DreamCoder: Bootstrapping Inductive Program Synthesis with Wake-Sleep Library Learning

Kevin Ellis[1] Catherine Wong[2] Maxwell Nye[2] Mathias Sablé-Meyer[3]
Lucas Morales[2] Luke Hewitt[2] Luc Cary[2]
Armando Solar-Lezama[2] Joshua B. Tenenbaum[2]

2021

PLDI. [1]Cornell; [2]MIT; [3]PSL/Collège de France & NeuroSpin

Inductive program synthesis

Program synthesis: automatically write computer programs from a specification of what they should do

Inductive program synthesis

Program synthesis: automatically write computer programs from a specification of what they should do

Inductive program synthesis: Learning programs from examples

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:

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

FlashFill (Gulwani 2012)

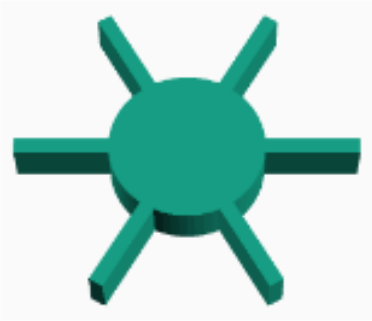
EXAMPLE 3 (Directory Name Extraction). *Consider the following example taken from an excel online help forum.*

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

String Program:

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

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.*

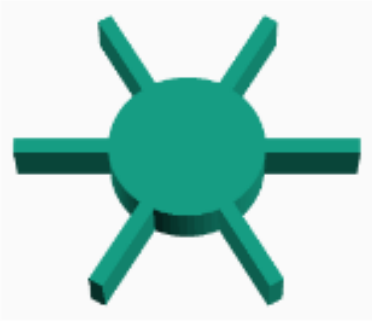
Input v_1	*Output*
Company\Code\index.html	*Company\Code*
Company\Docs\Spec\specs.doc	*Company\Docs\Spec*

String Program:

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

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

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

op	::=	+ | - | × | /
num	::=	$\mathbb{R}$ | ⟨*var*⟩ | ⟨*num*⟩ ⟨*op*⟩ ⟨*num*⟩
vec2	::=	[⟨*num*⟩, ⟨*num*⟩]
vec3	::=	[⟨*num*⟩, ⟨*num*⟩, ⟨*num*⟩]
affine	::=	Translate | Rotate | Scale | **TranslateSpherical**
binop	::=	Union | Difference | Intersection
cad	::=	(Cuboid ⟨*vec3*⟩) | (Sphere ⟨*num*⟩)
	|	(Cylinder ⟨*vec2*⟩) | (HexPrism ⟨*vec2*⟩) | …
	|	(⟨*affine*⟩ ⟨*vec3*⟩ ⟨*cad*⟩)
	|	(⟨*binop*⟩ ⟨*cad*⟩ ⟨*cad*⟩)
	|	**(Fold** ⟨*binop*⟩ ⟨*cad-list*⟩**)**
cad-list	::=	(List ⟨*cad*⟩+)
	|	(Concat ⟨*cad-list*⟩+)
	|	(Tabulate (⟨*var*⟩ $\mathbb{Z}^+$)+ ⟨*cad*⟩)
	|	(Map2 ⟨*affine*⟩ ⟨*vec3-list*⟩ ⟨*cad-list*⟩)
vec3-list	::=	(List ⟨*vec3*⟩+)
	|	(Concat ⟨*vec3-list*⟩+)
	|	(Tabulate (⟨*var*⟩ $\mathbb{Z}^+$)+ ⟨*vec3*⟩)

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

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]
...

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]
...

Library learning

Initial Primitives

```
⋮
map
fold
if
cons
>
⋮
```

Sample Problem: `sort list`

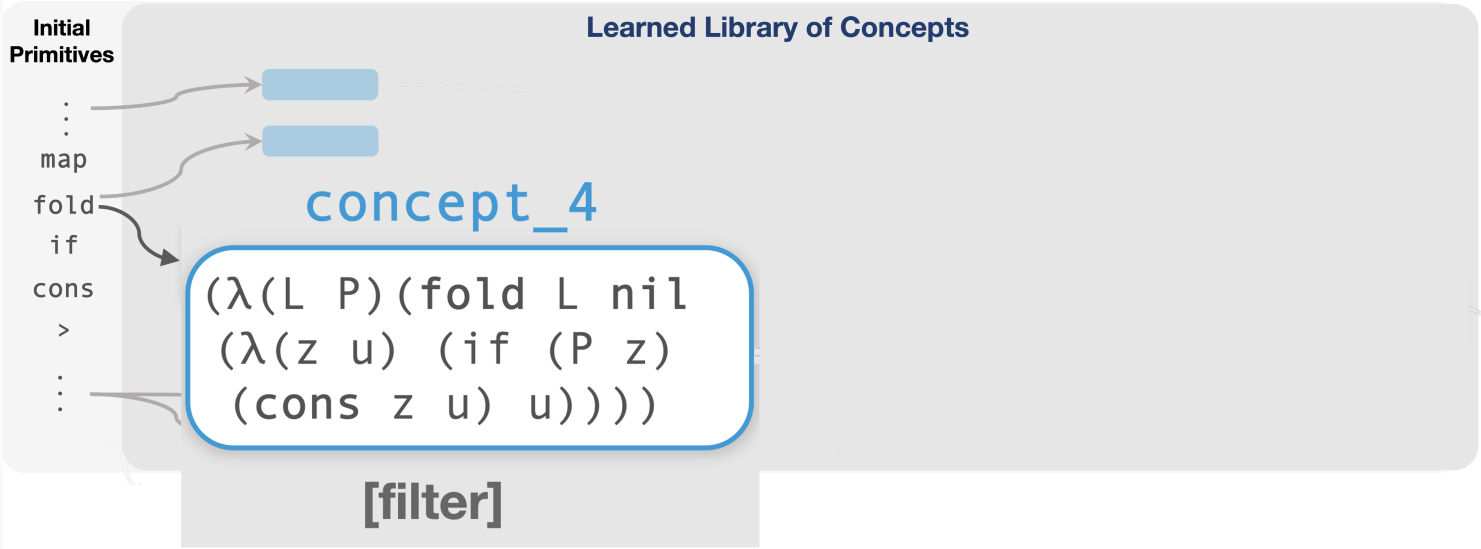
[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

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]
...

Library learning



Sample Problem: `sort list`

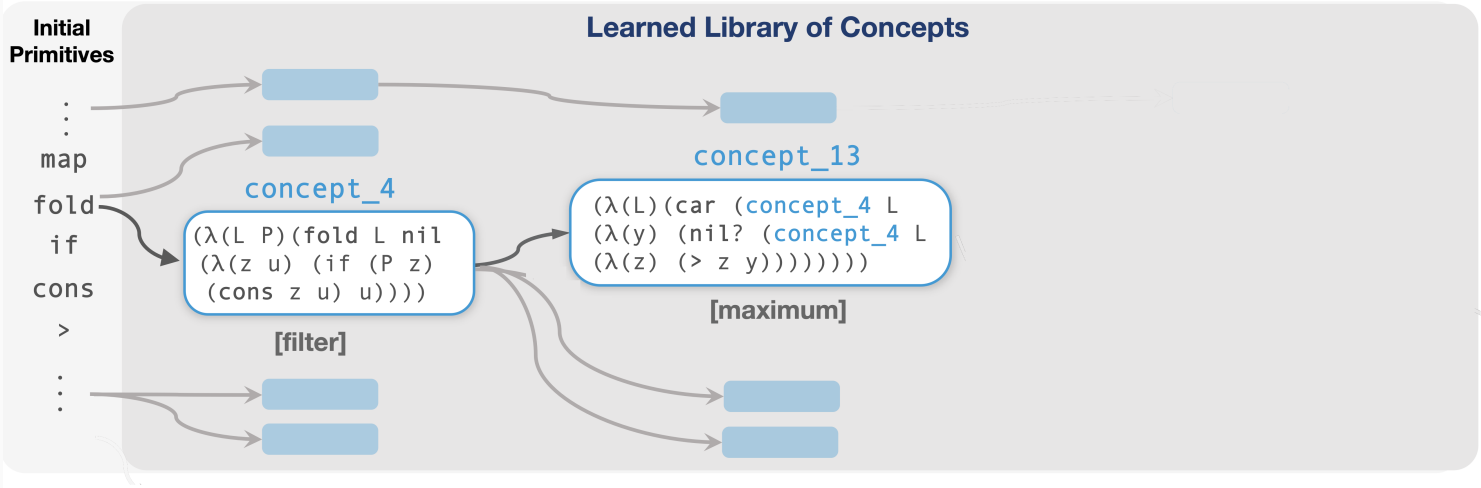
[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

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]
...

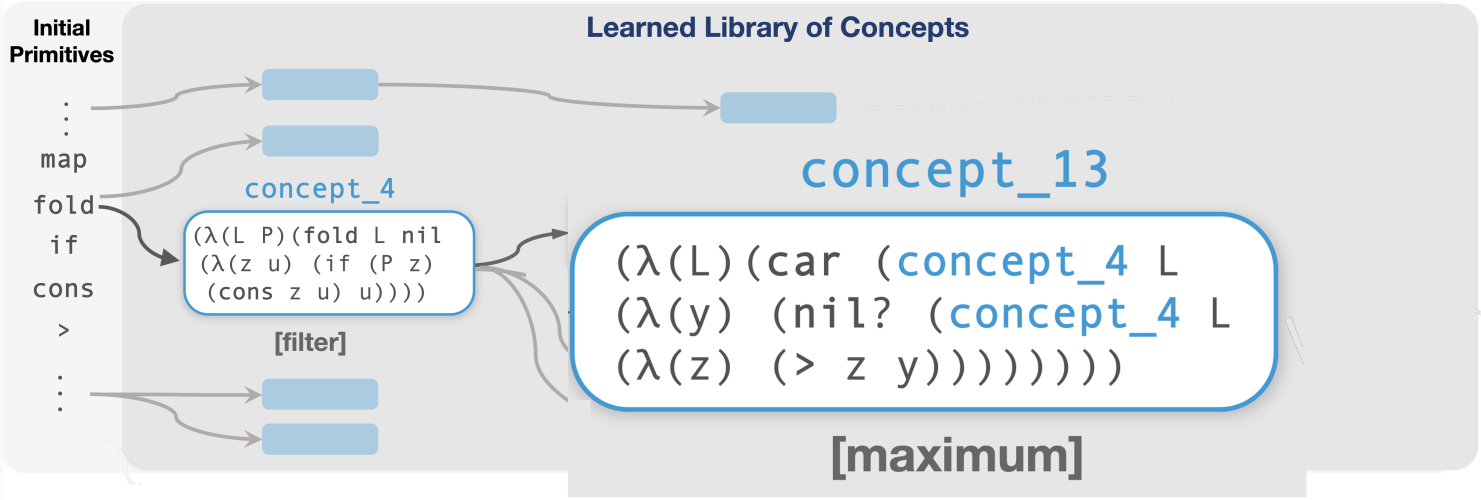
Library learning



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]
...

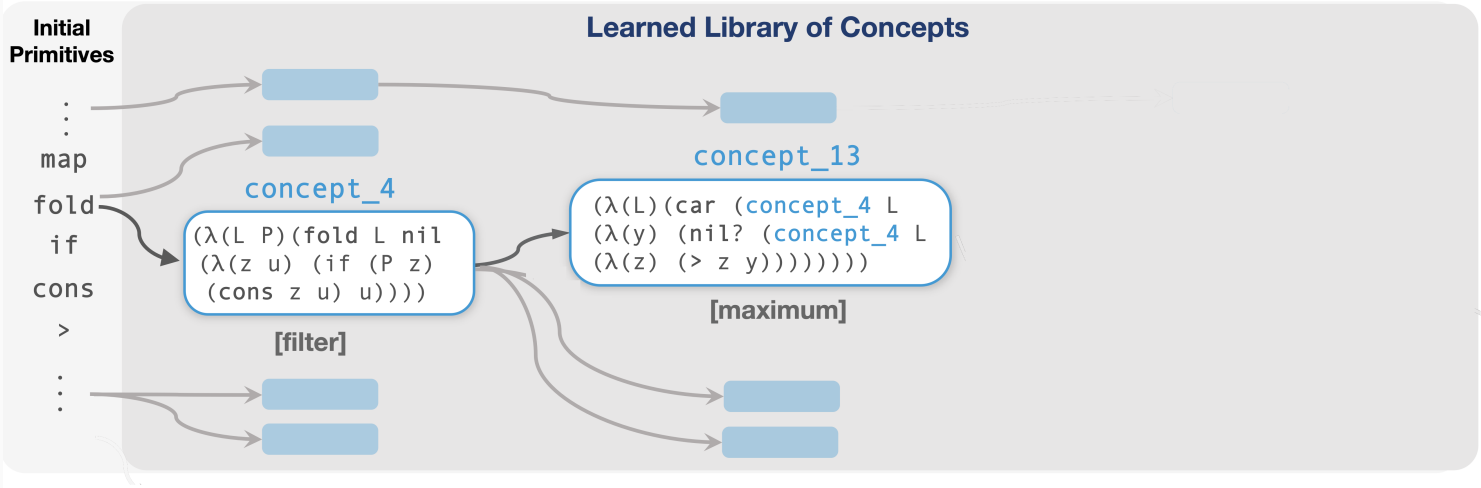
Library learning



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]
...

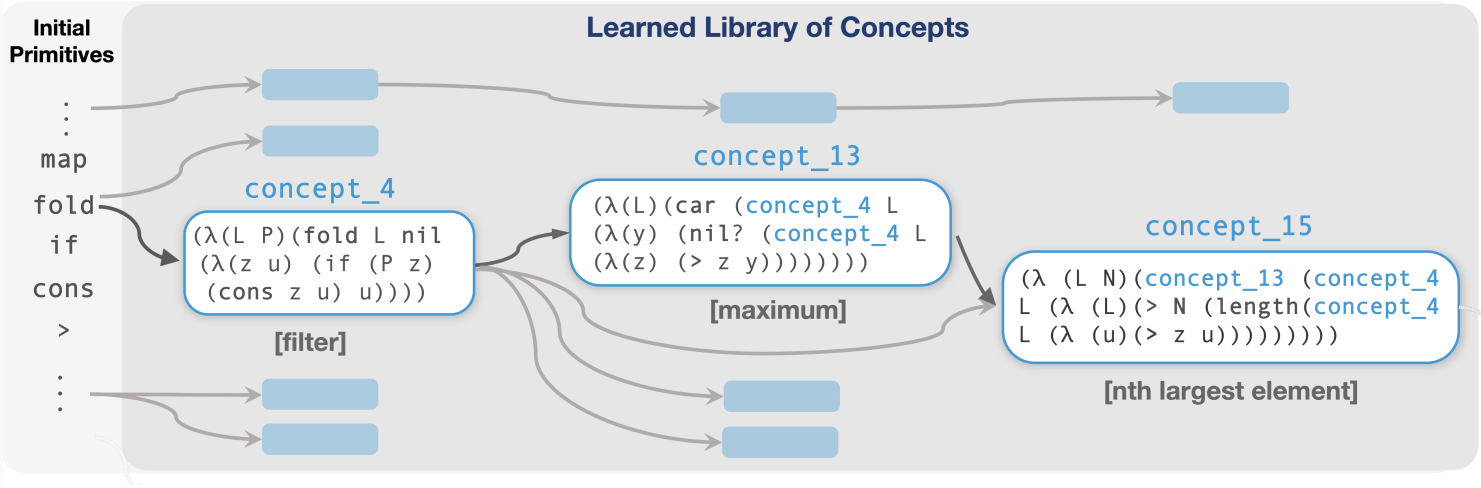
Library learning



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]
...

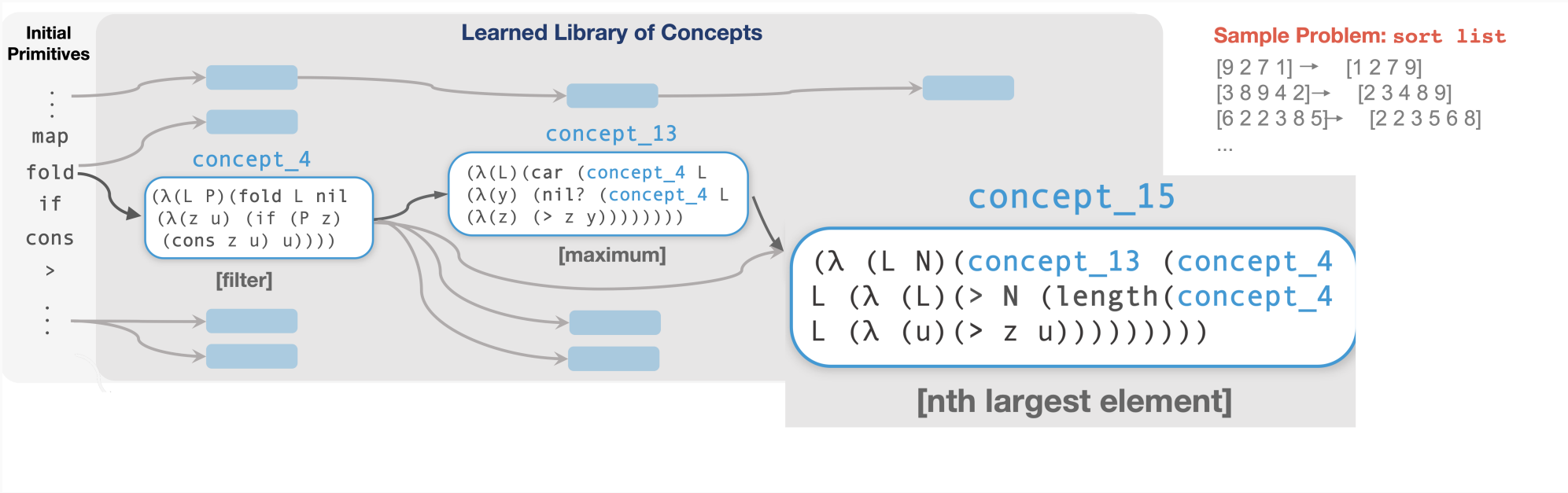
Library learning



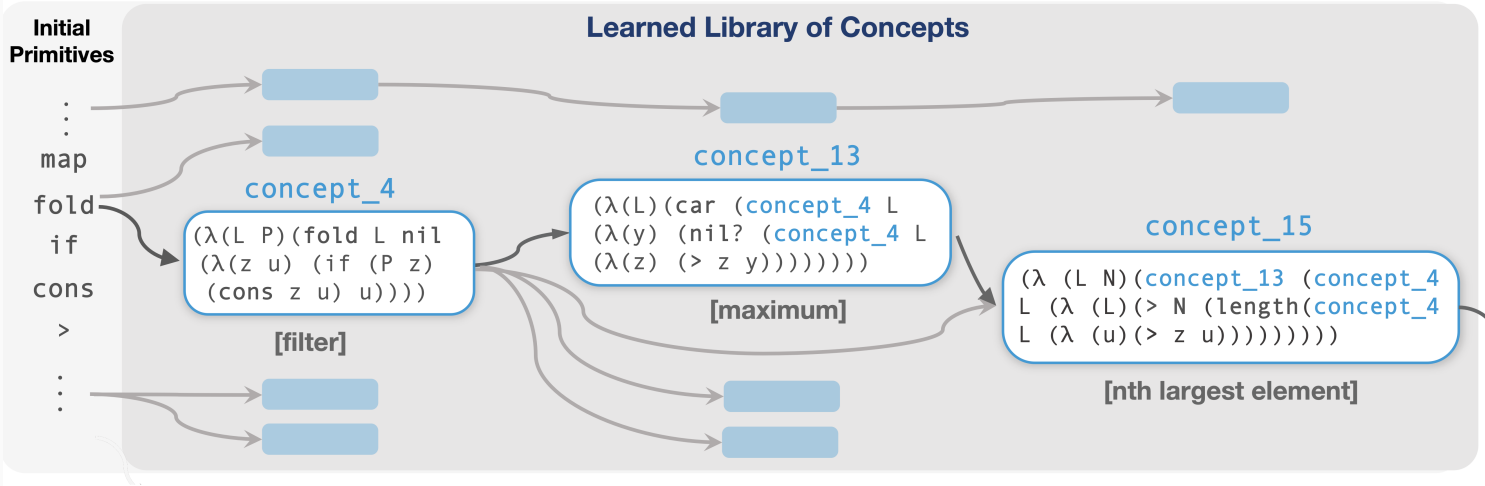
Sample Problem: `sort list`

[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]
...

Library learning



Library learning



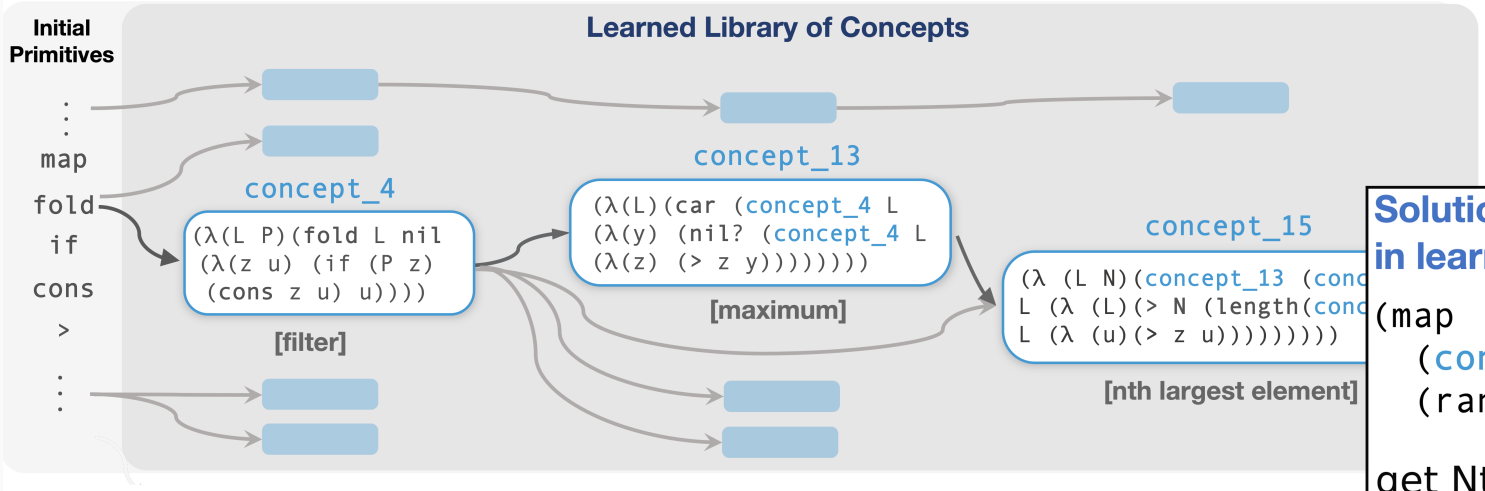
Sample Problem: `sort list`

[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]
...

Solution to `sort list` discovered in learned language:

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

Library learning



Sample Problem: `sort list`

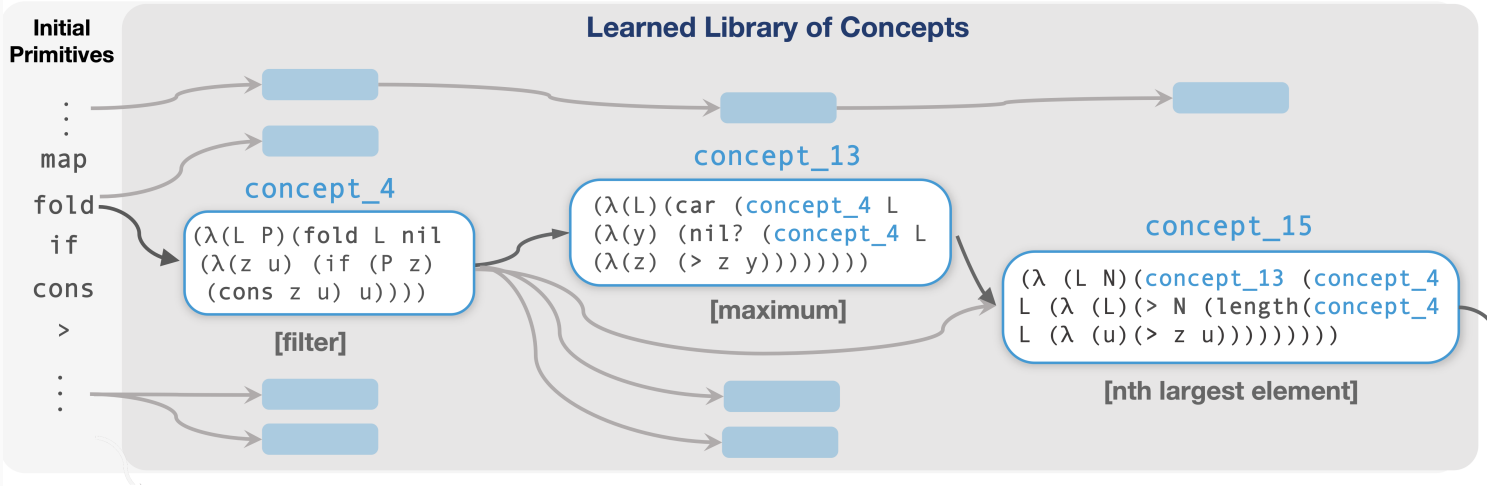
[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

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, ...

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]
...

Solution to `sort list` discovered in learned language:

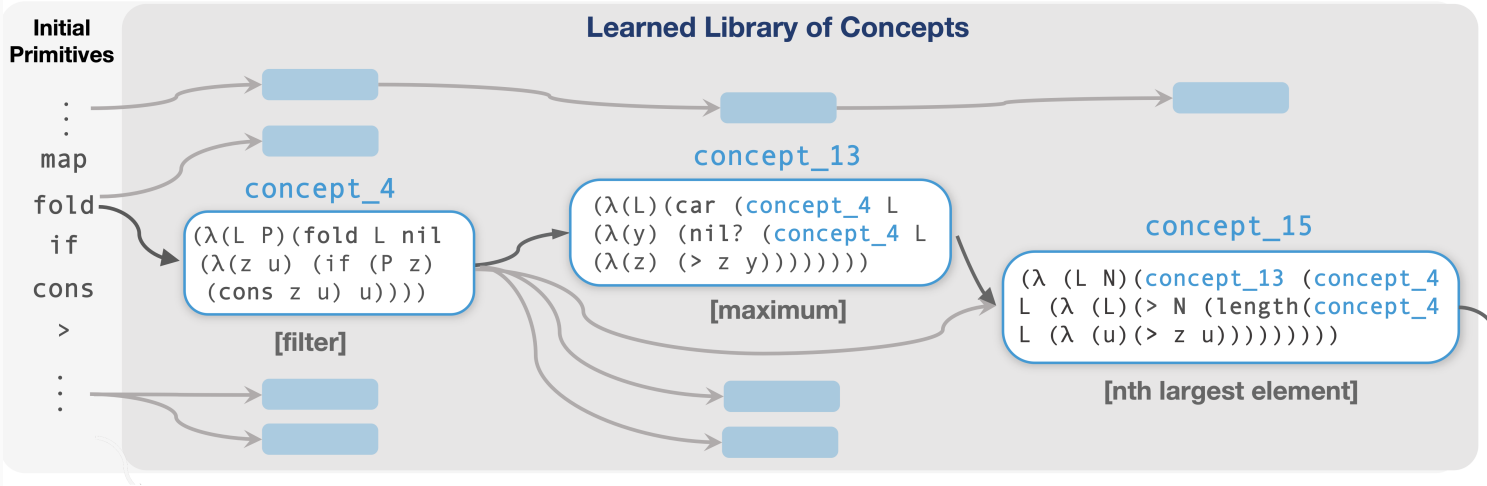
```
(map  (λ (n)
  (concept_15 L (+ 1 n)))
  (range (length L)))
```

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

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))))
```

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]
...

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, ...

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

DreamCoder

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

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

Three unknowns:

- programs
- library
- neural net weights

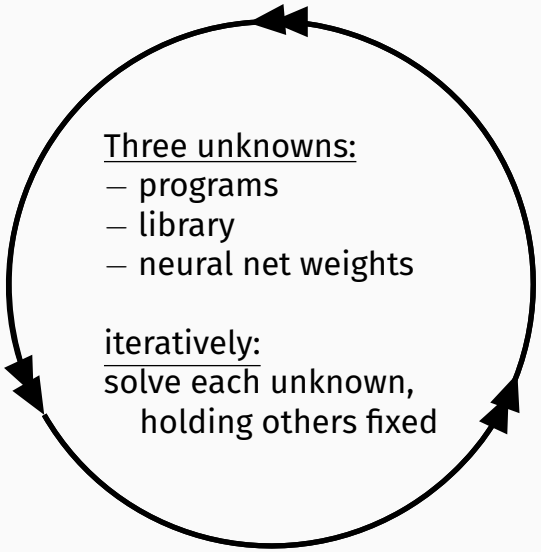
Three unknowns:
- programs
- library
- neural net weights

iteratively:
solve each unknown,
holding others fixed

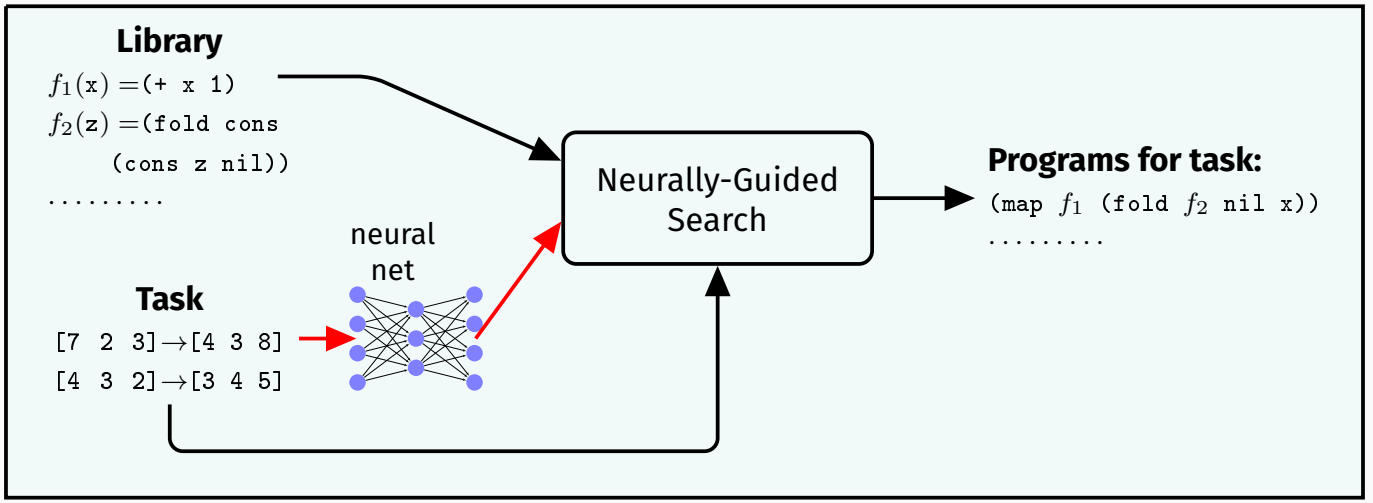
Wake

Sleep: Abstraction

Sleep: Dreaming

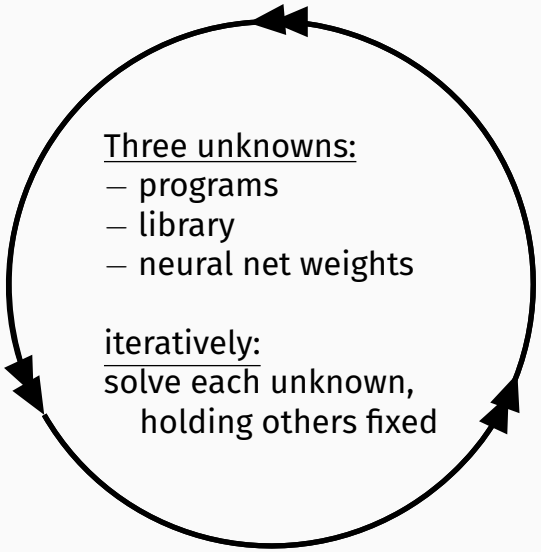


WAKE

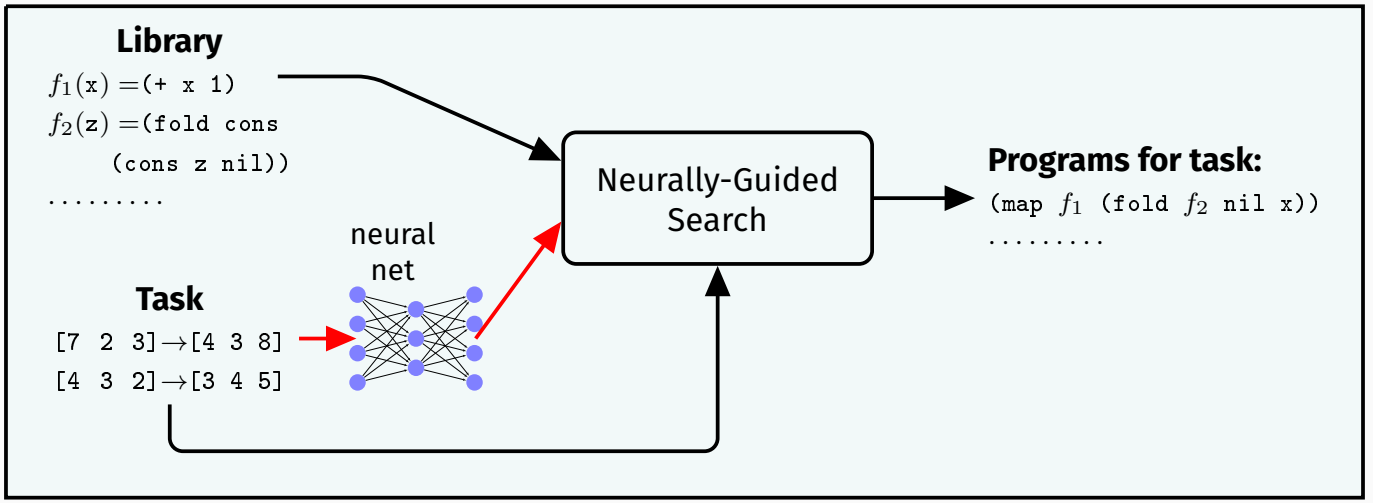


SLEEP: ABSTRACTION

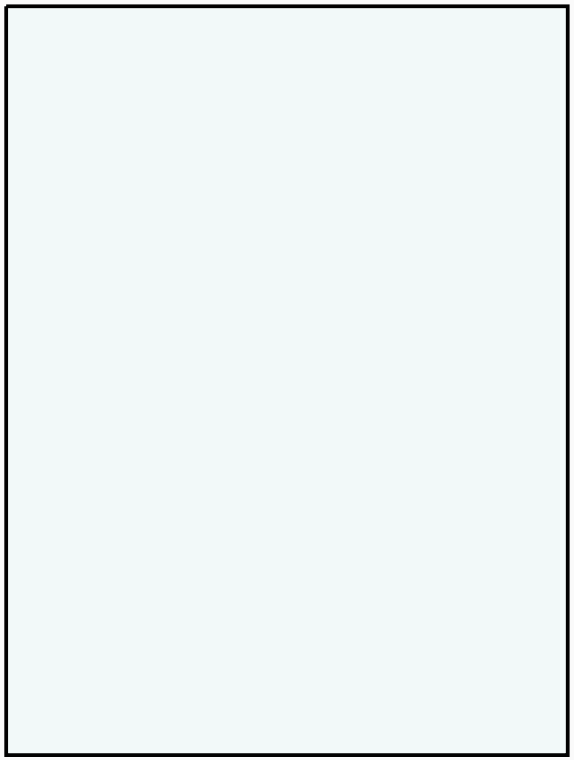
SLEEP: DREAMING

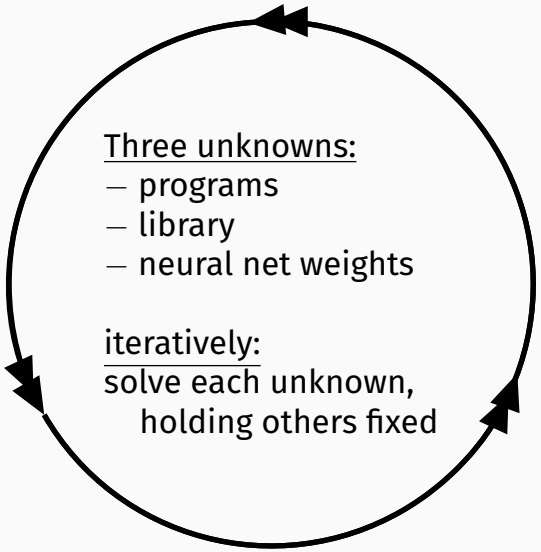


WAKE

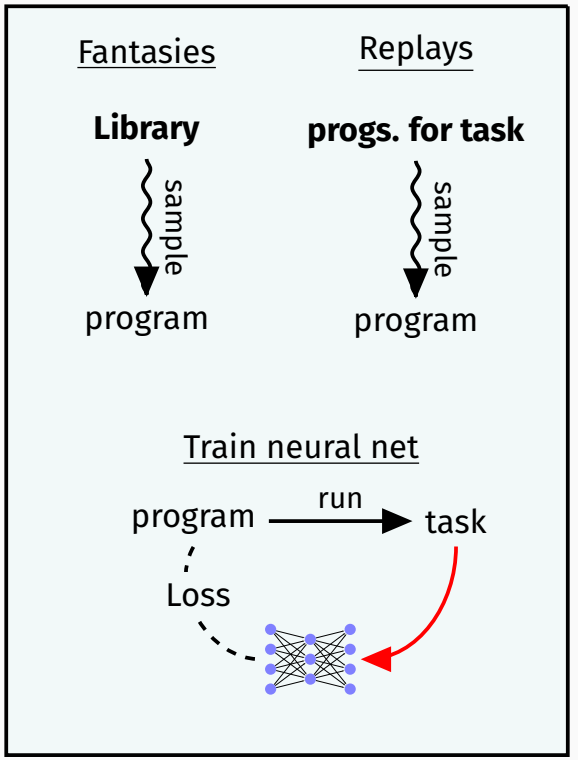


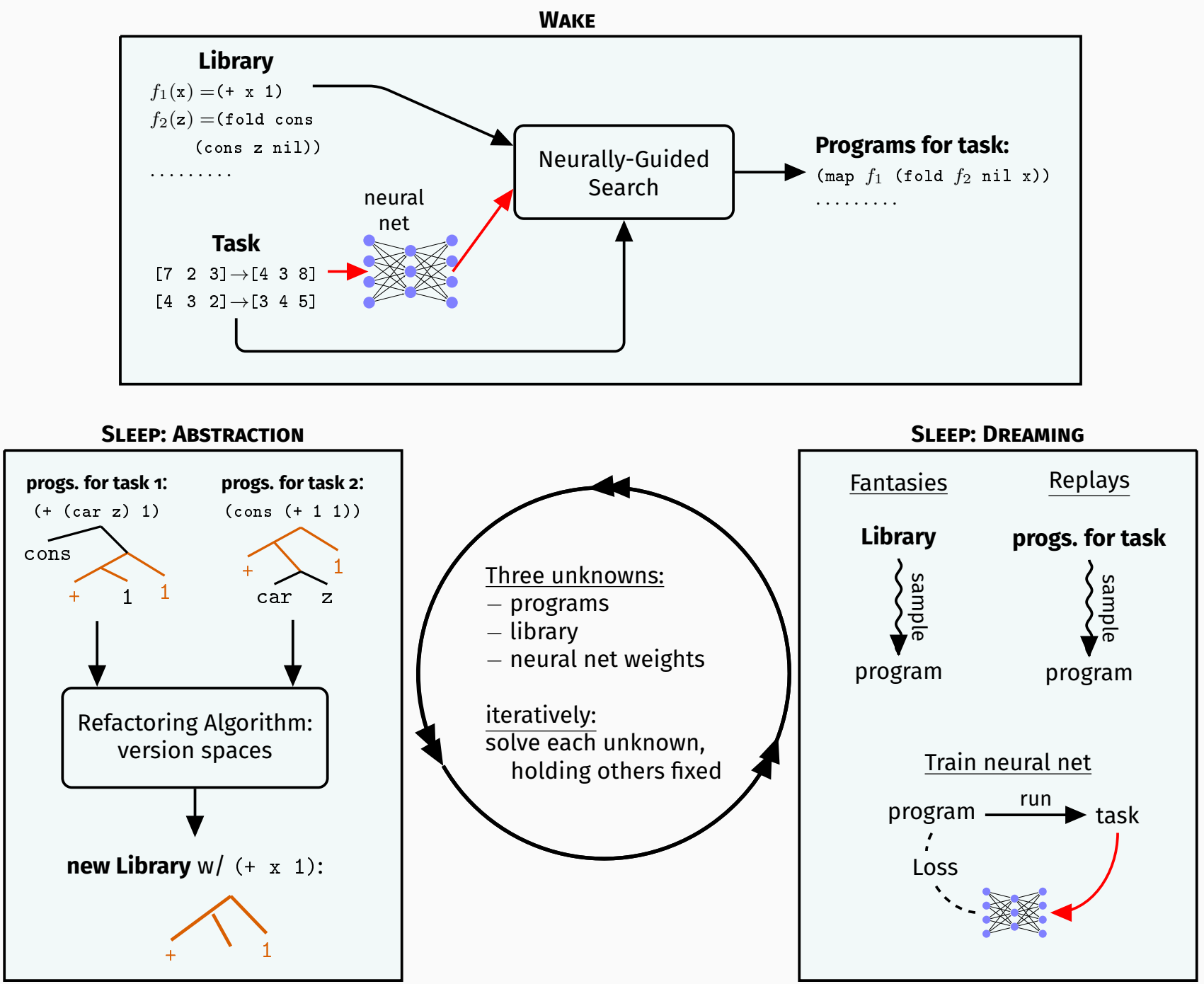
SLEEP: ABSTRACTION





SLEEP: DREAMING




WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
neural net
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(+ (car z) 1)
progs. for task 2:
(cons (+ 1 1))
cons
+
1
1
+
1
car
z
Refactoring Algorithm: version spaces
new Library w/ (+ x 1):
+
1
Three unknowns:
– programs
– library
– neural net weights
iteratively:
solve each unknown,
holding others fixed
SLEEP: DREAMING
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train neural net
program
run
task
Loss

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

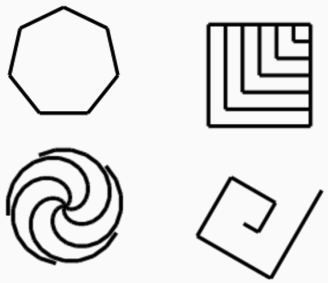
Phone numbers

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

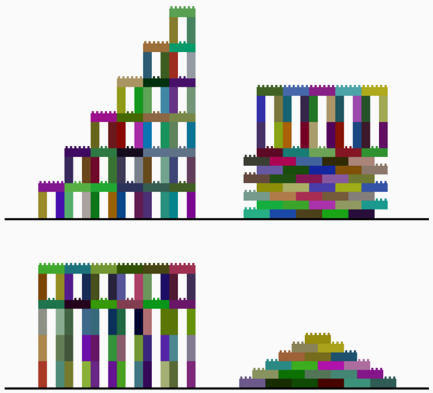
Currency

```
$100.25
$4.50
```

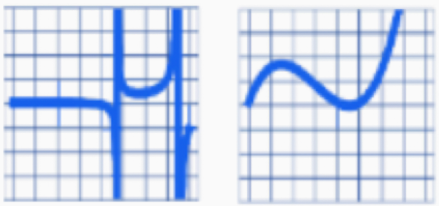
LOGO Graphics



Block Towers



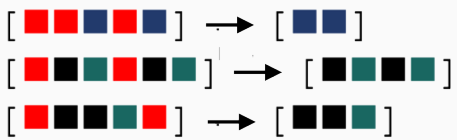
Symbolic Regression



$$y = f(x)$$

Recursive Programming

Filter Red



Physical Laws

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

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