DreamCoder: Growing domain-specific knowledge via wake-sleep program induction

Kevin Ellis
Collaborators: Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Armando Solar-Lezama, Joshua B. Tenenbaum

2020

Conceptual Abstraction and Analogy in Natural and Artificial Intelligence

The premise of program induction

1. Represent knowledge as programs: as symbolic code

The premise of program induction

1. Represent knowledge as programs: as symbolic code

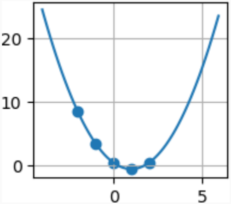
2. Learning=adding to that body of knowledge=
   making new programs=program synthesis

Why program induction?

Why program induction?

strong generalization

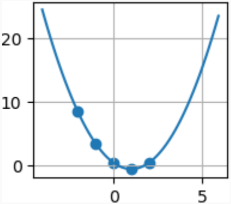
+data efficiency




```
f(x)=(x-1)**2 - 0.5
```

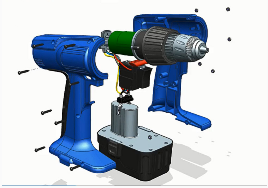
Why program induction?

strong generalization
+data efficiency

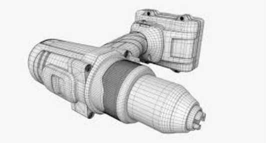


`f(x)=(x-1)**2 - 0.5`

compositional reuse
of abstractions

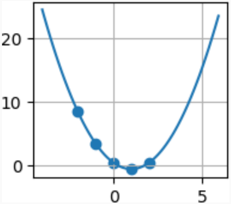


vs



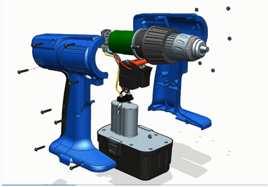
Why program induction?

strong generalization
+data efficiency

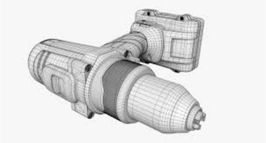


`f(x)=(x-1)**2 - 0.5`

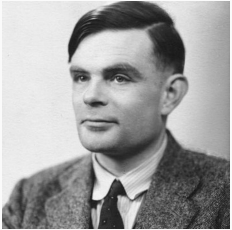
compositional reuse
of abstractions



vs

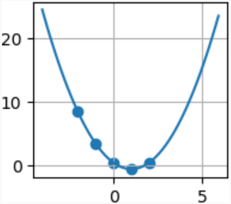


universal expressivity



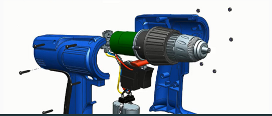
Why program induction?

strong generalization
+data efficiency



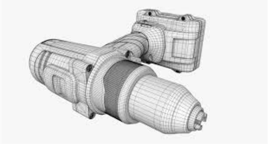
`f(x)=(x-1)**2 - 0.5`

compositional reuse
of abstractions

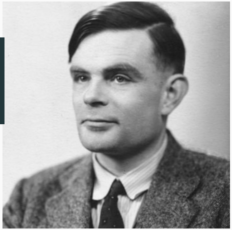


Challenge: Combinatorial search,
massive compute time

vs



universal expressivity



FlashFill (Gulwani 2012)

EXAMPLE 3 (Directory Name Extraction). *Consider the following example taken from an excel online help forum.*

Input v_1	*Output*
Company\Code\index.html	*Company\Code*
Company\Docs\Spec\specs.doc	*Company\Docs\Spec*

String Program:

$\mathtt{SubStr}(v_1, \mathtt{CPos}(0), \mathtt{Pos}(SlashTok, \epsilon, -1))$

FlashFill (Gulwani 2012)

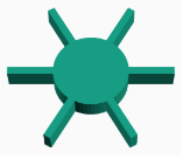
EXAMPLE 3 (Directory Name Extraction). *Consider the following example taken from an excel online help forum.*

Input v_1	*Output*
Company\Code\index.html	*Company\Code*
Company\Docs\Spec\specs.doc	*Company\Docs\Spec*

String Program:

$\mathtt{SubStr}(v_1, \mathtt{CPos}(0), \mathtt{Pos}(\mathit{SlashTok}, \epsilon, -1))$

Szalinski (Nandi 2020)



(a) CAD model of ship's wheel

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

(b) Caddy program

FlashFill (Gulwani 2012)

EXAMPLE 3 (Directory Name Extraction). *Consider the following example taken from an excel online help forum.*

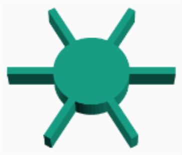
Input v_1	*Output*
Company\Code\index.html	*Company\Code*
Company\Docs\Spec\specs.doc	*Company\Docs\Spec*

String Program:

$\mathit{SubStr}(v_1, \mathit{CPos}(0), \mathit{Pos}(\mathit{SlashTok}, \epsilon, -1))$

$$
\begin{array}{rcl}
\text{String expr } P & := & \texttt{Switch}((\mathrm{b}_1, \mathrm{e}_1), \cdot\cdot, (\mathrm{b}_n, \mathrm{e}_n)) \\
\text{Bool b} & := & \mathrm{d}_1 \vee \cdot\cdot \vee \mathrm{d}_n \\
\text{Conjunct d} & := & \pi_1 \wedge \cdot\cdot \wedge \pi_n \\
\text{Predicate } \pi & := & \texttt{Match}(v_i, \mathrm{r}, k) \mid \neg\, \texttt{Match}(v_i, \mathrm{r}, k) \\
\text{Trace expr e} & := & \texttt{Concatenate}(\mathrm{f}_1, \cdot\cdot, \mathrm{f}_n) \\
\text{Atomic expr f} & := & \texttt{SubStr}(v_i, \mathrm{p}_1, \mathrm{p}_2) \\
 & \mid & \texttt{ConstStr}(s) \\
 & \mid & \texttt{Loop}(\lambda w : \mathrm{e}) \\
\text{Position p} & := & \texttt{CPos}(k) \mid \texttt{Pos}(\mathrm{r}_1, \mathrm{r}_2, \mathrm{c}) \\
\text{Integer expr c} & := & k \mid k_1 w + k_2 \\
\text{Regular Expression r} & := & \texttt{TokenSeq}(\mathrm{T}_1, \cdot\cdot, \mathrm{T}_m) \\
\text{Token T} & := & C+ \mid [\neg C]+ \\
 & \mid & \text{SpecialToken}
\end{array}
$$

Szalinski (Nandi 2020)



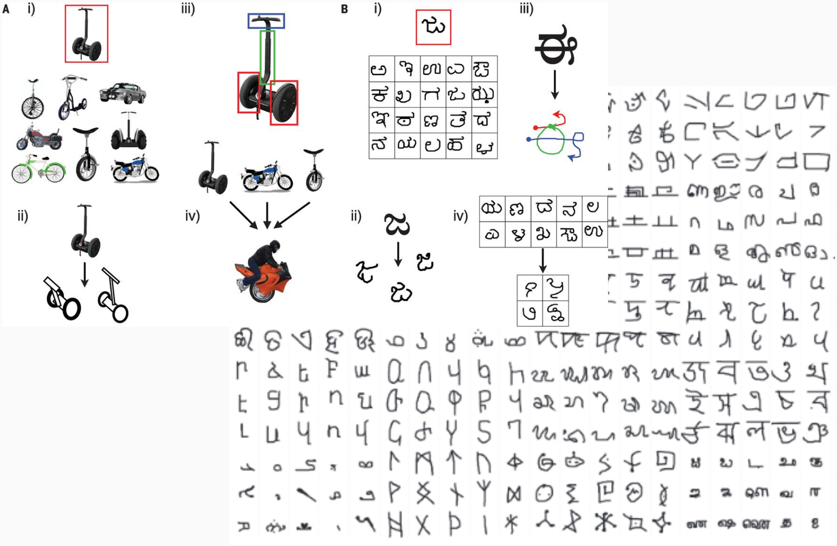
(a) CAD model of ship's wheel

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

(b) Caddy program

$$
\begin{array}{lll}
\text{op} & ::= & + \mid - \mid \times \mid / \qquad \text{num} ::= \mathbb{R} \mid \langle \mathit{var} \rangle \mid \langle \mathit{num} \rangle\, \langle \mathit{op} \rangle\, \langle \mathit{num} \rangle \\
\text{vec2} & ::= & [\langle \mathit{num} \rangle, \langle \mathit{num} \rangle] \qquad \text{vec3} ::= [\langle \mathit{num} \rangle, \langle \mathit{num} \rangle, \langle \mathit{num} \rangle] \\
\text{affine} & ::= & \text{Translate} \mid \text{Rotate} \mid \text{Scale} \mid \textbf{TranslateSpherical} \\
\text{binop} & ::= & \text{Union} \mid \text{Difference} \mid \text{Intersection} \\
\text{cad} & ::= & (\text{Cuboid } \langle \mathit{vec3} \rangle) \mid (\text{Sphere } \langle \mathit{num} \rangle) \\
 & \mid & (\text{Cylinder } \langle \mathit{vec2} \rangle) \mid (\text{HexPrism } \langle \mathit{vec2} \rangle) \mid \ldots \\
 & \mid & (\langle \mathit{affine} \rangle\ \langle \mathit{vec3} \rangle\ \langle \mathit{cad} \rangle) \\
 & \mid & (\langle \mathit{binop} \rangle\ \langle \mathit{cad} \rangle\ \langle \mathit{cad} \rangle) \\
 & \mid & (\textbf{Fold } \langle \mathit{binop} \rangle\ \langle \mathit{cad\text{-}list} \rangle) \\
\text{cad-list} & ::= & (\text{List } \langle \mathit{cad} \rangle +) \\
 & \mid & (\text{Concat } \langle \mathit{cad\text{-}list} \rangle +) \\
 & \mid & (\text{Tabulate } (\langle \mathit{var} \rangle\ \mathbb{Z}^+)+\ \langle \mathit{cad} \rangle) \\
 & \mid & (\text{Map2 } \langle \mathit{affine} \rangle\ \langle \mathit{vec3\text{-}list} \rangle\ \langle \mathit{cad\text{-}list} \rangle) \\
\text{vec3-list} & ::= & (\text{List } \langle \mathit{vec3} \rangle +) \\
 & \mid & (\text{Concat } \langle \mathit{vec3\text{-}list} \rangle +) \\
 & \mid & (\text{Tabulate } (\langle \mathit{var} \rangle\ \mathbb{Z}^+)+\ \langle \mathit{vec3} \rangle)
\end{array}
$$

Program induction for learning and perception



Lake et al 2015

Program Induction and learning to learn

learning a library

synergy between library & neural search

Learning to write code

Goal: acquire domain-specific knowledge needed to induce a class of programs

- Library of abstractions (domain specific language)
- Inference strategy (synthesis algorithm)

Library learning

Initial Primitives

⋮

`map`

`fold`

`if`

`cons`

`>`

⋮

Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Ellis, Morales, Sable-Meyer, Solar-Lezama, Tenenbaum. NeurIPS 2018.
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning

Initial Primitives

⋮

`map`

`fold`

`if`

`cons`

`>`

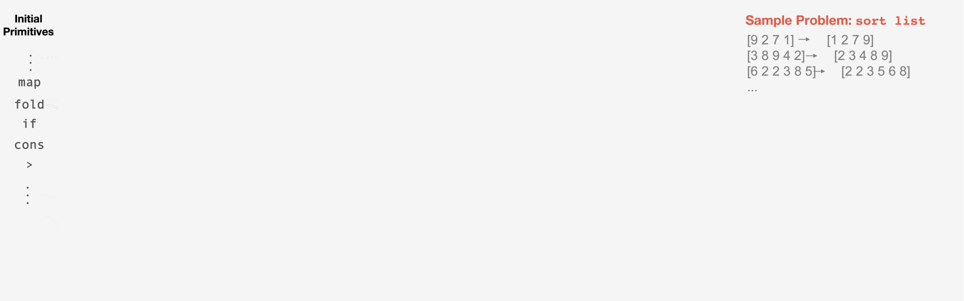
⋮

Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
…

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

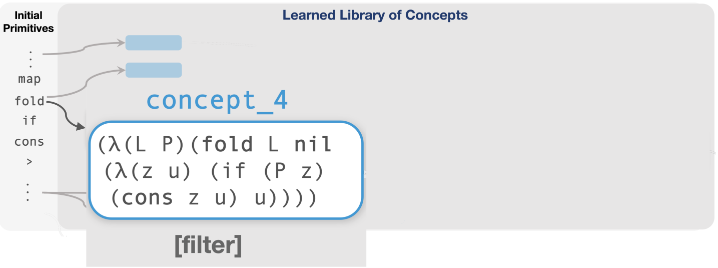
Library learning

Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

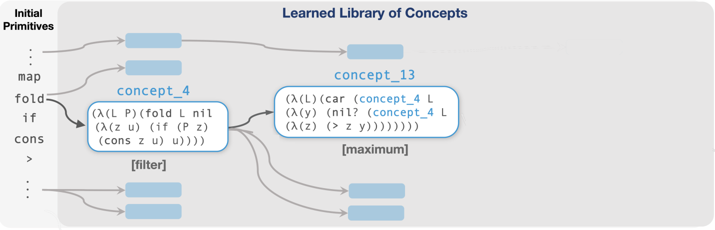
Library learning

Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning

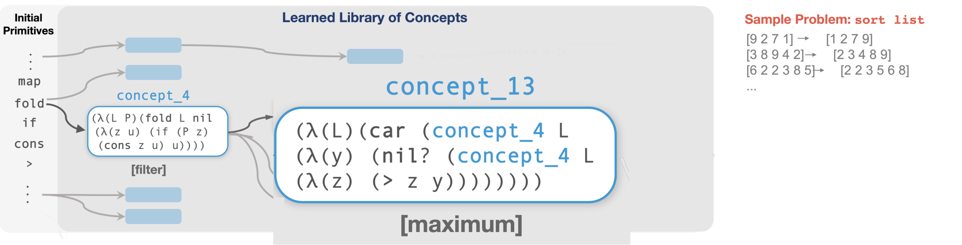


Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

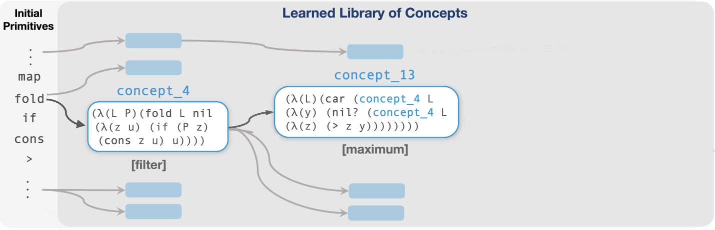
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning

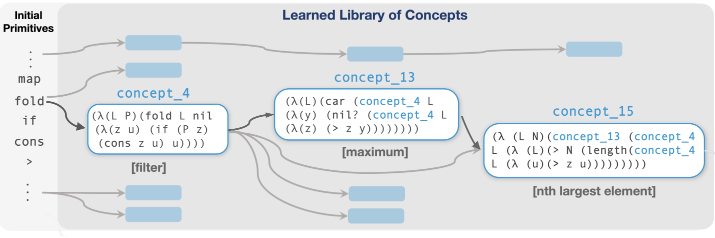


Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning

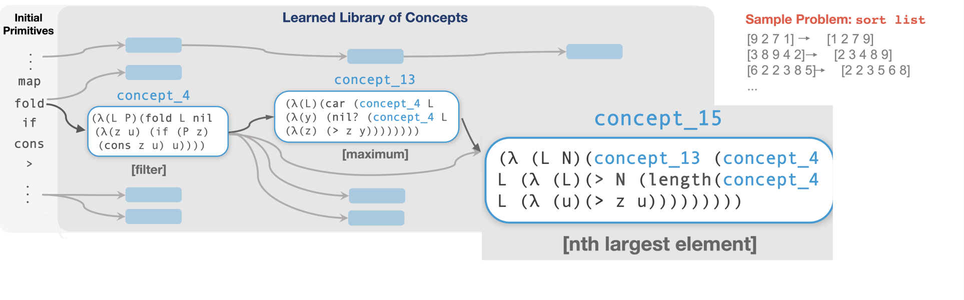


Sample Problem: `sort list`

[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

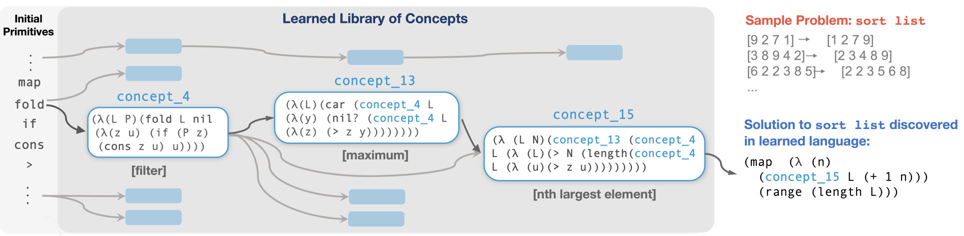
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



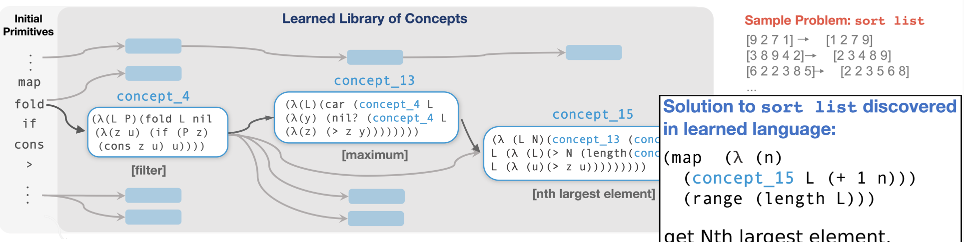
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



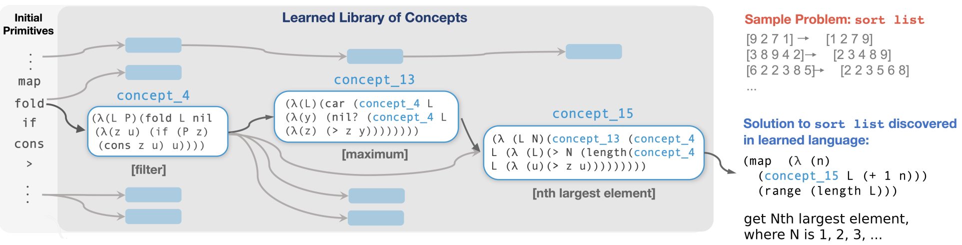
Solution to `sort list` discovered in learned language:

```
(map (λ (n)
  (concept_15 L (+ 1 n)))
  (range (length L)))
```

get Nth largest element, where N is 1, 2, 3, ...

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning

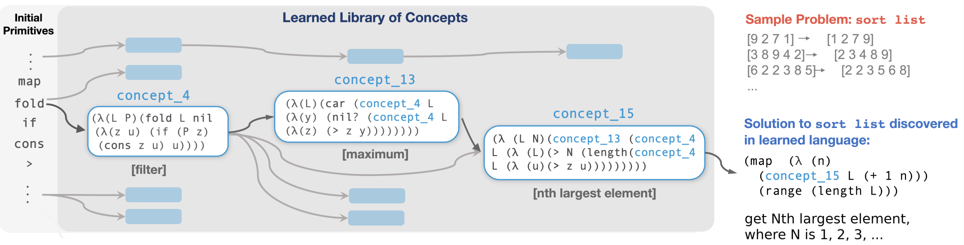


Solution rewritten in initial primitives:

```
(lambda (x) (map (lambda (y) (car (fold (fold x nil (lambda (z u) (if (gt? (+ y 1) (length
(fold x nil (lambda (v w) (if (gt? z v) (cons v w) w))))) (cons z u) u))) nil (lambda (a b) (if
(nil? (fold (fold x nil (lambda (c d) (if (gt? (+ y 1) (length (fold x nil (lambda (e f) (if
(gt? c e) (cons e f) f))))) (cons c d) d))) nil (lambda (g h) (if (gt? g a) (cons g h) h))))
(cons a b) b))))) (range (length x))))
```

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library learning



Solution rewritten in initial primitives:

```
(lambda (x) (map (lambda (y) (car (fold (fold x nil (lambda (z u) (if (gt? (+ y 1) (length
(fold x nil (lambda (v w) (if (gt? z v) (cons v w) w))))) (cons z u) u))) nil (lambda (a b) (if
(nil? (fold (fold x nil (lambda (c d) (if (gt? (+ y 1) (length (fold x nil (lambda (e f) (if
(gt? c e) (cons e f) f))))) (cons c d) d))) nil (lambda (g h) (if (gt? g a) (cons g h) h))))
(cons a b) b))))) (range (length x))))
```

induced sort program found in $\leq$ 10min. Brute-force search without learned library would take $\approx 10^{73}$ years

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

DreamCoder

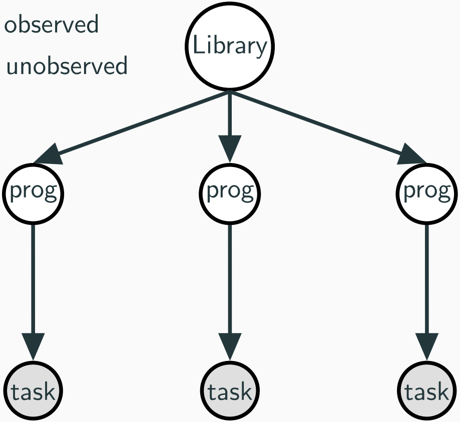
- **Wake:** Solve problems by writing programs
- **Sleep:** Improve library and neural recognition model:
  - **Abstraction sleep:** Improve library
  - **Dream sleep:** Improve neural recognition model

cf. Helmholtz machine, wake/sleep neural network training algorithms

Library learning as Bayesian inference

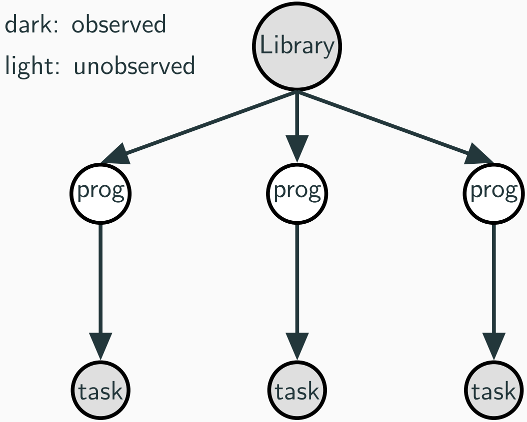
dark: observed

light: unobserved



[Dechter et al, 2013] [Liang et al, 2010] [Lake et al, 2015]

Library learning as Bayesian inference

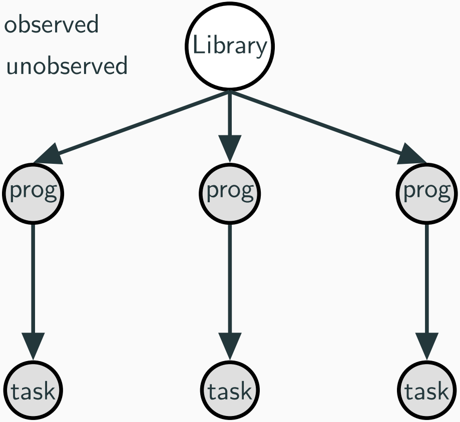


[Dechter et al, 2013] [Liang et al, 2010] [Lake et al, 2015]

Library learning as Bayesian inference

dark: observed

light: unobserved

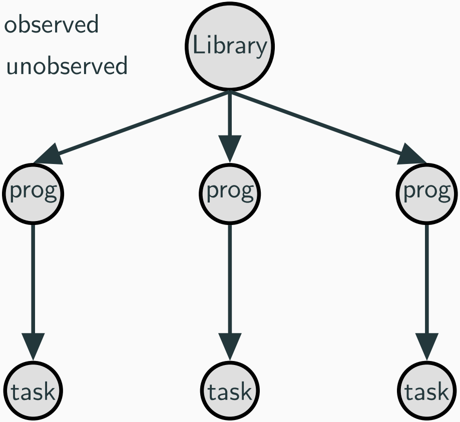


[Dechter et al, 2013] [Liang et al, 2010] [Lake et al, 2015]

Library learning as Bayesian inference

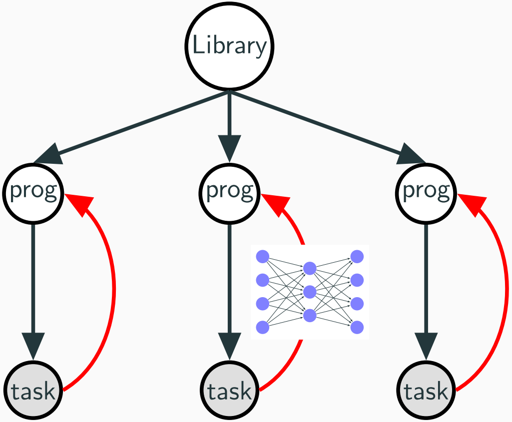
dark: observed

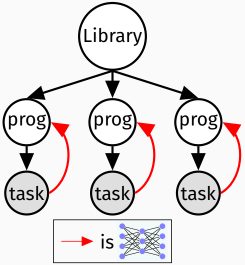
light: unobserved



[Dechter et al, 2013] [Liang et al, 2010] [Lake et al, 2015]

Library learning as neurally-guided Bayesian inference

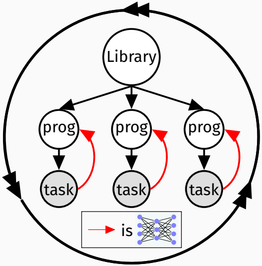


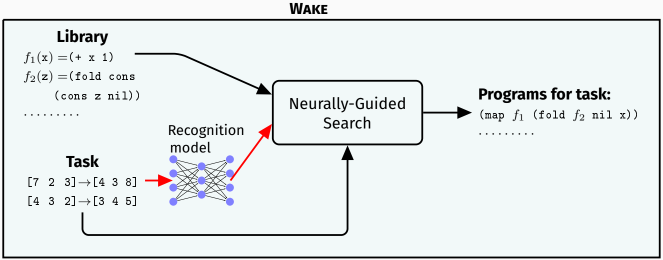

Library
prog
prog
prog
task
task
task
is

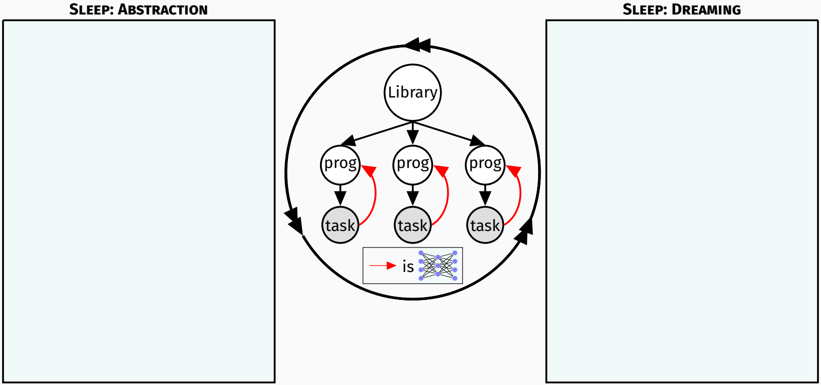
Wake

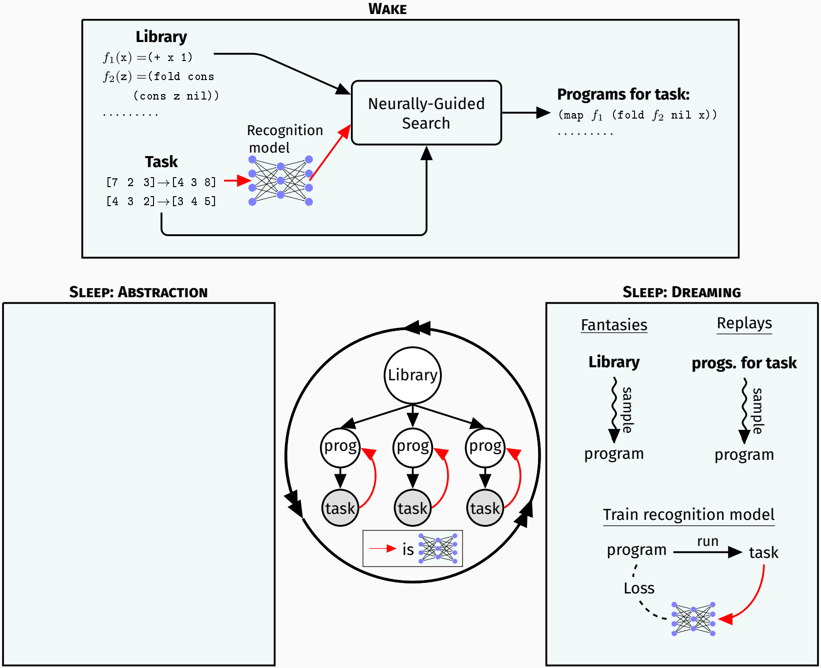
Sleep: Abstraction

Sleep: Dreaming

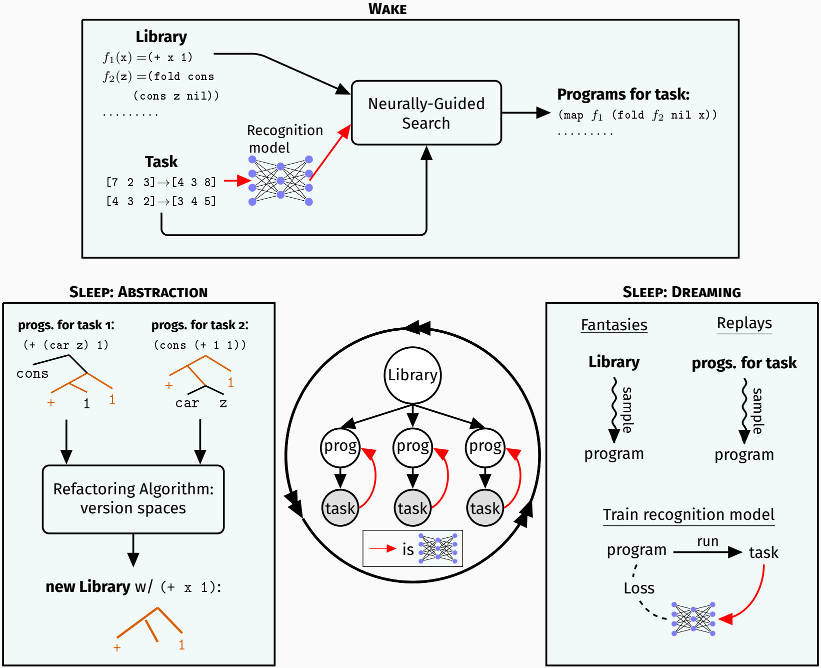


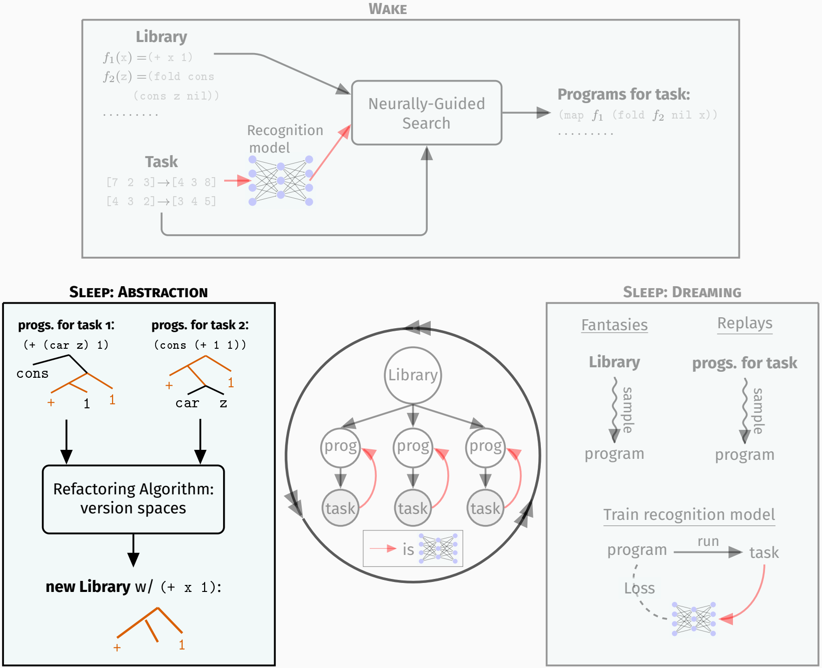

Wake
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
....
Recognition
model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided
Search
Programs for task:
(map f_1 (fold f_2 nil x))
....


Sleep: Abstraction
Sleep: Dreaming
Library
prog
prog
prog
task
task
task
is



WAKE
Library
f1(x) =(+ x 1)
f2(z) =(fold cons
(cons z nil))
....
Recognition model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided Search
Programs for task:
(map f1 (fold f2 nil x))
....
SLEEP: ABSTRACTION
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
sample
program
progs. for task
sample
program
Train recognition model
program
run
task
Loss


WAKE
Library
f1(x) =(+ x 1)
f2(z) =(fold cons (cons z nil))
....
Recognition model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided Search
Programs for task:
(map f1 (fold f2 nil x))
....
SLEEP: ABSTRACTION
progs. for task 1:
(+ (car z) 1)
cons
+
1
1
progs. for task 2:
(cons (+ 1 1))
+
1
car
z
Refactoring Algorithm: version spaces
new Library w/ (+ x 1):
+
1
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
sample
program
progs. for task
sample
program
Train recognition model
program
run
task
Loss


Wake
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons (cons z nil))
Recognition model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
Sleep: Abstraction
progs. for task 1:
(+ (car z) 1)
cons
+
1
1
progs. for task 2:
(cons (+ 1 1))
+
1
car
z
Refactoring Algorithm: version spaces
new Library w/ (+ x 1):
+
1
Library
prog
prog
prog
task
task
task
is
Sleep: Dreaming
Fantasies
Replays
Library
sample
program
progs. for task
sample
program
Train recognition model
program
run
task
Loss

Abstraction as analogy — from François Chollet, yesterday

Program-centric abstraction

- Graph of (usually discrete) operators where input nodes can take different values within a type
  - Example: function that sorts a list
- Abstract wrt input nodes values
- Obtained by merging specialized functions under a new abstract signature
  - This is a program analogy!

```
a = [4, 5, 2, 6]
result = 0
for i, e in enumerate(a):
  result += i * e


b = [6, 3, 4, 7]
result = 0
for i, e in enumerate(b):
  result += i * e
```

```
def process_item(x):
  result = 0
  for i, e in enumerate(x):
    result += i * e
  return result
```

Abstraction Sleep: Growing the library via refactoring

Task:	`[1 2 3]`$\to$`[2 4 6]`
	`[4 3 4]`$\to$`[8 6 8]`

Task:	`[1 2 3]`$\to$`[0 1 2]`
	`[4 3 4]`$\to$`[3 2 3]`

Abstraction Sleep: Growing the library via refactoring

Task: [1 2 3]→[2 4 6]
[4 3 4]→[8 6 8]

Wake: program search

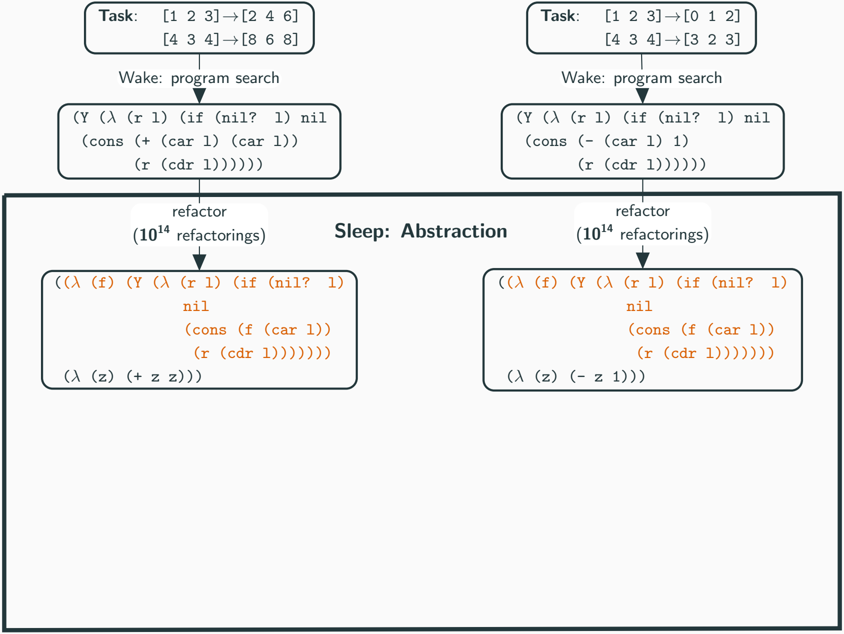
```
(Y (λ (r l) (if (nil? l) nil
 (cons (+ (car l) (car l))
       (r (cdr l))))))
```

Task: [1 2 3]→[0 1 2]
[4 3 4]→[3 2 3]

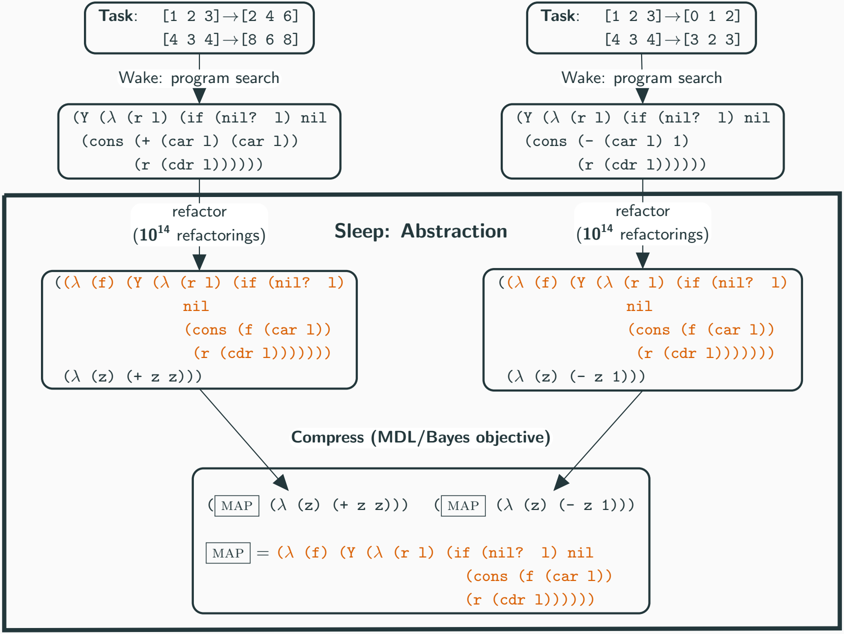
Wake: program search

```
(Y (λ (r l) (if (nil? l) nil
 (cons (- (car l) 1)
       (r (cdr l))))))
```

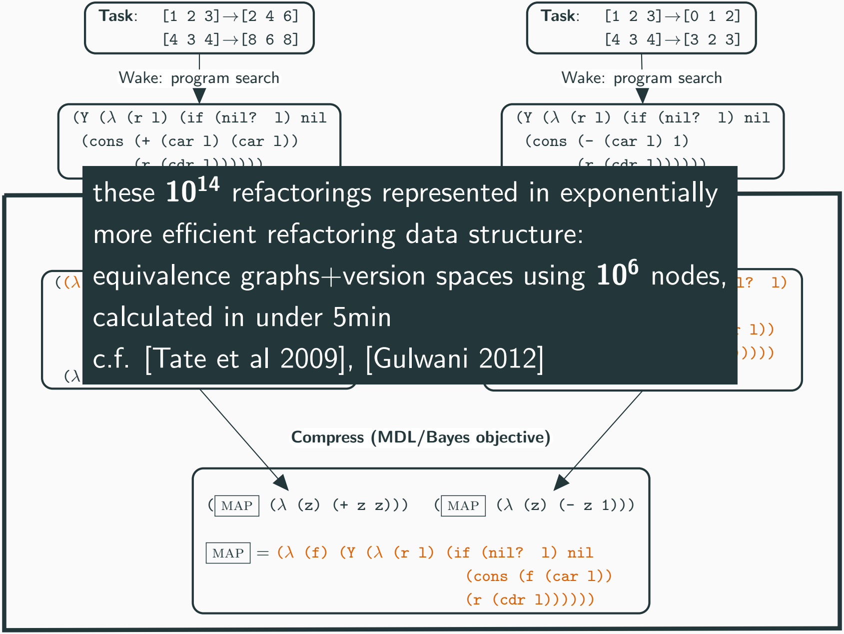
Abstraction Sleep: Growing the library via refactoring



Abstraction Sleep: Growing the library via refactoring



Abstraction Sleep: Growing the library via refactoring



these 10^{14} refactorings represented in exponentially more efficient refactoring data structure:

equivalence graphs+version spaces using 10^6 nodes, calculated in under 5min

c.f. [Tate et al 2009], [Gulwani 2012]

Program Induction and learning to learn

learning a library

synergy between library & neural search

DreamCoder Domains

List Processing

Sum List

`[1 2 3] → 6`
`[4 6 8 1] → 17`

Double

`[1 2 3] → [2 4 6]`
`[4 5 1] → [8 10 2]`

Text Editing

Abbreviate

`Allen Newell → A.N.`
`Herb Simon → H.S.`

Drop Last Three

`shrdlu → shr`
`shakey → sha`

Regexes

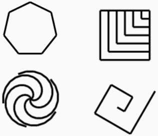
Phone numbers

`(555) 867-5309`
`(650) 555-2368`

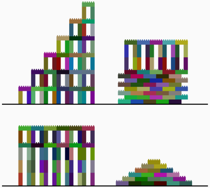
Currency

`$100.25`
`$4.50`

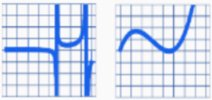
LOGO Graphics



Block Towers



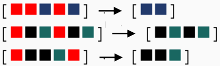
Symbolic Regression



$y = f(x)$

Recursive Programming

Filter Red



Physical Laws

$$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$$

$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r}|^2} \hat{r}$$

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

DreamCoder Domains

List Processing

Sum List

```
[1 2 3] → 6
[4 6 8 1] → 17
```

Double

```
[1 2 3] → [2 4 6]
[4 5 1] → [8 10 2]
```

Text Editing

Abbreviate

```
Allen Newell → A.N.
Herb Simon → H.S.
```

Drop Last Three

```
shrdlu → shr
shakey → sha
```

Regexes

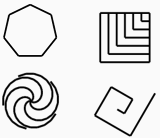
Phone numbers

```
(555) 867-5309
(650) 555-2368
```

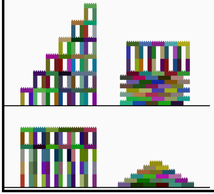
Currency

```
$100.25
$4.50
```

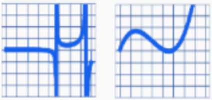
LOGO Graphics



Block Towers



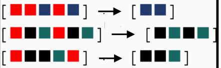
Symbolic Regression



$y = f(x)$

Recursive Programming

Filter Red



Physical Laws

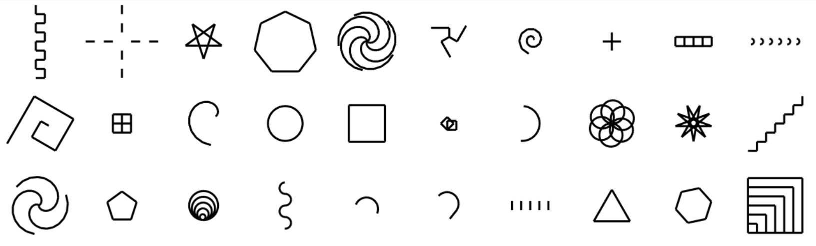
$$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$$

$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r}|^2} \hat{r}$$

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics

30 out of 160 tasks



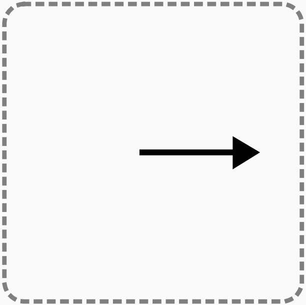
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics

DSL
`OP ::= FW x | RT x | UP | DOWN | SET state`

Tasks
`task :  image`

```
FW 1
```



LOGO Turtle Graphics

DSL

```
OP ::= FW x | RT x | UP | DOWN | SET state
```

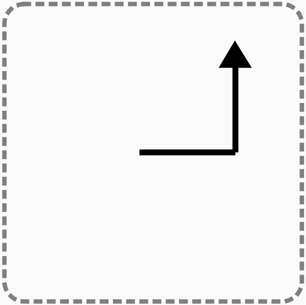
Tasks

```
task :  image
```

FW 1

RT $\frac{\pi}{2}$

FW 1



LOGO Turtle Graphics

DSL

```
OP ::= FW x | RT x | UP | DOWN | SET state
```

Tasks

```
task :  image
```

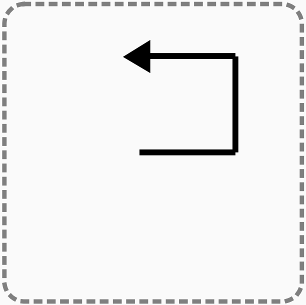
FW 1

RT $\frac{\pi}{2}$

FW 1

RT $\frac{\pi}{2}$

FW 1



LOGO Turtle Graphics

DSL

```
OP ::= FW x | RT x | UP | DOWN | SET state
```

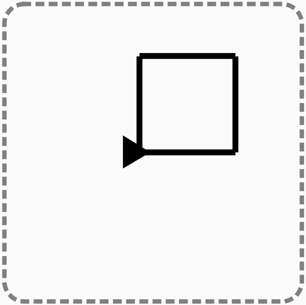
Tasks

```
task :  image
```

for i in range(4)

> FW 1

> RT $\frac{\pi}{2}$



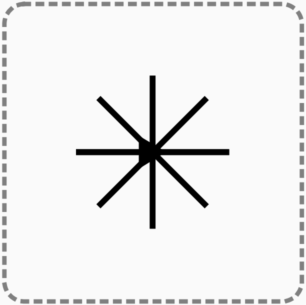
DSL

```
OP ::= FW x | RT x | UP | DOWN | SET state
```

Tasks

```
task :  image
```

```
for i in range(8)
> FW 1
> SET origin
> RT 2π/8
```



DSL

```
OP ::= FW x | RT x | UP | DOWN | SET state
```

Tasks

```
task :  image
```

for i in range(8)

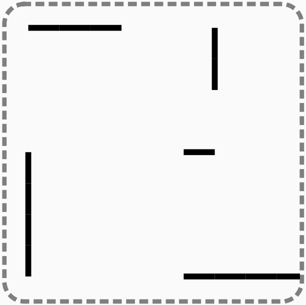
> PU

> FW $\frac{i}{2}$

> PD

> FW $\frac{i}{2}$

> RT $\frac{\pi}{2}$



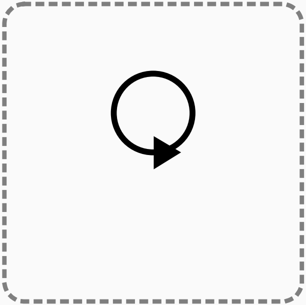
DSL

`OP ::= FW x | RT x | UP | DOWN | SET state`

Tasks

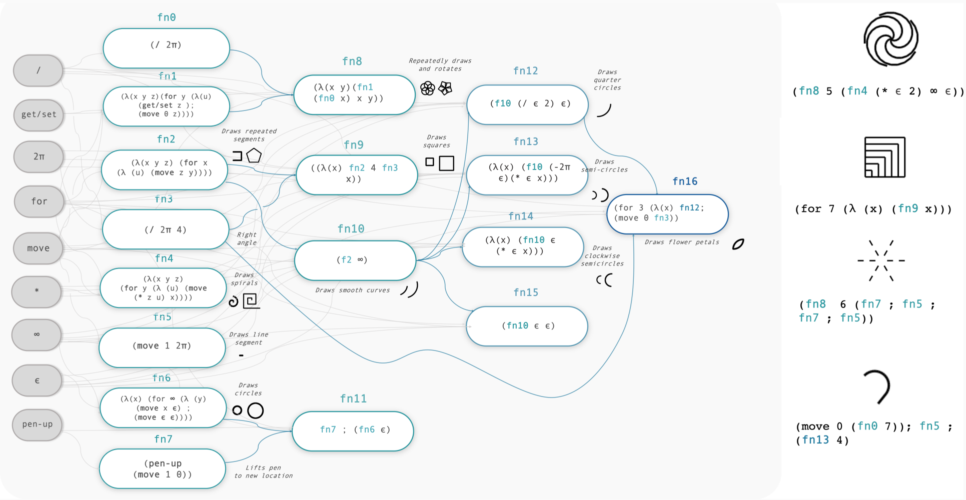
`task :  image`

```
for i in range(∞)
> FW ε
> RT ε
```



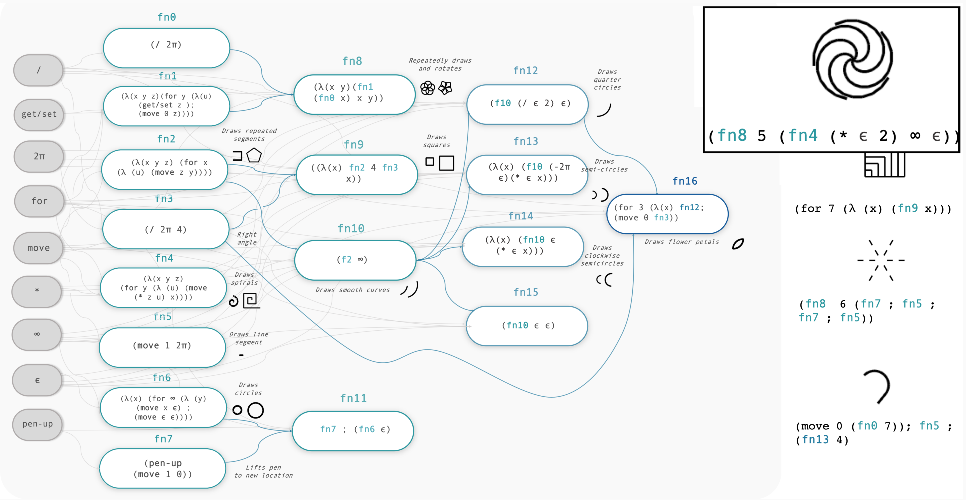
NUM ::= 1 | π | ∞ | ε | + | - | * | /

LOGO Turtle Graphics – learning an interpretable library



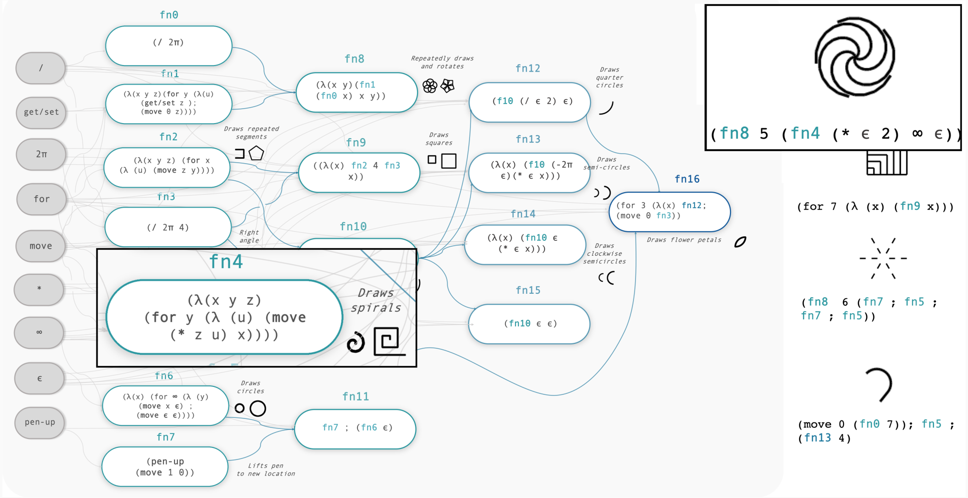
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics – learning an interpretable library



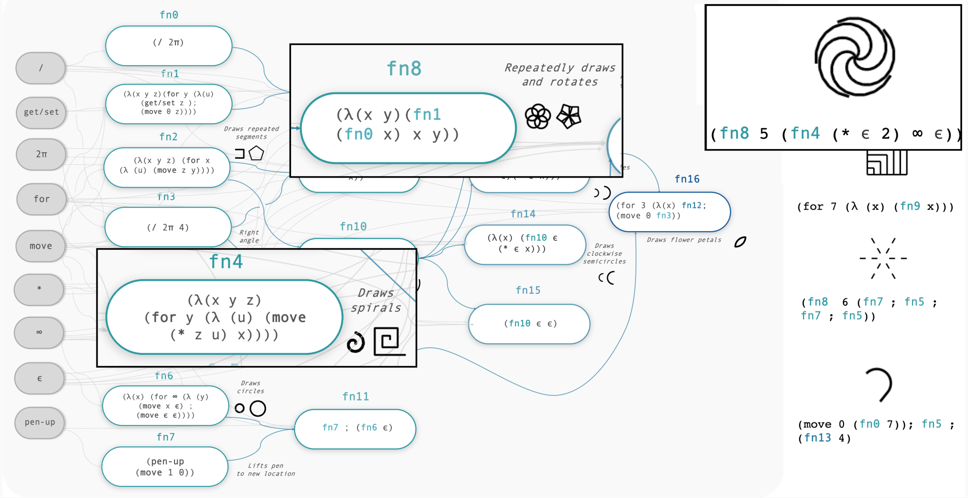
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics – learning an interpretable library



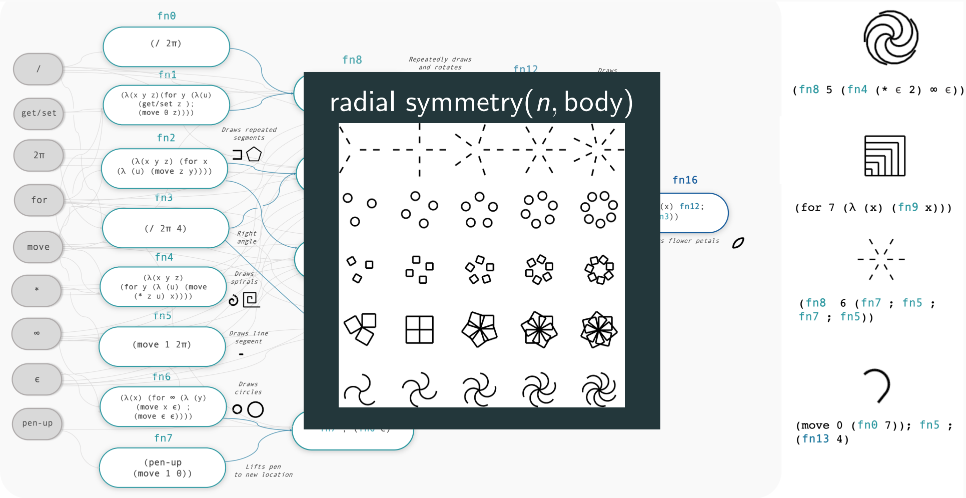
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics – learning an interpretable library



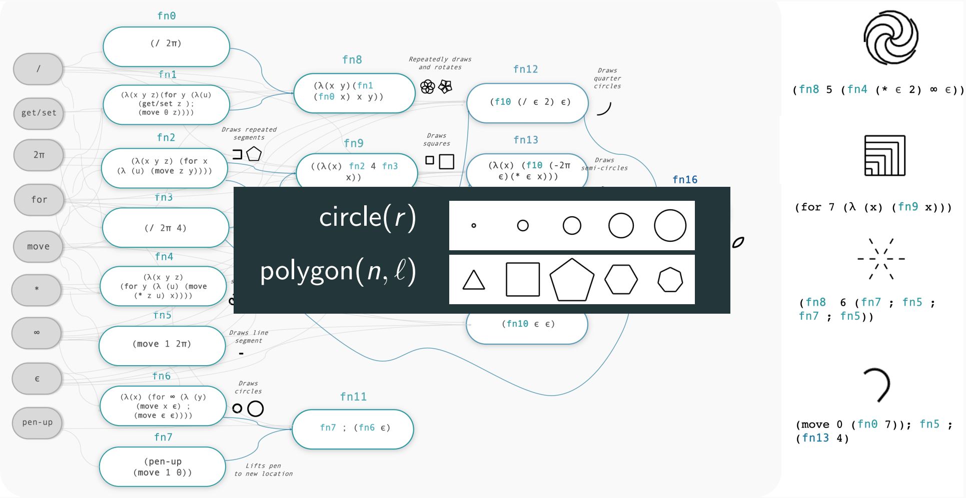
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics – learning an interpretable library



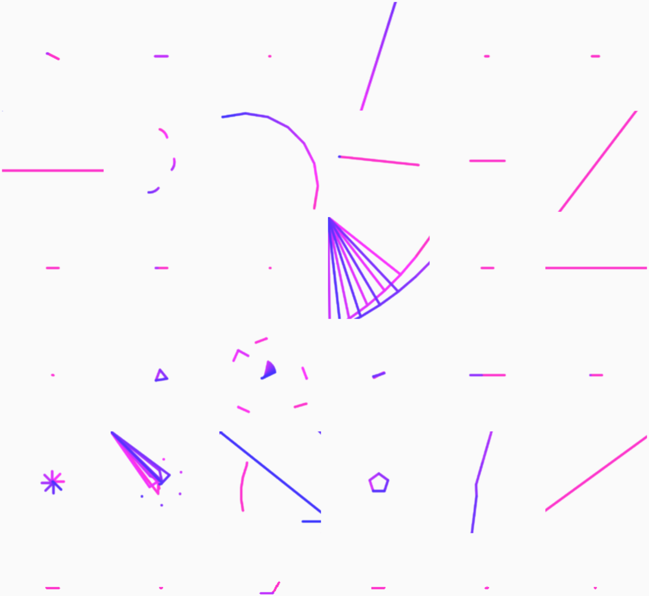
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

LOGO Turtle Graphics – learning an interpretable library

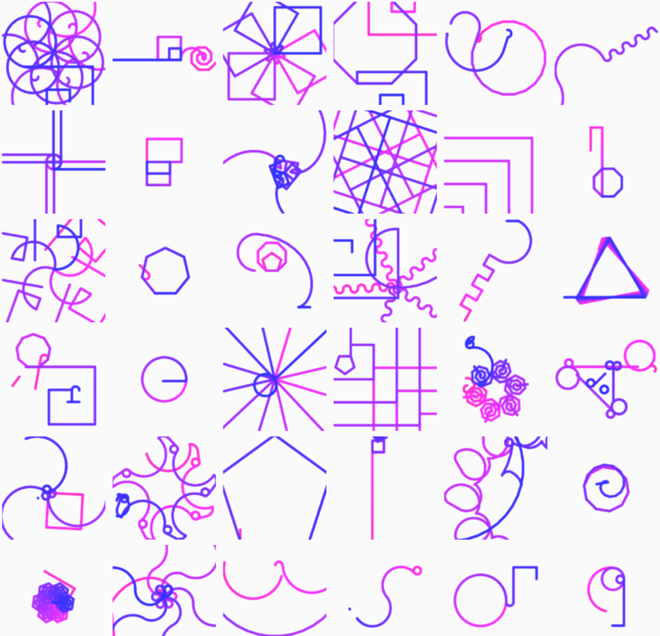


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

What does DreamCoder dream of? (before learning)

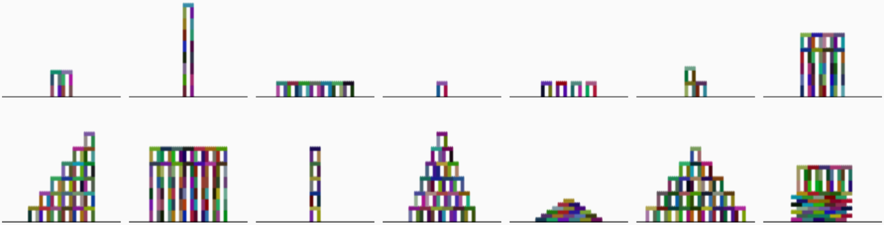


What does DreamCoder dream of? (after learning)



Planning to build towers

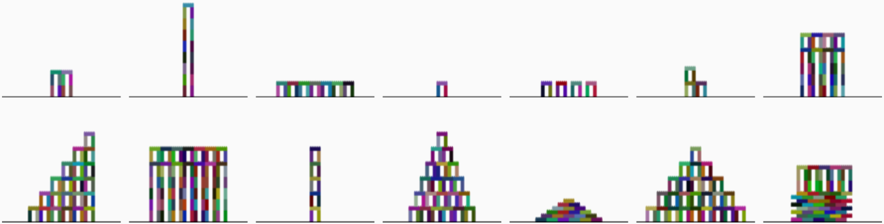
example tasks (112 total)



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Planning to build towers

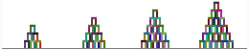
example tasks (112 total)



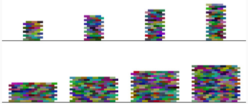
learned library routines (≈ 20 total)

arch(h)

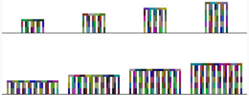
pyramid(h)



wall(w, h)

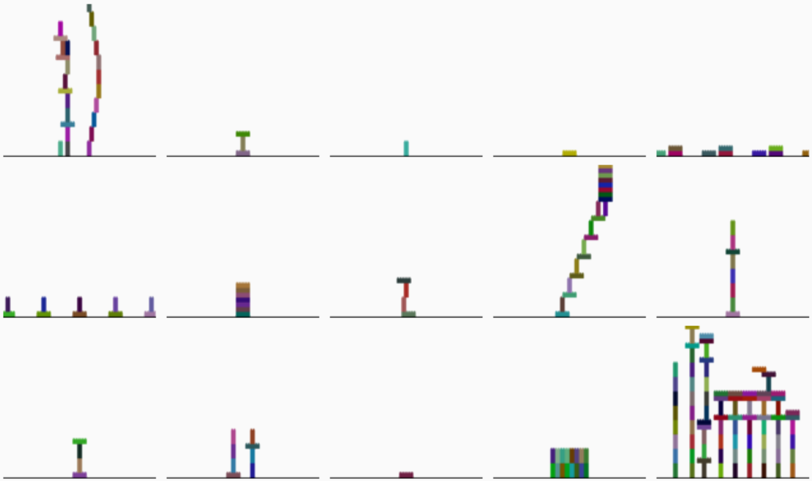


bridge(w, h)

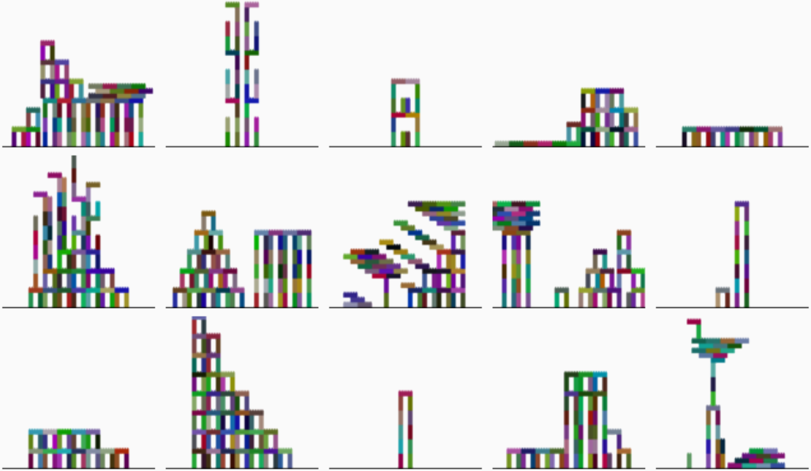


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

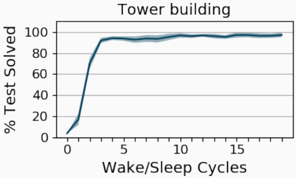
Dreams before learning



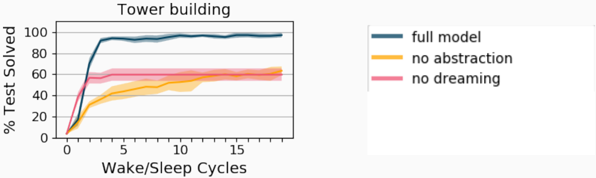
Dreams after learning



Learning dynamics

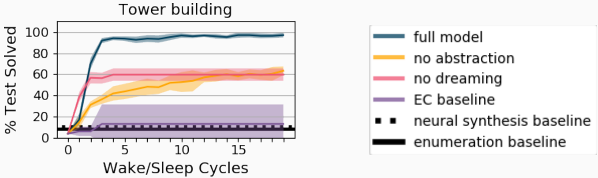


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

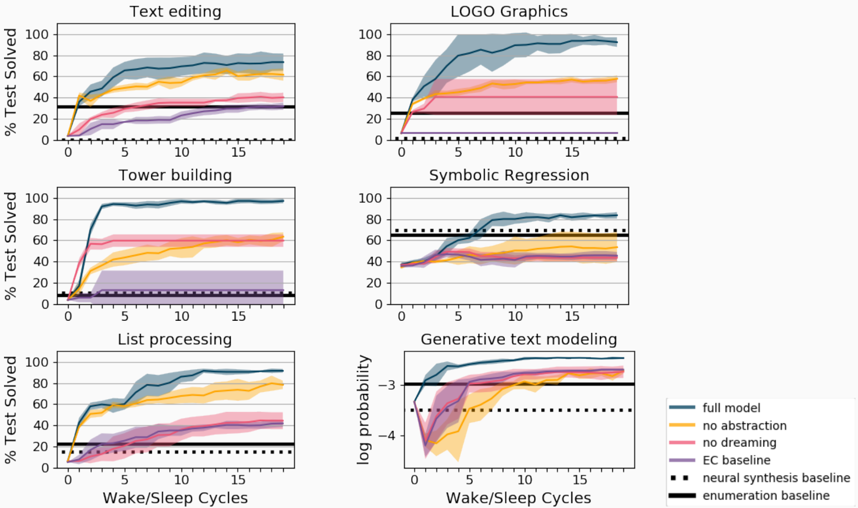
Learning dynamics



baselines: Exploration-Compression, EC [Dechter et al. 2013]
neural program synthesis, RobustFill [Devlin et al. 2017]
24 hours of brute-force enumeration

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Learning dynamics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

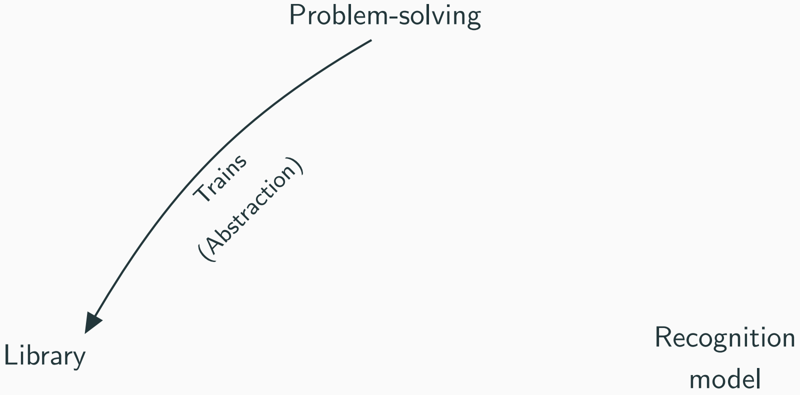
Synergy between recognition model and library learning

Problem-solving

Library

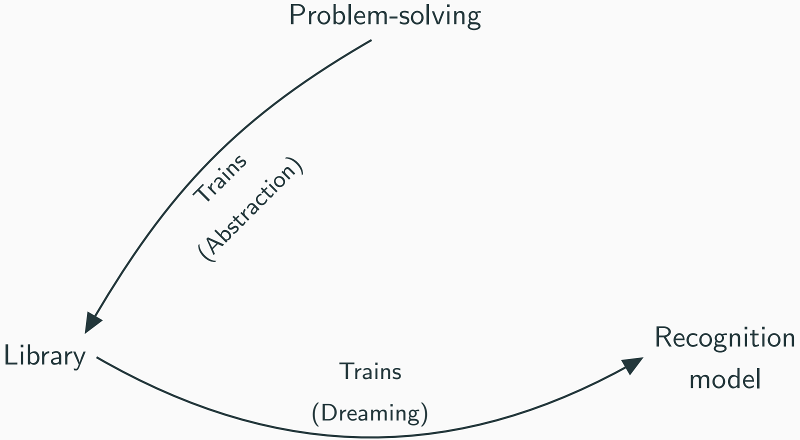
Recognition model

Synergy between recognition model and library learning



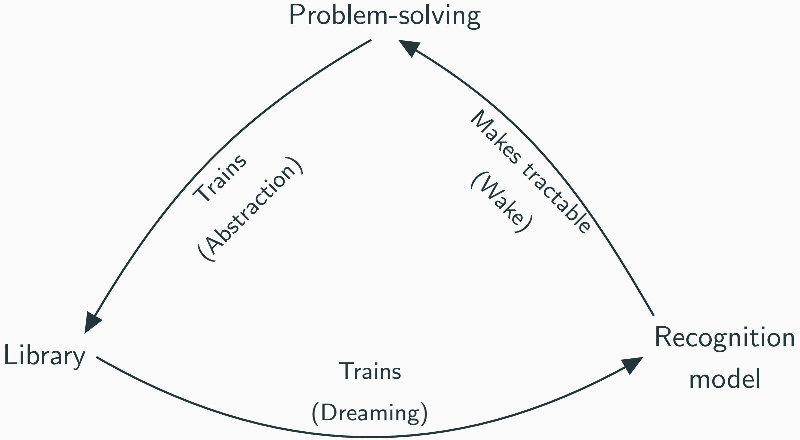
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Synergy between recognition model and library learning



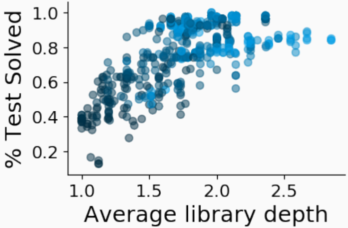
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Synergy between recognition model and library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Evidence for dreaming bootstrapping better libraries

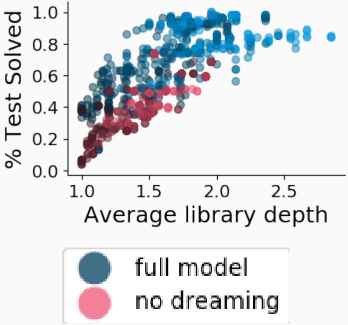


Darker: Early in learning

Brighter: Later in learning

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Evidence for dreaming bootstrapping better libraries



Darker: Early in learning

Brighter: Later in learning

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

From learning libraries,
to learning languages

From learning libraries, to learning languages

modern functional programming $\rightarrow$ physics

From learning libraries, to learning languages

1950's Lisp $\rightarrow$ modern functional programming $\rightarrow$ physics

Physics Formula Sheet

Mechanics

$x = x_0 + v_{x0}t + \frac{1}{2}a_x t^2$	$a_c = \frac{v^2}{r}$	$\lvert\vec{F}_{spring}\rvert = k\lvert\vec{x}\rvert$
$v = v_0 + at$	$\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$	$PE_{spring} = \frac{1}{2}kx^2$
$v_x^2 - v_{x0}^2 = 2a(x - x_0)$	$\omega = \omega_0 + \alpha t$	$T_{spring} = 2\pi\sqrt{\frac{m}{k}}$
$\vec{a} = \frac{\Sigma\vec{F}}{m} = \frac{\vec{F}_{net}}{m}$	$T = \frac{2\pi}{\omega} = \frac{1}{f}$	$T_{pendulum} = 2\pi\sqrt{\frac{\ell}{g}}$
$\lvert\vec{F}_{friction}\rvert \leq \mu\lvert\vec{F}_{Normal}\rvert$	$v = f\lambda$	
$\vec{p} = m\vec{v}$	$x = A\cos(2\pi f t)$	$\lvert\vec{F}_{gravity}\rvert = G\frac{m_1 m_2}{r^2}$
$\Delta\vec{p} = \vec{F}\Delta t$	$\vec{\alpha} = \frac{\Sigma\vec{\tau}}{I} = \frac{\vec{\tau}_{net}}{I}$	$\lvert\vec{F}_{gravity}\rvert = m\vec{g}$
$KE = \frac{1}{2}mv^2$	$\vec{\tau} = r \times F$	$PE_{gravity} = -G\frac{m_1 m_2}{r}$
$\Delta PE = mg\Delta y$	$L = I\omega$	$\rho = \frac{m}{V}$
$\Delta E = W = Fd\cos\theta$	$\Delta L = \tau\Delta t$	$KE = \frac{1}{2}I\omega^2$

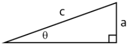
Electricity

$\lvert\vec{F}_E\rvert = k\left\lvert\frac{q_1 q_2}{r^2}\right\rvert$	$\Delta V = IR$	$R = \frac{\rho\ell}{A}$
$I = \frac{\Delta q}{\Delta t}$	$P = I\Delta V$	
$R_{series} = R_1 + R_2 + \ldots + R_n$	$\frac{1}{R_{Parallel}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$	

Geometry

Rectangle	$A = bh$	Rectangular Solid	$V = \ell wh$	Triangle	$A = \frac{1}{2}bh$
Circle	$A = \pi r^2$ $C = 2\pi r$	Cylinder	$V = \pi r^2\ell$ $S = 2\pi r\ell + 2\pi r^2$	Sphere	$V = \frac{4}{3}\pi r^3$ $S = 4\pi r^2$

Trigonometry



$c^2 = a^2 + b^2$ $\quad \sin\theta = \frac{a}{c}$ $\quad \cos\theta = \frac{b}{c}$ $\quad \tan\theta = \frac{a}{b}$

Variables

a = acceleration
A = amplitude
A = Area
b = base length
C = circumference
d = distance
E = energy
f = frequency
F = force
h = height
I = current
I = rotational inertia
KE = kinetic energy
k = spring constant
L = angular momentum
ℓ = length
m = mass
P = power
p = momentum
q = charge
r = radius
R = resistance
S = surface area
T = period
t = time
PE = potential energy
V = electric potential
V = volume
v = velocity
w = width
W = work
x = position
y = height
α = angular acceleration
λ = wavelength
μ = coefficient of friction

Growing languages for vector algebra and physics

Initial Primitives

map
zip
cons
empty
cdr
power
fold
car
+
-
*
/
0
1
π

Physics Equations

Newton's Second Law

$\vec{a} = \frac{1}{m}\sum_i \vec{F}_i$

Parallel Resistors

$R_{total} = \left(\Sigma_i \frac{1}{R_i}\right)^{-1}$

Work

$U = \vec{F} \cdot \vec{d}$

Force in a Magnetic Field

$|\vec{F}| = q|\vec{v} \times \vec{B}|$

Kinetic Energy

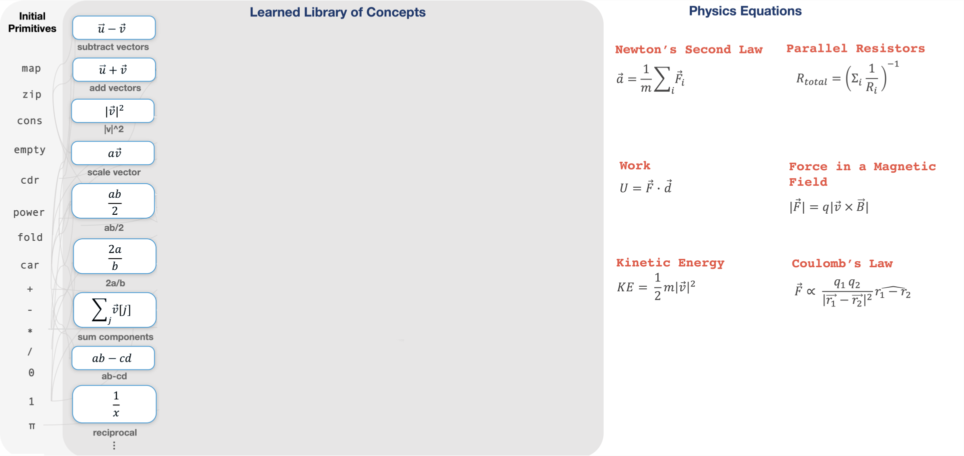
$KE = \frac{1}{2}m|\vec{v}|^2$

Coulomb's Law

$\vec{F} \propto \frac{q_1 q_2}{|\vec{r_1} - \vec{r_2}|^2}\widehat{r_1 - r_2}$

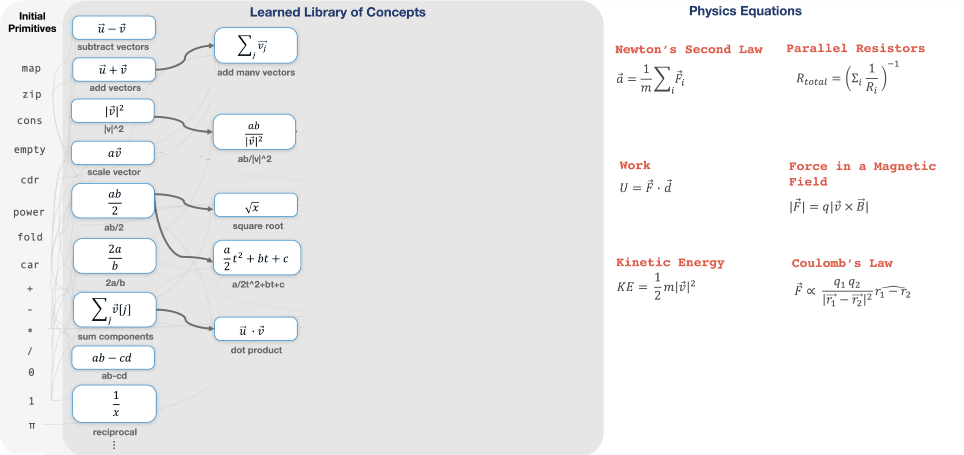
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing languages for vector algebra and physics



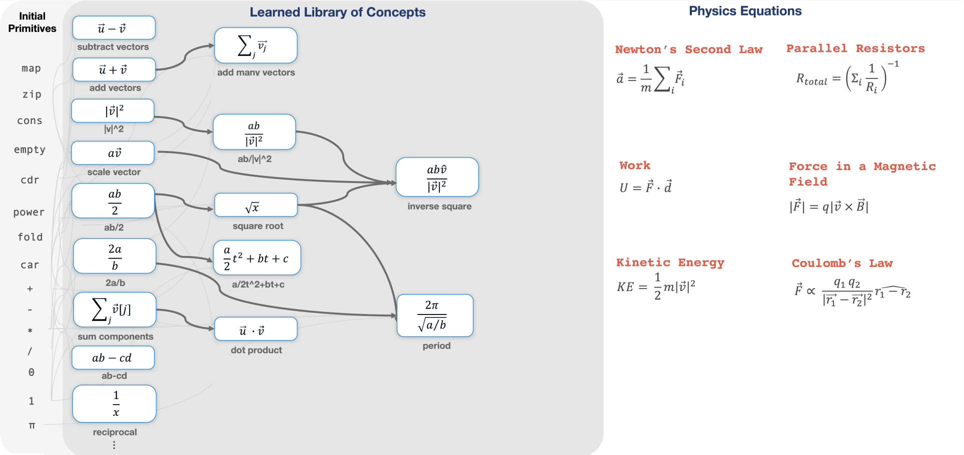
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing languages for vector algebra and physics



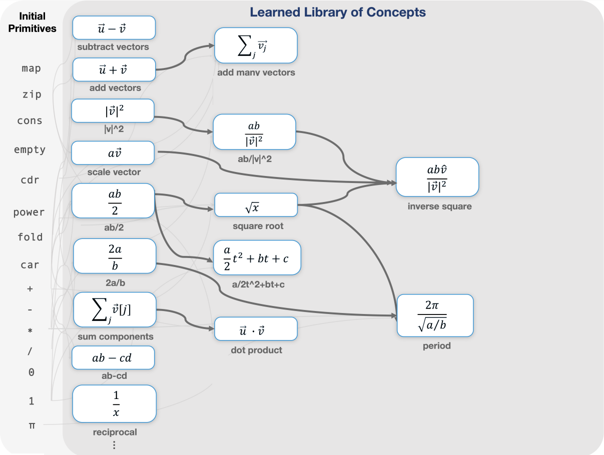
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing languages for vector algebra and physics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing languages for vector algebra and physics



Physics Equations

Newton's Second Law

$$\vec{a} = \frac{1}{m}\sum_i \vec{F_i}$$

```
(scale-vector(reciprocal m)
(add-many-vectors Fs))
```

Parallel Resistors

$$R_{total} = \left(\Sigma_i \frac{1}{R_i}\right)^{-1}$$

```
(reciprocal (sum-components
(map (λ(r) (reciprocal r))
Rs)))
```

Work

$$U = \vec{F} \cdot \vec{d}$$

```
(dot-product F d)
```

Force in a Magnetic Field

$$|\vec{F}| = q|\vec{v} \times \vec{B}|$$

```
(* q (ab-cd v_x b_y v_y b_x))
```

Kinetic Energy

$$KE = \frac{1}{2}m|\vec{v}|^2$$

```
(ab/2 m (|v|^2 v))
```

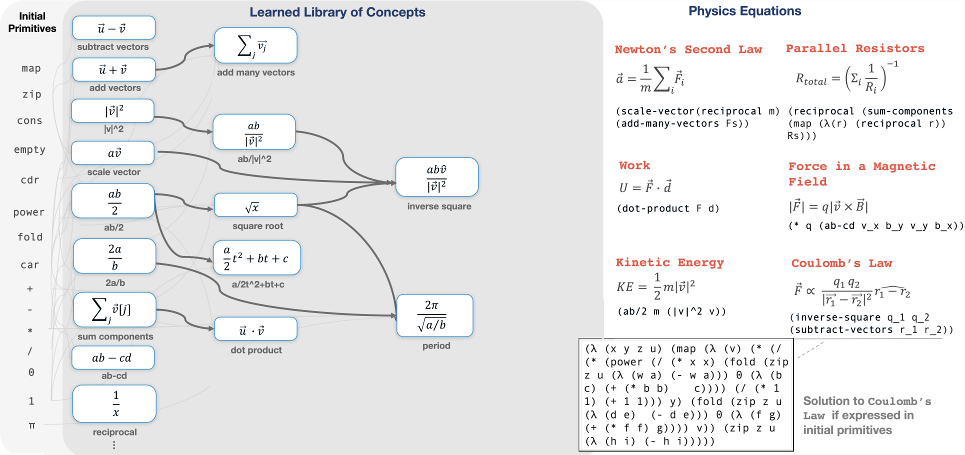
Coulomb's Law

$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r_1} - \vec{r_2}|^2}\widehat{r_1 - r_2}$$

```
(inverse-square q_1 q_2
(subtract-vectors r_1 r_2))
```

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing languages for vector algebra and physics



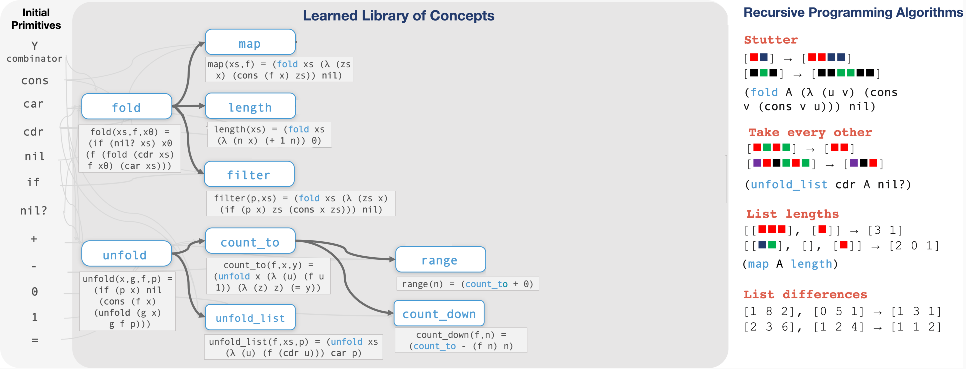
Growing a language for recursive programming

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing a language for recursive programming

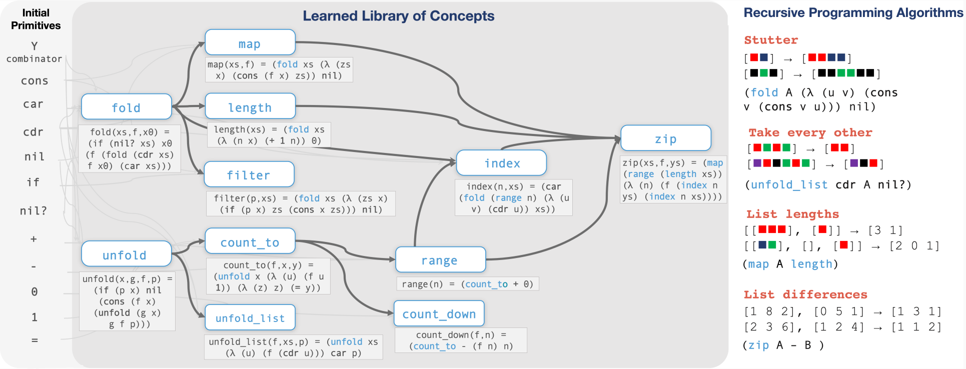
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing a language for recursive programming



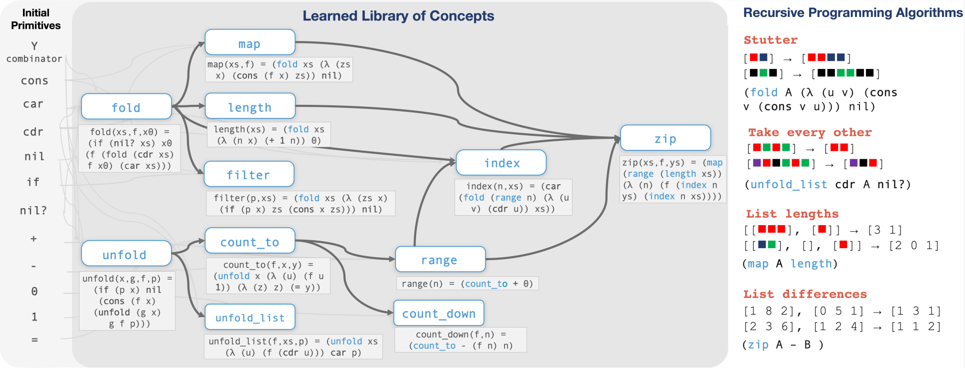
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

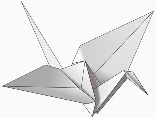
Growing a language for recursive programming



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing a language for recursive programming

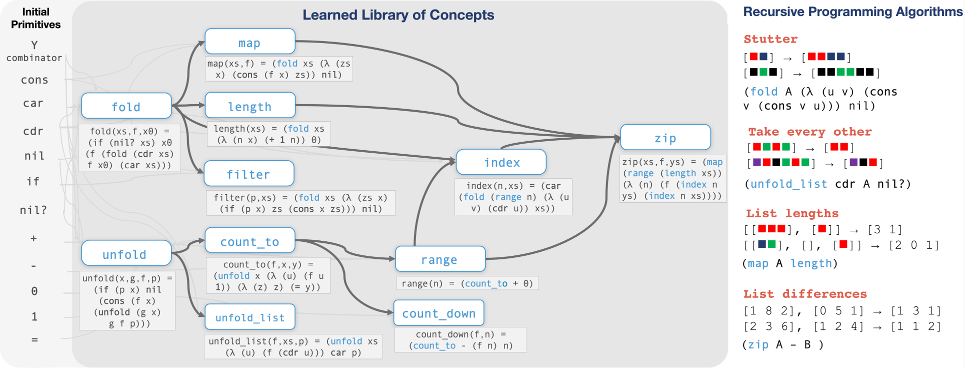




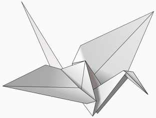
Origami Programming: Jeremy Gibbons, 2003

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Growing a language for recursive programming



1 year of compute. 5 days on 64 CPUs.



Origami Programming: Jeremy Gibbons, 2003

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Lessons

Growing a library of abstractions interacts synergistically with learning how to use those abstractions

Lessons

Growing a library of abstractions interacts synergistically with learning how to use those abstractions

Bayesian, symbolic learning algorithms mean you don't need massive data sets to learn libraries of abstractions

Lessons

Growing a library of abstractions interacts synergistically with learning how to use those abstractions

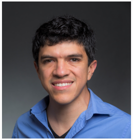
Bayesian, symbolic learning algorithms mean you don't need massive data sets to learn libraries of abstractions

Some abstractions can be bootstrapped without language—but surely natural language supervision would help...?

Collaborators

Josh Tenenbaum

Armando Solar-Lezama



Cathy Wong

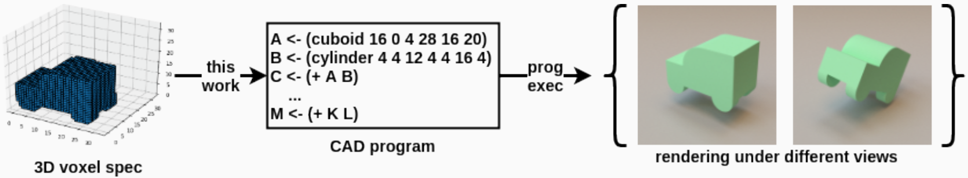
Max Nye

Mathias Sable-Meyer

Lucas Morales

thank you

3D program induction



Challenge: combinatorial search!

Branching factor: > 1.3 million per line of code, ≈ 20 lines of code

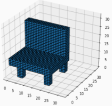
search space size: $(1.3 \text{ million})^{20} \approx 10^{122}$ programs

Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.

*equal contribution

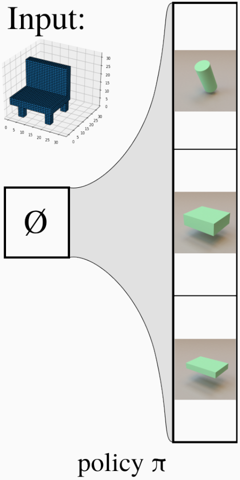
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]

Input:

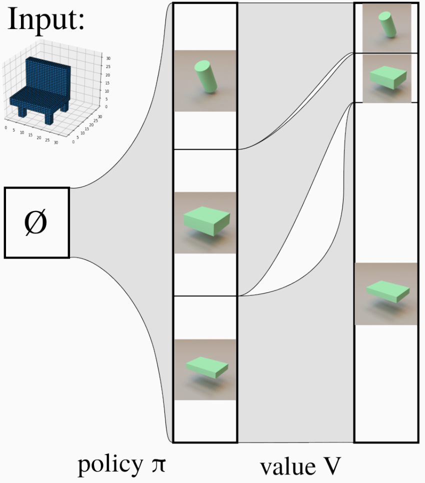


$\boxed{\emptyset}$

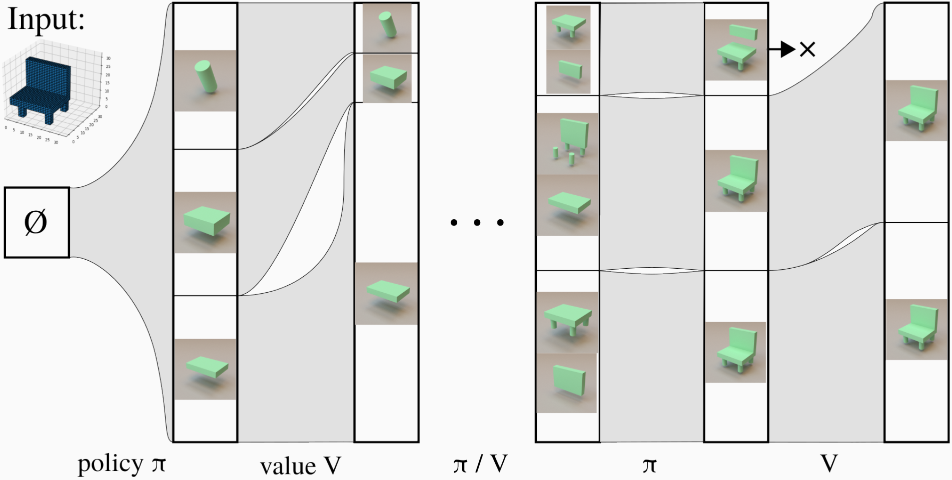
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]



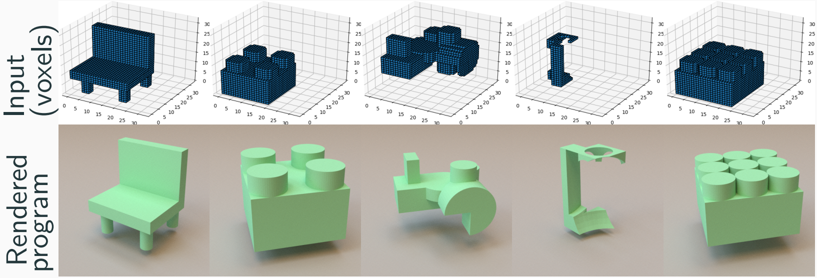
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]



Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]

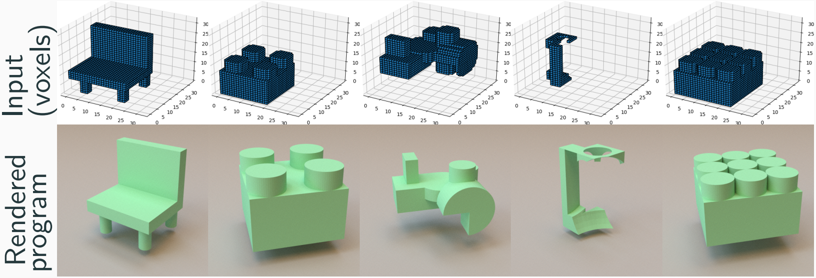


3D program induction



Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.
*equal contribution

3D program induction

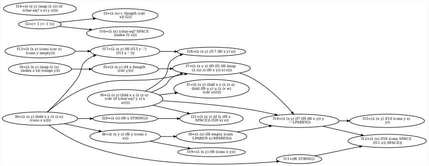


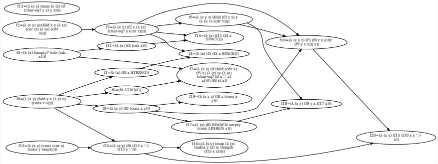
same architecture learns to synthesize text editing programs (FlashFill, Gulwani 2012)

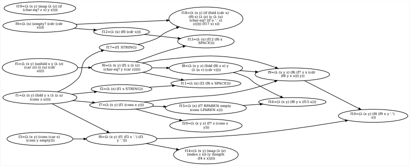
Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.
*equal contribution

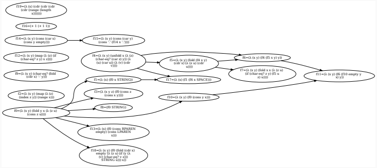
Library structure: Text Editing

DreamCoder learns libraries for FlashFill-style text editing [Gulwani 2012]





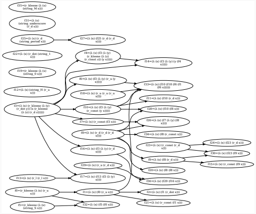


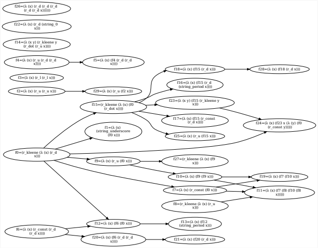


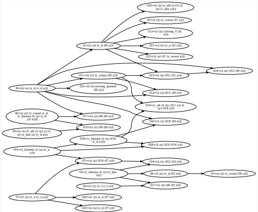
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. arxiv 2020.

Library structure: Generating Text

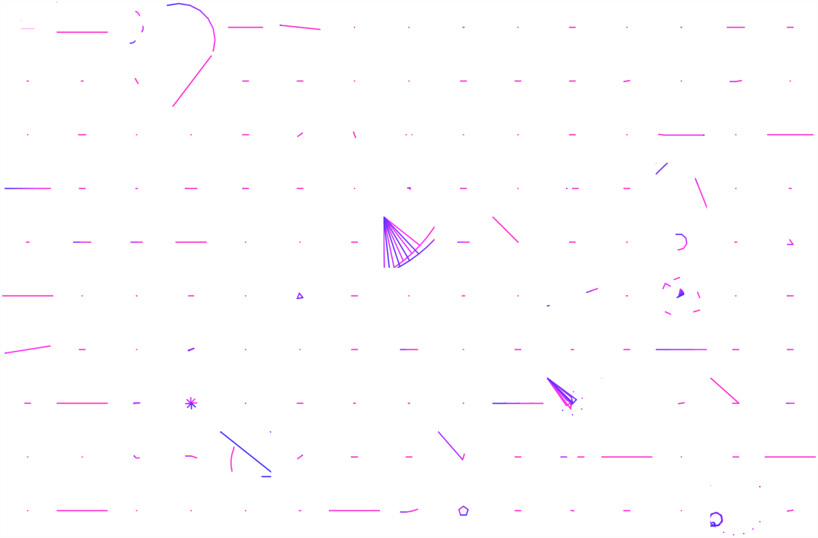
Libraries for probabilistic generative models over text:
data from crawling web for CSV files







150 random dreams before learning



150 random dreams after learning