]\

size: 50

\[
[\text{scale 5 circle,}
\text{move } -2 \text{ -1.5 (f 2)},
\text{move } 2 \text{ -1.5 (f 2)},
\text{move } 0 \text{ 2 (scale 1 circle)}]\

\[
[\text{scale 5 circle,}
\text{move } -2 \text{ -1.5 (rotate 90 (scale 1 circle))},
\text{move } 2 \text{ -1.5 (rotate 90 (scale 1 circle))},
\text{move } 0 \text{ 2 (scale 4 line)}]\

size: 51
DreamCoder: beta inversion

repeat

\[
f = \{x \rightarrow (\text{rotate } 90 \ (\text{scale } x \ \text{line}))\}
\]

size: 51

\[
f = \{x \ y \rightarrow (\text{rotate } 90 \ (\text{scale } x \ y))\}
\]

size: 45

[\text{scale } 5 \ \text{circle,}
\begin{align*}
\text{move} \ -2 & \ -1.5 \ (\text{rotate } 90 \ (\text{scale } 2 \ \text{line})), \\
\text{move} \ 2 & \ -1.5 \ (\text{rotate } 90 \ (\text{scale } 2 \ \text{line})), \\
\text{move} \ 0 & \ 2 \ (\text{scale } 1 \ \text{circle})
\end{align*}
\]

...
DreamCoder: limitations

repeat

\[ \text{f} = x \rightarrow (\text{rotate } 90 (\text{scale } x \ \text{line})) \quad \text{size: 51} \]

\[ \text{f} = x \ y \ a \ b \rightarrow (\text{rotate } 90 (\text{scale } x \ y)) \quad \text{size: 45} \]

\[ \text{f} = \text{scale } 5 \ \text{circle}, \ \text{move } -2 \ -1.5 \ (\text{rotate } 90 (\text{scale } x \ \text{line})), \ \text{move } 2 \ -1.5 \ (\text{rotate } 90 (\text{scale } 2 \ \text{line})), \ \text{move } 0 \ 2 \ (\text{scale } 4 \ \text{line}) \] \quad \text{size: 50} \]

1. inefficient / incomplete

2. requires syntactic alignment
syntactic alignment

\[
f = \langle x \ y \ a \ b \rangle \rightarrow \begin{array}{r}
\text{[scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 0 2 (scale 1 circle)]}
\end{array}
\]

\[
\text{[scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 0 2 (scale 4 line)]}
\]

\[
f = \langle x \ y \ a \ b \rangle \rightarrow \\
\begin{array}{r}
\text{[scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale y x))}, \\
\text{move 2 -1.5 (rotate 90 (scale y x))}, \\
\text{move 0 2 (scale b a)}
\end{array}
\]

\[
\text{[scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 0 2 (scale 4 line)]}
\]
syntactic alignment

\[ f = \langle x \ y \ a \ b \rangle \rightarrow \]

\[
\begin{array}{l}
[\text{scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 0 2 (scale 1 circle)}] \\
\end{array}
\]

\[
\begin{array}{l}
[\text{scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 2 -1.5 (rotate 90 (scale 1 circle))}, \\
\text{move 0 2 (scale 4 line)}] \\
\end{array}
\]

\[
\begin{array}{l}
[\text{scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 2 -1.5 (rotate 90 (scale 2 line))}, \\
\text{move 0 2 circle}] \\
\end{array}
\]

\[
\begin{array}{l}
[\text{scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 circle)}, \\
\text{move 2 -1.5 (rotate 90 circle)}, \\
\text{move 0 2 (scale 4 line)}] \\
\end{array}
\]

\[
\text{f = \langle x \ y \ a \ b \rangle \rightarrow} \\
[\text{scale 5 circle,} \\
\text{move -2 -1.5 (rotate 90 (scale y x))}, \\
\text{move 2 -1.5 (rotate 90 (scale y x))}, \\
\text{move 0 2 (scale b a)}] \\
\]
syntactic alignment

\[ f = \langle x \ y \ a \ b \rangle \to \]
\[ \begin{align*}
[\text{scale 5 circle,} \\
\text{move } -2 \ -1.5 \ (\text{rotate } 90 \ (\text{scale 2 line}))], \\
\text{move } 2 \ -1.5 \ (\text{rotate } 90 \ (\text{scale 2 line})), \\
\text{move } 0 \ 2 \ (\text{scale 1 circle})]
\end{align*} \]

\[ \begin{align*}
[\text{scale 5 circle,} \\
\text{move } -2 \ -1.5 \ (\text{rotate } 90 \ (\text{scale 1 circle})), \\
\text{move } 2 \ -1.5 \ (\text{rotate } 90 \ (\text{scale 1 circle})), \\
\text{move } 0 \ 2 \ (\text{scale 4 line})]
\end{align*} \]
syntactic alignment

\[ f = \langle x \ y \ a \ b \rangle \]

DreamCoder

\[
\begin{aligned}
\text{[scale 5 circle,} \\
\text{move} \ 2 \ -1.5 \ (\text{rotate} \ 90 \ (\text{scale} \ 2 \ \text{line})),} \\
\text{move} \ 0 \ 2 \ (\text{scale} \ 1 \ \text{circle})]
\end{aligned}
\]

\[
\begin{aligned}
\text{[scale 5 circle,} \\
\text{move} \ 2 \ -1.5 \ (\text{rotate} \ 90 \ (\text{scale} \ 2 \ \text{line})),} \\
\text{move} \ 0 \ 2 \ (\text{scale} \ 4 \ \text{line})]
\end{aligned}
\]

\[
\begin{aligned}
\text{[scale 5 circle,} \\
\text{move} \ 2 \ -1.5 \ (\text{rotate} \ 90 \ (\text{scale} \ 2 \ \text{line})),} \\
\text{move} \ 0 \ 2 \ (\text{scale} \ 1 \ \text{circle})]
\end{aligned}
\]

\[
\begin{aligned}
\text{[scale 5 circle,} \\
\text{move} \ 2 \ -1.5 \ (\text{rotate} \ 90 \ (\text{scale} \ 2 \ \text{line})),} \\
\text{move} \ 0 \ 2 \ (\text{scale} \ 1 \ \text{circle})]
\end{aligned}
\]

\[
\begin{aligned}
\text{[scale 5 circle,} \\
\text{move} \ 2 \ -1.5 \ (\text{rotate} \ 90 \ (\text{scale} \ 2 \ \text{line})),} \\
\text{move} \ 0 \ 2 \ (\text{scale} \ 4 \ \text{line})]
\end{aligned}
\]
syntactic alignment

\[ \text{DreamCoder} \]
\[
\text{f} = \{(x, y, a, b) \mapsto \}
\]
\[
[\text{scale 5 circle,}
\text{move -2 -1.5 (rotate 90 (scale 2 line)),}
\text{move 2 -1.5 (rotate 90 (scale 2 line)),}
\text{move 0 2 (scale 1 circle)}]}
\]

\[ \text{babble} \]
\[
\text{f (rotate 90 (scale 2 line)) circle}
\]
\[
\text{f (rotate 90 (scale 2 line)) circle (scale 4 line)}
\]
beyond syntactic alignment

```plaintext
[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

```plaintext
[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

rewrite rules:

\[ \text{circle} \Rightarrow (\text{scale 1 circle}) \]
\[ \text{circle} \Rightarrow (\text{rotate 90 circle}) \]
equality saturation

[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 \( ? \times \text{line} \)

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

idea: generate candidate patterns by anti-unification
term anti-unification

two terms

rotate

90

scale

4

line

rotate

90

scale

1

circle

anti-unify

pattern
term anti-unification

two terms

rotate

90

scale

4

line

rotate

90

scale

1

circle

anti-unify

rotate

pattern
term anti-unification

two terms

rotate
90
scale
4
line

rotate
90
scale
1
circle

anti-unify

rotate 90

pattern
term anti-unification

two terms

rotate
down
90
down
down
4
down
down
line
down

down
down

down
down

down
down

down
down

down

down
down

scale
down
down

scale
down
down

rotate
down
90
down
down
1
down
down
circle
down
down

anti-unify

rotate 90 (scale)
term anti-unification

two terms

pattern

rotate 90 (scale ?x  )
term anti-unification

two terms

rotate

90

scale

4

line

rotate

90

scale

1

circle

anti-unify

rotate 90 (scale ?x ?y)

pattern

challenge: how to extend this to e-graphs?
e-graph anti-unification

two e-classes

rotate

scale

4

line

rotate

circle

scale

90

1

anti-unify

pattern(s)
e-graph anti-unification

two e-classes

rotate

scale

4

line

90

rotate

circle

scale

1

pattern(s)

anti-unify

rotate
e-graph anti-unification

two e-classes

rotate

scale

4

line

rotate

circle

scale

90

1

anti-unify

rotate

pattern(s)
e-graph anti-unification

two e-classes

rotate

scale

90

4

line

rotate
circle
scale

rotate 90

anti-unify

pattern(s)
e-graph anti-unification

Two e-classes:
- rotate
- scale

Pattern(s):
- rotate 90
- circle
- scale

Anti-unify: rotate 90
e-graph anti-unification

two e-classes

rotate

circle

scale

90

1

4

line

anti-unify

rotate 90 (scale )

pattern(s)
e-graph anti-unification

two e-classes

rotate

scale

90

4

line

rotate
circle

scale

1

rotate 90 (scale)

anti-unify

pattern(s)
e-graph anti-unification

two e-classes

rotate

scale

90

4

line

rotate
circle

scale

1

pattern(s)

anti-unify

rotate 90 (scale ?x 1)
e-graph anti-unification

two e-classes

rotate
circle

scale

90

1

4

line

pattern(s)

rotate 90 (scale ?x )

anti-unify
e-graph anti-unification

two e-classes

rotate
circle

scale
90

1

4
line

anti-unify

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

pattern(s)
e-graph anti-unification

two e-classes

pattern(s)

rotate
circle
scale

4
line

more efficient with dynamic programming!

rotate 90 (scale ?x ?y)

anti-unify
learning better abstractions

programs \quad \rightarrow \quad \text{babble} \quad \rightarrow \quad \text{library} \quad \rightarrow \quad \text{smaller programs}