

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 24, 2018

MIT

The Language of Thought

Copyrighted Material

The Language of Thought

JERRY A. FODOR

The Language and Thought Series

Jerrold J. Katz

D. Terence Langendoen

George A. Miller

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.

Engineering the language of thought

Universal
Theory



Theory



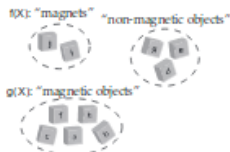
Model



Data

Magnetism
Core Predicates: $f(X)$, $g(X)$
Surface Predicates: $interacts(X,Y)$

Laws:
 $interacts(X,Y) \leftarrow f(X) \wedge f(Y)$
 $interacts(X,Y) \leftarrow f(X) \wedge g(Y)$
 $interacts(X,Y) \leftarrow interacts(Y,X)$



Probabilistic Horn Clause Grammar

Taxonomy

Core Predicates: $s(X,Y)$, $t(X,Y)$
Surface Predicates: $has_a(X,Y)$, $is_a(X,Y)$

Laws:
 $is_a(X,Y) \leftarrow s(X,Y)$
 $has_a(X,Y) \leftarrow t(X,Y)$
 $has_a(X,Y) \leftarrow is_a(X,Z) \wedge has_a(Z,Y)$
 $is_a(X,Y) \leftarrow is_a(X,Z) \wedge is_a(Z,Y)$

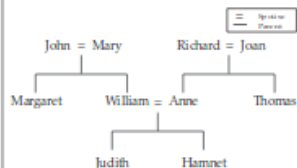


"a shark is a fish"
"a bird can fly"
"a canary can fly"
"a salmon can breathe"
...

Kinship

Core Predicates: $u(X)$, $v(X,Y)$, $w(X,Y)$
Surface Predicates: $female(X)$, $child(X,Y)$, $parent(X,Y)$, $spouse(X,Y)$, $father(X,Y)$, ...

Laws:
 $female(X) \leftarrow u(X)$
 $spouse(X,Y) \leftarrow v(X,Y)$
 $spouse(X,Y) \leftarrow v(Y,X)$
 $child(X,Y) \leftarrow w(X,Y)$
 $child(X,Y) \leftarrow child(X,Z) \wedge spouse(Z,Y)$
 $father(X,Y) \leftarrow \neg female(X) \wedge child(X,Y)$
...



"John is William's father"
"John is Judith's grandfather"
"Judith is Hamnet's sister"
"Margaret is Judith's aunt"
...

Engineering the language of thought

A

i) 

iii) 

ii) 

iv) 

B

i) 

iii) 

ii) 

iv) 



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] → [7 3]
[1 2 3 4] → [3 4]
[4 3 2 1] → [4 3]    [7 3] → False
f(ℓ) = (f1 ℓ (λ (x)    [3] → False
    (> x 2)))          [9 0 0] → True
                        [0] → True
                        [0 7 3] → True
                        f(ℓ) = (f3 ℓ 0)
[2 7 8 1] → 8
[3 19 14] → 19
f(ℓ) = (f2 ℓ)
```

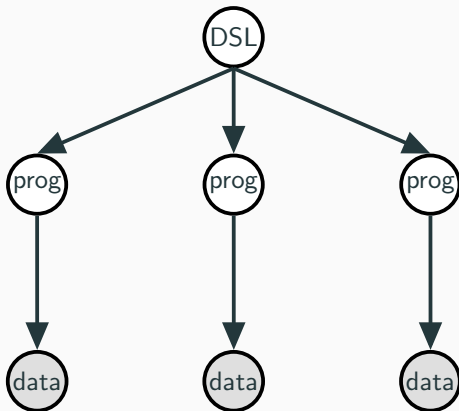
DSL

```
f0(ℓ, r) = (foldr r ℓ cons)
           (f0: Append lists r and ℓ)
f1(ℓ, p) = (foldr ℓ nil (λ (x a)
    (if (p x) (cons x a) a)))
           (f1: Higher-order filter function)
f2(ℓ) = (foldr ℓ 0 (λ (x a)
    (if (> a x) a x)))
           (f2: Maximum element in list ℓ)
f3(ℓ, k) = (foldr ℓ (is-nil ℓ)
    (λ (x a) (if a a (= k x))))
           (f3: Whether ℓ 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

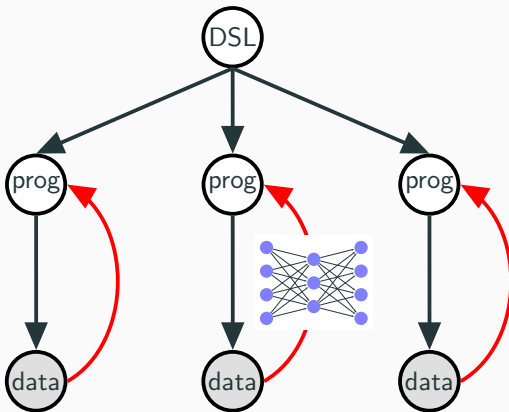
Combines ideas from Wake-Sleep & Exploration-Compression algorithm by Eyal Dechter

DSL learning as Bayesian inference

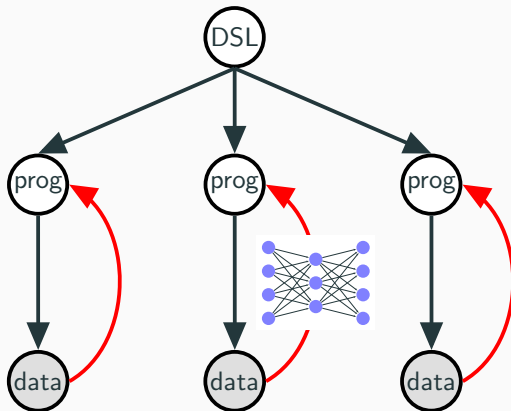


[Dechter et al., 2013] [Liang et al, 2010]; [Lake et al, 2015]

DSL learning as **amortized** Bayesian inference

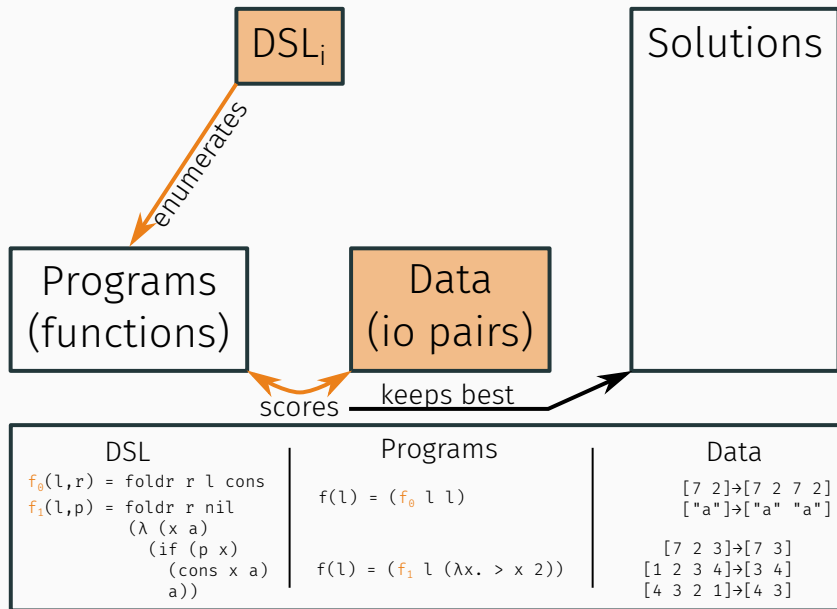


DSL learning as **amortized** Bayesian inference

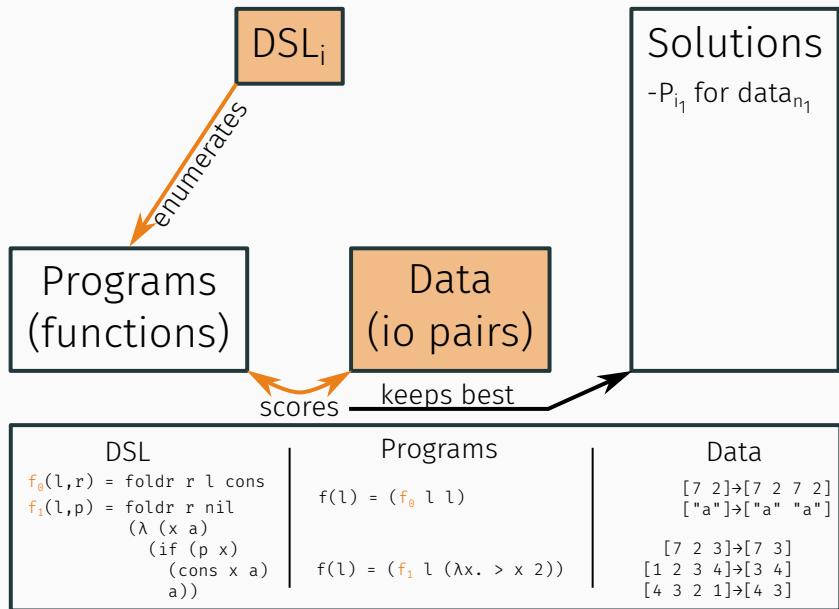


New: amortized inference +
better program representation (Lisp) +
better DSL inference

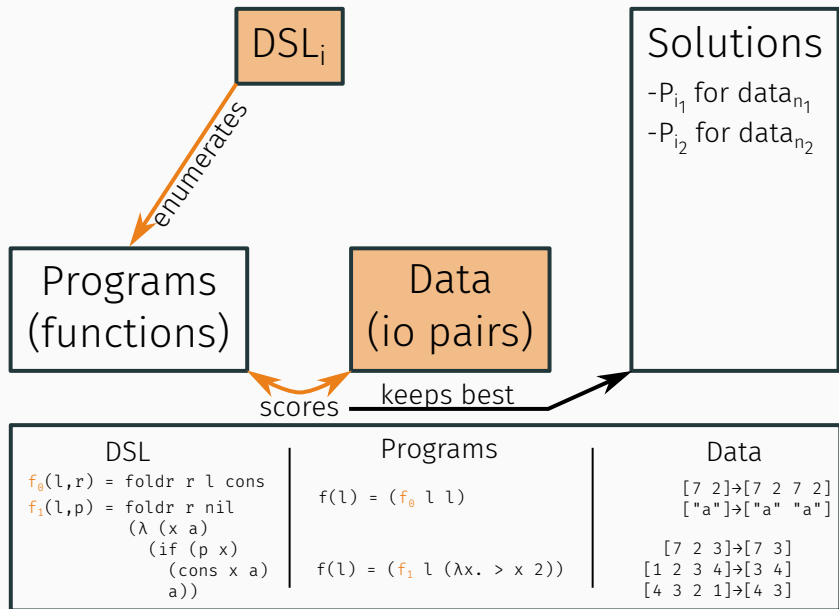
Wake — as in Exploration/Compression Algorithm



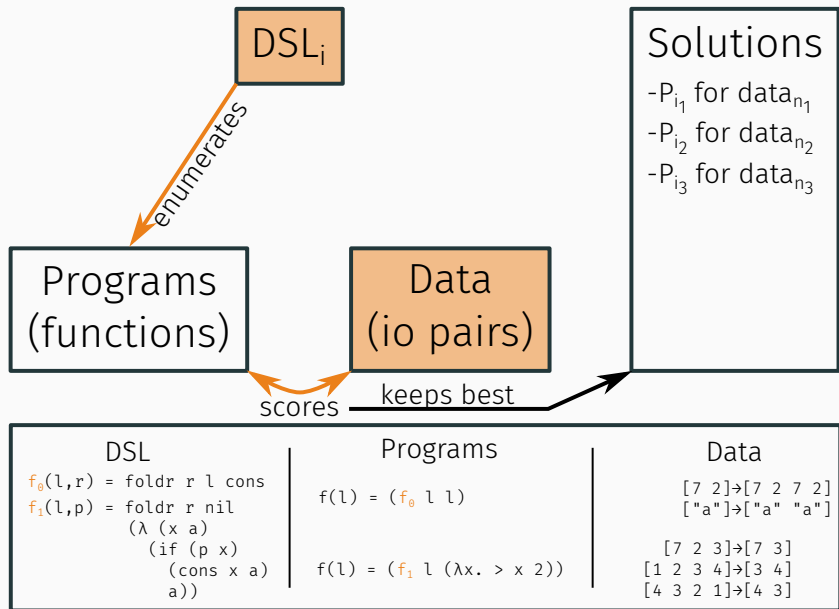
Wake — as in Exploration/Compression Algorithm



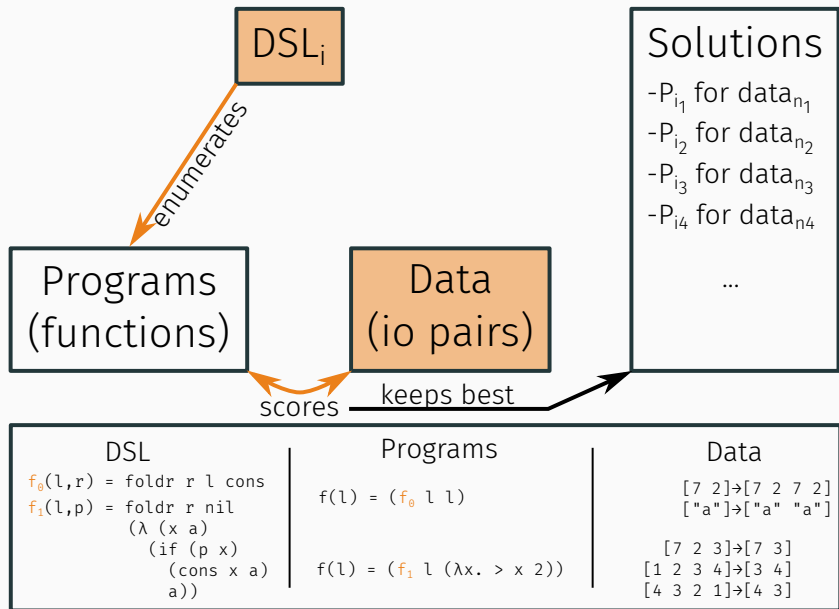
Wake — as in Exploration/Compression Algorithm

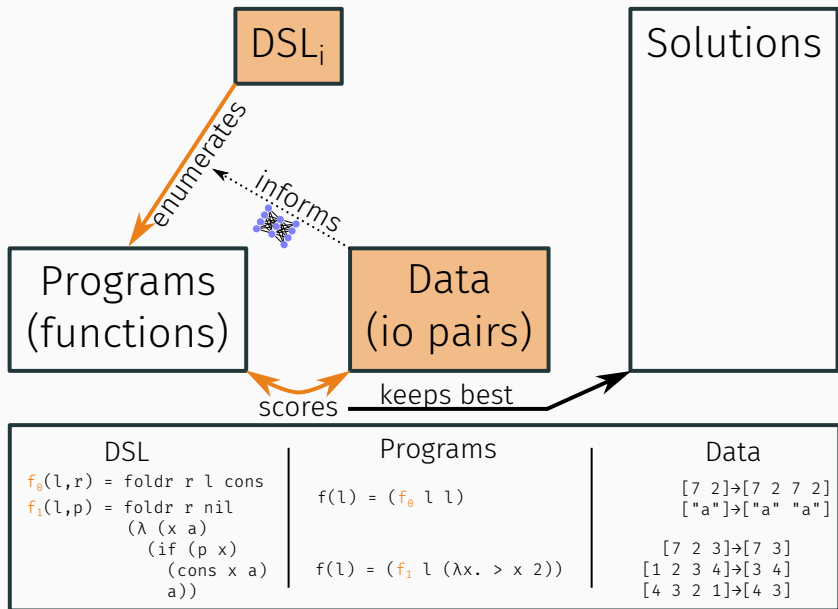


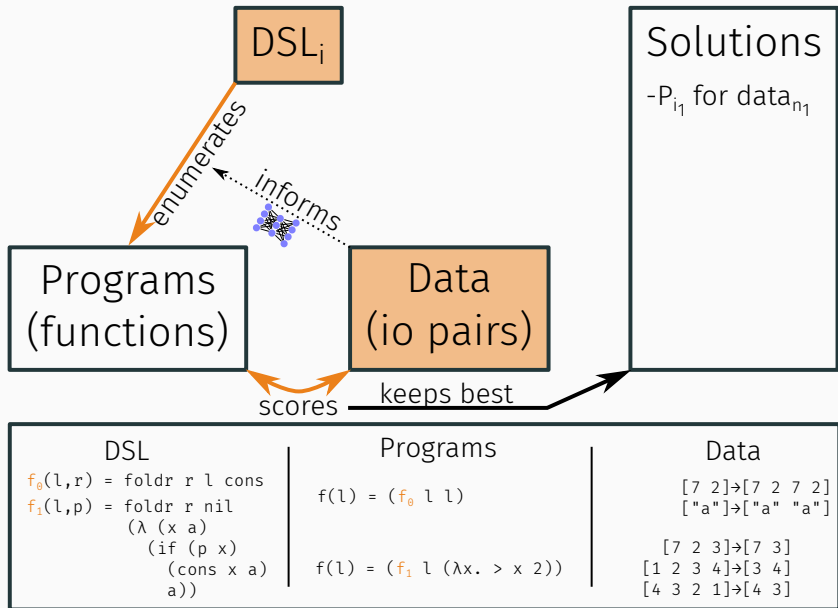
Wake — as in Exploration/Compression Algorithm

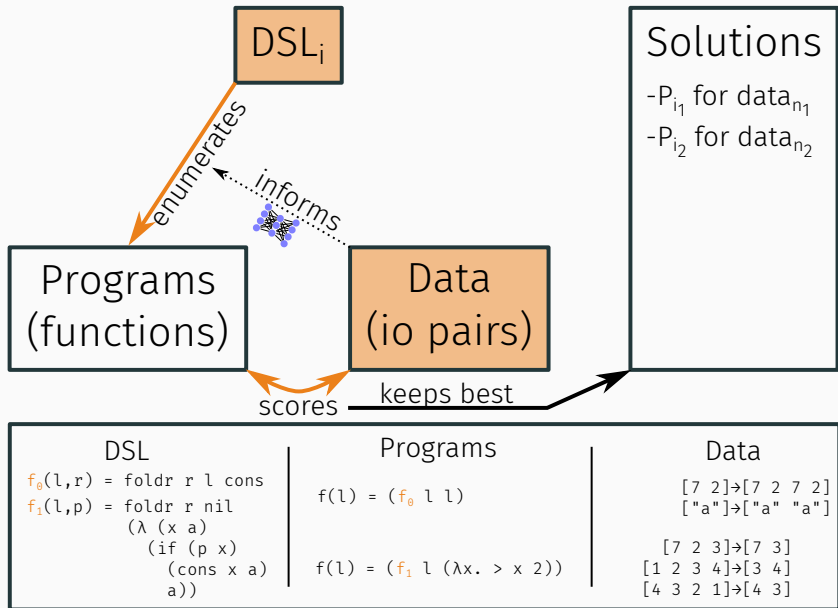


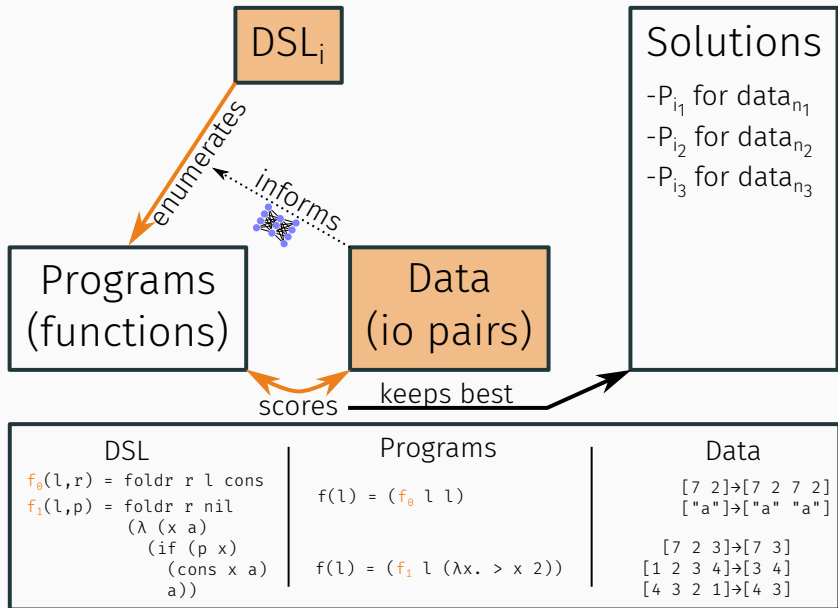
Wake — as in Exploration/Compression Algorithm

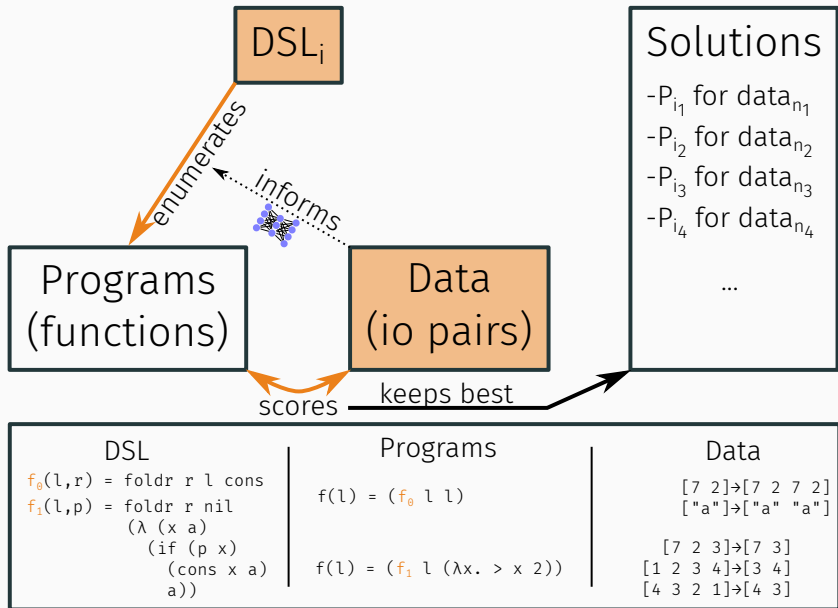




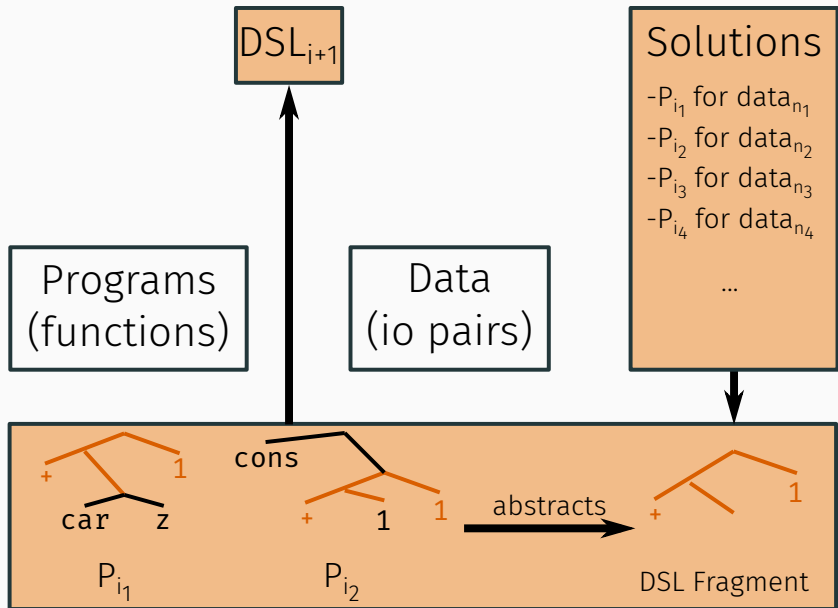




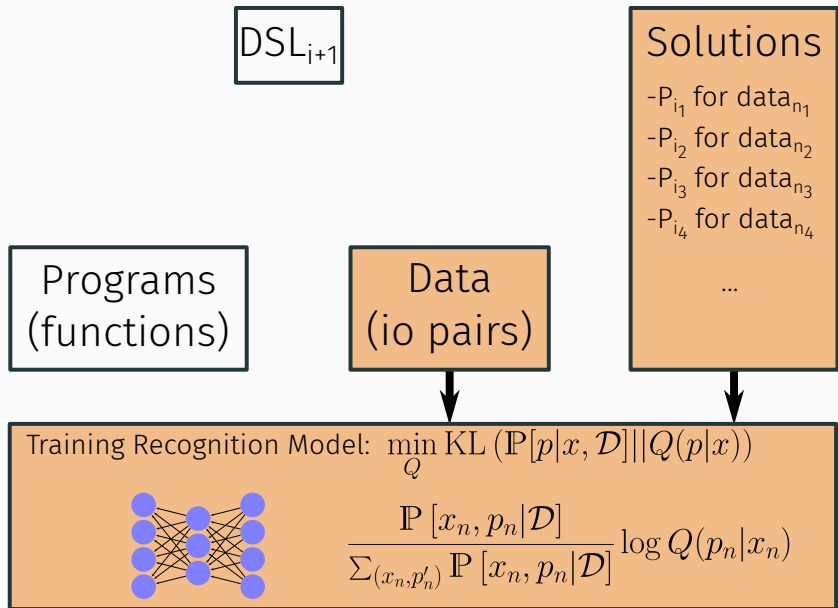




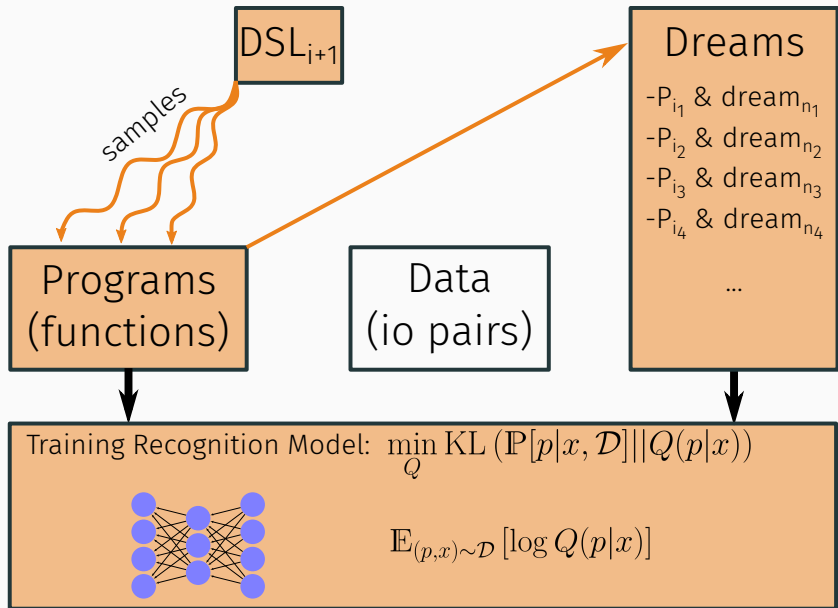
DreamCoder — Sleep-G



DreamCoder — Sleep-R (Experience Replay)



DreamCoder — Sleep-R (Dreaming)

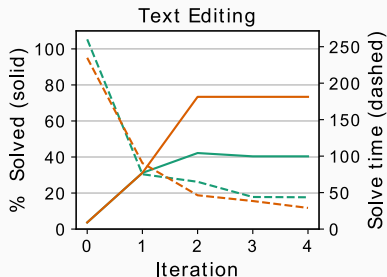
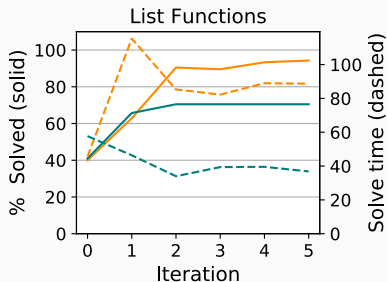


List functions — Created & investigated by Lucas Morales

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.

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

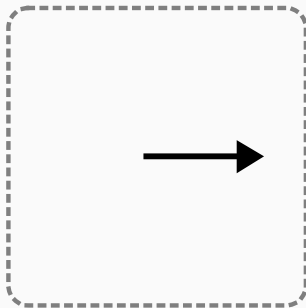
OP ::= FW x | RT x | UP | DOWN | SET state

Tasks

task : image



FW 1



DSL

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

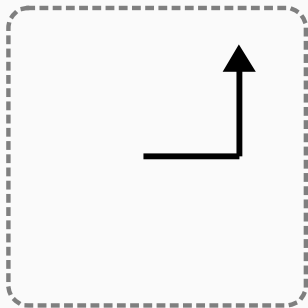
Tasks

task : image

```
FW 1
```

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

```
FW 1
```



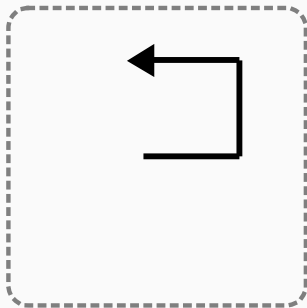
DSL

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
```



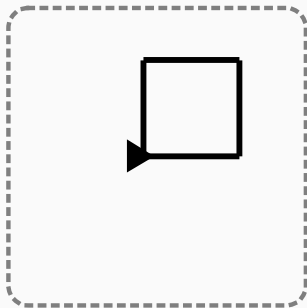
DSL

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

Tasks

task : image

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



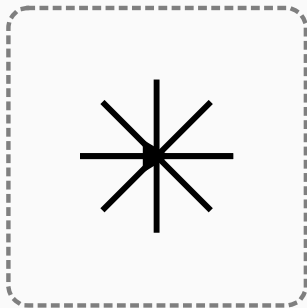
DSL

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

Tasks

task : image

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



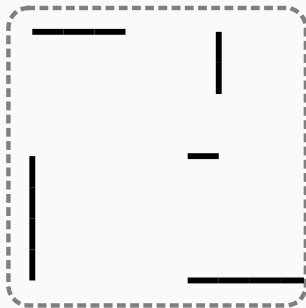
DSL

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



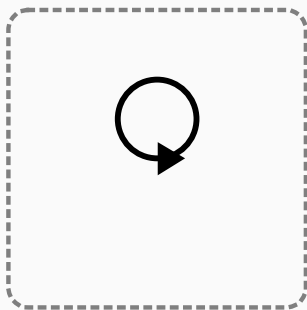
DSL

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

Tasks

task : image

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



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

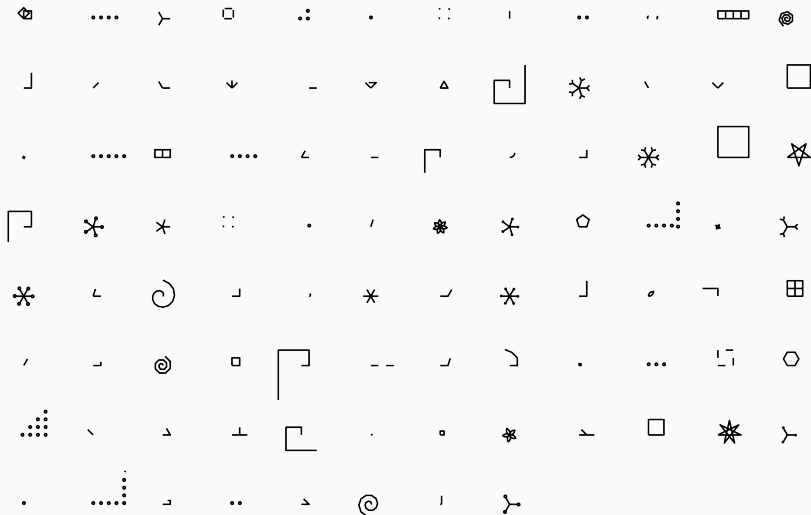
task : image

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

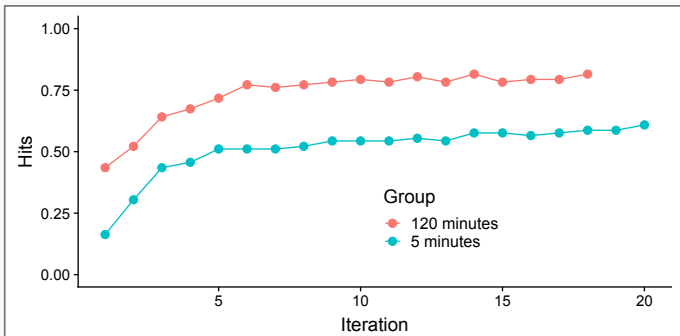


NUM ::= 1 | π | ∞ | ε | + | - | * | /

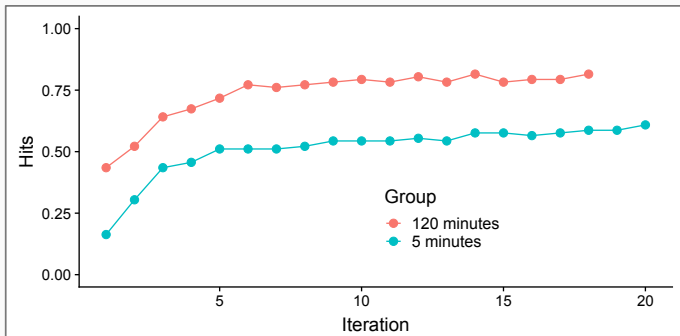
Turtle graphics — Training tasks



Turtle graphics — Learning curves



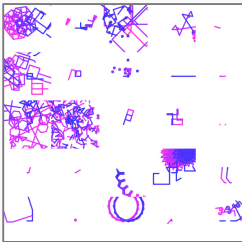
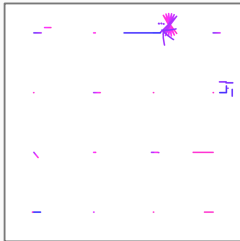
Turtle graphics — Learning curves



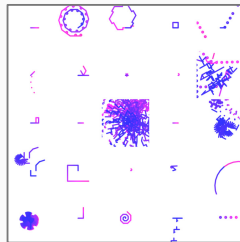
- $\frac{\pi}{2}$ and $\frac{\pi}{4}$ from π , 2, + and /
- A line of length n followed with a right angle
- Loops of length n that uses the number n inside.
- Unit line then teleport back to origin
- ...

Turtle graphics — Dreams

Before training

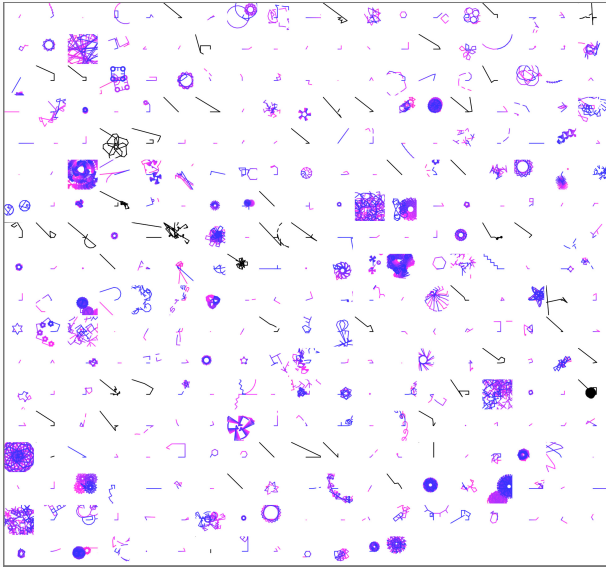


Plateau 5 minutes

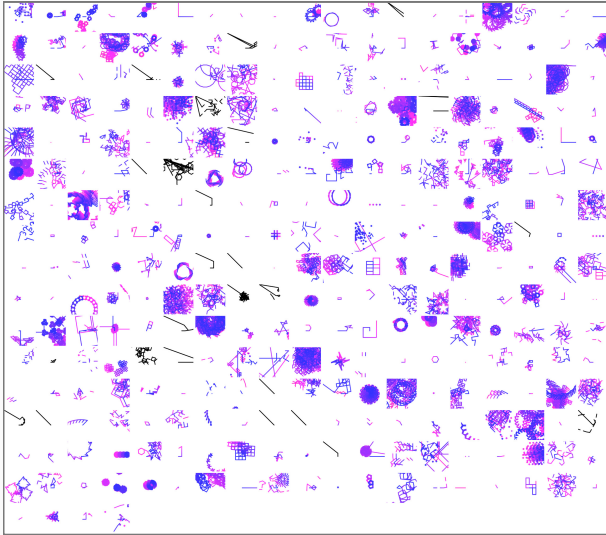


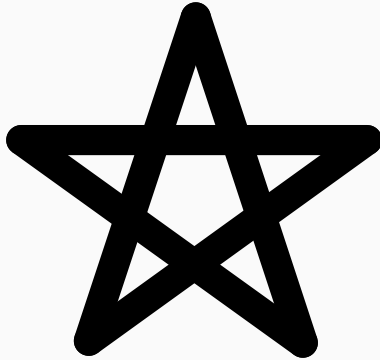
Plateau 2 hours

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

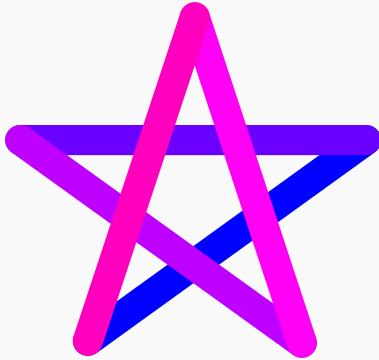


Turtle graphics — More dreams, 5 minutes, last iteration

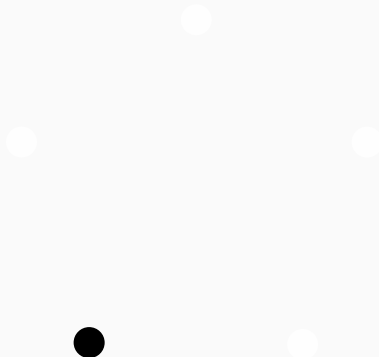




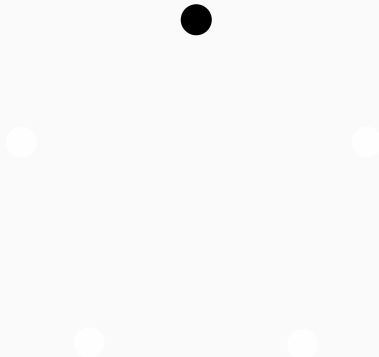
Turtle graphics — All the way down



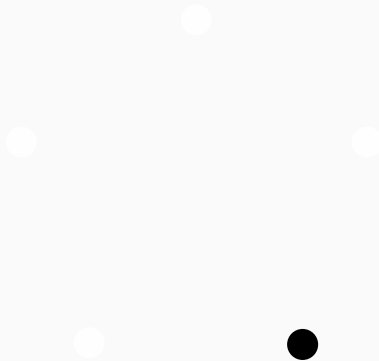
Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



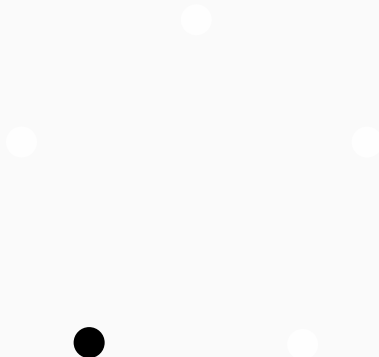
Turtle graphics — All the way down



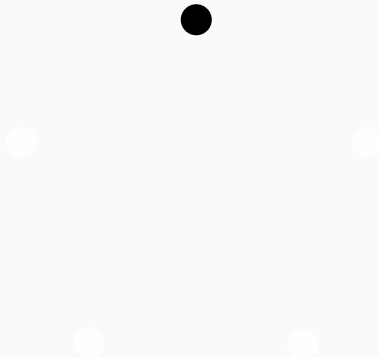
Turtle graphics — All the way down



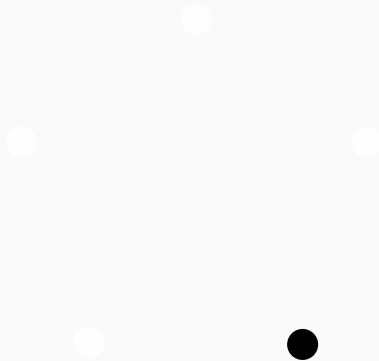
Turtle graphics — All the way down



Turtle graphics — All the way down



Turtle graphics — All the way down



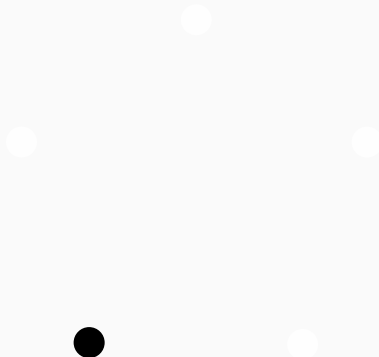
Turtle graphics — All the way down



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))))

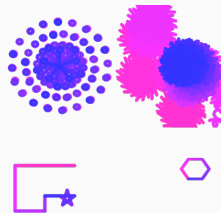
(f2: unfold)
f3(i,l) = (if (= i 0) (car l)
              (f3 (f1 i) (cdr l)))

(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	
	
$f(x) = (f_1 \ x)$	$f(x) = (f_6 \ x)$
	
$f(x) = (f_4 \ x)$	$f(x) = (f_3 \ x)$
$f_0(x) = (+ \ x \ \text{real})$ $f_1(x) = (f_0 \ (* \ \text{real} \ x))$ $f_2(x) = (f_1 \ (* \ x \ (f_0 \ x)))$ $f_3(x) = (f_0 \ (* \ x \ (f_2 \ x)))$ $f_4(x) = (f_0 \ (* \ x \ (f_3 \ x)))$ <i>(f4: 4th order polynomial)</i> $f_5(x) = (/ \ \text{real} \ x)$ $f_6(x) = (f_5 \ (f_0 \ x))$ <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)

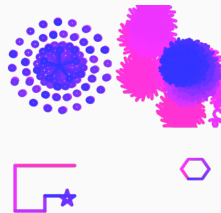
f4(f,l,x) = (if (empty? l) x
                (f (car l) (f4 (cdr l))))

(f4: fold)

f5(f,l) = (if (empty? l) nil
              (cons (f (car l)) (f5 (cdr l))))

(f5: map)
```

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



The End.