

Growing libraries of concepts with wake-sleep program induction

Kevin Ellis & Mathias Sablé Meyer

Joint with: Lucas Morales, Armando Solar-Lezama, Joshua B. Tenenbaum

Heavy inspiration from: Eyal Dechter

July 25, 2018

MIT

The Language of Thought

Copyrighted Material

The Language of Thought

JERRY A. FODOR

A FORMAL THEORY OF INDUCTIVE
INFERENCE, Part I*†

Ray J. Solomonoff

Visiting Professor, Computer Learning Research Center
Royal Holloway, University of London
Mailing Address: P.O.B. 400404, Cambridge, Ma. 02140, U.S.A.

Information and Control, Volume 7, No. 1, Pp. 1-22, March 1964 Copyright
by Academic Press Inc.

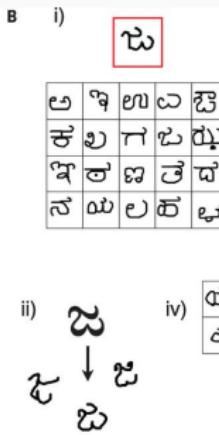
The Language and Thought Series

Jerrold J. Katz

D. Terence Langendoen

George A. Miller

Engineering the language of thought



ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಶಿಕ್ಷಣಾ ತ್ವಾ ಪರಾ
ಧಾರ್ಮಿಕಾ ವಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ

ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಶಿಕ್ಷಣಾ ತ್ವಾ ಪರಾ
ಧಾರ್ಮಿಕಾ ವಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ
ಬ್ರಹ್ಮಾ ಸೂರ್ಯಾ ಪರಾ
ಉತ್ತರಾ ಮಾಸಾ ಪರಾ

Growing a domain-specific language of thought

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

Growing a domain-specific language of thought

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

- Library of concepts (declarative knowledge)
- Inference strategy (procedural knowledge)

DSL: Library of concepts

Tasks and Programs

$[7 \ 2 \ 3] \rightarrow [7 \ 3]$

$[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$

$[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

$f(\ell) = (f_1 \ \ell \ (\lambda \ (x) \ (> \ x \ 2)))$

$[2 \ 7 \ 8 \ 1] \rightarrow 8$

$[3 \ 19 \ 14] \rightarrow 19$

$f(\ell) = (f_2 \ \ell)$

$[7 \ 3] \rightarrow \text{False}$

$[3] \rightarrow \text{False}$

$[9 \ 0 \ 0] \rightarrow \text{True}$

$[0] \rightarrow \text{True}$

$[0 \ 7 \ 3] \rightarrow \text{True}$

$f(\ell) = (f_3 \ \ell \ 0)$

$f_0(\ell, r) = (\text{foldr } r \ \ell \ \text{cons})$

$(f_0: \text{Append lists } r \text{ and } \ell)$

$f_1(\ell, p) = (\text{foldr } \ell \ \text{nil} \ (\lambda \ (x \ a)$

$\text{if } (p \ x) \ (\text{cons } x \ a) \ a))$

$(f_1: \text{Higher-order filter function})$

$f_2(\ell) = (\text{foldr } \ell \ 0 \ (\lambda \ (x \ a)$

$\text{if } (> \ a \ x) \ a \ x))$

$(f_2: \text{Maximum element in list } \ell)$

$f_3(\ell, k) = (\text{foldr } \ell \ (\text{is-nil } \ell)$

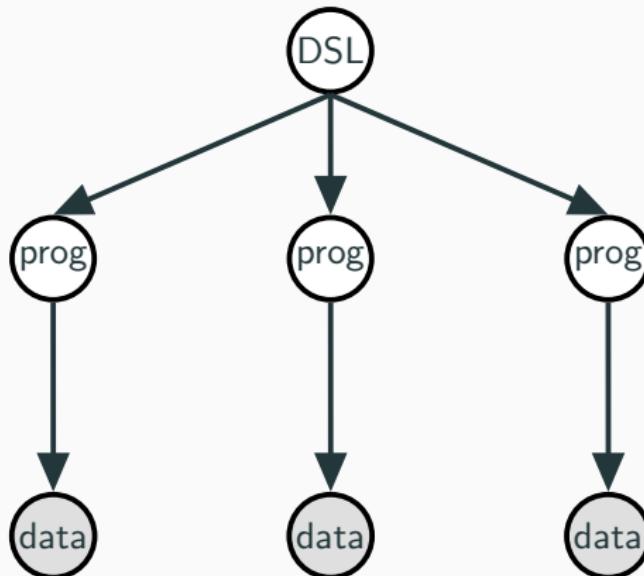
$(\lambda \ (x \ a) \ (\text{if } a \ a \ (= \ k \ x))))$

$(f_2: \text{Whether } \ell \text{ contains } k)$

- **Wake:** Solve problems by writing programs
- **Sleep:** Improve DSL and neural recognition model:
 - **Sleep-G:** Improve DSL (**G**enerative model)
 - **Sleep-R:** Improve **R**ecognition model

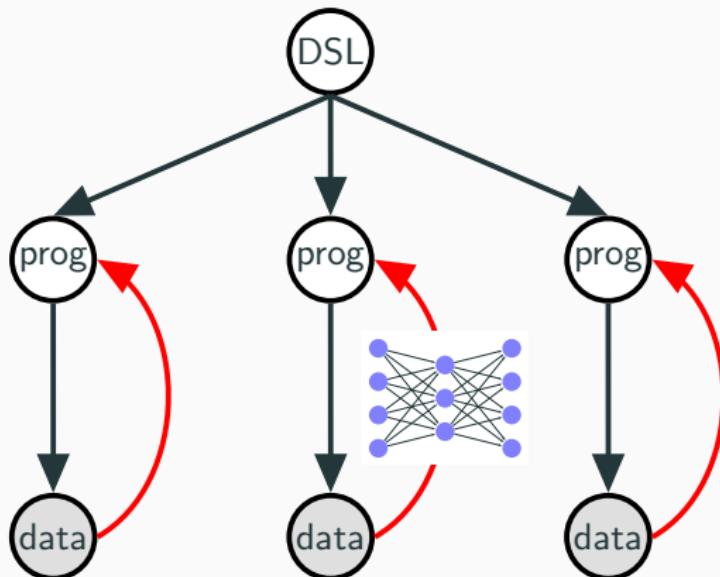
Combines ideas from Wake-Sleep & Exploration-Compression algorithm by Eyal Dechter

DSL learning as Bayesian inference

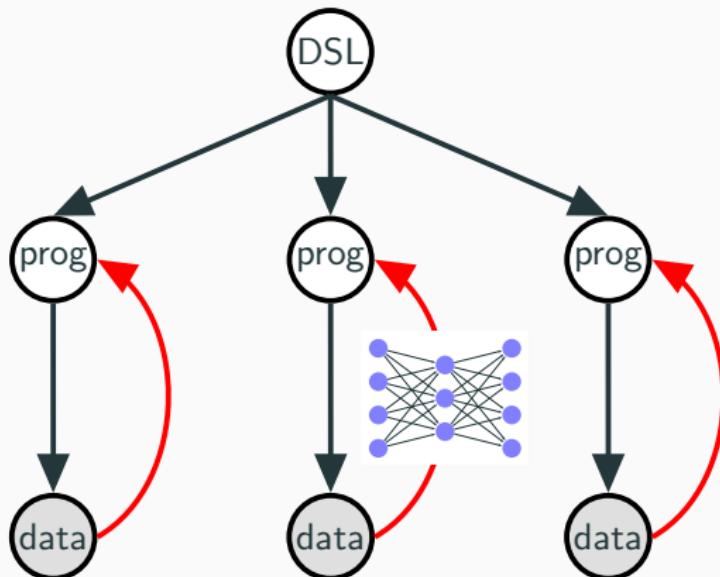


[Dechter et al., 2013] [Liang et al, 2010]; [Lake et al, 2015]

DSL learning as amortized Bayesian inference

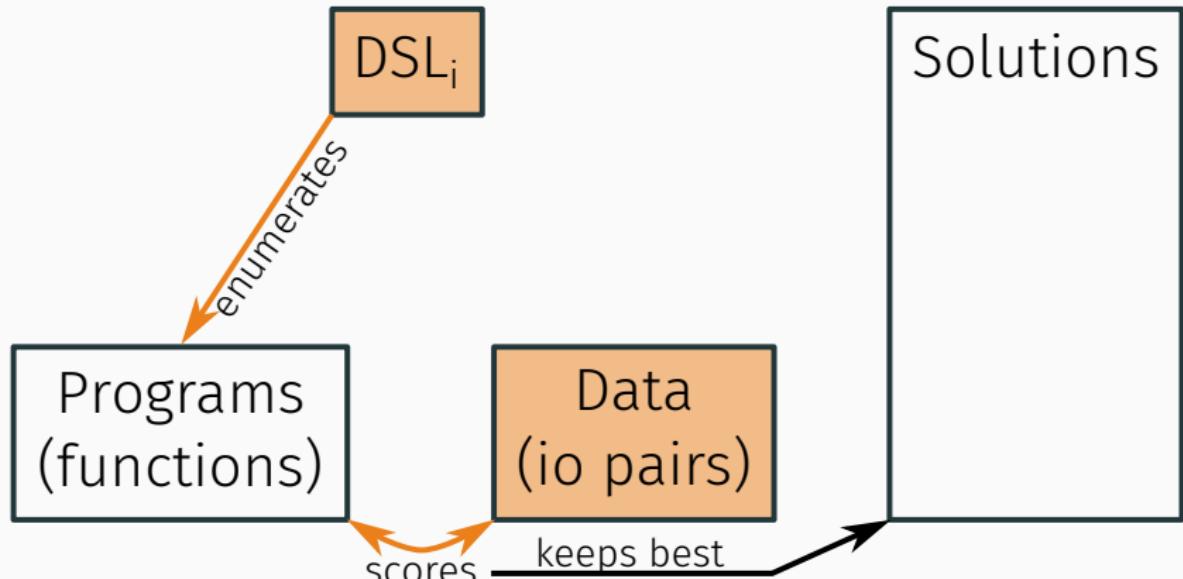


DSL learning as amortized Bayesian inference



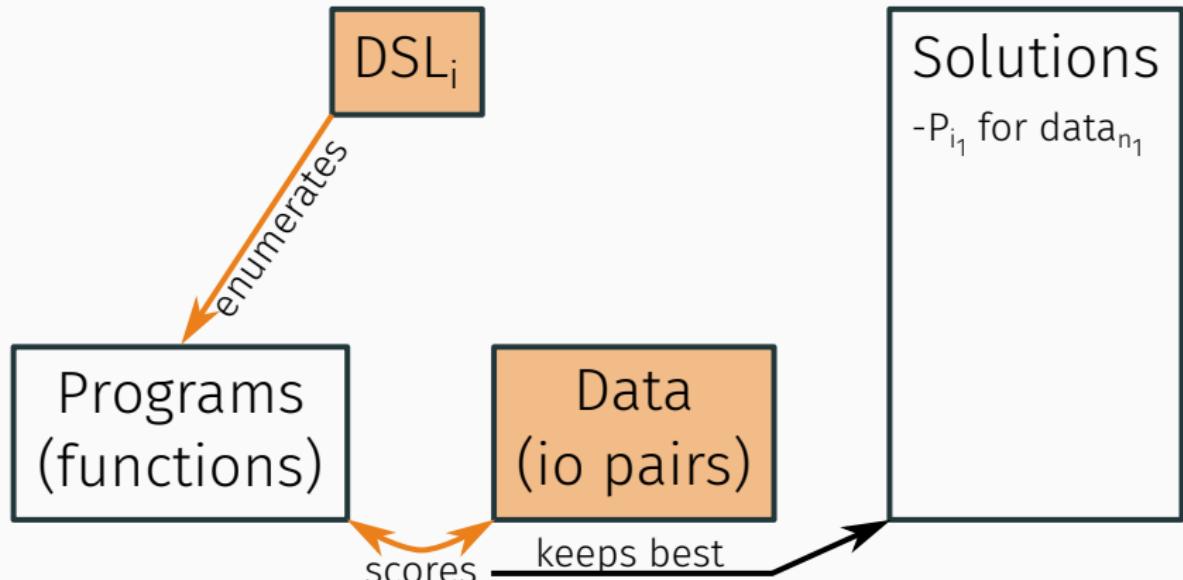
New: amortized inference +
better program representation (Lisp) +
better DSL inference

Wake — as in Exploration/Compression Algorithm



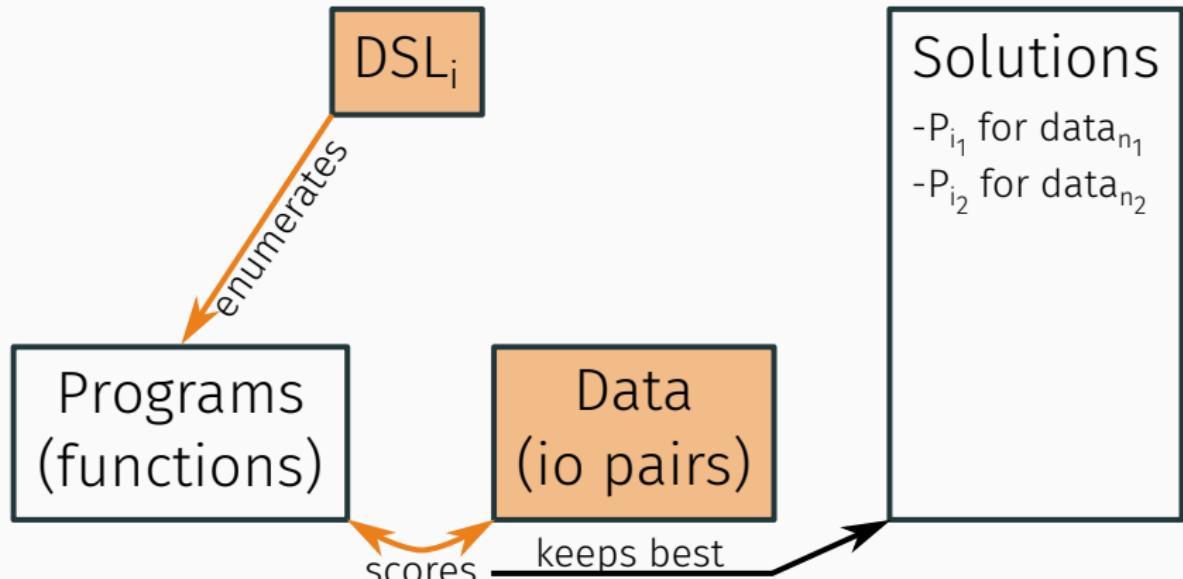
DSL	Programs	Data
$f_0(l,r) = \text{foldr } r\ l\ \text{cons}$ $f_1(l,p) = \text{foldr } r\ \text{nil}$ $\quad (\lambda\ (x\ a)$ $\quad \quad (\text{if}\ (p\ x)$ $\quad \quad \quad (\text{cons}\ x\ a)$ $\quad \quad \quad a))$	$f(l) = (f_0\ l\ l)$ $f(l) = (f_1\ l\ (\lambda x. x > 2))$	$[7\ 2] \rightarrow [7\ 2\ 7\ 2]$ $["a"] \rightarrow ["a"\ "a"]$ $[7\ 2\ 3] \rightarrow [7\ 3]$ $[1\ 2\ 3\ 4] \rightarrow [3\ 4]$ $[4\ 3\ 2\ 1] \rightarrow [4\ 3]$

Wake — as in Exploration/Compression Algorithm



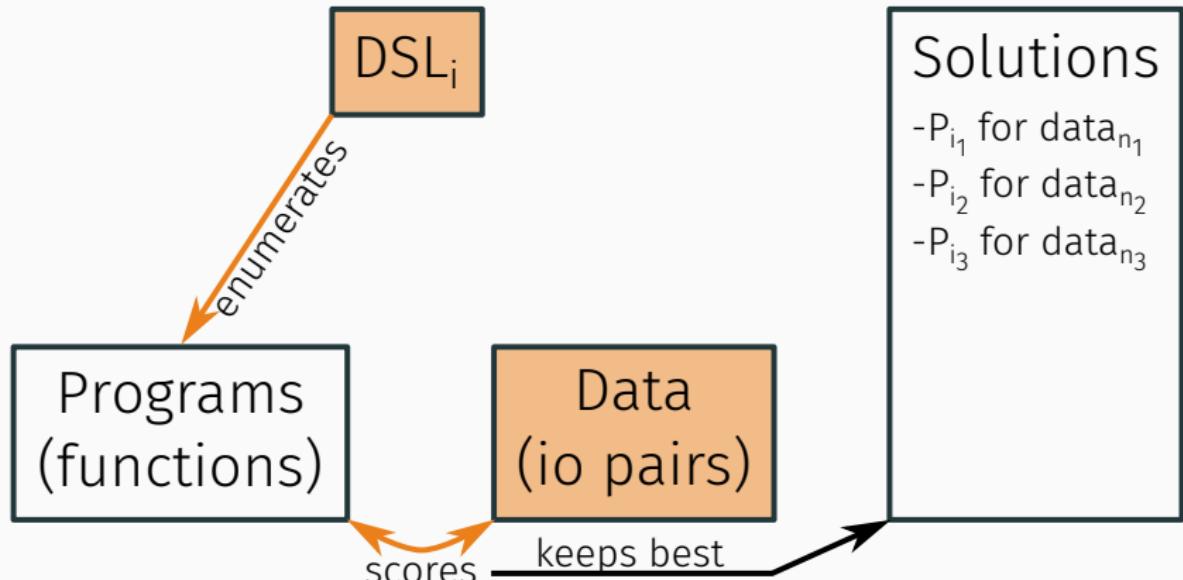
DSL	Programs	Data
$f_0(l,r) = \text{foldr } r\ l\ \text{cons}$ $f_1(l,p) = \text{foldr } r\ \text{nil}\ (\lambda(x)\ a)$ $\quad (\text{if } (p\ x)\ (\text{cons } x\ a))$	$f(l) = (f_0\ l\ l)$ $f(l) = (f_1\ l\ (\lambda x.\ x > 2))$	$[7\ 2] \rightarrow [7\ 2\ 7\ 2]$ $["a"] \rightarrow ["a"\ "a"]$ $[7\ 2\ 3] \rightarrow [7\ 3]$ $[1\ 2\ 3\ 4] \rightarrow [3\ 4]$ $[4\ 3\ 2\ 1] \rightarrow [4\ 3]$

Wake — as in Exploration/Compression Algorithm



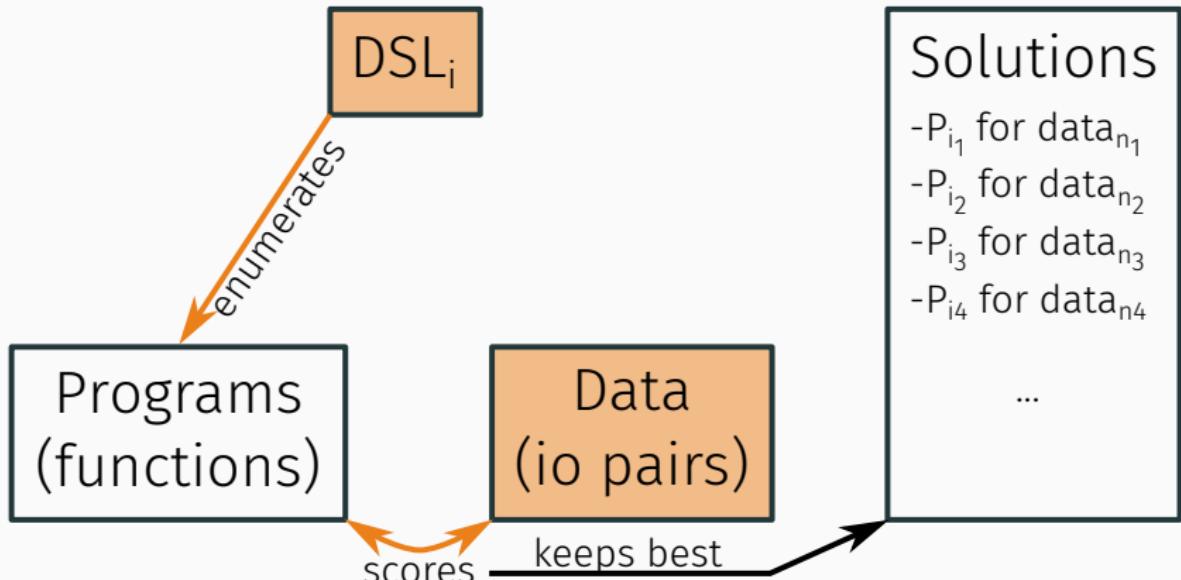
DSL	Programs	Data
$f_0(l,r) = foldr r l cons$ $f_1(l,p) = foldr r nil (\lambda (x a) (if (p x) (cons x a) a))$	$f(l) = (f_0 l l)$ $f(l) = (f_1 l (\lambda x. > x 2))$	$[7\ 2] \rightarrow [7\ 2\ 7\ 2]$ $["a"] \rightarrow ["a"\ "a"]$ $[7\ 2\ 3] \rightarrow [7\ 3]$ $[1\ 2\ 3\ 4] \rightarrow [3\ 4]$ $[4\ 3\ 2\ 1] \rightarrow [4\ 3]$

Wake — as in Exploration/Compression Algorithm



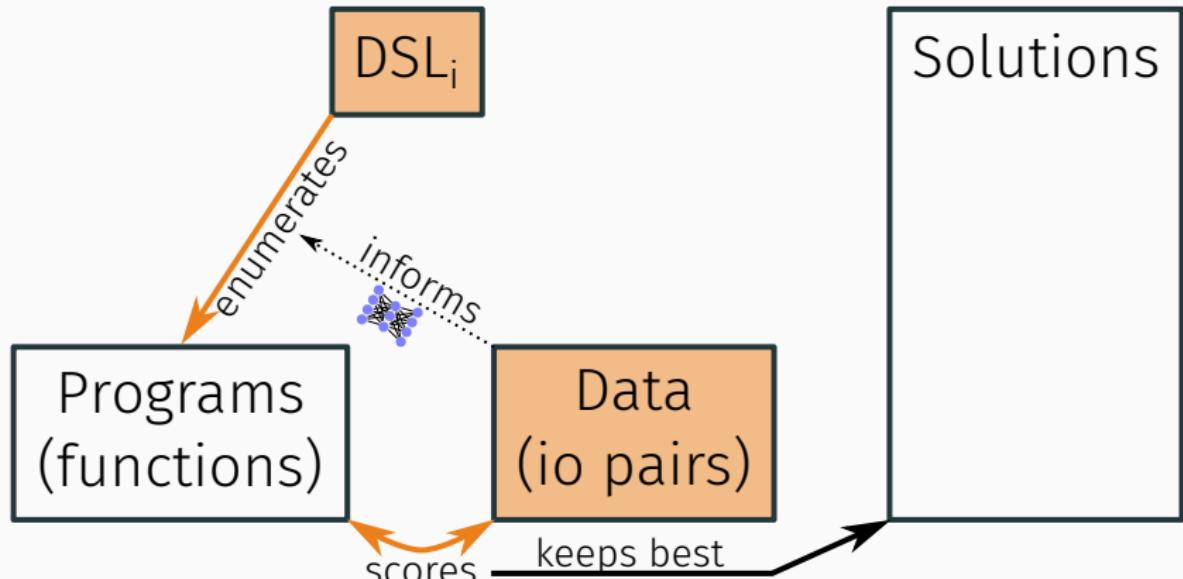
DSL	Programs	Data
$f_0(l,r) = foldr r l cons$ $f_1(l,p) = foldr r nil (\lambda (x a) (if (p x) (cons x a) a))$	$f(l) = (f_0 l l)$ $f(l) = (f_1 l (\lambda x. > x 2))$	$[7\ 2] \rightarrow [7\ 2\ 7\ 2]$ $["a"] \rightarrow ["a"\ "a"]$ $[7\ 2\ 3] \rightarrow [7\ 3]$ $[1\ 2\ 3\ 4] \rightarrow [3\ 4]$ $[4\ 3\ 2\ 1] \rightarrow [4\ 3]$

Wake — as in Exploration/Compression Algorithm



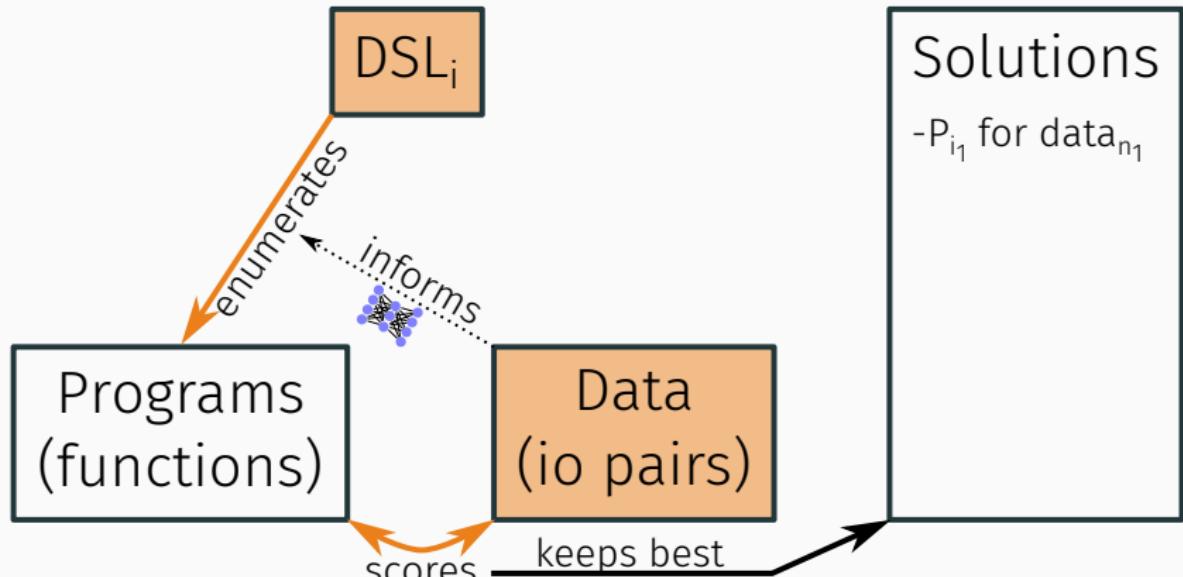
DSL	Programs	Data
$f_0(l,r) = \text{foldr } r\ l\ \text{cons}$ $f_1(l,p) = \text{foldr } r\ \text{nil}\ (\lambda(x)\ a)$ $\quad (\text{if } (p\ x)\ (\text{cons } x\ a)\ a))$	$f(l) = (f_0\ l\ l)$ $f(l) = (f_1\ l\ (\lambda x.\ x > 2))$	$[7\ 2] \rightarrow [7\ 2\ 7\ 2]$ $["a"] \rightarrow ["a"\ "a"]$ $[7\ 2\ 3] \rightarrow [7\ 3]$ $[1\ 2\ 3\ 4] \rightarrow [3\ 4]$ $[4\ 3\ 2\ 1] \rightarrow [4\ 3]$

DreamCoder — Wake



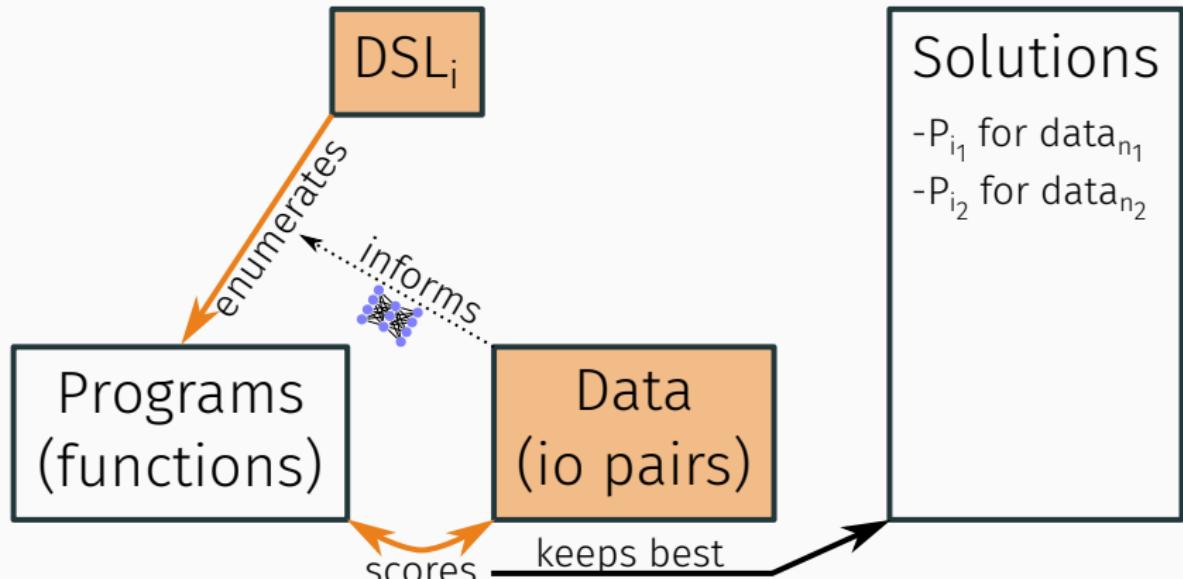
DSL	Programs	Data
$f_0(l, r) = \text{foldr } r \ l \ \text{cons}$ $f_1(l, p) = \text{foldr } r \ \text{nil}$ $\quad (\lambda (x \ a)$ $\quad \quad (\text{if} (p \ x)$ $\quad \quad \quad (\text{cons} \ x \ a)$ $\quad \quad \quad a))$	$f(l) = (f_0 \ l \ l)$ $f(l) = (f_1 \ l \ (\lambda x. \ x > 2))$	$[7 \ 2] \rightarrow [7 \ 2 \ 7 \ 2]$ $["a"] \rightarrow ["a" \ "a"]$ $[7 \ 2 \ 3] \rightarrow [7 \ 3]$ $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$ $[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

DreamCoder — Wake



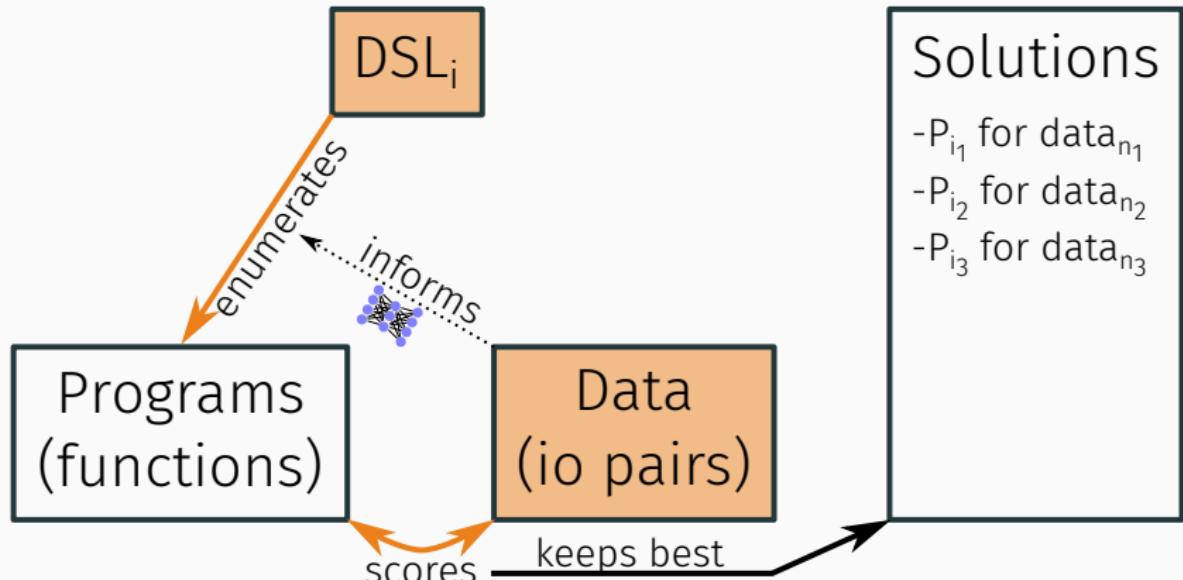
DSL	Programs	Data
$f_0(l, r) = \text{foldr } r \ l \ \text{cons}$ $f_1(l, p) = \text{foldr } r \ \text{nil} \quad (\lambda (x \ a) \quad (if \ (p \ x) \quad (\text{cons} \ x \ a) \quad a))$	$f(l) = (f_0 \ l \ l)$ $f(l) = (f_1 \ l \ (\lambda x. \ x > 2))$	$[7 \ 2] \rightarrow [7 \ 2 \ 7 \ 2]$ $["a"] \rightarrow ["a" \ "a"]$
		$[7 \ 2 \ 3] \rightarrow [7 \ 3]$ $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$ $[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

DreamCoder — Wake



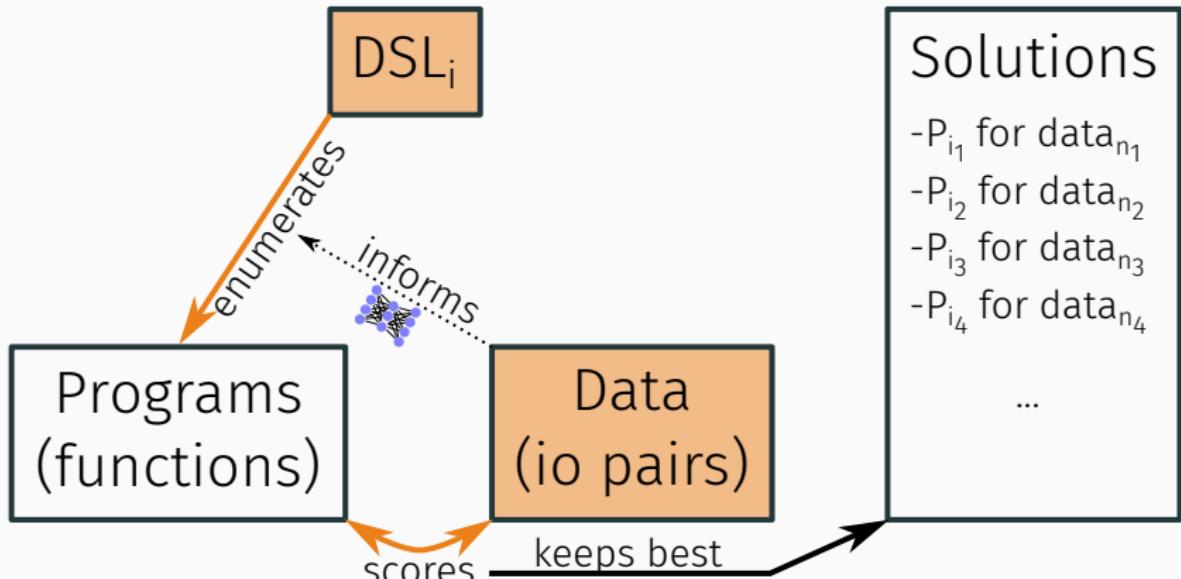
DSL	Programs	Data
$f_0(l, r) = \text{foldr } r \ l \ \text{cons}$ $f_1(l, p) = \text{foldr } r \ \text{nil}$ $\quad (\lambda (x \ a)$ $\quad \quad (\text{if} (p \ x)$ $\quad \quad \quad (\text{cons} \ x \ a)$ $\quad \quad \quad a))$	$f(l) = (f_0 \ l \ l)$ $f(l) = (f_1 \ l \ (\lambda x. \ x > 2))$	$[7 \ 2] \rightarrow [7 \ 2 \ 7 \ 2]$ $["a"] \rightarrow ["a" \ "a"]$ $[7 \ 2 \ 3] \rightarrow [7 \ 3]$ $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$ $[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

DreamCoder — Wake



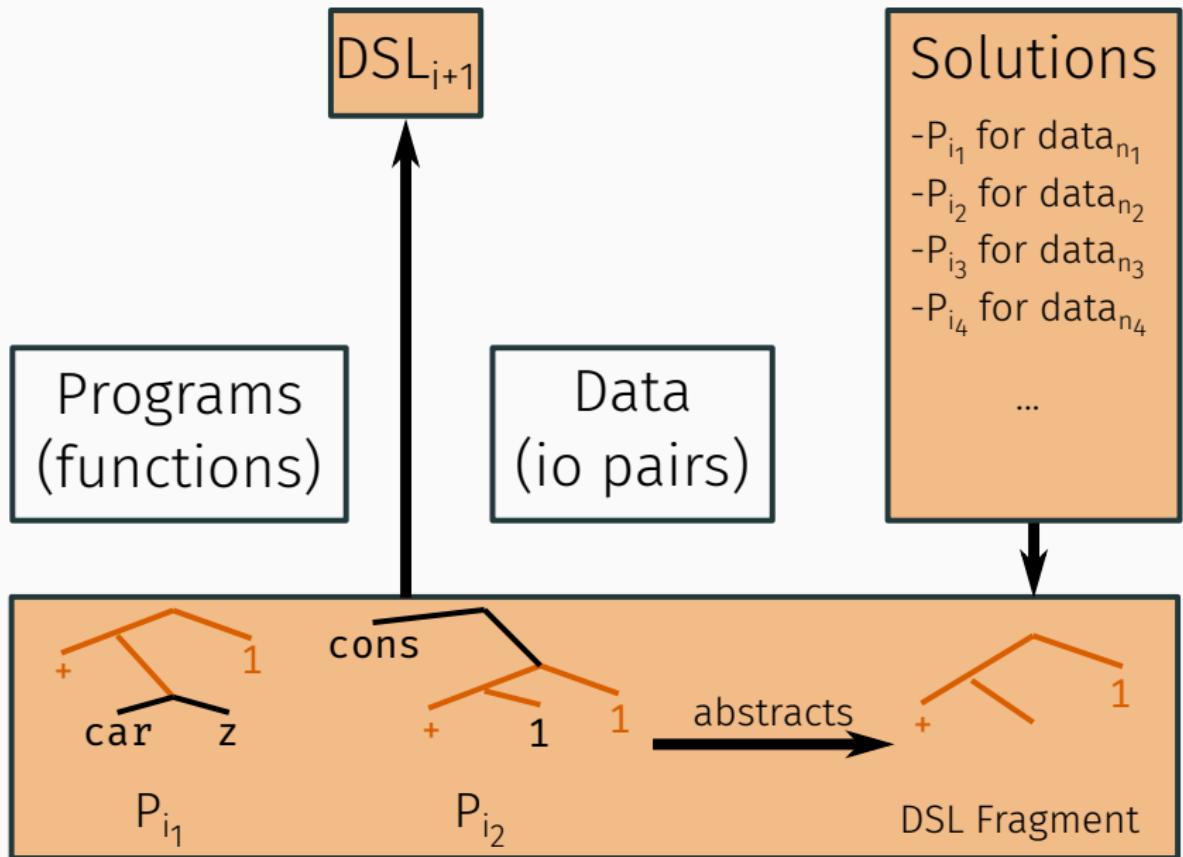
DSL	Programs	Data
$f_0(l, r) = \text{foldr } r \ l \ \text{cons}$ $f_1(l, p) = \text{foldr } r \ \text{nil}$ $(\lambda (x \ a)$ $(\text{if} (p \ x)$ $(\text{cons} \ x \ a)$ $a))$	$f(l) = (f_0 \ l \ l)$ $f(l) = (f_1 \ l \ (\lambda x. \ x > 2))$	$[7 \ 2] \rightarrow [7 \ 2 \ 7 \ 2]$ $["a"] \rightarrow ["a" \ "a"]$ $[7 \ 2 \ 3] \rightarrow [7 \ 3]$ $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$ $[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

DreamCoder — Wake

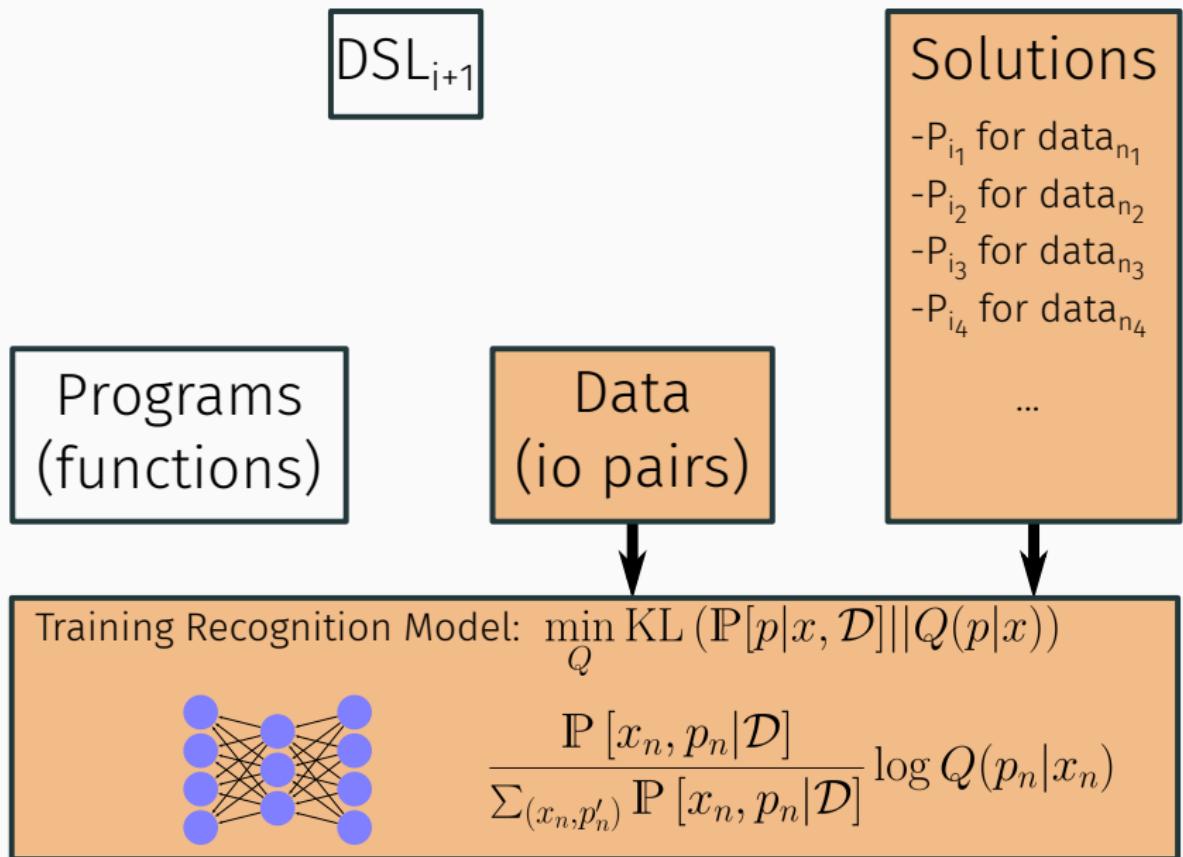


DSL	Programs	Data
$f_0(l, r) = \text{foldr } r \ l \ \text{cons}$ $f_1(l, p) = \text{foldr } r \ \text{nil}$ $(\lambda (x \ a)$ $(\text{if} (p \ x)$ $(\text{cons} \ x \ a)$ $a))$	$f(l) = (f_0 \ l \ l)$ $f(l) = (f_1 \ l \ (\lambda x. \ x > 2))$	$[7 \ 2] \rightarrow [7 \ 2 \ 7 \ 2]$ $["a"] \rightarrow ["a" \ "a"]$ $[7 \ 2 \ 3] \rightarrow [7 \ 3]$ $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4]$ $[4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3]$

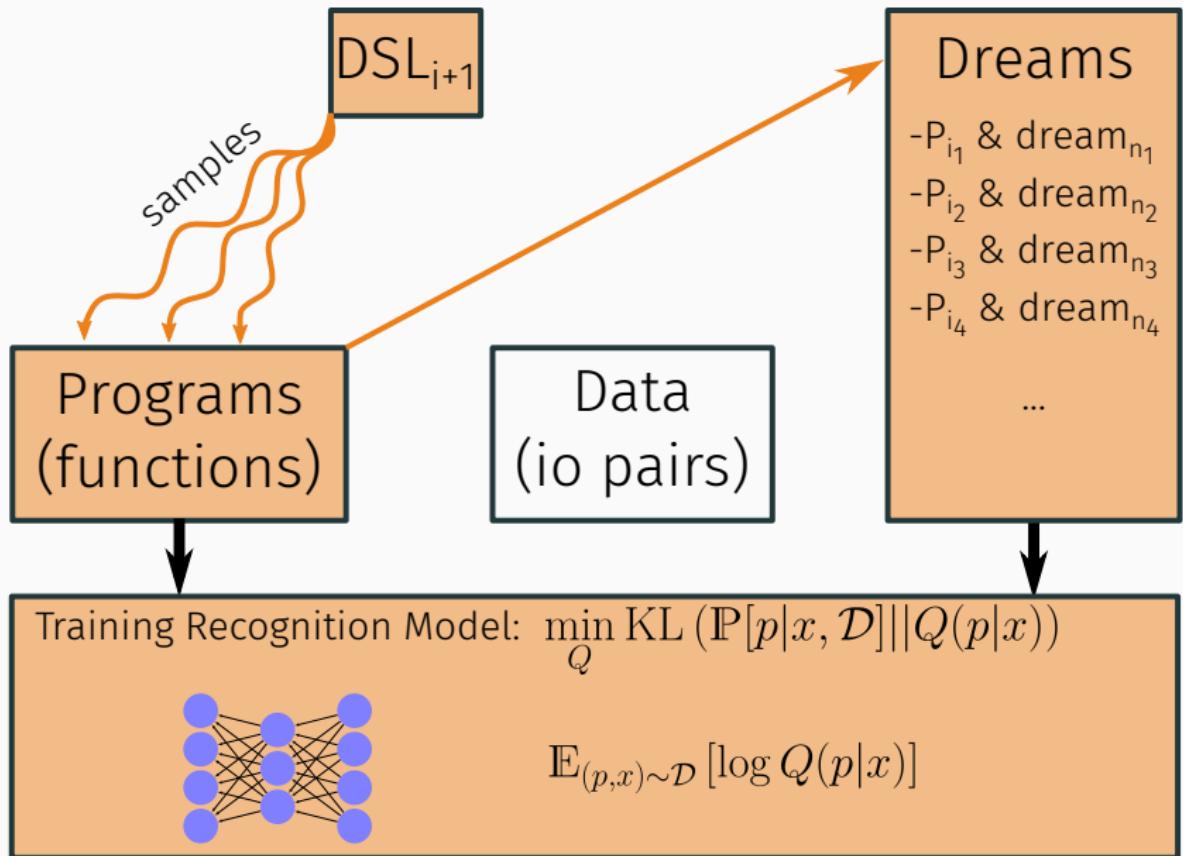
DreamCoder — Sleep-G



DreamCoder — Sleep-R (Experience Replay)



DreamCoder — Sleep-R (Dreaming)

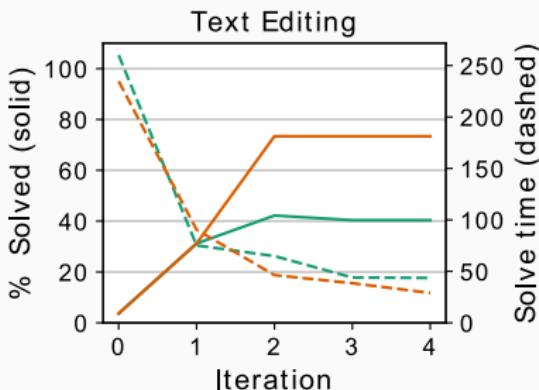
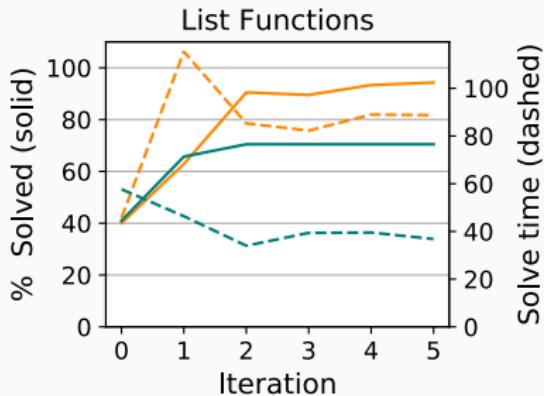


List functions — Created & investigated by Lucas Morales

Name	Input	Output
repeat-3	[7 0]	[7 0 7 0 7 0]
drop-3	[0 3 8 6 4]	[6 4]
rotate-2	[8 14 1 9]	[1 9 8 14]
count-head-in-tail	[1 2 1 1 3]	2
keep-div-5	[5 9 14 6 3 0]	[5 0]
product	[7 1 6 2]	84

Discovers 38 concepts, including ‘filter’. With suitable curriculum can also learn ‘map’, ‘fold’, etc. starting with 1950’s Lisp

List functions & Text editing: Learning curves on hold out tasks



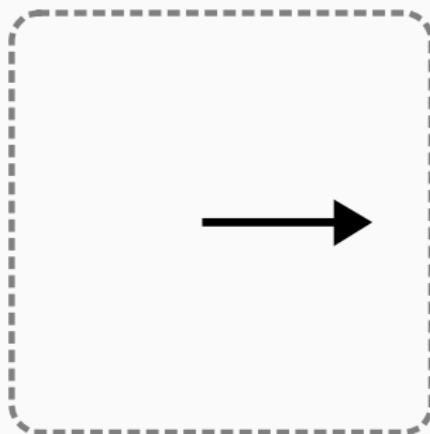
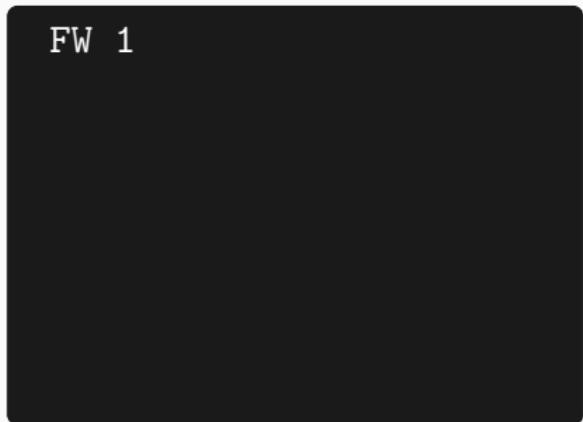
Learning curves for DreamCoder both with (**in orange**) and without (**in teal**) the recognition model. Solid lines: % holdout testing tasks solved w/ 10m timeout. Dashed lines: Average solve time, averaged only over tasks that are solved.

DSL

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

Tasks

task : image



DSL

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

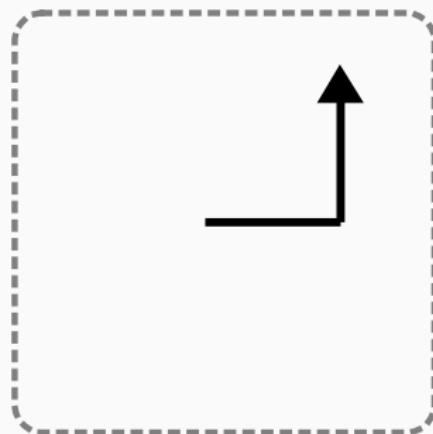
Tasks

task : image

FW 1

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

FW 1



DSL

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

Tasks

task : image

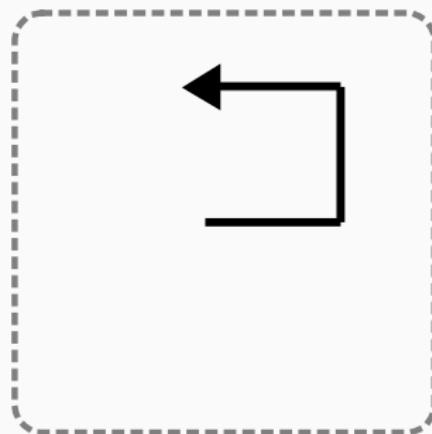
FW 1

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

FW 1

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

FW 1



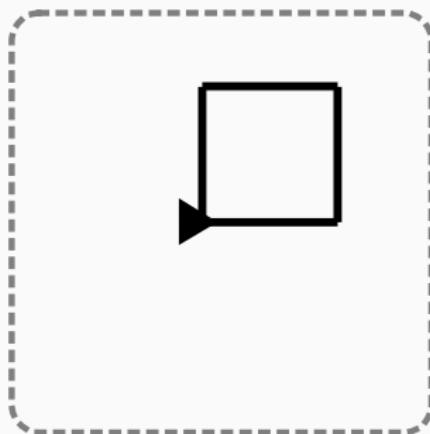
DSL

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

Tasks

task : image

```
for i in range(4)
> FW 1
> RT  $\frac{\pi}{2}$ 
```



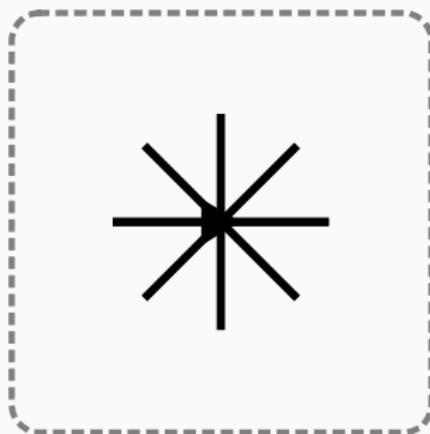
DSL

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

Tasks

task : image

```
for i in range(8)
> FW 1
> SET origin
> RT  $\frac{2\pi}{8}$ 
```



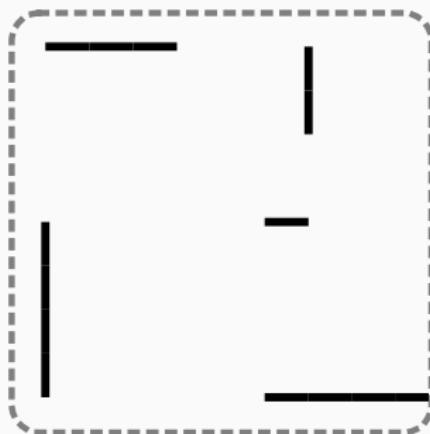
DSL

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  $\frac{\pi}{2}$ 
```



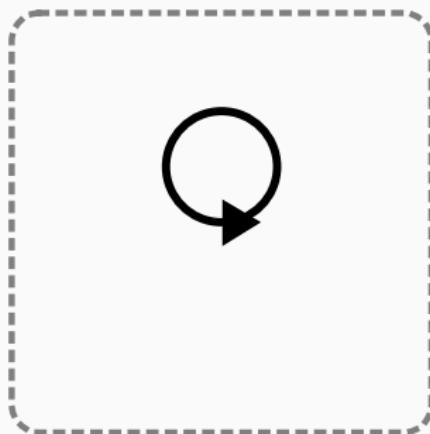
DSL

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

Tasks

task : image

```
for i in range( $\infty$ )
> FW ε
> RT ε
```



DSL

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

Tasks

task : image

```
for i in range(5 × ∞)
> FW i × ε
> RT ε
```



DSL

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

Tasks

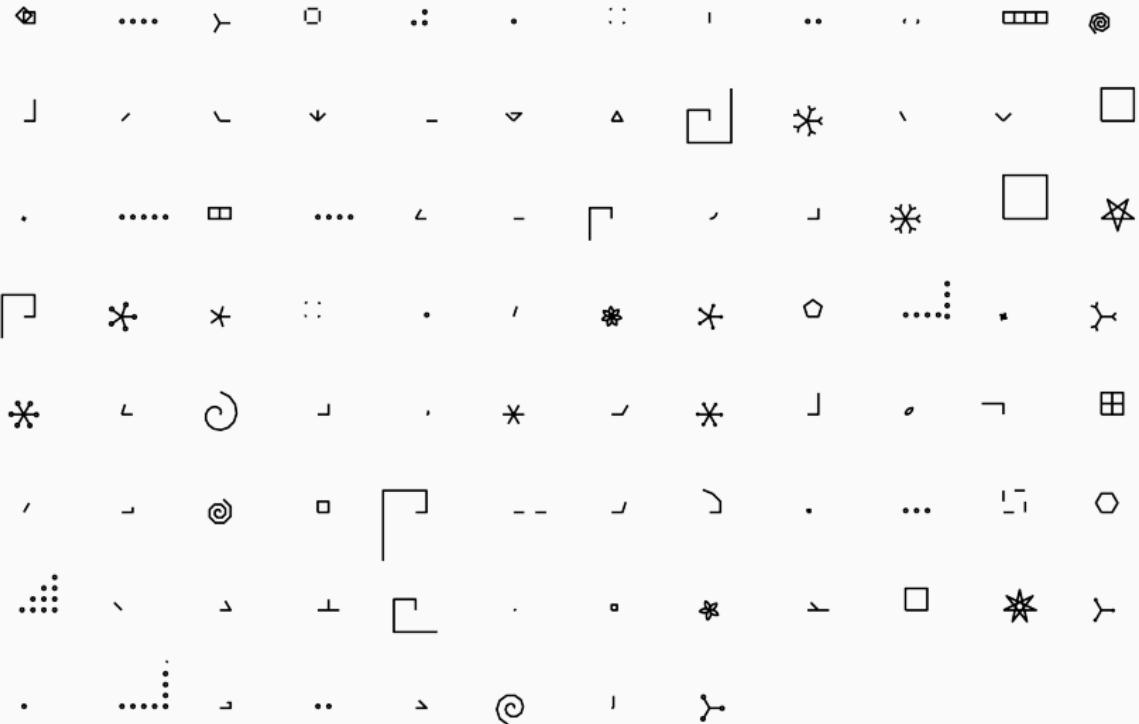
task : image

```
for i in range(5 × ∞)
> FW i × ε
> RT ε
```

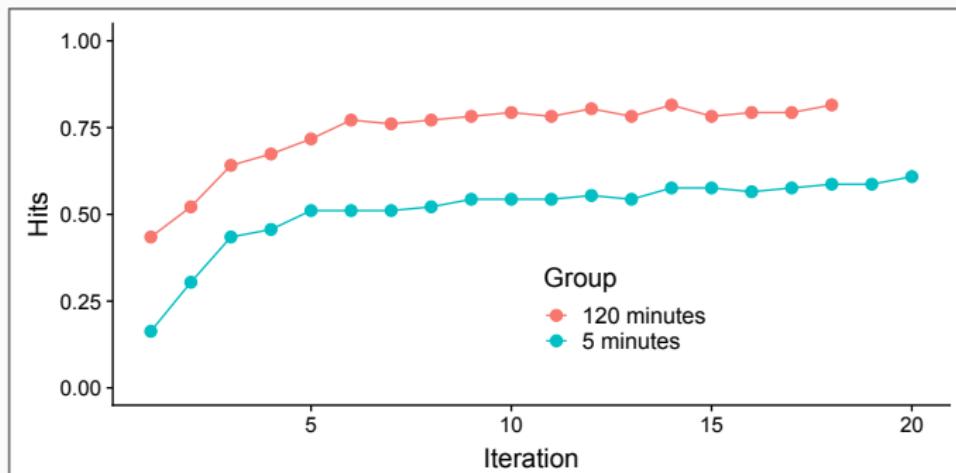


NUM ::= 1 | π | ∞ | ε | + | - | * | /

Turtle graphics — Training tasks



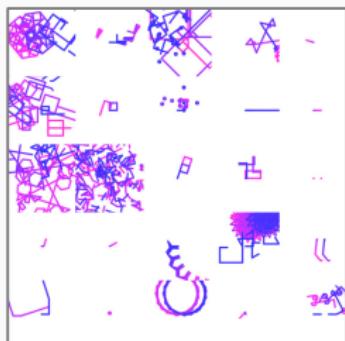
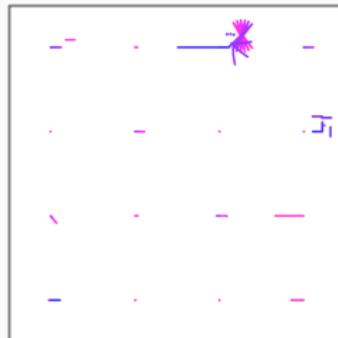
Turtle graphics — Learning curves



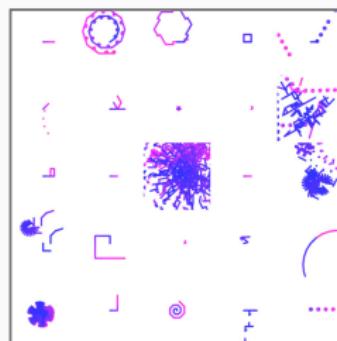
- $\frac{\pi}{2}$ and $\frac{\pi}{4}$ from π , 2, + and /
- A line of length n followed with a right angle
- Loops of length n that uses the number n inside.
- Unit line then teleport back to origin
- ...

Turtle graphics — Dreams

Before training

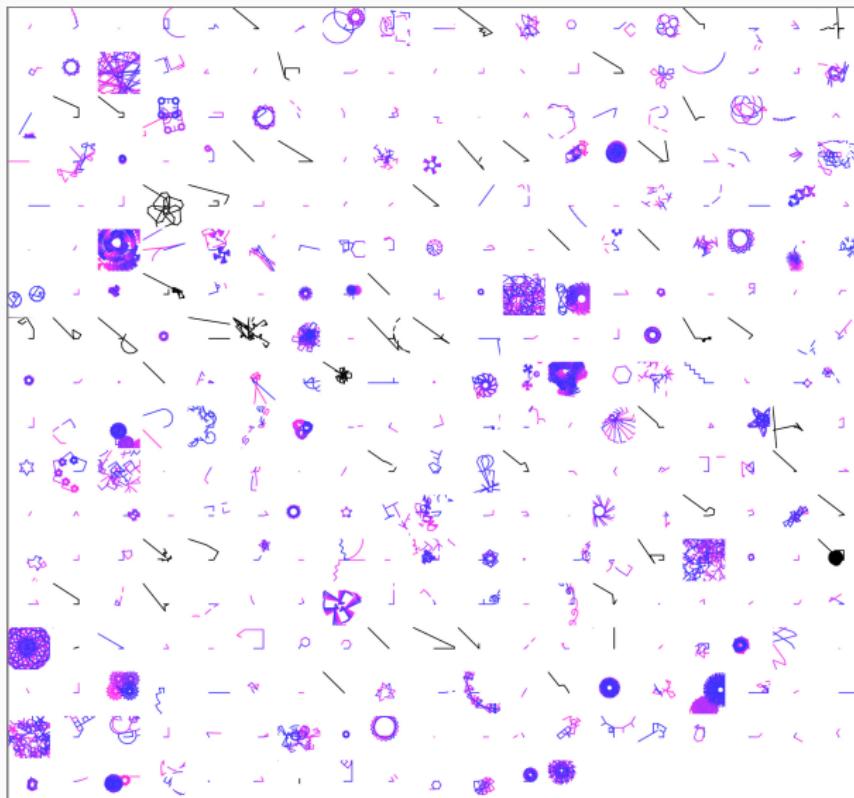


Plateau 5 minutes

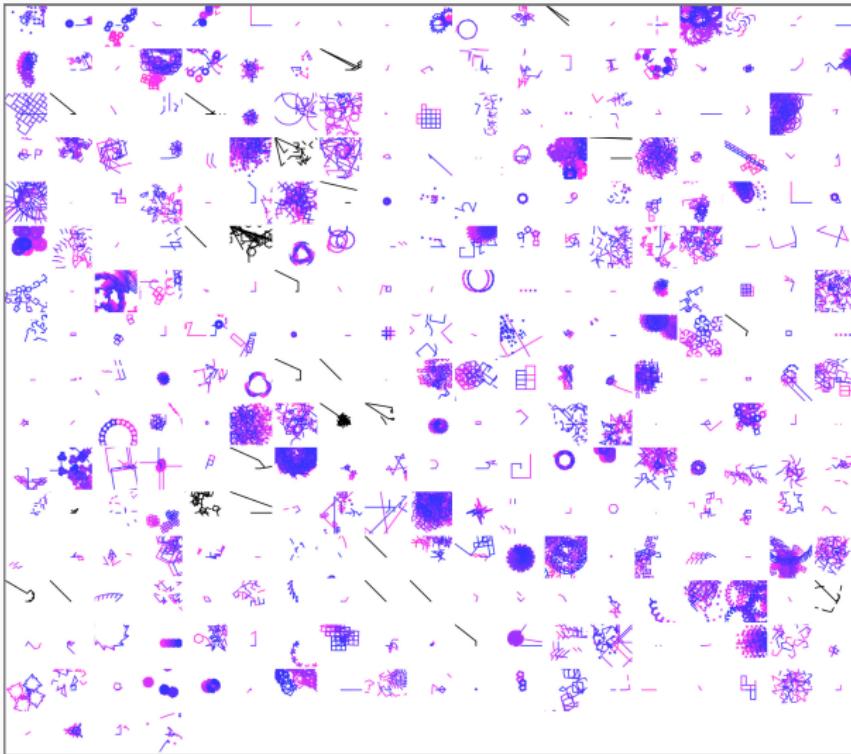


Plateau 2 hours

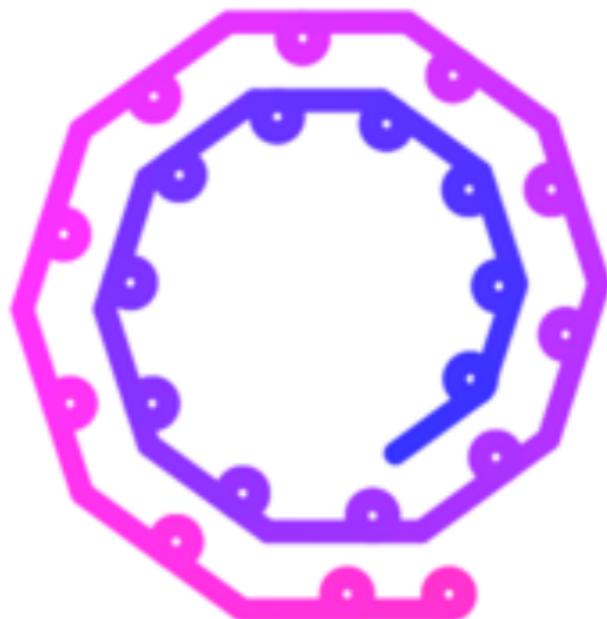
Turtle graphics — More dreams, 5 minutes, 1st iteration



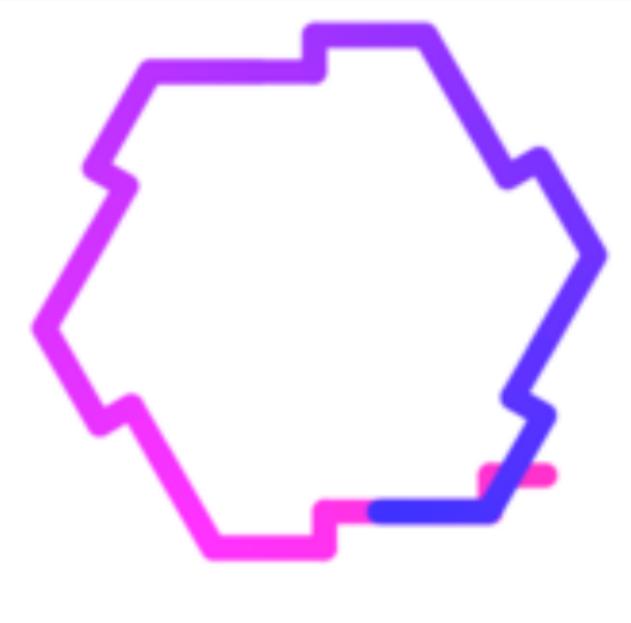
Turtle graphics — More dreams, 5 minutes, last iteration



Turtle graphics — Generative model



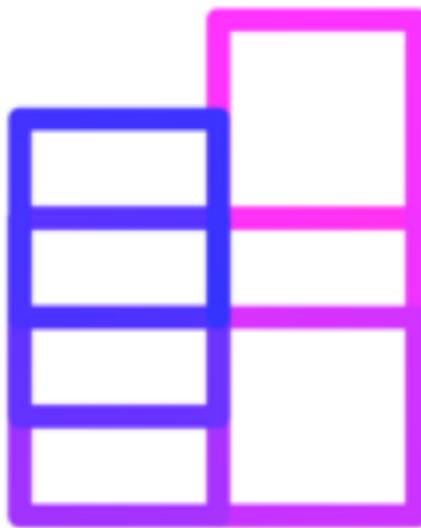
Turtle graphics — Generative model



Turtle graphics — Generative model



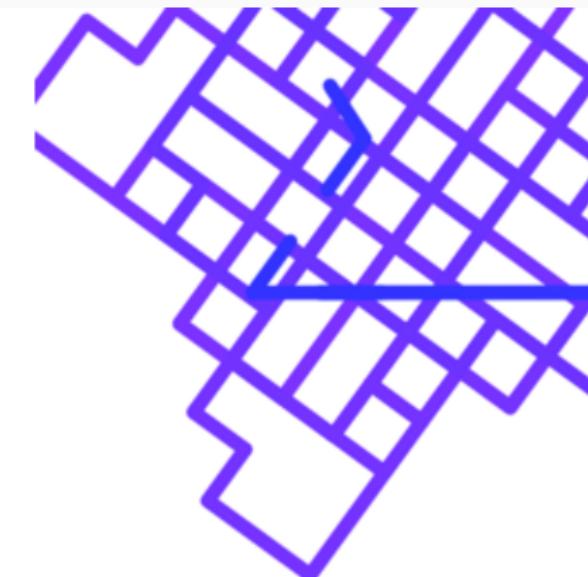
Turtle graphics — Generative model



Turtle graphics — Generative model



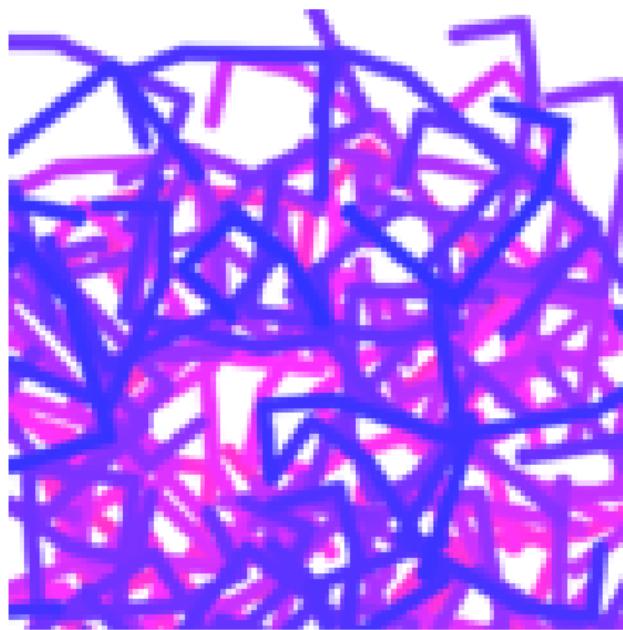
Turtle graphics — Generative model



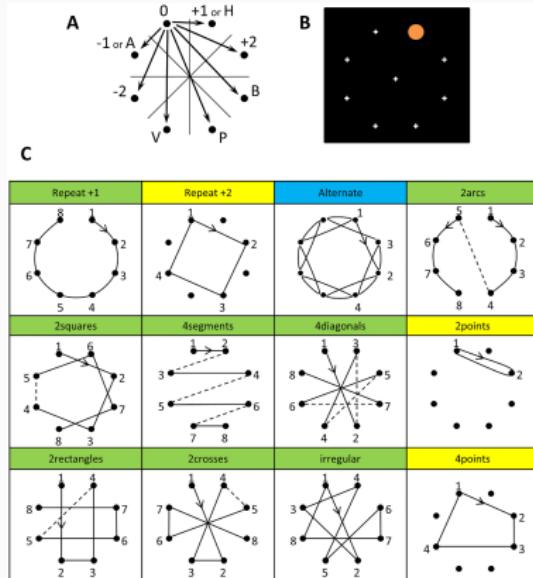
Turtle graphics — Generative model



Turtle graphics — Generative model

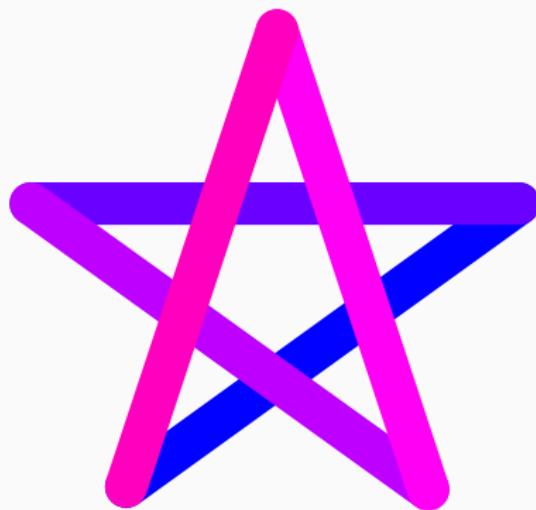


Turtle graphics — All the way down



Marie Amalric, Stanislas Dehaene et al.

Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down

Turtle graphics — All the way down



Turtle graphics — All the way down



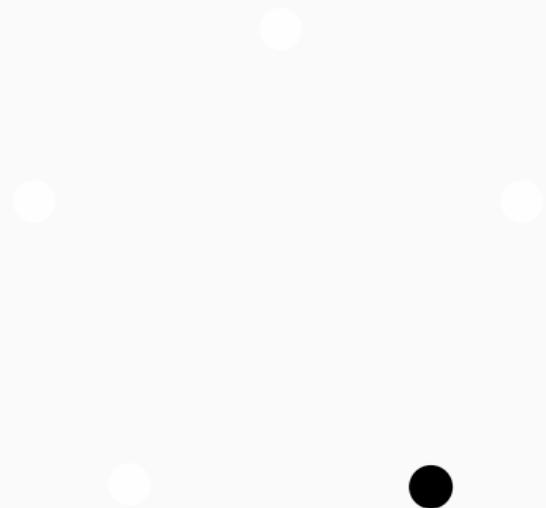
Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



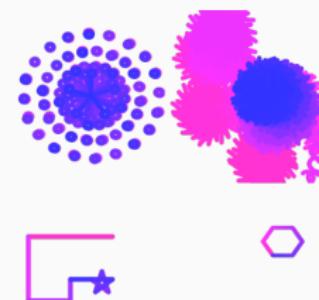
More human-like machine intelligence

- Acquiring a domain-specific language-of-thought (DSL)
- Learning to use the language of thought (recognition model)

DreamCoder: an algorithm for jointly realizing these goals

```
f2(p,f,n,x) = (if (p x) nil  
                  (cons (f x) (f2 (n x))))  
(f2: unfold)  
f3(i,l) = (if (= i 0) (car l)  
              (f3 (f1 i) (cdr l)))  
(f3: index)  
f4(f,l,x) = (if (empty? l) x  
                  (f (car l) (f4 (cdr l))))  
(f4: fold)  
f5(f,l) = (if (empty? l) nil  
              (cons (f (car l)) (f5 (cdr l))))  
(f5: map)
```

Symbolic Regression	
	$f(x) = (f_1 \times)$
	$f(x) = (f_6 \times)$
	$f(x) = (f_4 \times)$
	$f(x) = (f_3 \times)$
$f_0(x) = (+ \times \text{real})$	
$f_1(x) = (f_0 (\star \text{real} \times))$	
$f_2(x) = (f_1 (\star \times (f_0 \times)))$	
$f_3(x) = (f_0 (\star \times (f_2 \times)))$	
$f_4(x) = (f_0 (\star \times (f_3 \times)))$	<i>(f₄: 4th order polynomial)</i>
$f_5(x) = (/ \text{real} \times)$	
$f_6(x) = (f_5 (f_0 \times))$	<i>(f₆: rational function)</i>



More human-like machine intelligence

- Acquiring a domain-specific language-of-thought (DSL)
- Learning to use the language of thought (recognition model)

DreamCoder: an algorithm for jointly realizing these goals

```
f2(p,f,n,x) = (if (p x) nil  
                  (cons (f x) (f2 (n x))))  
(f2: unfold)  
f3(i,l) = (if (= i 0) (car l)  
            (f3 (f1 i) (cdr l)))  
(f3: index)  
f4(f,l,x) = (if (empty? l) x  
              (f (car l) (f4 (cdr l))))  
(f4: fold)  
f5(f,l) = (if (empty? l) nil  
              (cons (f (car l)) (f5 (cdr l))))  
(f5: map)
```

Symbolic Regression	
	$f(x) = (f_1 \times)$
	$f(x) = (f_6 \times)$
	$f(x) = (f_4 \times)$
	$f(x) = (f_3 \times)$
$f_0(x) = (+ \times \text{real})$	
$f_1(x) = (f_0 (\star \text{real} \times))$	
$f_2(x) = (f_1 (\star x (f_0 \times)))$	
$f_3(x) = (f_0 (\star x (f_2 \times)))$	
$f_4(x) = (f_0 (\star x (f_3 \times)))$	<i>(f4: 4th order polynomial)</i>
$f_5(x) = (/ \text{real} \times)$	
$f_6(x) = (f_5 (f_0 \times))$	<i>(f6: rational function)</i>

