

# The Role of Higher-level Knowledge in Discovery Problems: Programs and Hierarchical Bayes

---

Kevin Ellis

2021 Workshop on Artificial Scientific Discovery

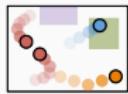
Cornell

1. Scientific models
2. Discovery
3. Higher-level knowledge

1. Scientific models as **generative programs**
2. Discovery
3. Higher-level knowledge

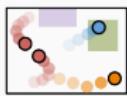
1. Scientific models as **generative programs**
2. Discovery as **program synthesis**
3. Higher-level knowledge

1. Scientific models as **generative programs**
2. Discovery as **program synthesis**
3. Higher-level knowledge as **Bayesian prior** over programs



Ullman et al 2014:  
Learning physics from dynamical scenes

*Initial conditions*



Property  
values

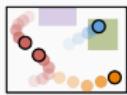
- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

Force  
parameters

*Force b/w reds: attract*

Ullman et al 2014:  
Learning physics from dynamical scenes

*Initial conditions*



Property  
values

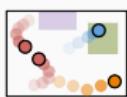
- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

Force  
parameters

*Force b/w reds: attract*

Ullman et al 2014:  
Learning physics from dynamical scenes

*Initial conditions*



```
(define scenario
  (let* (
    (init-cond (sample-init world-entities))
    (run-dynamics world-entities
      world-forces init-cond steps dt)))
```

Property  
values

- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

```
(define world-entities
  (map sample-values enti...))
```

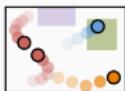
Force  
parameters

```
Force b/w reds: attract
(define world-forces
  (map sample-parameters :...))
```

Ullman et al 2014:  
Learning physics from dynamical scenes

## Level 0 (data)

### Initial conditions



```
(define scenario
  (let* (
    (init-cond (sample-init world-entities))
    (run-dynamics world-entities
      world-forces init-cond steps dt)))
```

## Level 1

### Property values

- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

```
(define world-entities
  (map sample-values enti:
```

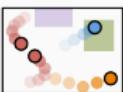
### Force parameters

Force b/w reds: attract  
(define world-forces  
 (map sample-parameters :

Ullman et al 2014:  
Learning physics from dynamical scenes

## Level 0 (data)

### Initial conditions



```
(define scenario
  (let* (
    (init-cond (sample-init world-entities))
    (run-dynamics world-entities
      world-forces init-cond steps dt)))
```

## Level 1

### Property values

- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

Force b/w reds: attract (define world-forces  
(map sample-parameters :

### Force parameters

## Level 2

### Entity types

*Puck:* ○ (define puck (make-entity  
pos shape mass vel ...))  
*Surface:* □ (define surface (make-entity  
pos shape friction ...))

### Force classes

### Properties

*Mass* (define (mass)  
(pair "mass" (uniform '(1 3 9))))  
*Friction* (define (friction)  
(pair "friction" (uniform '(0 5 20)))

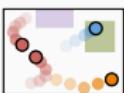
*Pairwise:* ○ → (define (pairwise-force c1 c2)  
(let\* ((d (uniform-draw '(-1 0 1)))  
(lambda (c1 c2)  
(let ((r (euc-dist c1 c2))  
(/ (\* (del(c1,col(c1)) del(c2,col  
(power r 2)))))))

*Global:* ○ → (define (global-force)  
(let\* ((d (uniform-draw compass-dir  
(lambda (o) (\* k d)))))

Ullman et al 2014:  
Learning physics from dynamical scenes

## Level 0 (data)

### Initial conditions



```
(define scenario
  (let* (
    (init-cond (sample-init world-entities))
    (run-dynamics world-entities
      world-forces init-cond steps dt)))
```

## Level 1

### Property values

- : large mass
- : medium mass
- : small mass
- : high friction
- : no friction

```
(define world-entities
  (map sample-values enti...))
```

### Force parameters

```
Force b/w reds: attract (define world-forces
  (map sample-parameters :...))
```

## Level 2

### Entity types

*Puck:* ○ (define puck (make-entity  
pos shape mass vel ...))  
*Surface:* □ (define surface (make-entity  
pos shape friction ...))

### Force classes

*Pairwise:* ○ → (define (pairwise-force c1 c2)  
 (let\* ((d (uniform-draw '(-1 0 1))))  
 (lambda (c1 c2)  
 (let ((r (euc-dist c1 c2)))  
 (/ (\* del(c1,col(c1)) del(c2,col  
 (power r 2)))))))

### Properties

*Mass* (define (mass)  
 (pair "mass" (uniform '(1 3 9))))  
*Friction* (define (friction)  
 (pair "friction" (uniform '(0 5 20))))

*Global:* ○ → (define (global-force)  
 (let\* ((d (uniform-draw compass-dir  
 (lambda (o) (\* k d)))))

## Level N

*Entity* (define (make-entity property1 property2 ...)  
 (list property1 property2 ...))

Ullman et al 2014:

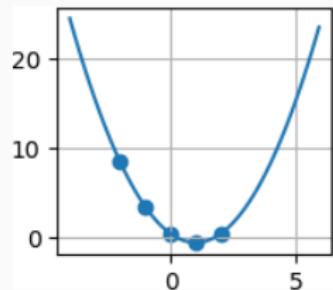
*Newtonian Dynamics* (define (run-dynamics entities forces  
 init-cond steps dt)  
 (if (= steps 0) '()  
 (let\* ((m (get-mass entities))  
 (F (apply-forces forces entities))  
 (a (/ F m))  
 (new-cond (integrate init-cond  
 a dt noise)))  
 (pair new-cond (run-dynamics entities

Learning physics from dynamical scenes

# Why programs?

# Why programs?

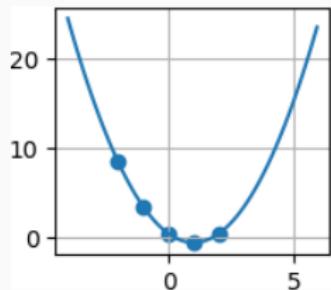
strong generalization



$$f(x) = (x-1)^2 - 0.5$$

# Why programs?

strong generalization

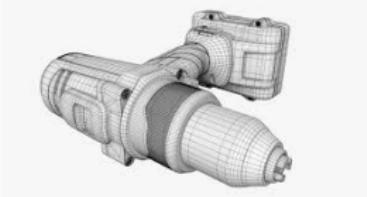


$$f(x) = (x-1)^{**2} - 0.5$$

interpretability

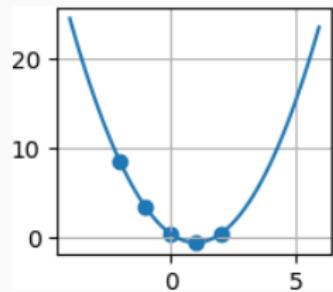


vs



# Why programs?

strong generalization

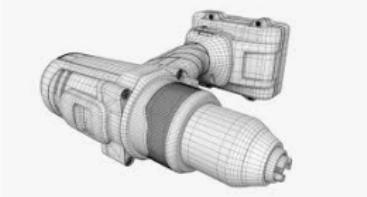


$$f(x) = (x-1)^2 - 0.5$$

interpretability



vs

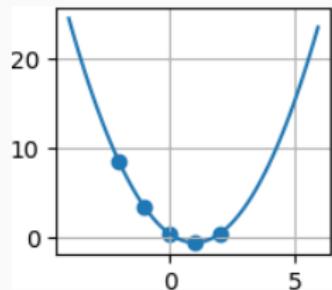


universal expressivity



# Why programs?

strong generalization



interpretability



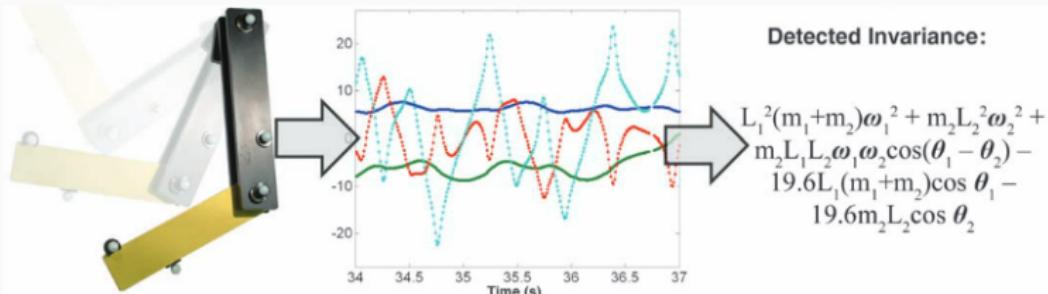
universal expressivity



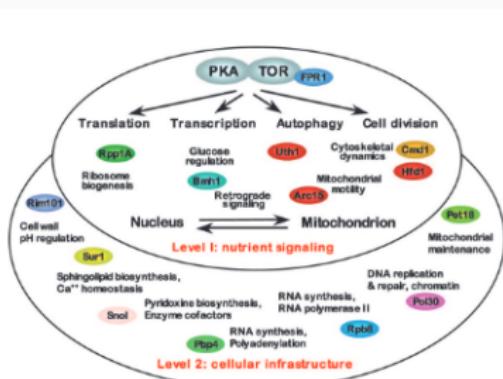
Higher-level Knowledge:

model discovery  
learning to learn

# Scientific discovery

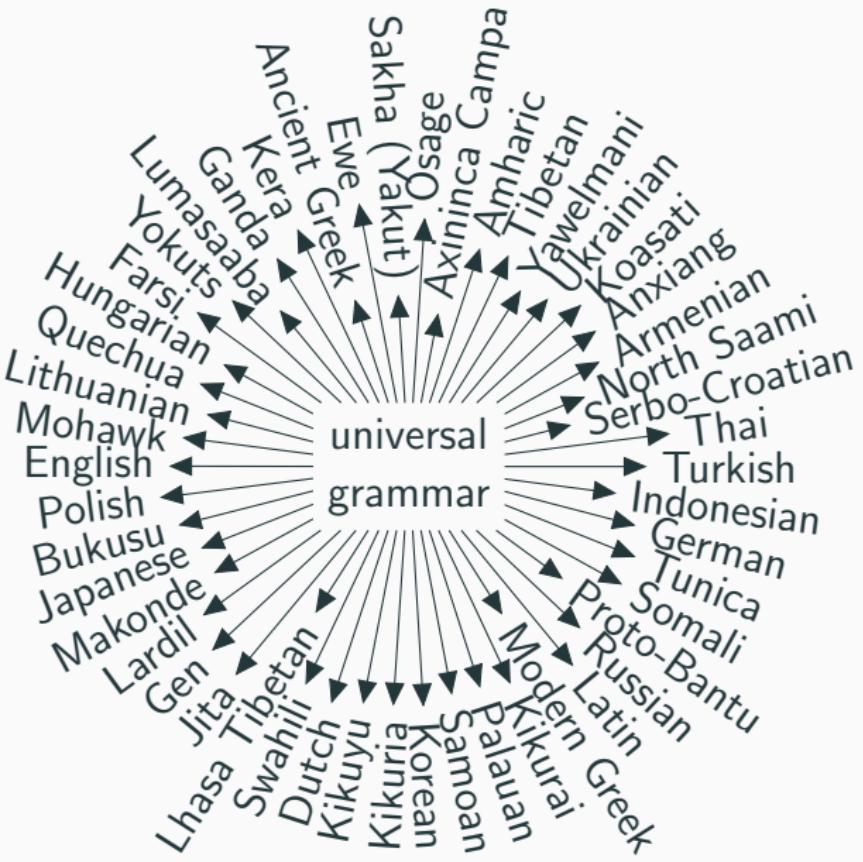


Schmidt & Lipson: "Distilling Free-Form Natural Laws from Experimental Data"



Lezon et al. 2006  
inferring genetic interaction networks

# Discovering human-understandable models of language



Tim O'Donnell



# Few-shot language learning experiment

Mandarin:

	adjective	adverb
“slow”	man	manmandə
“small”	xiao	xiaoxiaodə
“fast”	kuai	???

# Few-shot language learning experiment

Mandarin:

	adjective	adverb
“slow”	man	manmandə
“small”	xiao	xiaoxiaodə
“fast”	kuai	kuaikuaidə

# Few-shot language learning experiment

Mandarin:

	adjective	adverb
“slow”	man	manmandə
“small”	xiao	xiaoxiaodə
“fast”	kuai	kuaikuaidə

stem+stem+də

# Few-shot language learning experiment

Serbo-Croatian:

	masculine	feminine
“rich”	bogat	bogata
“mild”	blag	blaga
“green”	zelen	???

# Few-shot language learning experiment

Serbo-Croatian:

	masculine	feminine
“rich”	bogat	bogata
“mild”	blag	blaga
“green”	zelen	zelena

# Few-shot language learning experiment

Serbo-Croatian:

	mASCULINE	fEMININE
“rich”	bogat	bogata
“mild”	blag	blaga
“green”	zelen	zelena

*add “a” to stem to make feminine*

# Few-shot language learning experiment

Serbo-Croatian:

	mASCULINE	fEMININE
“rich”	bogat	bogata
“mild”	blag	blaga
“green”	zelen	zelena
“clear”	???	yasna

*add “a” to stem to make feminine*

# Few-shot language learning experiment

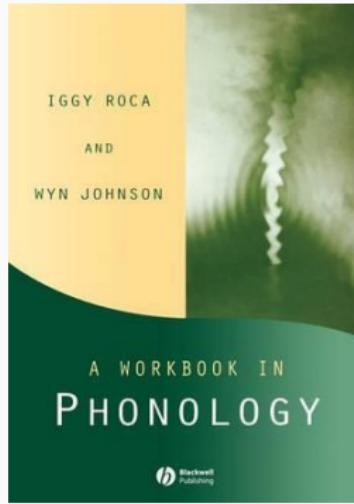
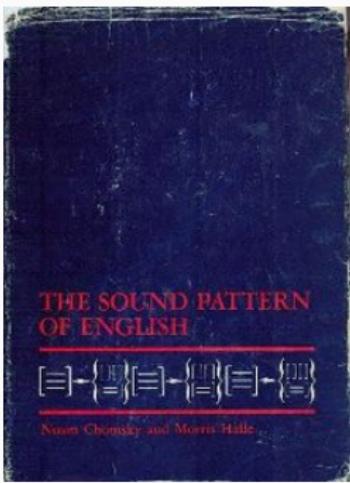
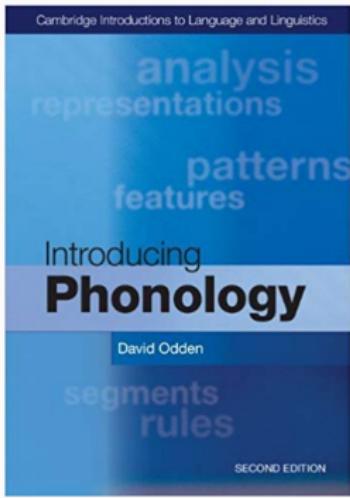
Serbo-Croatian:

	masculine	feminine
“rich”	bogat	bogata
“mild”	blag	blaga
“green”	zelen	zelena
“clear”	<b>yasan</b>	yasna

*add “a” to stem to make feminine*

*insert “a” between two word-final consonants*

$\emptyset \rightarrow a / C_C\#$



## 10 Sakha (Yakut)

Give a phonological analysis of the following case-marking paradigms of nouns in Sakha.

<i>Noun</i>	<i>Plural</i>	<i>Associative</i>	<i>oyuur</i>	<i>oyurdar</i>	<i>oyuurduun</i>	<i>'forest'</i>	
aýa	aýalar	aýaliin	'father'	üçügey	üçügeyde	'good person'	
paarta	paartalar	paartaliin	'school desk'	ejiy	ejiyde	'elder sister'	
tia	tialar	tialiin	'forest'	tomtor	tomtordor	'knob'	
kinige	kinigeler	kinigeliiñ	'book'	moyotoy	moyotoydor	'chipmunk'	
Jie	jieler	Jieliiñ	'house'	kötör	kötördör	'bird'	
iyé	iyeler	iyeliin	'mother'	bölköy	bölköydör	'islet'	
kini	kiniler	kiniliin	'3rd person'	xatiiñ	xatignar	'birch'	
bie	bieler	bieliin	'mare'	aan	aannar	'doo'	
oyo	oyolor	oyoluun	'child'	tiig	tiigner	'squirrel'	
xopto	xoptolor	xoptoluun	'gull'	sordog	sordognor	'pike'	
börö	börölör	böröliün	'wolf'	olom	olomnor	'ford'	
tial	tiallar	tialiin	'wind'	oron	oronnor	'bed'	
ial	iallar	ialliin	'neighbor'	bödög	bödögör	'strong one'	
kuul	kuullar	kuulluuñ	'sack'	<i>Noun</i>	<i>Partitive</i>	<i>Comparative</i>	<i>Ablative</i>
at	attar	attiiñ	'horse'	aýa	ayataa	ayattan	'father'
balik	baliktar	balikiin	'fish'	paarta	paartata	paartattan	'school desk'
iskaap	iskaaptar	iskaaptiin	'cabinet'	tia	tiata	tiattan	'forest'
oyus	oyustar	oyustuuñ	'bull'	kinige	kinigete	kinigetten	'book'
kus	kustar	kustuuñ	'duck'	Jie	jiete	jietten	'house'
tünnük	tünnükter	tünnüktüün	'window'	iye	iyyete	iyyetten	'mother'
sep	septer	septiiñ	'tool'	kini	kinite	kinitten	'3rd person'
et	etter	ettiiñ	'meat'	bie	biete	bietten	'mare'
örüs	örüster	örüstüün	'river'	oyo	oyoto	oyotton	'child'
tis	tiister	tiistiin	'tooth'	xopto	xoptoto	xoptotton	'gull'
sorox	soroxtor	soroxtuuñ	'some person'	börö	börötö	böröttön	'wolf'
ox	oxtor	oxtuun	'arrow'	tial	tialla	tialtan	'wind'
oloppos	oloppstor	oloppstuun	'chair'	ial	ialla	ialtan	'neighbor'
ötöx	ötöxtör	ötöxtüün	'abandoned farm'	kuul	kuulla	kuultan	'sack'
ubay	ubaydar	ubaydiin	'elder brother'	moxsoyol	moxsoyollo	moxsoyolton	'falcon'
asaray	saraydar	saraydiin	'bam'	at	atta	attan	'horse'
tiy	tiydar	tiydiin	'foal'	balik	balikta	baliktan	'fish'
atiir	atiirdar	atiirdiin	'stallion'	iskaap	iskaapta	iskaaptan	'cabinet'
			tünnük	oyus	oyusta	oyustan	'bul'
				kus	kusta	kustan	'duck'
				tünnükte	tünnükte	tünnükten	'window'

# Turkic Sakha (Yakut)

## observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	is̥kaap	is̥kaaptar

138 total examples

# Turkic Sakha (Yakut)

grammar (unobserved)

SINGULAR→stem  
PLURAL→stem+lar

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	is̥kaap	is̥kaaptar

138 total examples

# Turkic Sakha (Yakut)

constraint-based  
synthesis  
[Solar-Lezama 2008]  
test-driven synthesis  
[Perelman et al. 2016]

## grammar (unobserved)

SINGULAR → stem  
PLURAL → stem + lar

$r_1: l \rightarrow d / [ -\text{lateral} \ -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
do not voice next to voiceless

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant} \ -\text{high}] \rightarrow [ -\text{rounded}] / u \ C_0$   
“harmonize” round vowels like “u”, “o”

$r_5: V \rightarrow [ -\text{back} \ -\text{low}] / [ -\text{back} \ +\text{vowel}] [ ]_0$   
“harmonize” vowels to be not at back of mouth

$r_6: [ -\text{sonorant} \ +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
“nasalize” consonant next to a nasal, like “m”

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	iskaap	iskaaptar

138 total examples

# Turkic Sakha (Yakut)

constraint-based  
synthesis  
[Solar-Lezama 2008]  
test-driven synthesis  
[Perelman et al. 2016]

## grammar (unobserved)

SINGULAR → stem  
PLURAL → stem + lar

$r_1: l \rightarrow d / [ -\text{lateral} \ -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
do not voice next to voiceless

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant} \ -\text{high}] \rightarrow [ -\text{rounded}] / u \ C_0$   
“harmonize” round vowels like “u”, “o”

$r_5: V \rightarrow [ -\text{back} \ -\text{low}] / [ -\text{back} \ +\text{vowel}] [ ]_0$   
“harmonize” vowels to be not at back of mouth

$r_6: [ -\text{sonorant} \ +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
“nasalize” consonant next to a nasal, like “m”

## stems (unobserved)

BED : oron  
MARE : bie  
CABINET : ̄skaap

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	̄skaap	̄skaaptar

138 total examples

# Turkic Sakha (Yakut)

## grammar (unobserved)

SINGULAR→stem

PLURAL→stem+lar

$r_1: l \rightarrow d / [-\text{lateral } -\text{tense}]$   
"l" becomes "d" next to "r", "t", but not "l"

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
**do not voice next to voiceless**

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant } -\text{high}] \rightarrow [ -\text{rounded}] / u C_0$   
**"harmonize" round vowels like "u", "o"**

$r_5: V \rightarrow [ -\text{back } -\text{low}] / [ -\text{back } +\text{vowel}] [ ]_0$   
**"harmonize" vowels to be not at back of mouth**

$r_6: [ -\text{sonorant } +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
**"nasalize" consonant next to a nasal, like "m"**

stems  
(unobserved)

observed data

BED : oron

BEDS

# Turkic Sakha (Yakut)

## grammar (unobserved)

SINGULAR→stem

PLURAL→stem+lar

$r_1: l \rightarrow d / [-\text{lateral } -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
**do not voice next to voiceless**

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant } -\text{high}] \rightarrow [ -\text{rounded}] / u C_0$   
“harmonize” round vowels like “u”, “o”

$r_5: V \rightarrow [ -\text{back } -\text{low}] / [ -\text{back } +\text{vowel}] [ ]_0$   
“harmonize” vowels to be not at back of mouth

$r_6: [ -\text{sonorant } +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
“nasalize” consonant next to a nasal, like “m”

stems  
(unobserved)

observed data

BED : oron

BEDS → oron+lar → oronlar

# Turkic Sakha (Yakut)

## grammar (unobserved)

SINGULAR→stem

PLURAL→stem+lar

$r_1: l \rightarrow d / [-\text{lateral } -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [-\text{voice}] / [-\text{voice}]$   
**do not voice next to voiceless**

$r_3: V \rightarrow [+ \text{rounded}] / [+ \text{rounded}] [- \text{low}]_0$

$r_4: [+ \text{continuant } -\text{high}] \rightarrow [- \text{rounded}] / u C_0$   
“harmonize” round vowels like “u”, “o”

$r_5: V \rightarrow [-\text{back } -\text{low}] / [-\text{back } +\text{vowel}] [ ]_0$   
“harmonize” vowels to be not at back of mouth

$r_6: [-\text{sonorant } +\text{voice}] \rightarrow [+ \text{nasal}] / [+ \text{nasal}]$   
“nasalize” consonant next to a nasal, like “m”

stems  
(unobserved)

observed data

BED : oron

BEDS → oron+lar → oronlar  $\xrightarrow{r_1}$  orondar

# Turkic Sakha (Yakut)

## grammar (unobserved)

SINGULAR→stem

PLURAL→stem+lar

$r_1: l \rightarrow d / [-\text{lateral } -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
**do not voice next to voiceless**

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant } -\text{high}] \rightarrow [ -\text{rounded}] / u C_0$   
“harmonize” round vowels like “u”, “o”

$r_5: V \rightarrow [ -\text{back } -\text{low}] / [ -\text{back } +\text{vowel}] [ ]_0$   
“harmonize” vowels to be not at back of mouth

$r_6: [ -\text{sonorant } +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
“nasalize” consonant next to a nasal, like “m”

stems  
(unobserved)

observed data

BED : oron

BEDS → oron+lar → oronlar  $\xrightarrow{r_1}$  orondar  $\xrightarrow{r_3}$  orondor

# Turkic Sakha (Yakut)

## grammar (unobserved)

SINGULAR→stem

PLURAL→stem+lar

$r_1: l \rightarrow d / [-\text{lateral } -\text{tense}]$   
“l” becomes “d” next to “r”, “t”, but not “l”

$r_2: C \rightarrow [ -\text{voice}] / [ -\text{voice}]$   
**do not voice next to voiceless**

$r_3: V \rightarrow [ +\text{rounded}] / [ +\text{rounded}] [ -\text{low}]_0$

$r_4: [ +\text{continuant } -\text{high}] \rightarrow [ -\text{rounded}] / u C_0$   
**“harmonize” round vowels like “u”, “o”**

$r_5: V \rightarrow [ -\text{back } -\text{low}] / [ -\text{back } +\text{vowel}] [ ]_0$   
**“harmonize” vowels to be not at back of mouth**

$r_6: [ -\text{sonorant } +\text{voice}] \rightarrow [ +\text{nasal}] / [ +\text{nasal}]$   
**“nasalize” consonant next to a nasal, like “m”**

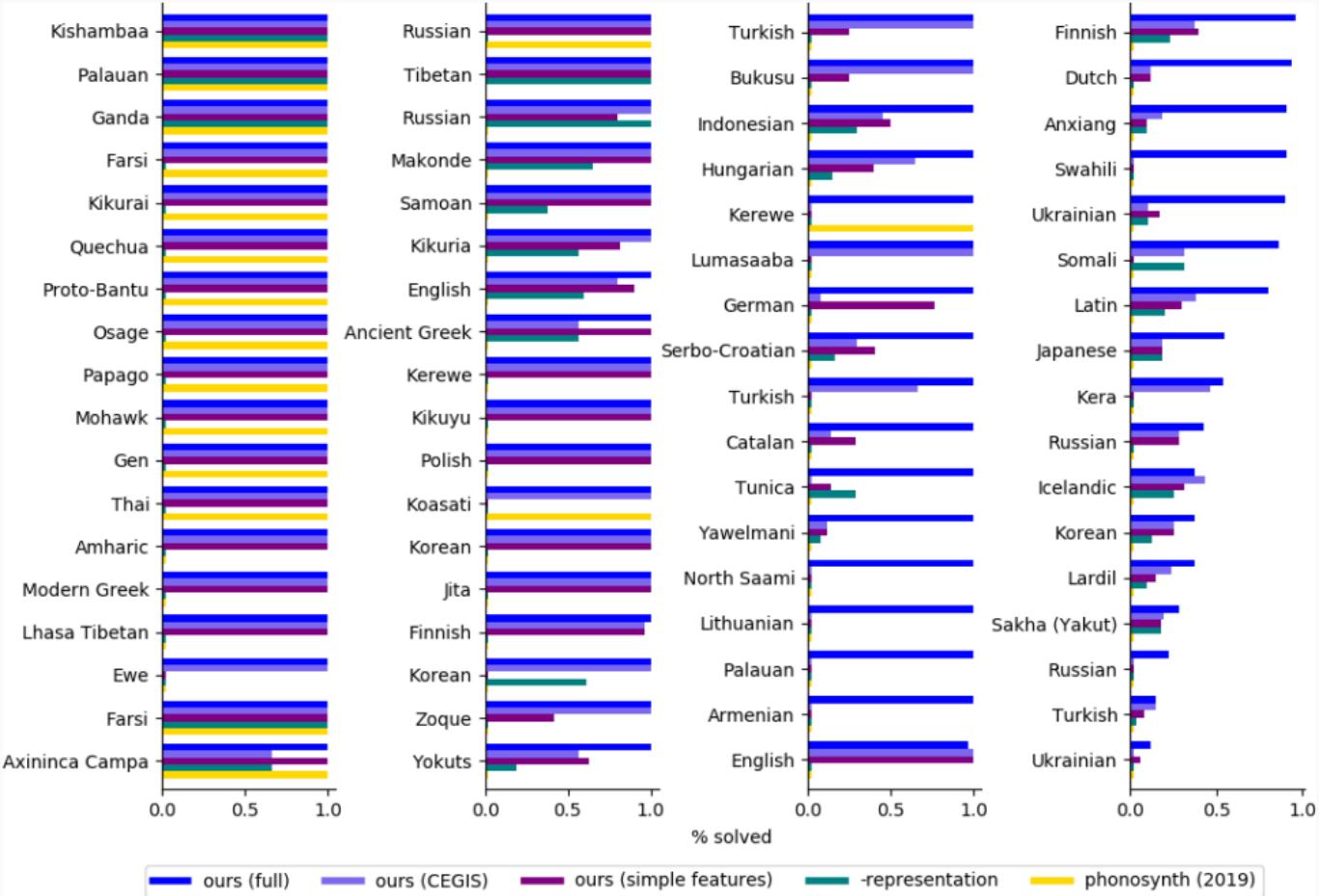
stems  
(unobserved)

observed data

BED : oron

BEDS → oron+lar → oronlar  $\xrightarrow{r_1}$  orondar  $\xrightarrow{r_3}$  orondor  $\xrightarrow{r_6}$  oronnor

Kishambaa	Russian	Turkish	Finnish
Palauan	Tibetan	Bukusu	Dutch
Ganda	Russian	Indonesian	Anxiang
Farsi	Makonde	Hungarian	Swahili
Kikurai	Samoan	Kerewe	Ukrainian
Quechua	Kikuria	Lumasaaba	Somali
Proto-Bantu	English	German	Latin
Osage	Ancient Greek	Serbo-Croatian	Japanese
Papago	Kerewe	Turkish	Kera
Mohawk	Kikuyu	Catalan	Russian
Gen	Polish	Tunica	Icelandic
Thai	Koasati	Yawelmani	Korean
Amharic	Korean	North Saami	Lardil
Modern Greek	Jita	Lithuanian	Sakha (Yakut)
Lhasa Tibetan	Finnish	Palauan	Russian
Ewe	Korean	Armenian	Turkish
Farsi	Zoque	English	Ukrainian
Axininca Campa	Yokuts		



# Distilling higher-level knowledge

Ewe  
data

Gen  
data

Jita  
data

Thai  
data

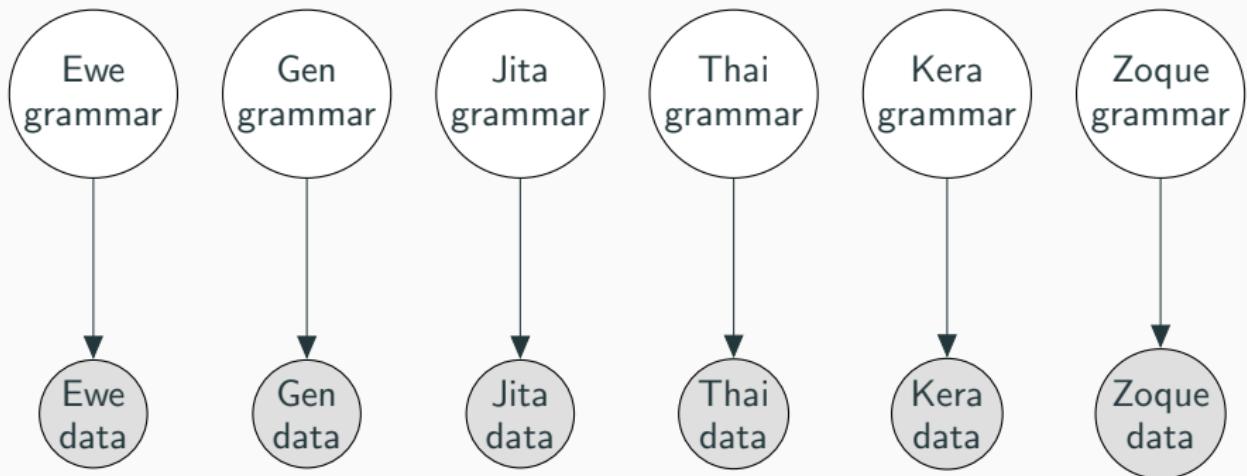
Kera  
data

Zoque  
data

# Distilling higher-level knowledge

**dark:** we know it

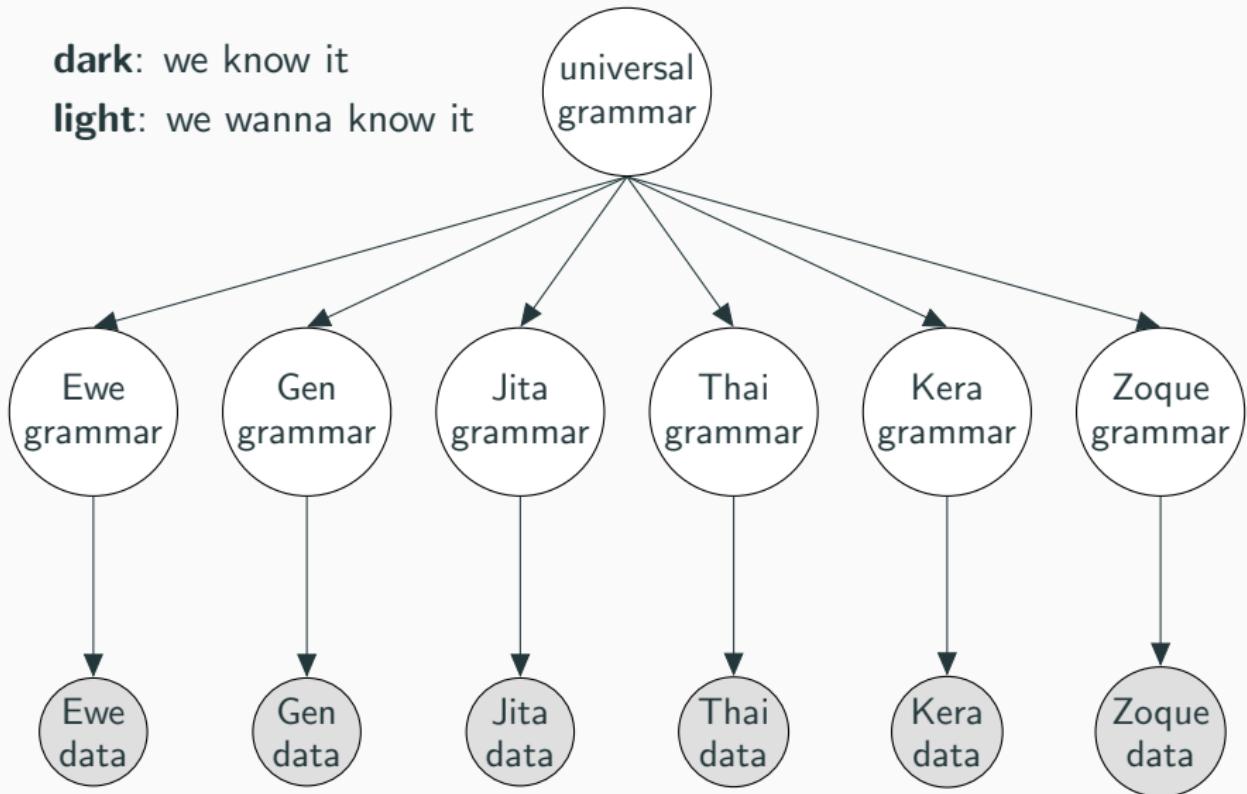
**light:** we wanna know it



# Distilling higher-level knowledge

**dark:** we know it

**light:** we wanna know it

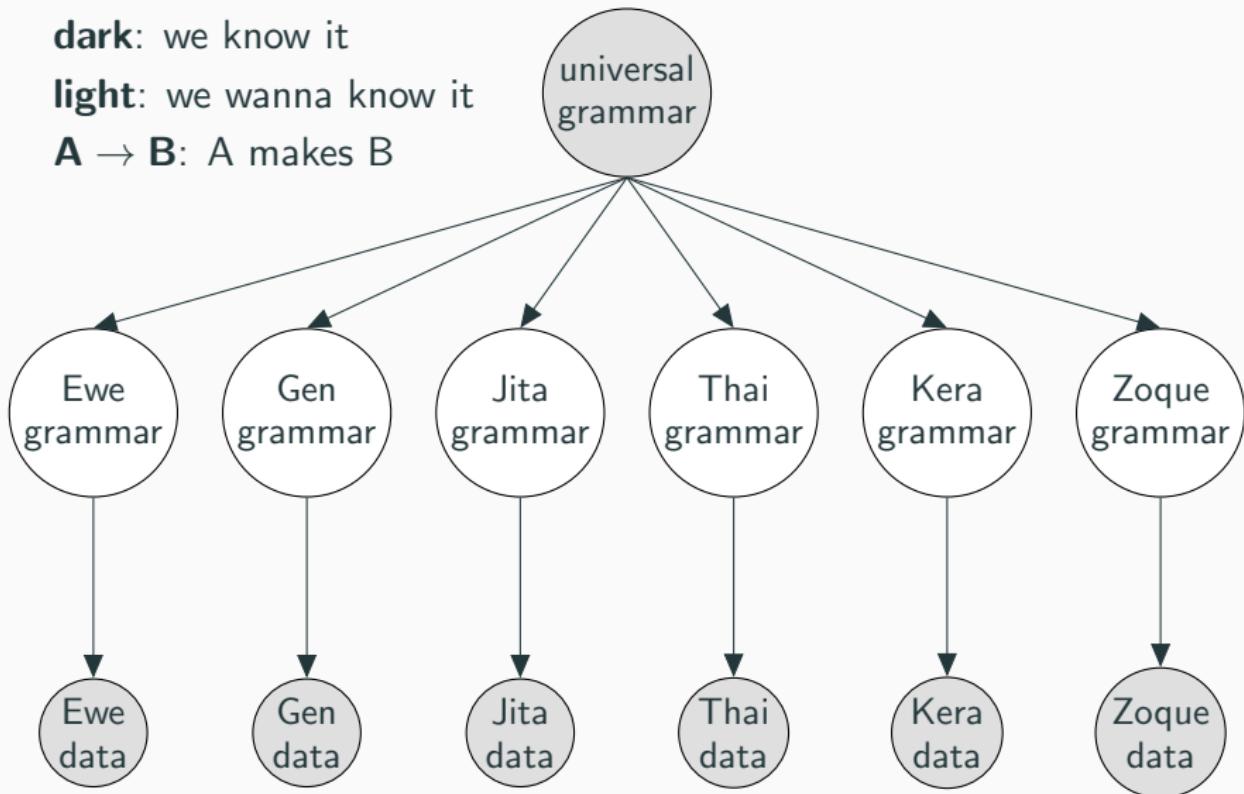


# Distilling higher-level knowledge

**dark:** we know it

**light:** we wanna know it

**A → B:** A makes B

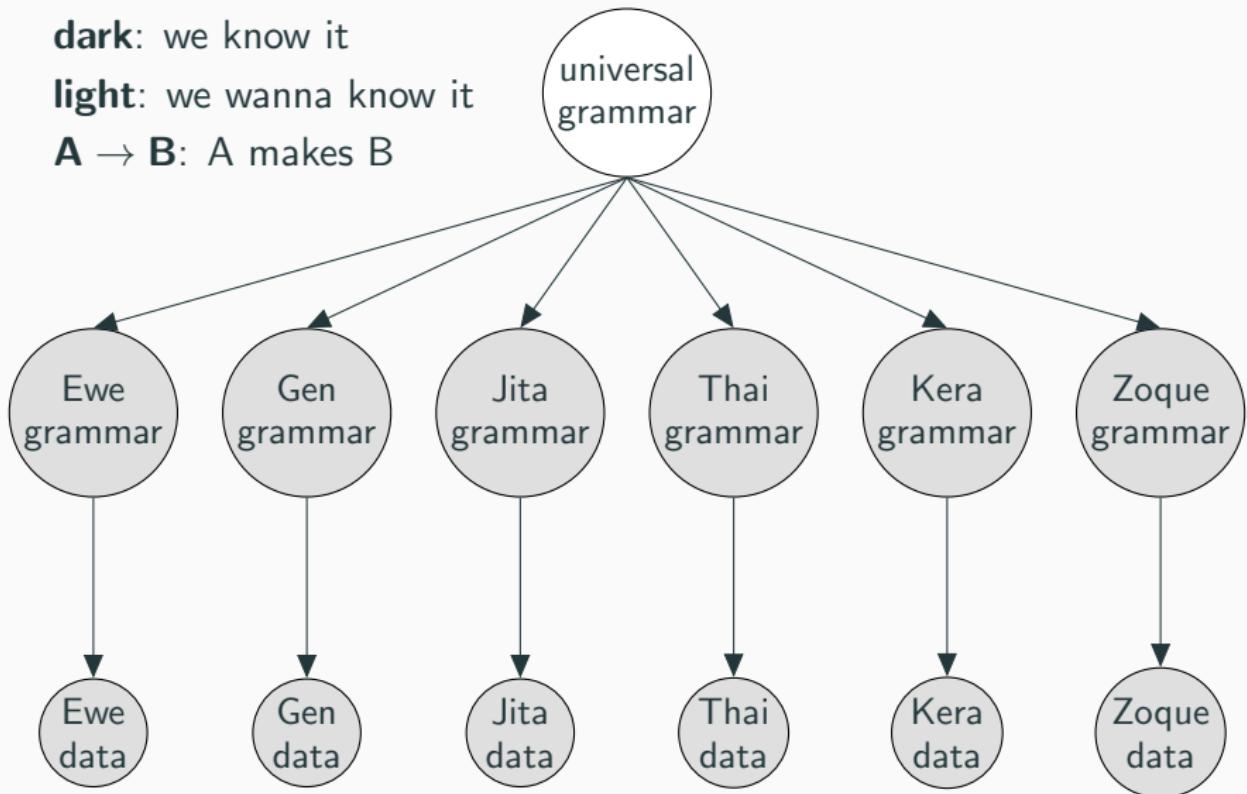


# Distilling higher-level knowledge

**dark:** we know it

**light:** we wanna know it

**A → B:** A makes B

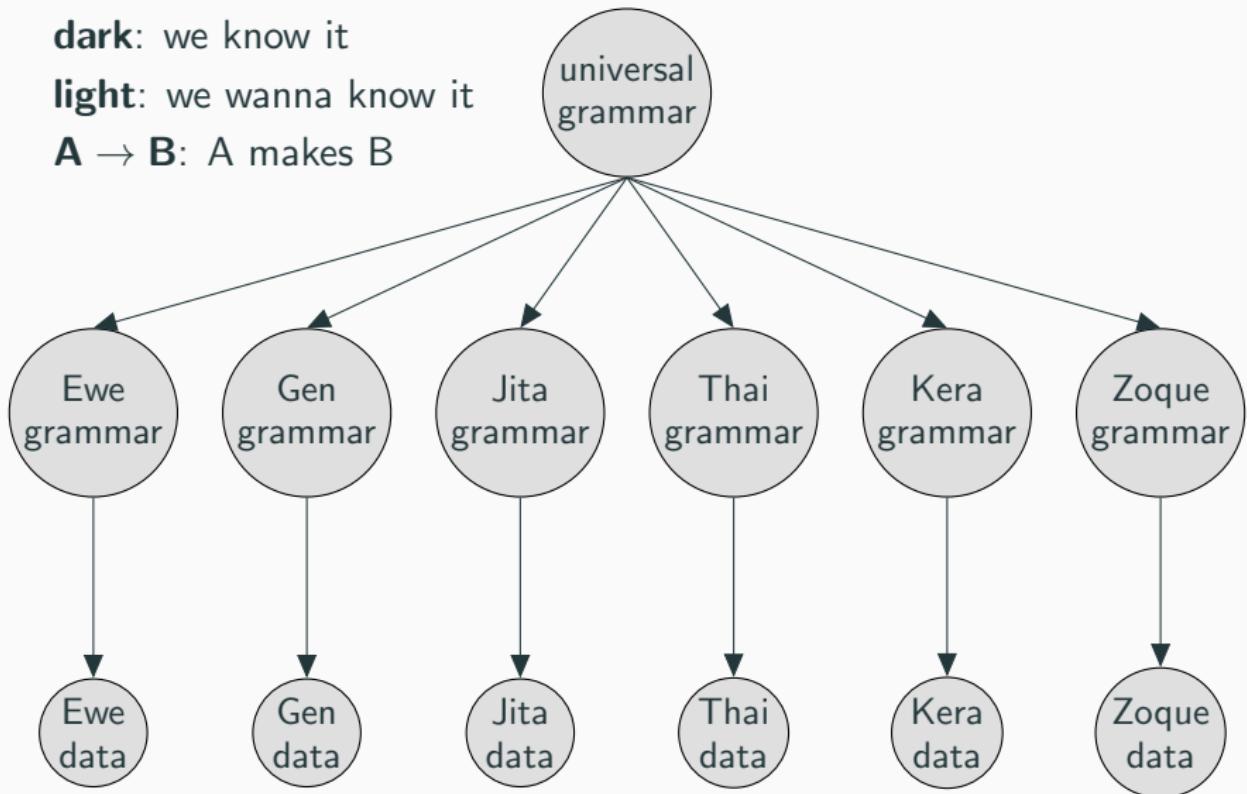


# Distilling higher-level knowledge

**dark:** we know it

**light:** we wanna know it

**A → B:** A makes B



## Lessons

Higher-level knowledge matters (“universal grammar”). Get the basics of the representation correct

But *some* of this higher-level knowledge can be learned. You don’t need millions of examples to learn it. But it’s not a one-shot learning problem either

Program Induction and perception  
model discovery  
learning to learn

# Learning to write code

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

- Library of concepts (declarative knowledge; domain specific language)
- Inference strategy (procedural knowledge; synthesis algorithm)

Cathy Wong



Max Nye



Mathias  
Sable-Meyer



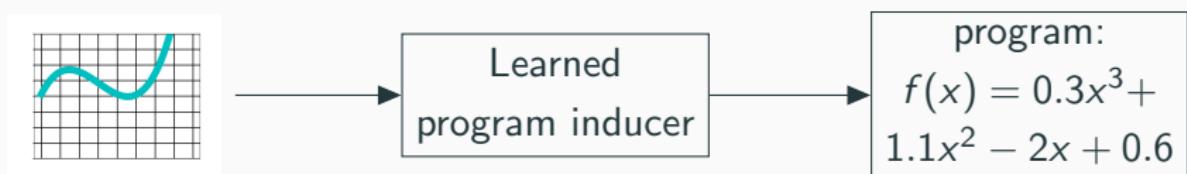
Lucas Morales



# Learning to write code

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

- Library of concepts (declarative knowledge; domain specific language)
- Inference strategy (procedural knowledge; synthesis algorithm)



Concepts:  $x^3$ ,  $\alpha x + \beta$ , etc

Inference strategy: neurosymbolic search for programs

# 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

## 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]  
...
```

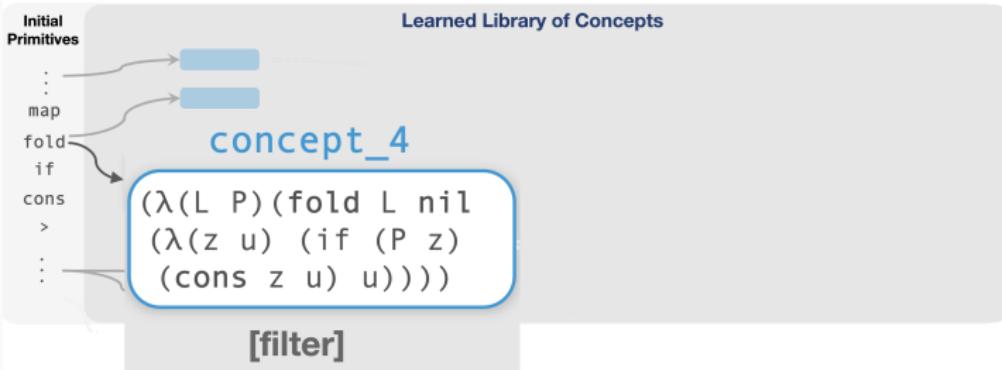
# Library learning



## Sample Problem: sort list

$[9\ 2\ 7\ 1] \rightarrow [1\ 2\ 7\ 9]$   
 $[3\ 8\ 9\ 4\ 2] \rightarrow [2\ 3\ 4\ 8\ 9]$   
 $[6\ 2\ 2\ 3\ 8\ 5] \rightarrow [2\ 2\ 3\ 5\ 6\ 8]$   
...

# 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]  
...

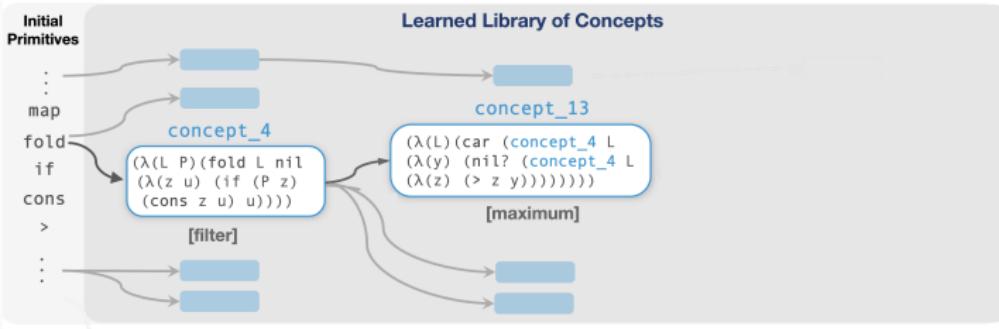
# Library learning



## Sample Problem: sort list

$[9\ 2\ 7\ 1] \rightarrow [1\ 2\ 7\ 9]$   
 $[3\ 8\ 9\ 4\ 2] \rightarrow [2\ 3\ 4\ 8\ 9]$   
 $[6\ 2\ 2\ 3\ 8\ 5] \rightarrow [2\ 2\ 3\ 5\ 6\ 8]$   
...

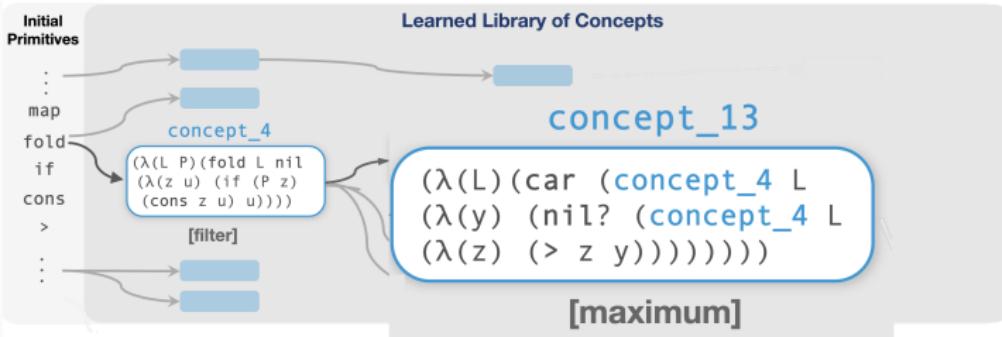
# Library learning



## 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]  
...

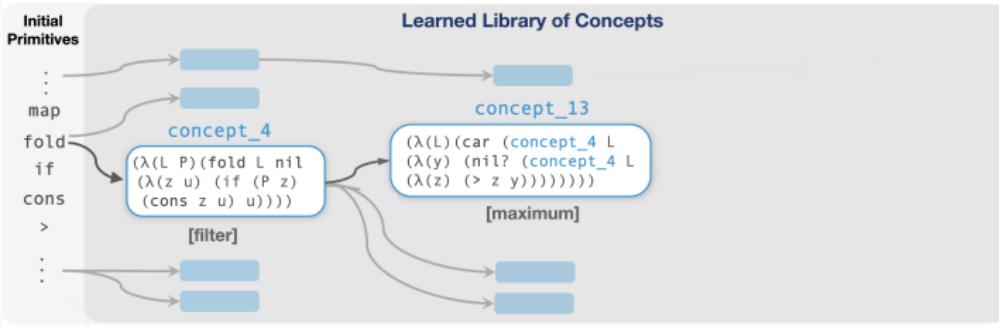
# Library learning



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]  
...

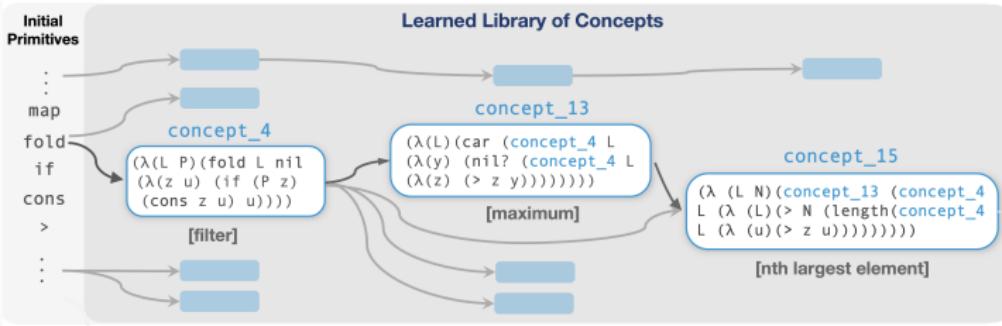
# Library learning



## 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]  
...

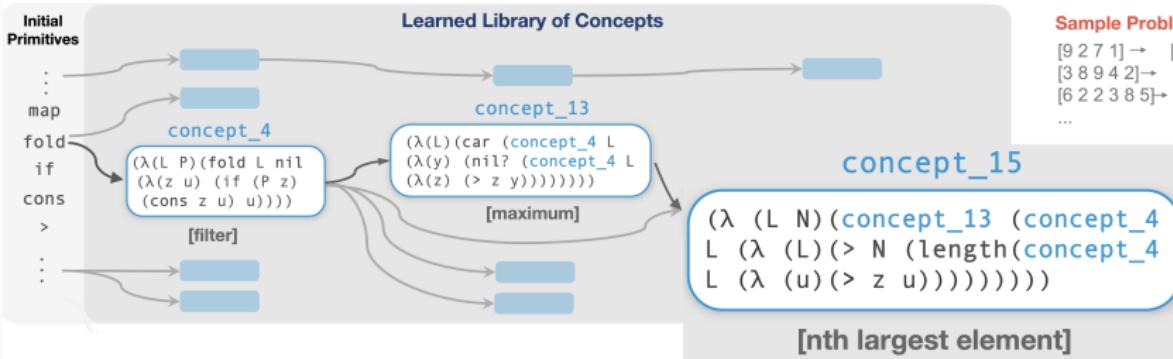
# Library learning



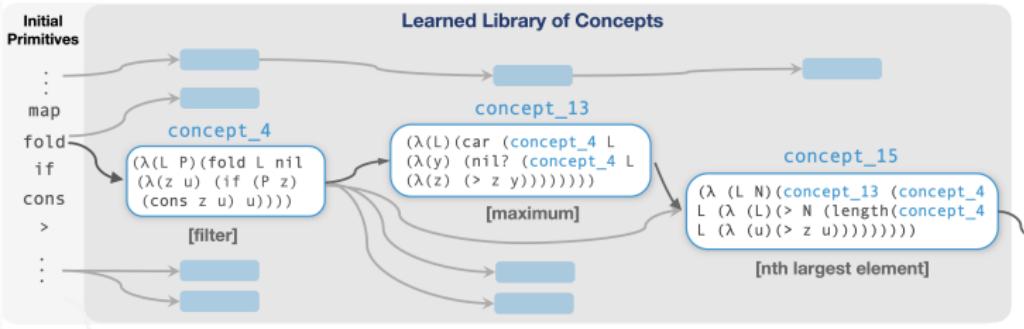
Sample Problem: sort list

[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



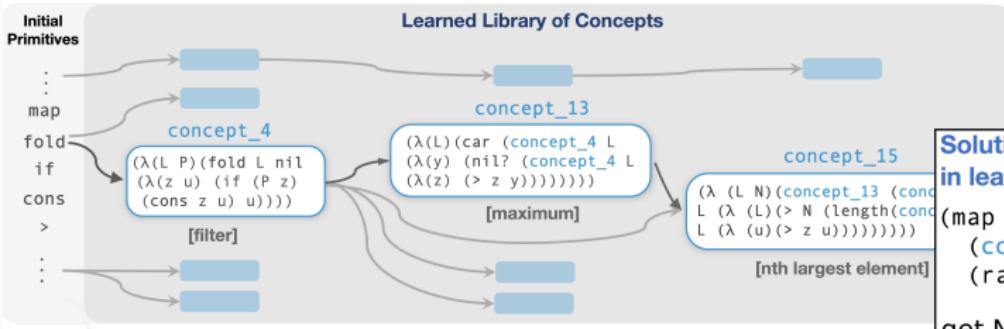
Sample Problem: sort list

[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

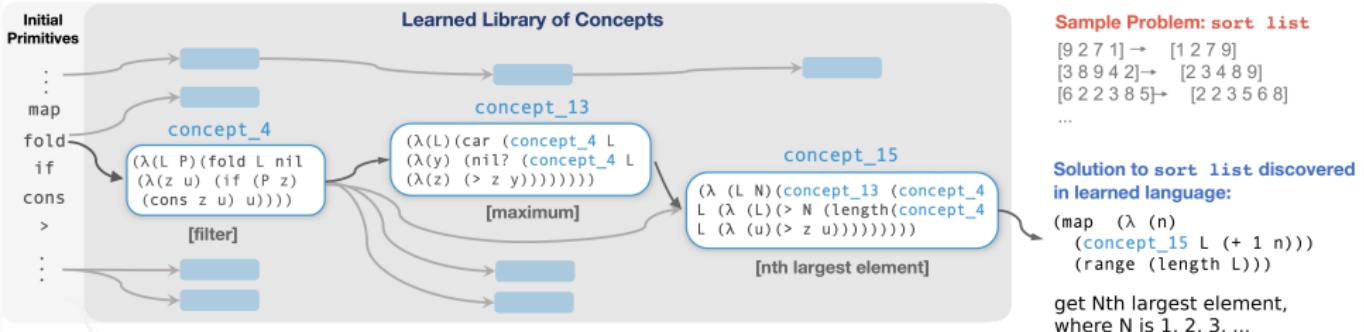
[9 2 7 1] → [1 2 7 9]  
[3 8 9 4 2] → [2 3 4 8 9]  
[6 2 2 3 8 5] → [2 2 3 5 6 8]  
...

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, ...

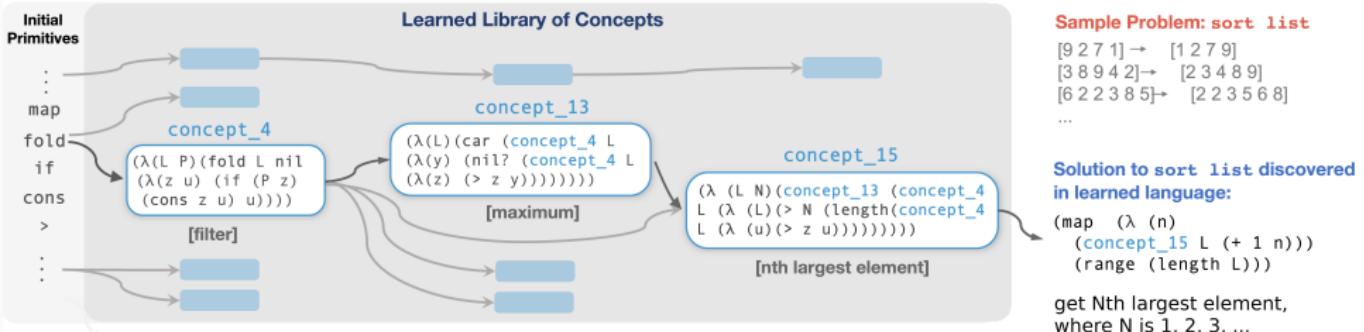
# Library learning



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



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  $\leq 10\text{min}$ . Brute-force search  
without learned library would take  $\approx 10^{73}$  years

# 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

# DreamCoder

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



## List Processing

### Sum List

$[1\ 2\ 3] \rightarrow 6$

$[4\ 6\ 8\ 1] \rightarrow 17$

### Double

$[1\ 2\ 3] \rightarrow [2\ 4\ 6]$

$[4\ 5\ 1] \rightarrow [8\ 10\ 2]$

## Text Editing

### Abbreviate

Allen Newell → A.N.

Herb Simon → H.S.

### Drop Last Three

shrdlu → shr

shakey → sha

## Regexes

### Phone numbers

(555) 867-5309

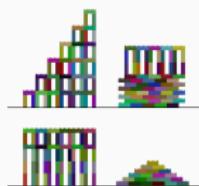
(650) 555-2368

### Currency

\$100.25

\$4.50

## Block Towers



## Symbolic Regression



$$y = f(x)$$

## Recursive Programming

### Filter Red

[■■■■■■] → [■■■■■]

[■■■■■■■■] → [■■■■■■■]

[■■■■■■■■■] → [■■■■■■■■]

## Physical Laws

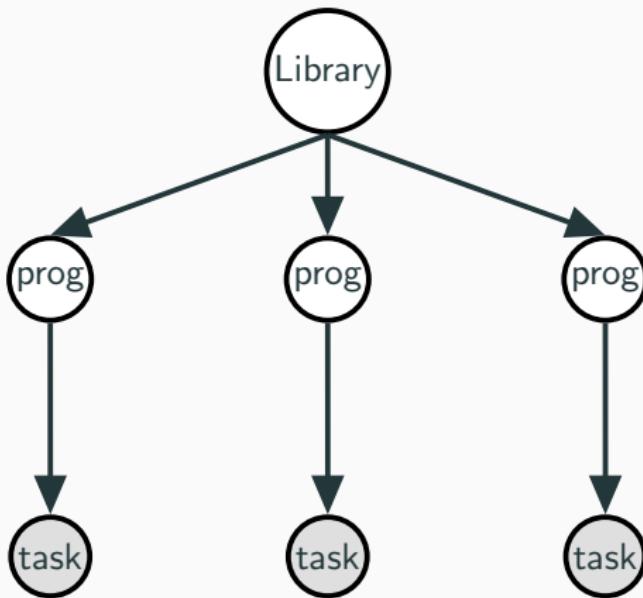
$$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$$



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

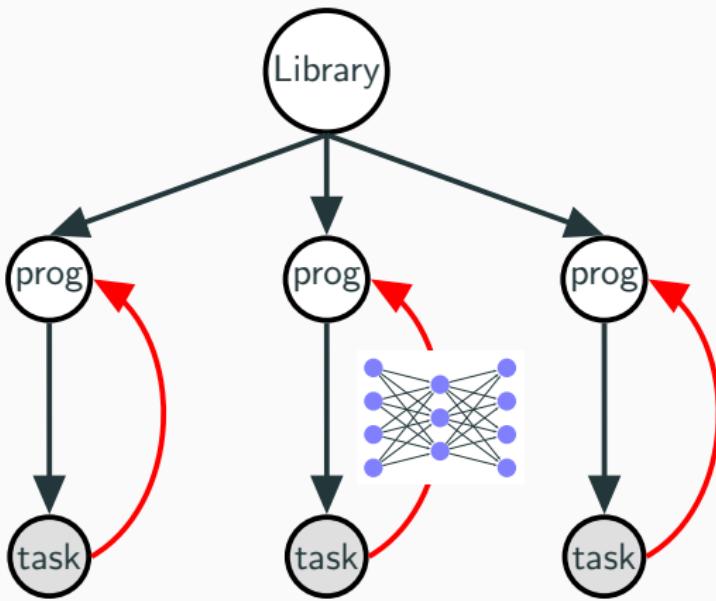
cf. Helmholtz machine, wake/sleep neural network training algorithms

## Library learning as Bayesian inference

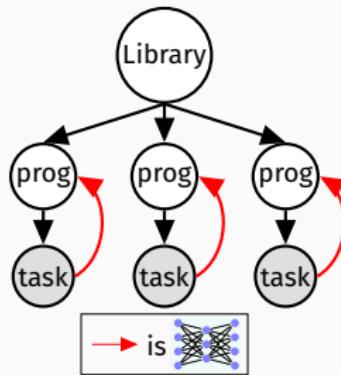


[Dechter et al, 2013] [Liang et al, 2010] [Lake et al, 2015]

# Library learning as neurally-guided Bayesian inference



library learning via program analysis +  
new neural inference network for program synthesis +  
better program representation (Lisp+polymorphic types [Milner 1978])



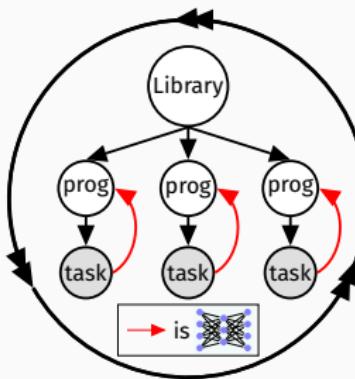
**WAKE**

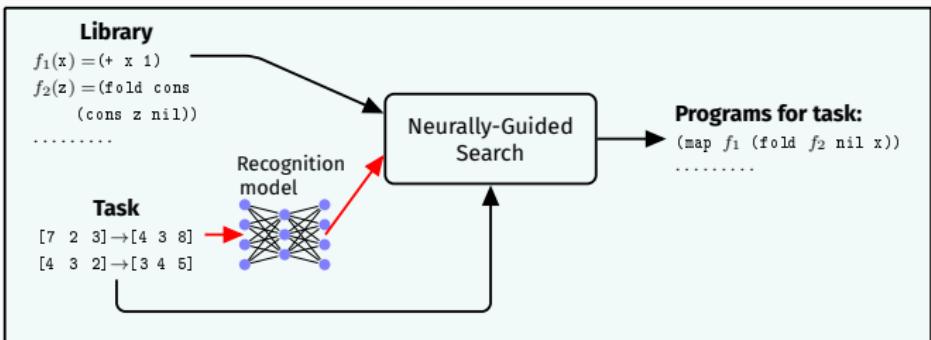
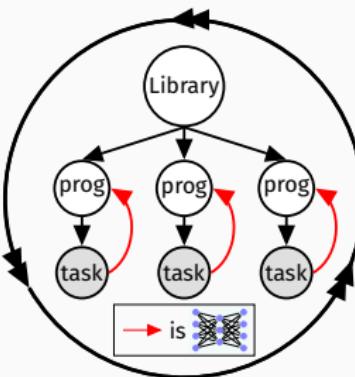


**SLEEP: ABSTRACTION**

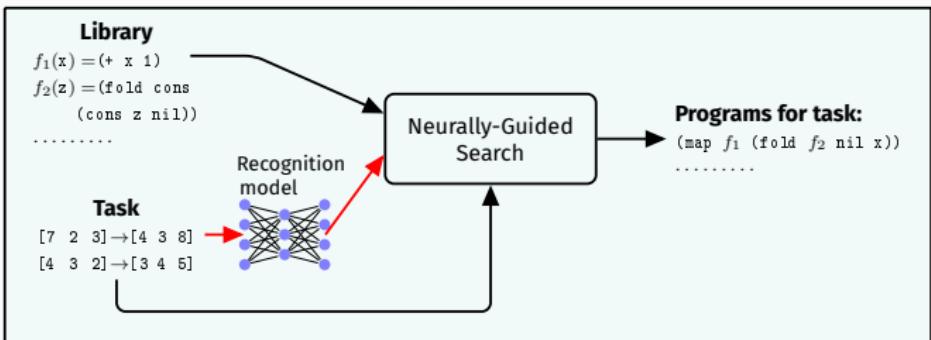


**SLEEP: DREAMING**

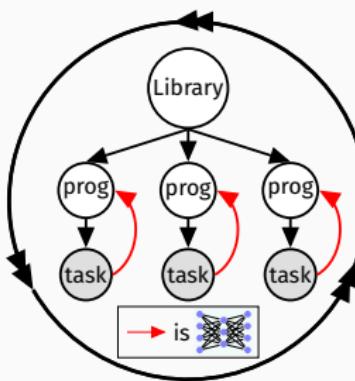


**WAKE****SLEEP: ABSTRACTION****SLEEP: DREAMING**

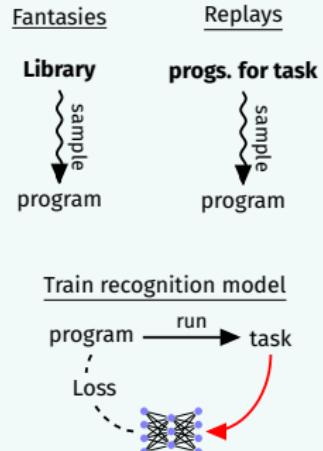
### WAKE

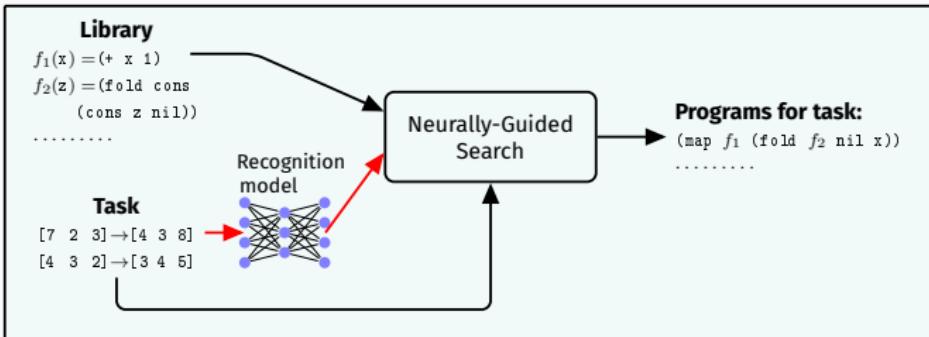
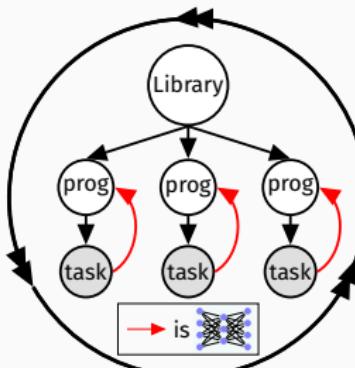
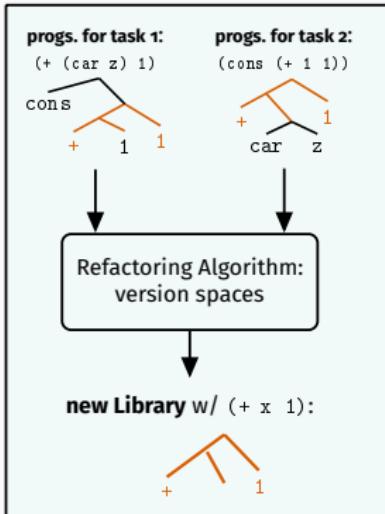
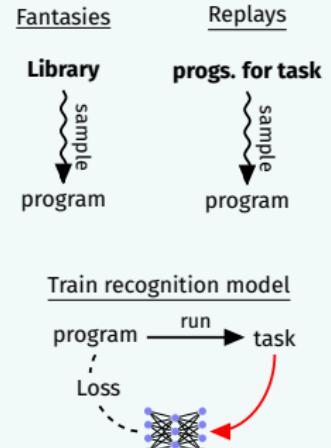


### SLEEP: ABSTRACTION

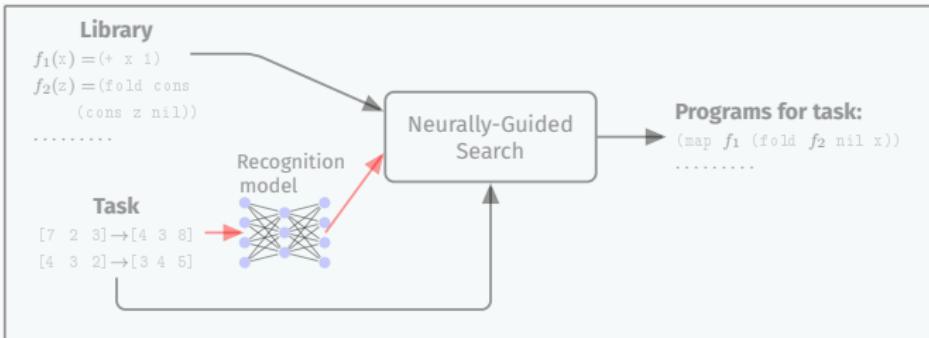


### SLEEP: DREAMING

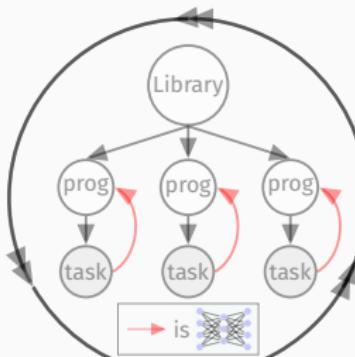
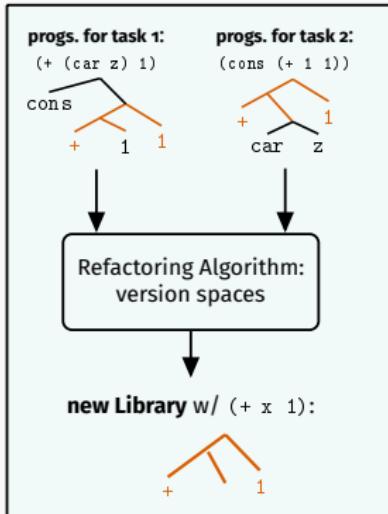


**WAKE****SLEEP: ABSTRACTION****SLEEP: DREAMING**

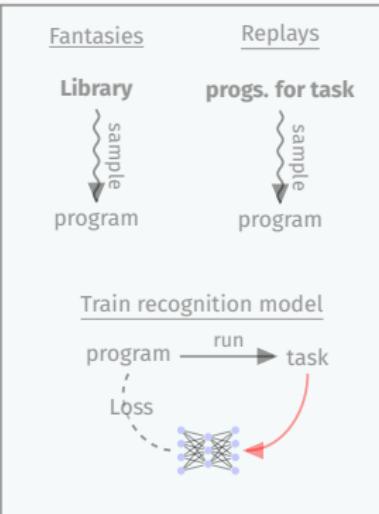
## WAKE



## SLEEP: ABSTRACTION



## SLEEP: DREAMING



# DreamCoder Domains

## List Processing

**Sum List**

$[1 \ 2 \ 3] \rightarrow 6$   
 $[4 \ 6 \ 8 \ 1] \rightarrow 17$

**Double**

$[1 \ 2 \ 3] \rightarrow [2 \ 4 \ 6]$   
 $[4 \ 5 \ 1] \rightarrow [8 \ 10 \ 2]$

## Block Towers



## Text Editing

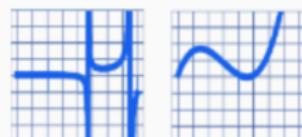
**Abbreviate**

$\text{Allen Newell} \rightarrow \text{A.N.}$   
 $\text{Herb Simon} \rightarrow \text{H.S.}$

**Drop Last Three**

$\text{shrdlu} \rightarrow \text{shr}$   
 $\text{shakey} \rightarrow \text{sha}$

## Symbolic Regression



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.  
 $y = f(x)$

## Regexes

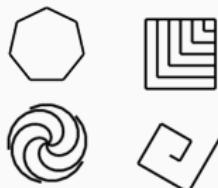
**Phone numbers**

$(555) \ 867-5309$   
 $(650) \ 555-2368$

**Currency**

\$100.25  
\$4.50

## LOGO Graphics



## Recursive Programming

**Filter Red**

$[ \text{red} \ \text{black} \ \text{red} \ \text{black} ] \rightarrow [ \text{black} \ \text{black} ]$   
 $[ \text{red} \ \text{black} \ \text{red} \ \text{black} \ \text{red} \ \text{black} ] \rightarrow [ \text{black} \ \text{black} \ \text{red} \ \text{black} ]$   
 $[ \text{red} \ \text{black} \ \text{red} \ \text{black} \ \text{red} \ \text{black} \ \text{red} ] \rightarrow [ \text{black} \ \text{black} \ \text{black} \ \text{black} ]$

## Physical Laws

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

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

# DreamCoder Domains

## List Processing

**Sum List**

$[1 \ 2 \ 3] \rightarrow 6$   
 $[4 \ 6 \ 8 \ 1] \rightarrow 17$

**Double**

$[1 \ 2 \ 3] \rightarrow [2 \ 4 \ 6]$   
 $[4 \ 5 \ 1] \rightarrow [8 \ 10 \ 2]$

## Text Editing

**Abbreviate**

$\text{Allen Newell} \rightarrow \text{A.N.}$   
 $\text{Herb Simon} \rightarrow \text{H.S.}$

**Drop Last Three**

$\text{shrdlu} \rightarrow \text{shr}$   
 $\text{shakey} \rightarrow \text{sha}$

## Regexes

**Phone numbers**

$(555) \ 867-5309$   
 $(650) \ 555-2368$

**Currency**

\$100.25  
\$4.50

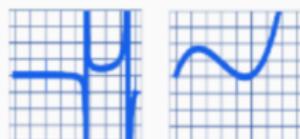
## LOGO Graphics



## Block Towers



## Symbolic Regression



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

## Recursive Programming

**Filter Red**

$[ \text{red} \ \text{red} \ \text{blue} ] \rightarrow [ \text{blue} ]$   
 $[ \text{red} \ \text{black} \ \text{red} \ \text{black} ] \rightarrow [ \text{black} \ \text{black} ]$   
 $[ \text{red} \ \text{black} \ \text{red} \ \text{green} ] \rightarrow [ \text{black} \ \text{black} ]$

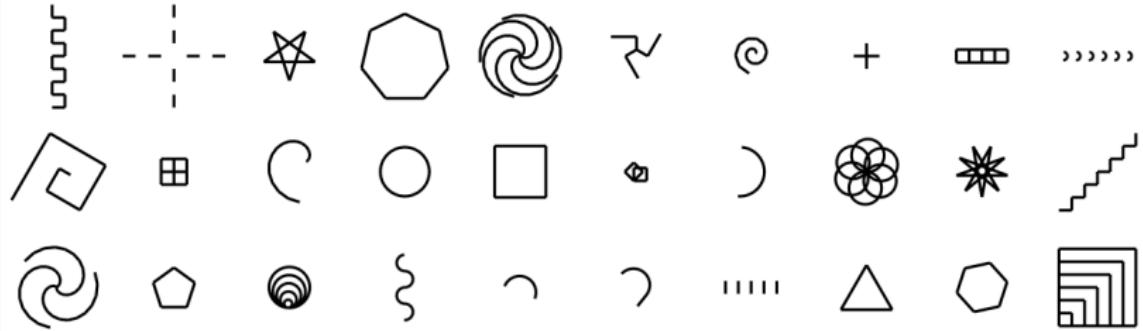
## Physical Laws

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

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

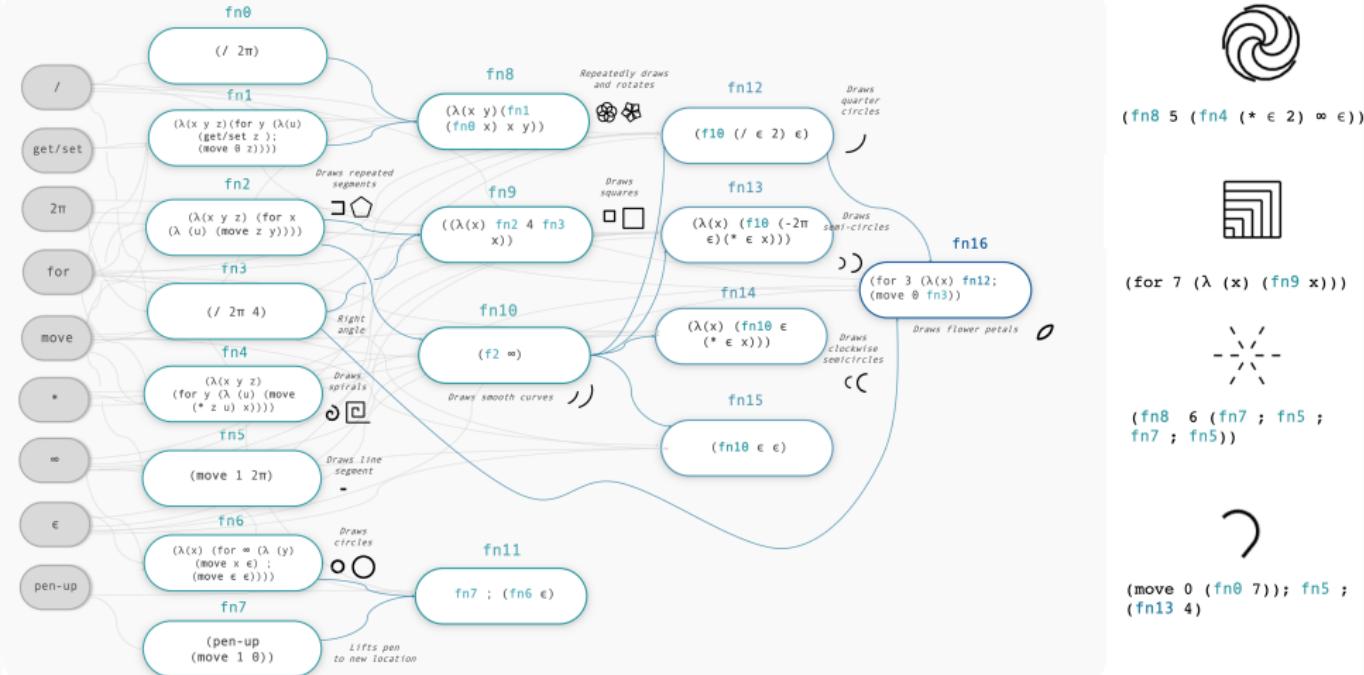
# LOGO Turtle Graphics

30 out of 160 tasks



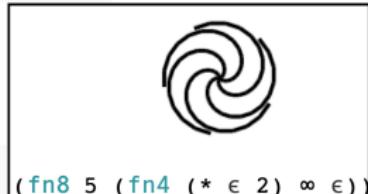
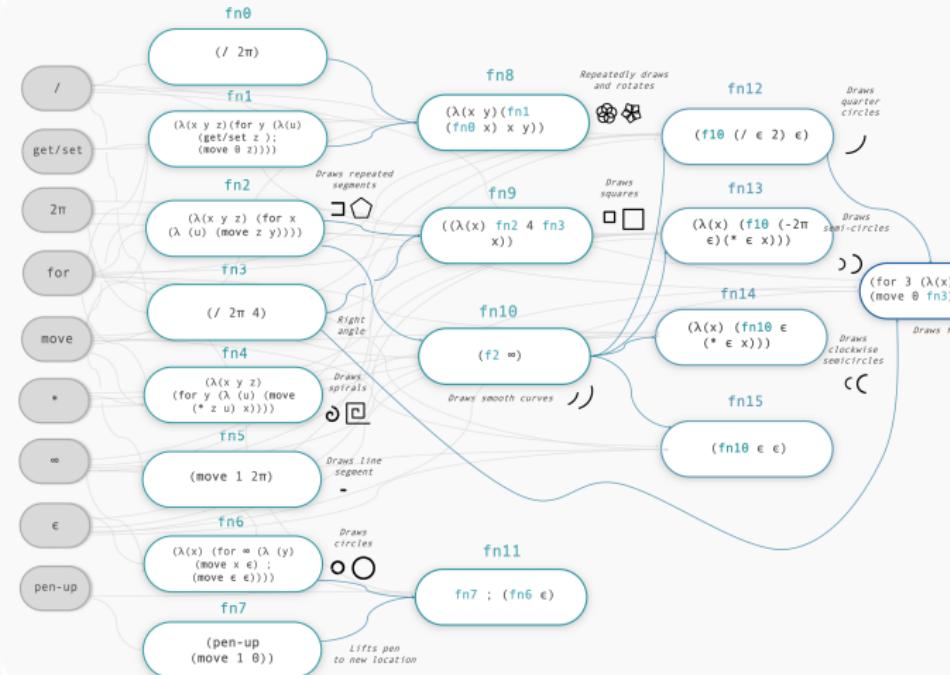
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# LOGO Turtle Graphics – learning an interpretable library



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# LOGO Turtle Graphics – learning an interpretable library



```
(fn8 5 (fn4 (* \epsilon 2) \infty \epsilon))
```

```
(for 7 (\lambda (x) (fn9 x)))
```

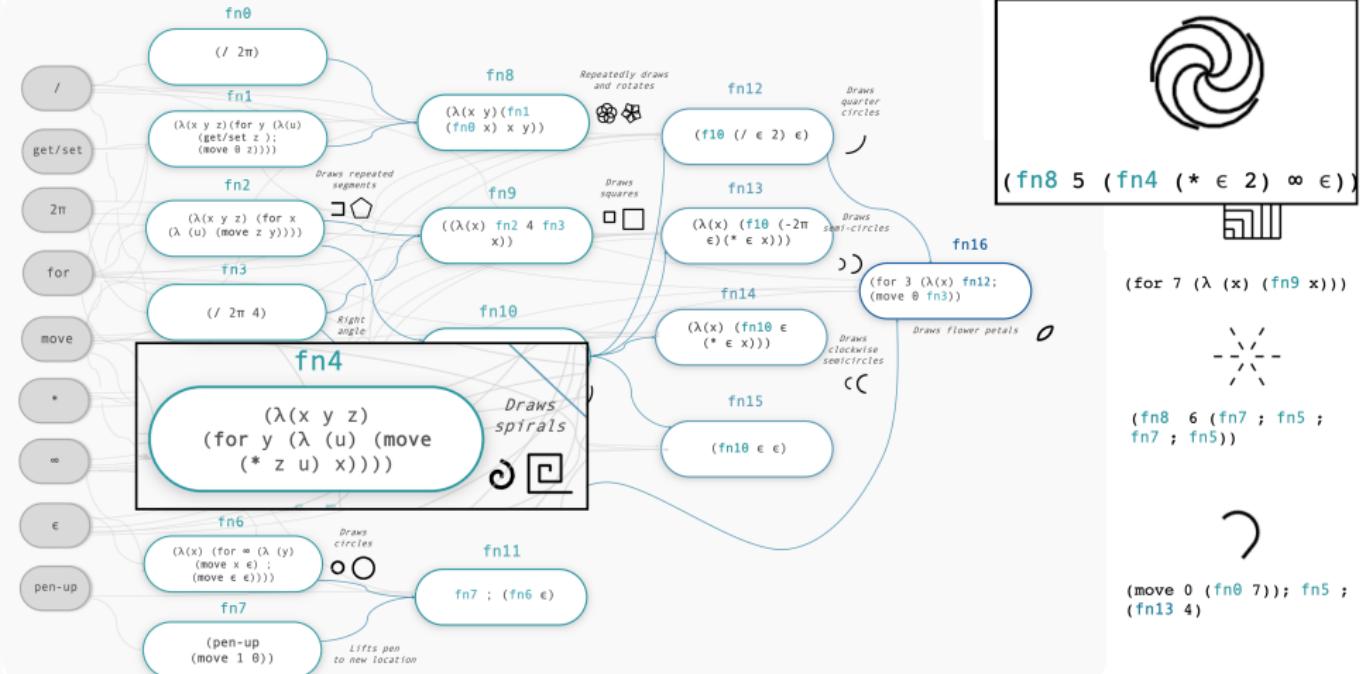


```
(fn8 6 (fn7 ; fn5 ; fn7 ; fn5))
```



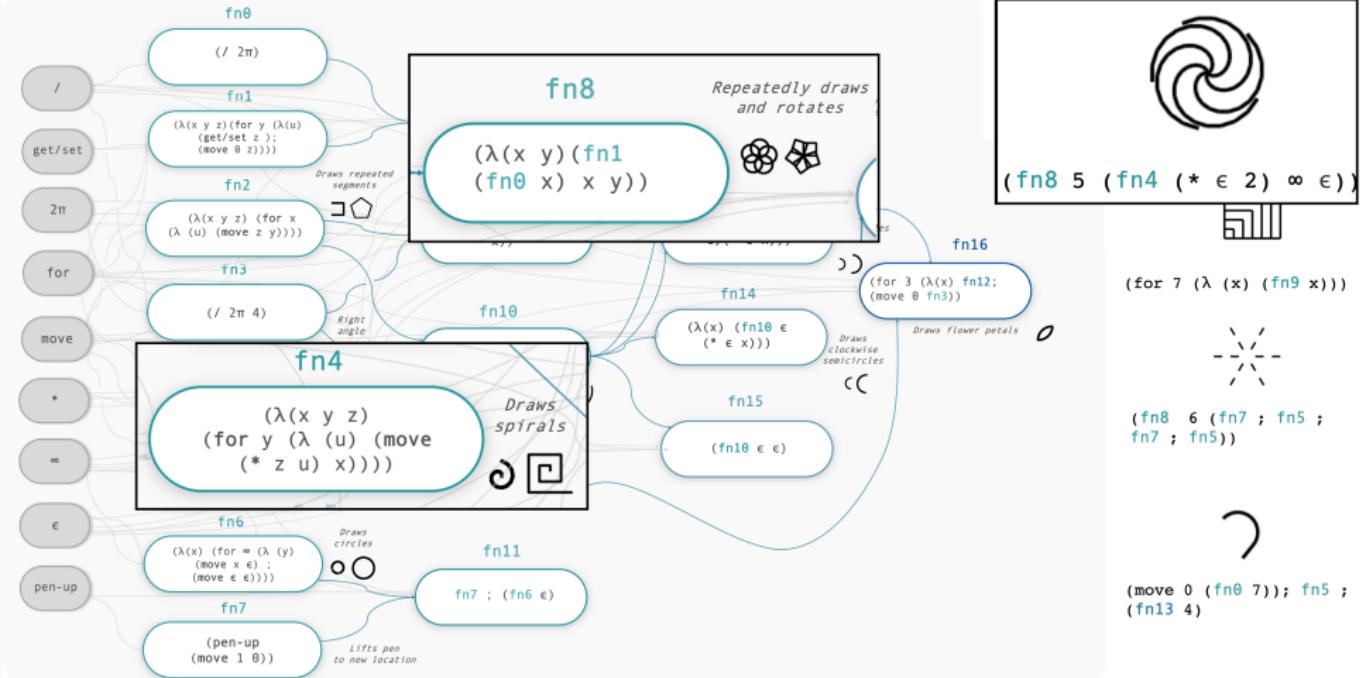
```
(move 0 (fn0 7)); fn5 ; (fn13 4)
```

# LOGO Turtle Graphics – learning an interpretable library

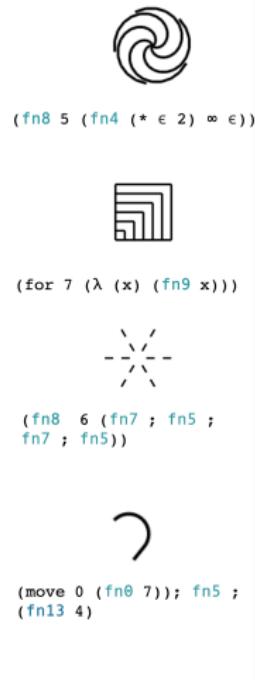
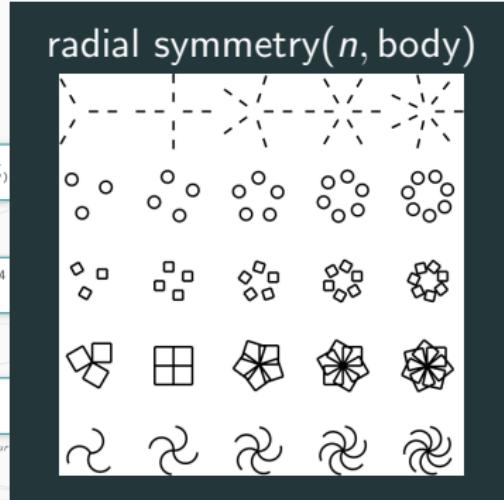
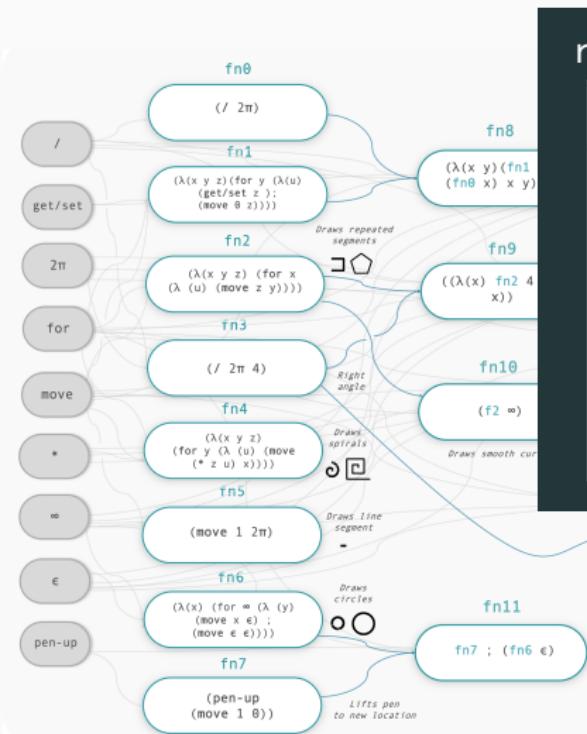


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# LOGO Turtle Graphics – learning an interpretable library

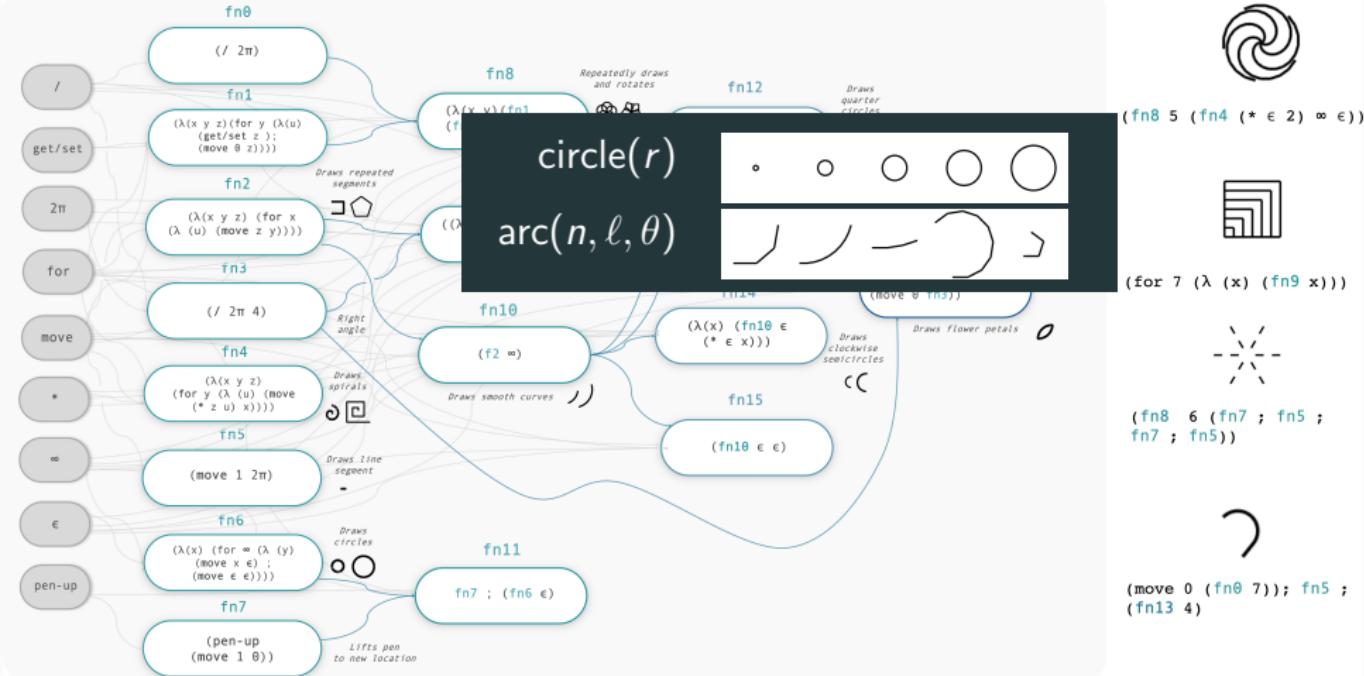


# LOGO Turtle Graphics – learning an interpretable library

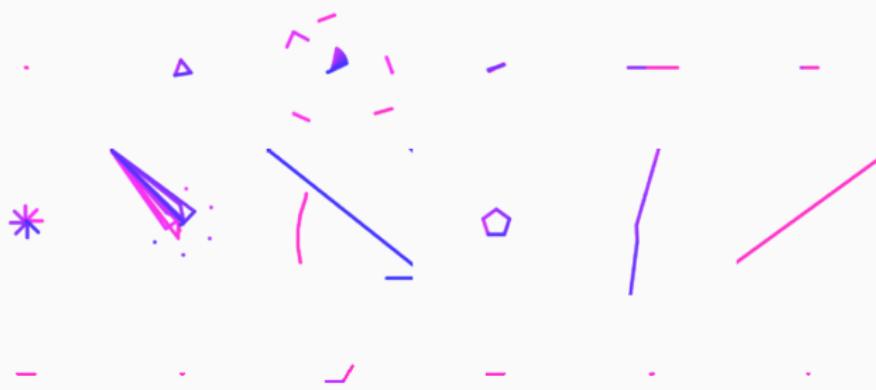
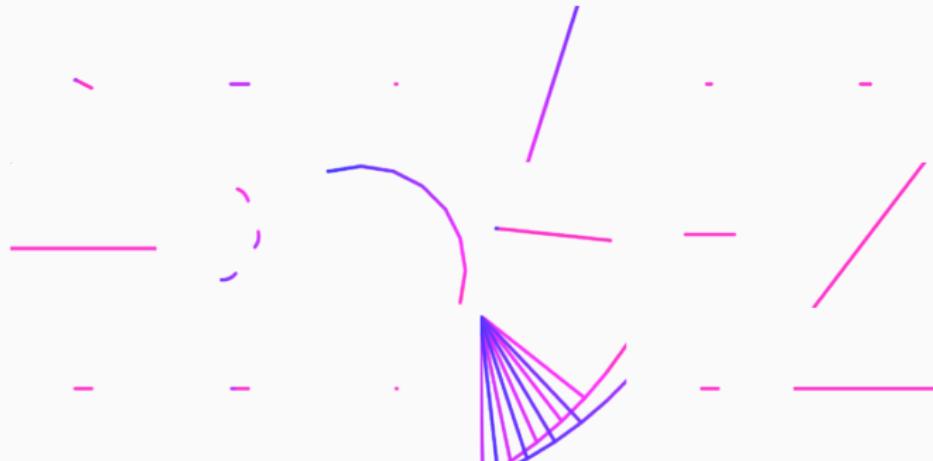


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

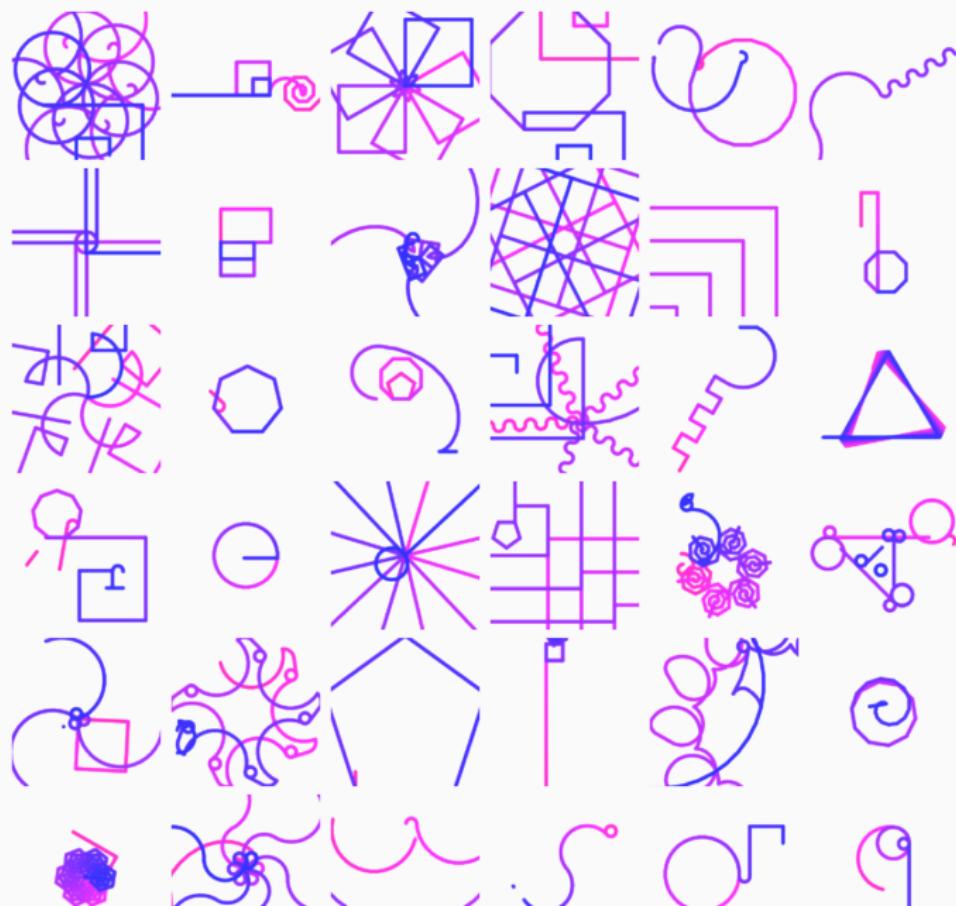
# LOGO Turtle Graphics – learning an interpretable library



# What does DreamCoder dream of? (before learning)



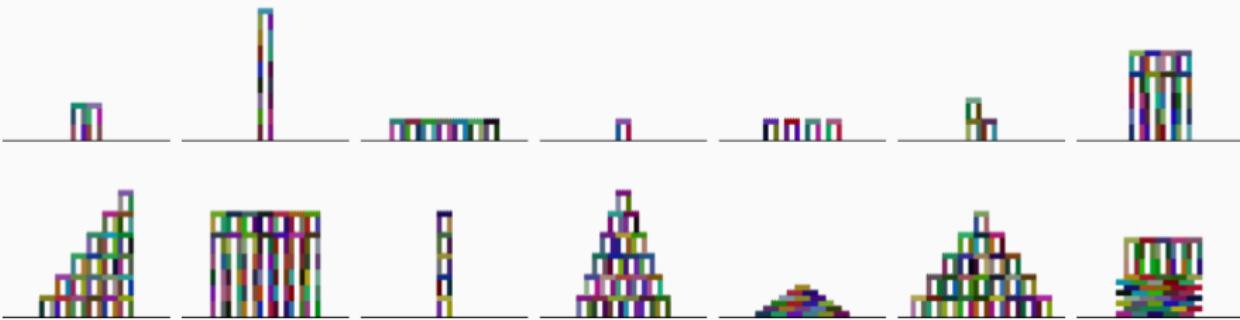
## What does DreamCoder dream of? (after learning)



# Planning to build towers

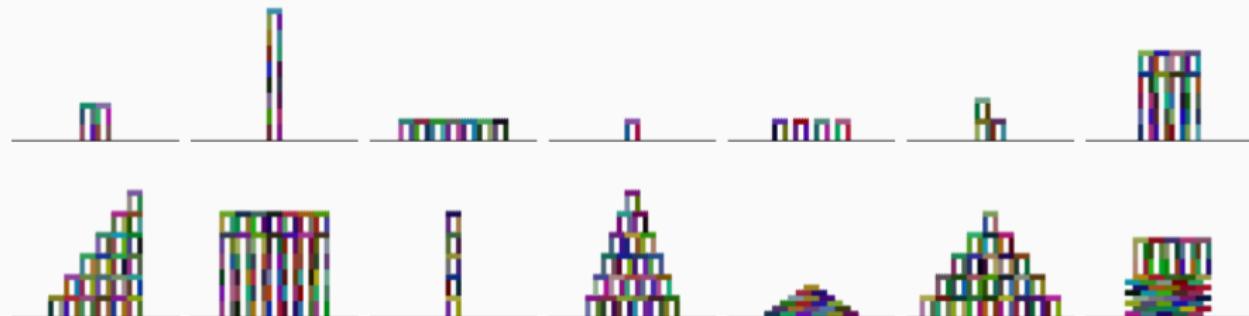
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

example tasks (112 total)

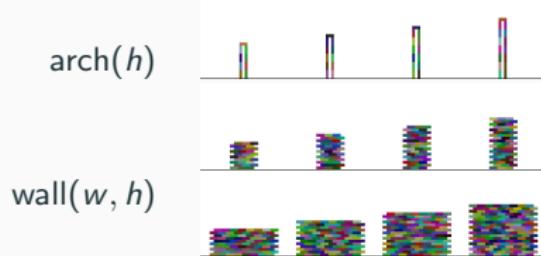


# Planning to build towers

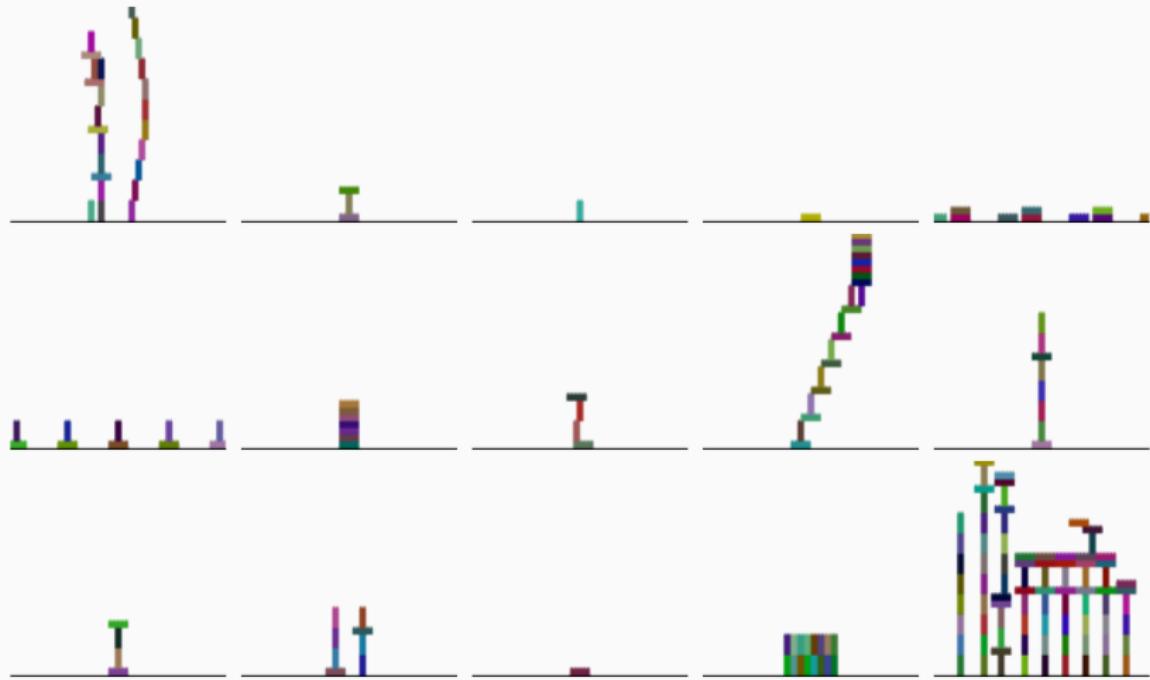
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.  
example tasks (112 total)



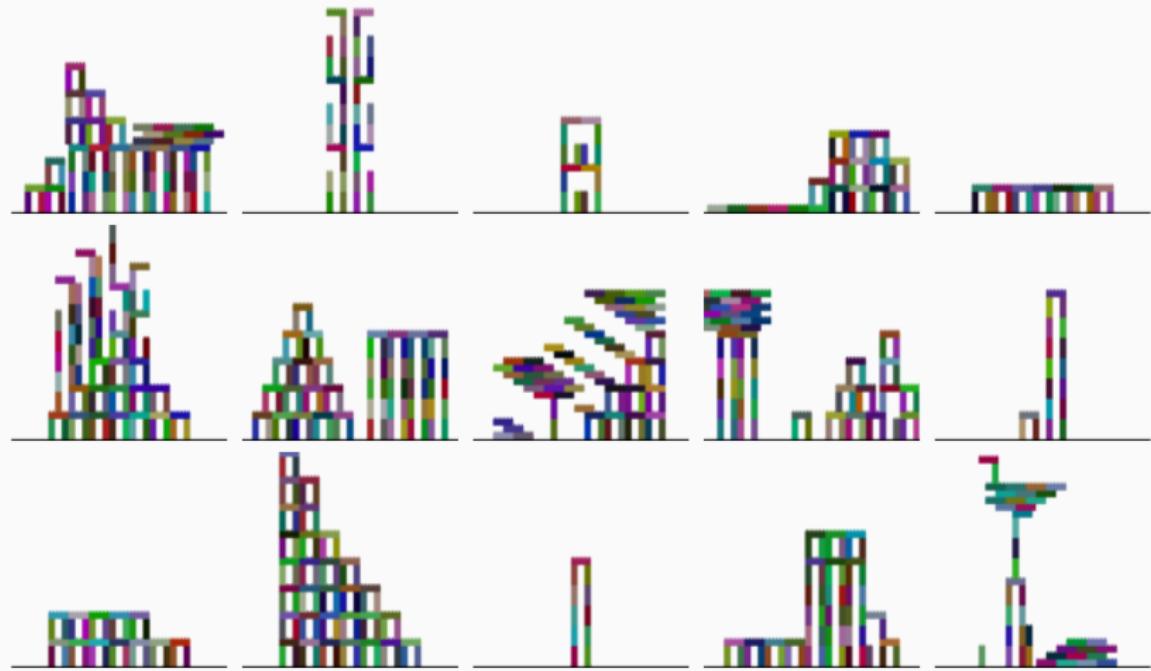
learned library routines ( $\approx 20$  total)



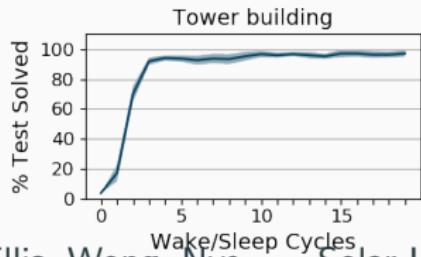
# Dreams before learning



# Dreams after learning

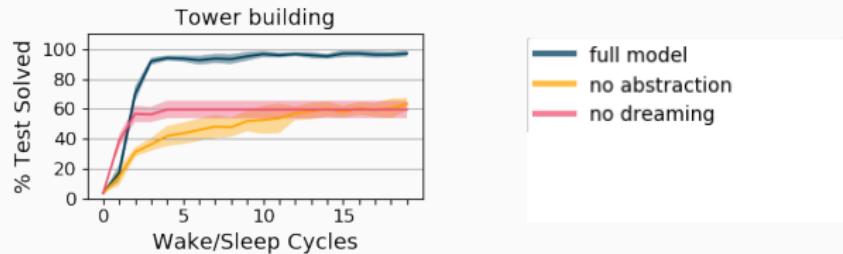


# Learning dynamics



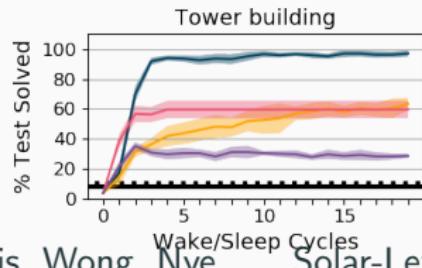
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Learning dynamics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Learning dynamics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Synergy between dreaming and library learning (higher-level knowledge)

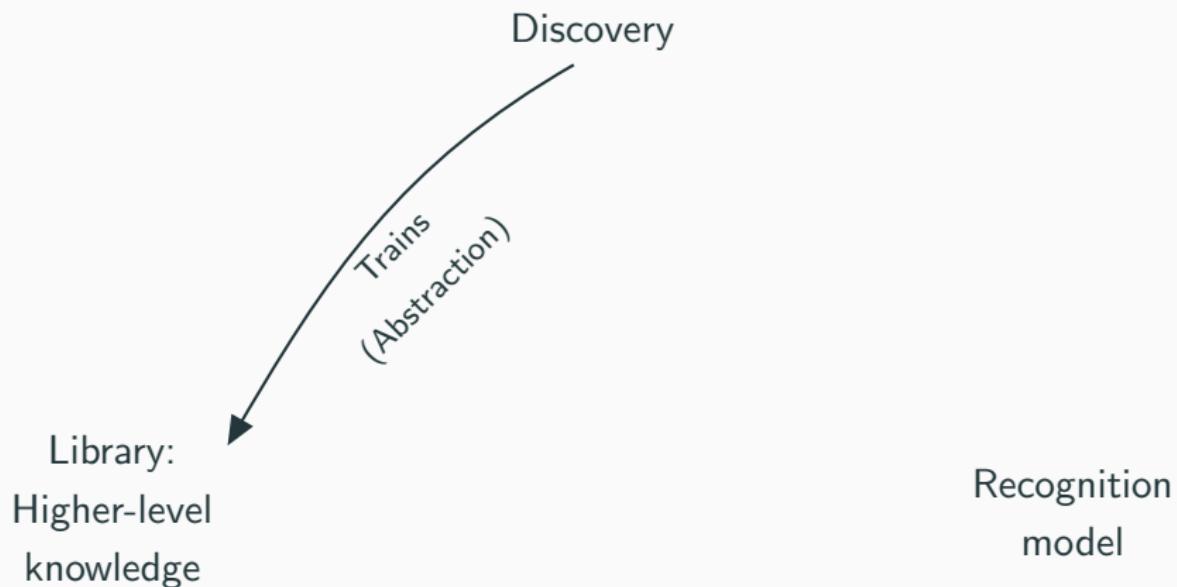
Discovery

Library:  
Higher-level  
knowledge

Recognition  
model

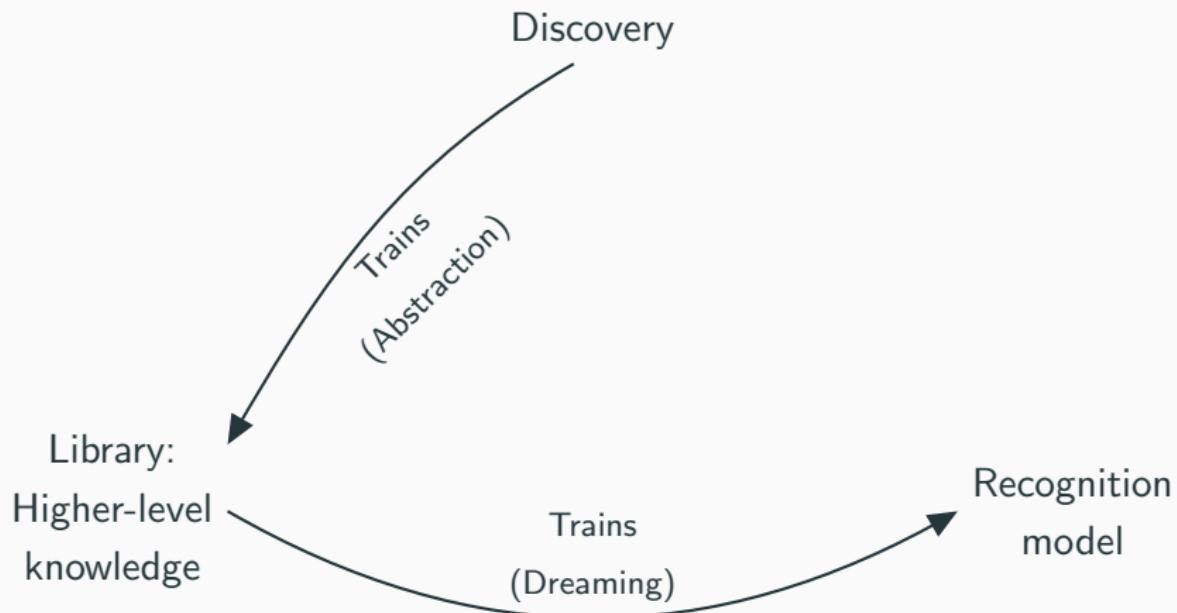
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Synergy between dreaming and library learning (higher-level knowledge)



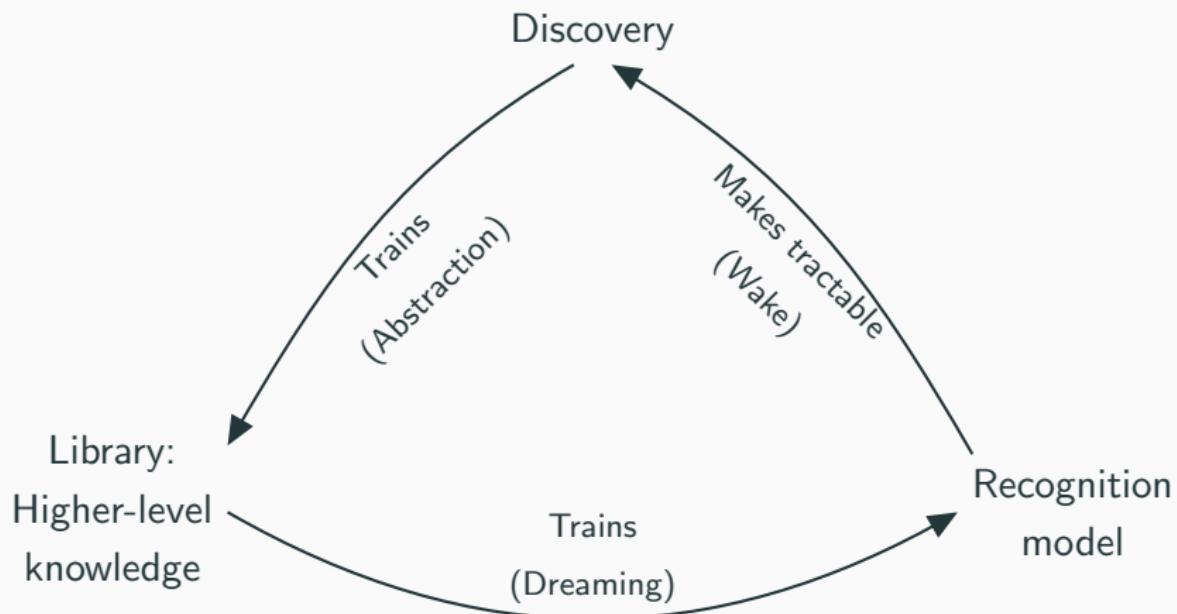
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

## Synergy between dreaming and library learning (higher-level knowledge)



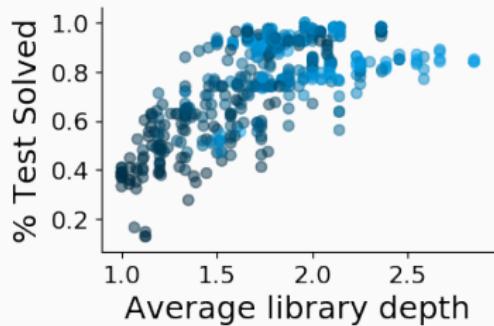
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Synergy between dreaming and library learning (higher-level knowledge)



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# Evidence for dreaming bootstrapping better libraries

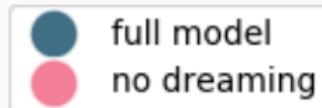
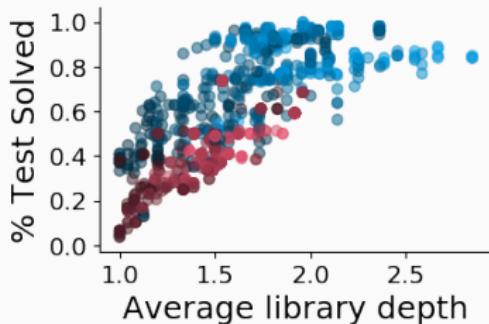


Darker: Early in learning

Brighter: Later in learning

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2013

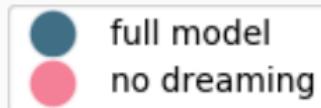
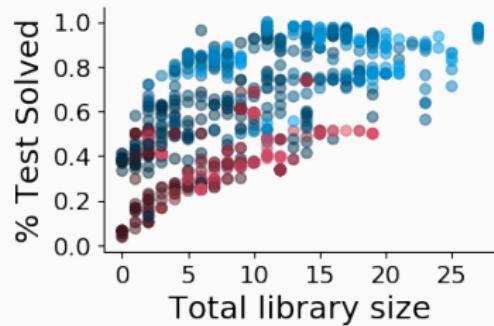
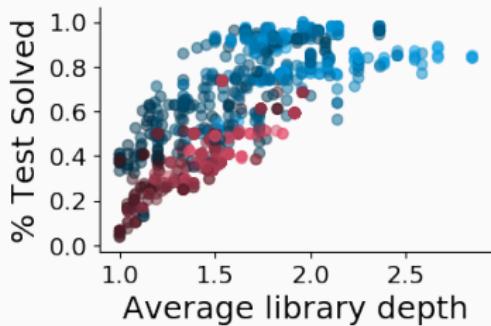
# Evidence for dreaming bootstrapping better libraries



Darker: Early in learning

Brighter: Later in learning

# Evidence for dreaming bootstrapping better libraries



Darker: Early in learning

Brighter: Later in learning

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

From learning libraries,  
to learning languages

# From learning libraries, to learning languages

modern functional programming → physics

From learning libraries,  
to learning languages

1950's Lisp → modern functional programming → physics

# Physics Formula Sheet

## Mechanics

$x = x_0 + v_{x0}t + \frac{1}{2}a_xt^2$	$a_t = \frac{v^2}{r}$	$ \vec{F}_{\text{spring}}  = k  \vec{x} $
$v = v_0 + at$	$\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$	$\text{PE}_{\text{spring}} = \frac{1}{2}kx^2$
$v_s^2 - v_{s0}^2 = 2a(x - x_0)$	$\omega = \omega_0 + \alpha t$	$T_{\text{spring}} = 2\pi \sqrt{\frac{m}{k}}$
$\bar{a} = \frac{\sum \vec{F}}{m} = \frac{\vec{F}_{\text{net}}}{m}$	$T = \frac{2\pi}{\omega} = \frac{1}{f}$	$T_{\text{pendulum}} = 2\pi \sqrt{\frac{l}{g}}$
$ \vec{F}_{\text{friction}}  \leq \mu  \vec{F}_{\text{Normal}} $	$v = f\lambda$	
$\bar{p} = m\bar{v}$	$x = A \cos(2\pi ft)$	$ \vec{F}_{\text{gravity}}  = G \frac{m_1 m_2}{r^2}$
$\Delta \bar{p} = \vec{F} \Delta t$	$\bar{a} = \frac{\sum \vec{F}}{l} = \frac{\vec{F}_{\text{net}}}{l}$	$ \vec{F}_{\text{gravity}}  = m\bar{g}$
$KE = \frac{1}{2}mv^2$	$\vec{r} = r \times \vec{F}$	$\text{PE}_{\text{gravity}} = -G \frac{m_1 m_2}{r}$
$\Delta PE = mg\Delta y$	$L = I\omega$	$p = \frac{m}{V}$
$\Delta E = W = Fd \cos\theta$	$\Delta L = \tau \Delta t$	$KE = \frac{1}{2}I\omega^2$

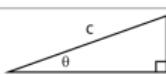
## Electricity

$ \vec{F}_E  = k \left  \frac{q_1 q_2}{r^2} \right $	$\Delta V = IR$	$R = \frac{\rho l}{A}$
$I = \frac{\Delta q}{\Delta t}$		$P = I \Delta V$
$R_{\text{series}} = R_1 + R_2 + \dots + R_n$	$\frac{1}{R_{\text{parallel}}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n}$	

## Geometry

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

## Trigonometry

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

## 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  
 $\alpha$  = angular acceleration  
 $\lambda$  = wavelength  
 $\mu$  = coefficient of friction

# Growing languages for vector algebra and physics

## Initial Primitives

map  
zip

cons

empty

cdr

power

fold

car

+

-

\*

/

0

1

$\pi$

## Physics Equations

### Newton's Second Law

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

### Parallel Resistors

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

### 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

$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r}_1 - \vec{r}_2|^2} \widehat{\vec{r}_1 - \vec{r}_2}$$

# Growing languages for vector algebra and physics

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

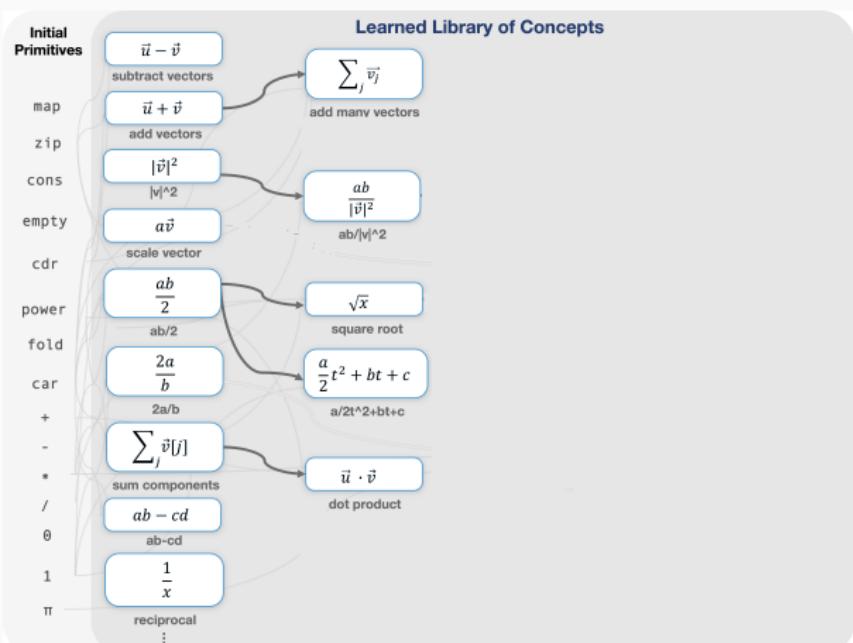
## Initial Primitives

	$\vec{u} - \vec{v}$
map	subtract vectors
zip	$\vec{u} + \vec{v}$
cons	add vectors
empty	$ \vec{v} ^2$
cdr	$ v ^2$
power	$a\vec{v}$
fold	scale vector
car	$\frac{ab}{2}$
+	$\frac{ab}{2}$
-	$\frac{2a}{b}$
*	$ab/2$
/	$2a/b$
θ	$\sum_j \vec{v}[j]$
1	sum components
π	$ab - cd$
	$ab - cd$
	$\frac{1}{x}$
	reciprocal
	⋮

## Learned Library of Concepts

Physics Equations
<b>Newton's Second Law</b>
$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$
<b>Parallel Resistors</b>
$R_{total} = \left( \sum_i \frac{1}{R_i} \right)^{-1}$
<b>Work</b>
$U = \vec{F} \cdot \vec{d}$
<b>Force in a Magnetic Field</b>
$ \vec{F}  = q \vec{v} \times \vec{B} $
<b>Kinetic Energy</b>
$KE = \frac{1}{2} m  \vec{v} ^2$
<b>Coulomb's Law</b>
$\vec{F} \propto \frac{q_1 q_2}{ \vec{r}_1 - \vec{r}_2 ^2} \widehat{\vec{r}_1 - \vec{r}_2}$

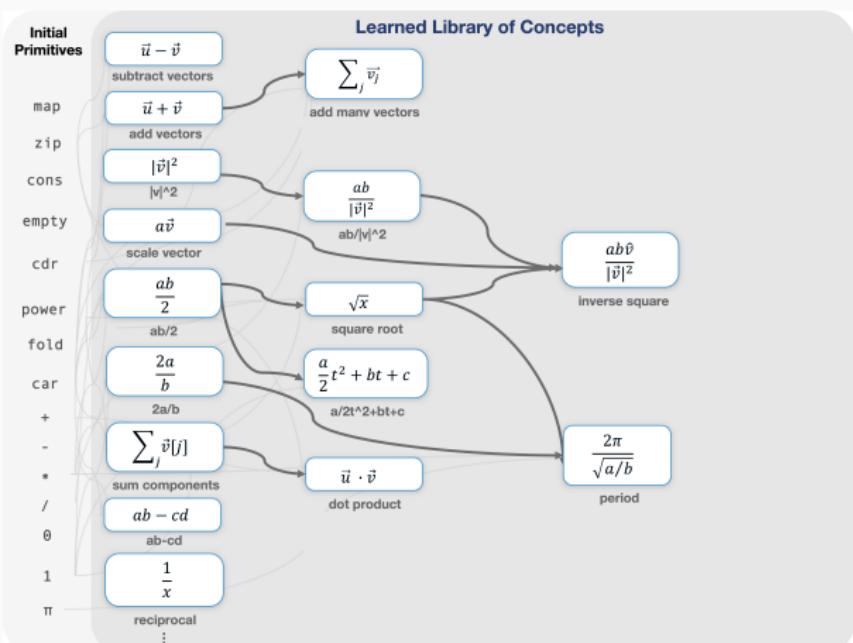
# Growing languages for vector algebra and physics



Physics Equations	
Newton's Second Law	Parallel Resistors
$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$	$R_{total} = \left( \sum_i \frac{1}{R_i} \right)^{-1}$
Work	Force in a Magnetic Field
$U = \vec{F} \cdot \vec{d}$	$ \vec{F}  = q \vec{v} \times \vec{B} $
Kinetic Energy	Coulomb's Law
$KE = \frac{1}{2} m  \vec{v} ^2$	$\vec{F} \propto \frac{q_1 q_2}{ \vec{r}_1 - \vec{r}_2 ^2} \hat{r}_1 - \hat{r}_2$

# Growing languages for vector algebra and physics

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.



## Physics Equations

### Newton's Second Law

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

### Parallel Resistors

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

### Work

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

### Force in a Magnetic Field

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

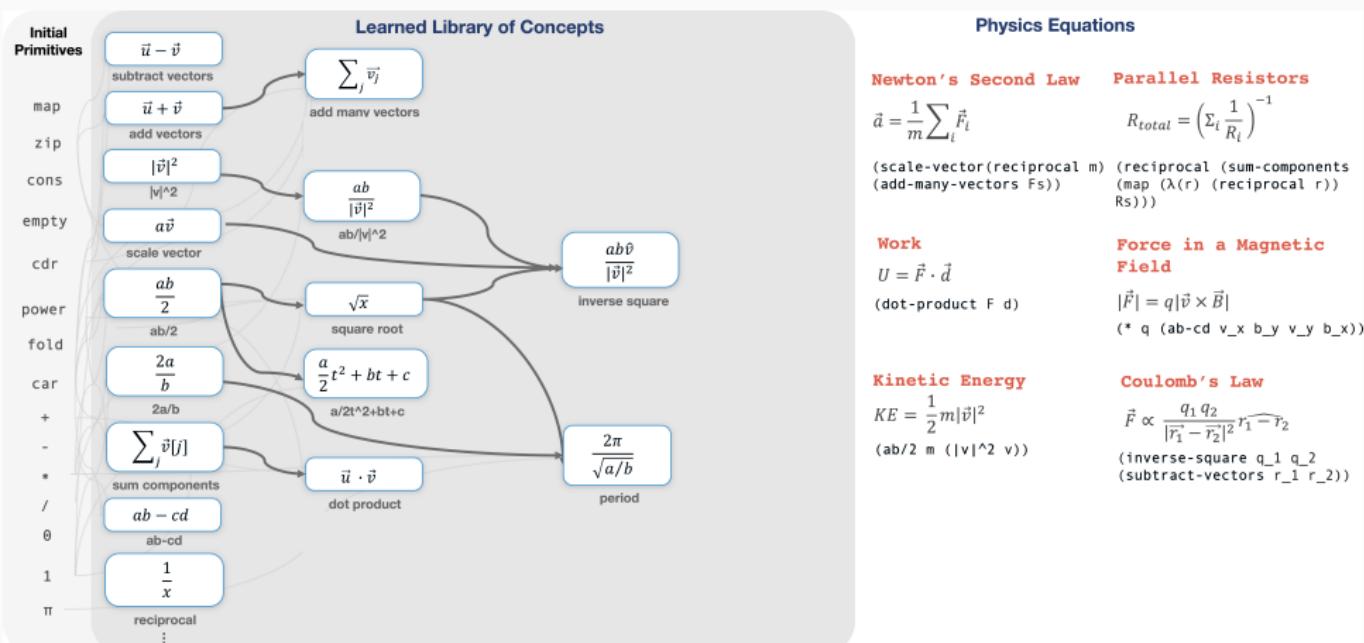
### Kinetic Energy

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

### Coulomb's Law

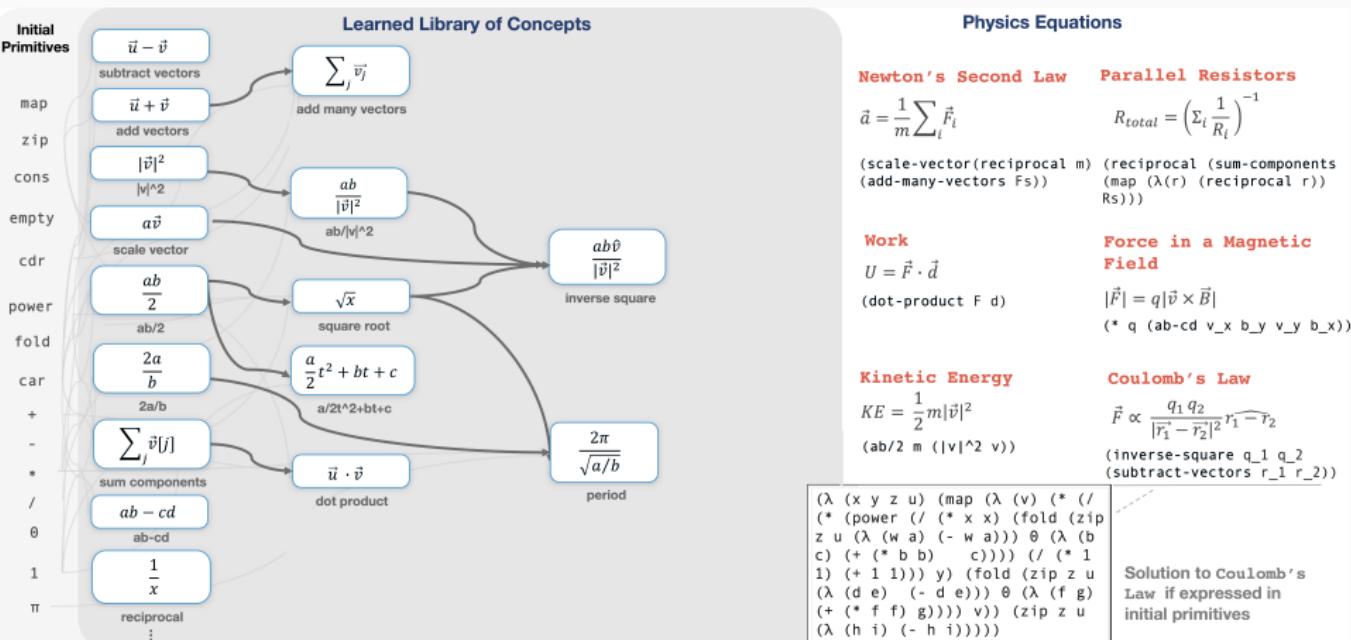
$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r}_1 - \vec{r}_2|^2} \hat{r}_1 - \hat{r}_2$$

# Growing languages for vector algebra and physics



# Growing languages for vector algebra and physics

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.



# Growing a language for recursive programming

## Initial Primitives

Y

combinator

cons

car

cdr

nil

if

nil?

+

-

0

1

=

## Recursive Programming Algorithms

### Stutter

[ ] → [ ]  
[ ] → [ ]

### Take every other

[ ] → [ ]  
[ ] → [ ]

### List lengths

[ , []] → [3 1]  
[[ ], [], []] → [2 0 1]

### List differences

[1 8 2], [0 5 1] → [1 3 1]  
[2 3 6], [1 2 4] → [1 1 2]

# Growing a language for recursive programming

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

**Initial Primitives**

Y  
combinator  
  
cons  
car  
  
cdr  
nil  
  
if  
  
nil?  
  
+  
  
-  
  
0  
1  
  
=



**fold**

```
fold(xs,f,x0) =  
(if (nil? xs) x0  
(f (fold (cdr xs)  
f x0) (car xs)))
```

**unfold**

```
unfold(x,g,f,p) =  
(if (p x) nil  
(cons (f x)  
(unfold (g f x)  
g f p)))
```

**Learned Library of Concepts**

**Recursive Programming Algorithms**

**Stutter**

```
[■■] → [■■■■]  
[■■■] → [■■■■■■]
```

`(fold A (λ (u v) (cons v (cons v u))) nil)`

**Take every other**

```
[■■■■] → [■■]  
[■■■■■■■■] → [■■■■]
```

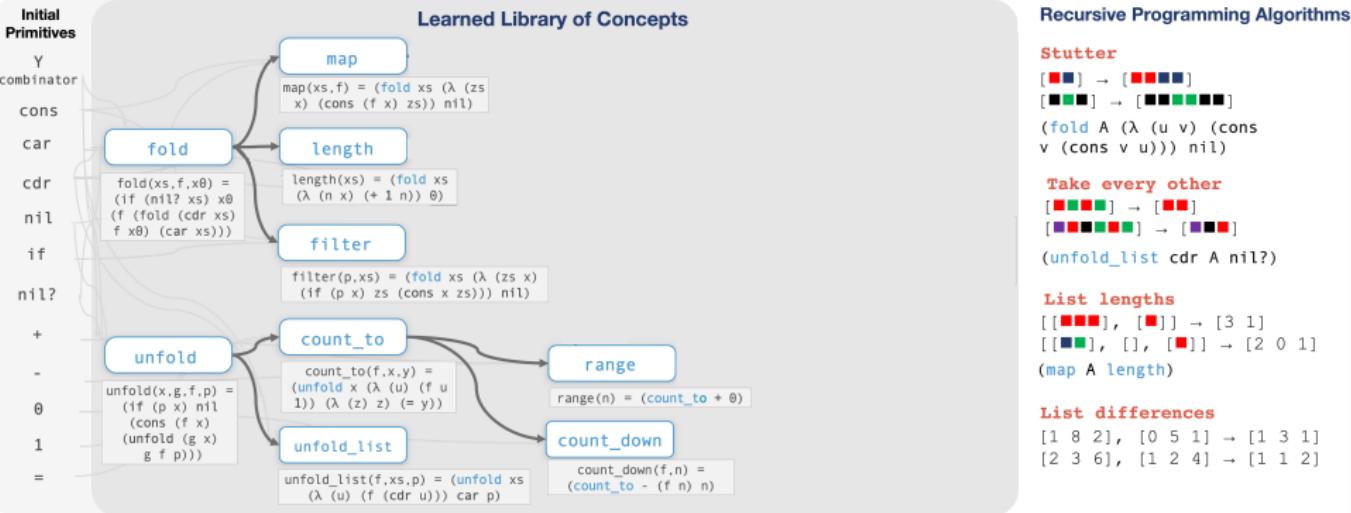
**List lengths**

```
[[■■■], [■]] → [3 1]  
[[■■■], [], [■]] → [2 0 1]
```

**List differences**

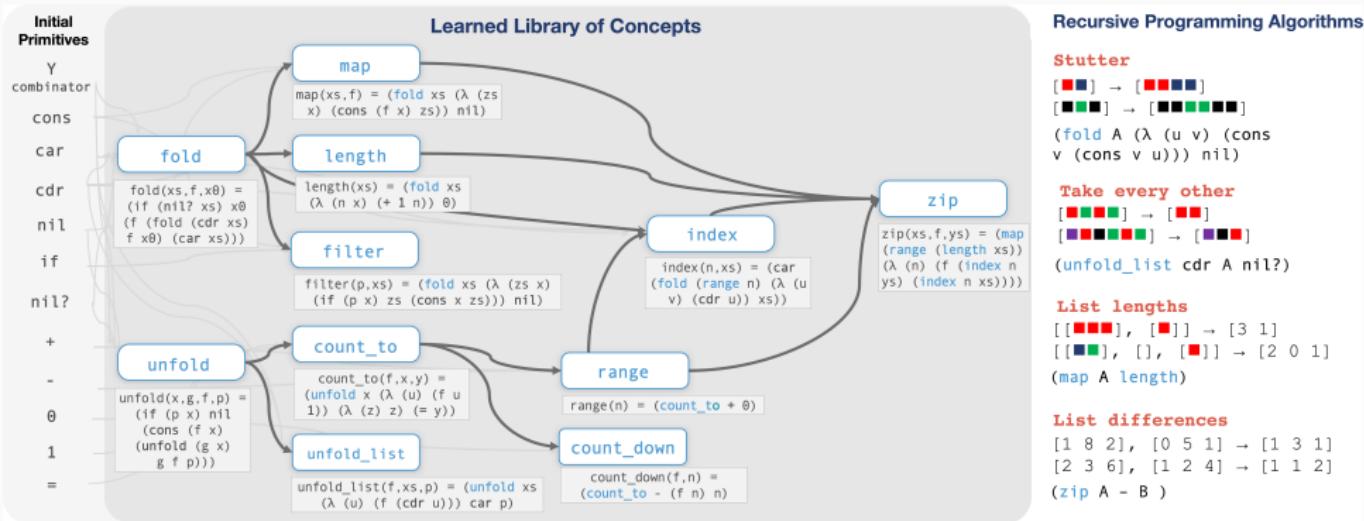
```
[1 8 2], [0 5 1] → [1 3 1]  
[2 3 6], [1 2 4] → [1 1 2]
```

# Growing a language for recursive programming



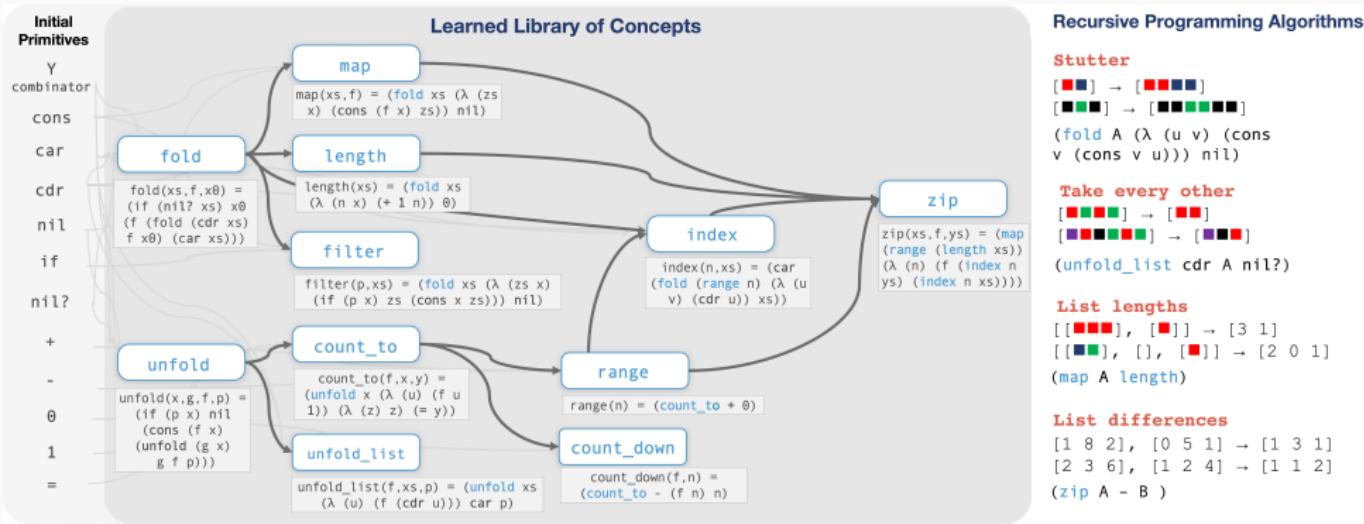
# Growing a language for recursive programming

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.



# Growing a language for recursive programming

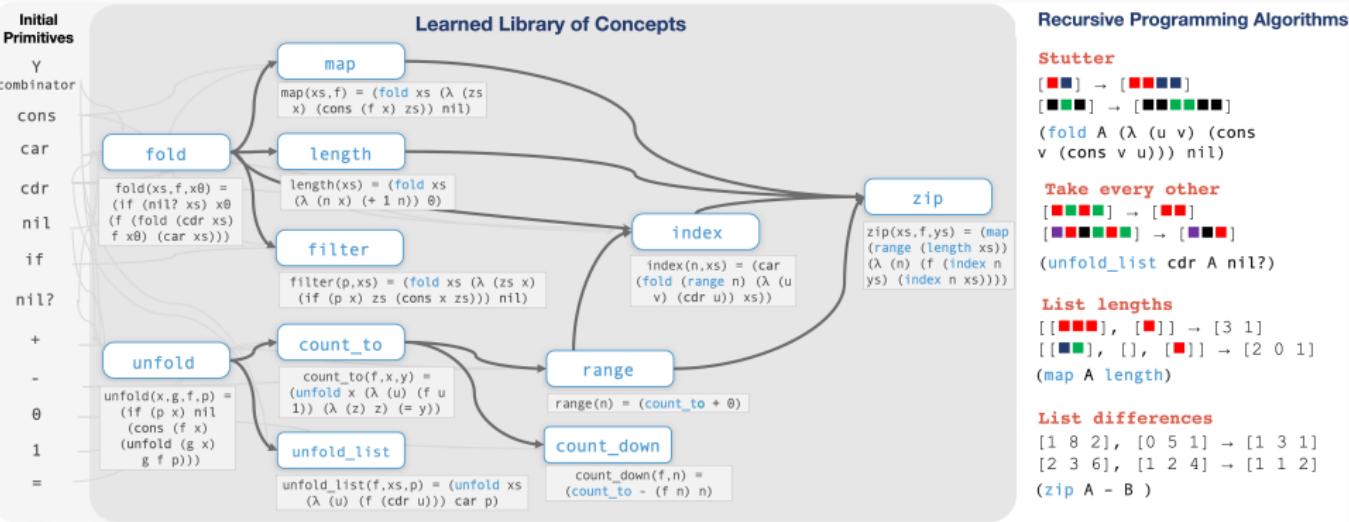
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.



Origami Programming: Jeremy Gibbons, 2003

# Growing a language for recursive programming

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.



1 year of compute. 5 days on 64 CPUs.



Origami Programming: Jeremy Gibbons, 2003

## Lessons

Symbols aren't necessarily interpretable. Flexibly grow the language based on experience to make it more powerful *and* more human understandable

Learning-from-scratch is possible in principle. Don't do it. But program induction makes it convenient to build in what we know how to build in, and then learn and adapt on top of that

Program Induction and perception  
learning to learn  
model discovery  
the future

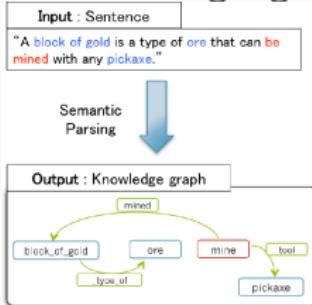
What we've got now:

a toolkit for program induction,  
addressing combinatorial program search via learning, integrating  
techniques for machine learning, AI, and programming languages

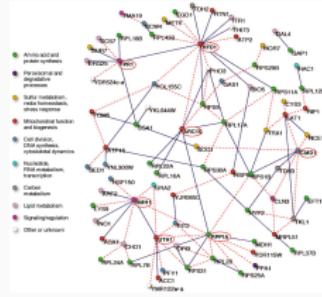
Where will this toolkit take us?

# What's in reach

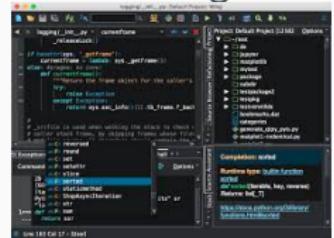
## Library learning+ Natural language



## Computer-Aided Science

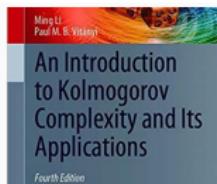


## Synthesis for software engineers



## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$



## Modeling the physical world

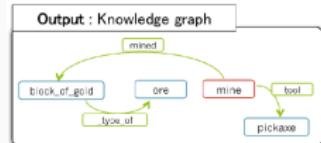


# What's in reach

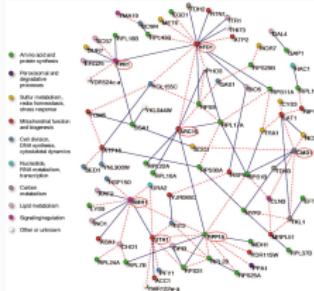
## Library learning+ Natural language

Input : Sentence  
"A block of gold is a type of ore that can be mined with any pickaxe."

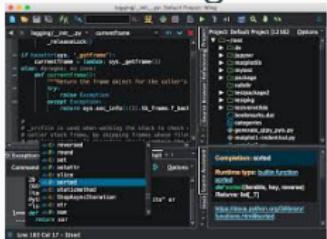
Semantic  
Parsing



## Computer-Aided Science

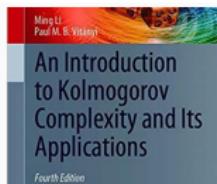


## Synthesis for software engineers



## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$

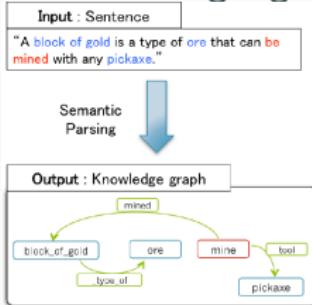


## Modeling the physical world

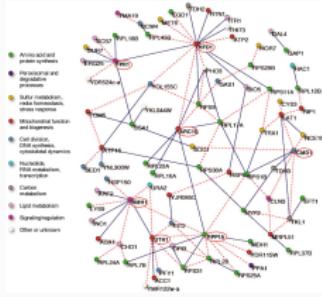


# What's in reach

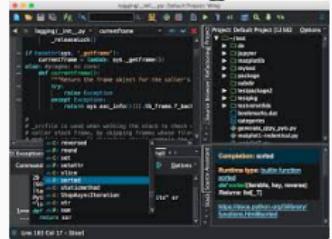
## Library learning + Natural language



## Computer-Aided Science

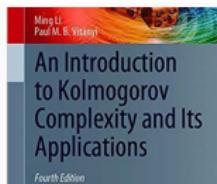


## Synthesis for software engineers



## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$

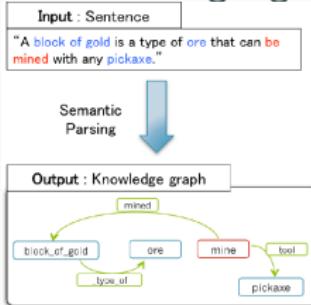


## Modeling the physical world

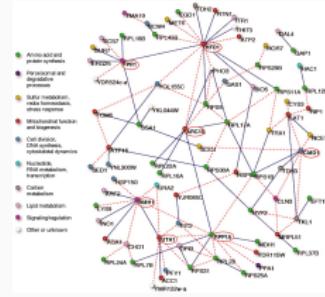


# What's in reach

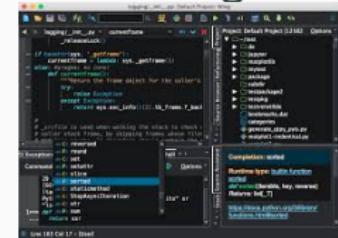
## Library learning+ Natural language



## Computer-Aided Science

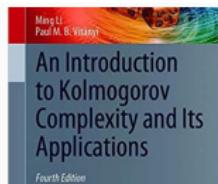


## Synthesis for software engineers



## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$

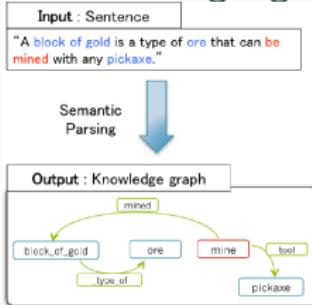


## Modeling the physical world

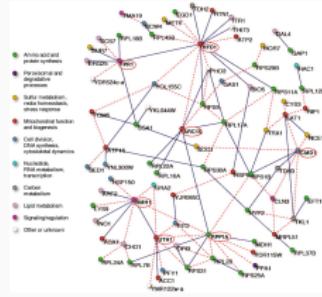


# What's in reach

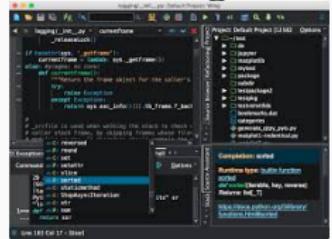
## Library learning+ Natural language



## Computer-Aided Science

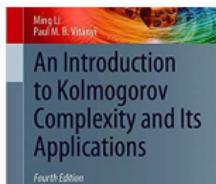


## Synthesis for software engineers

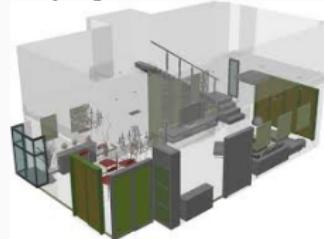


## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$

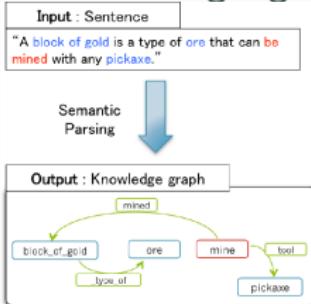


## Modeling the physical world

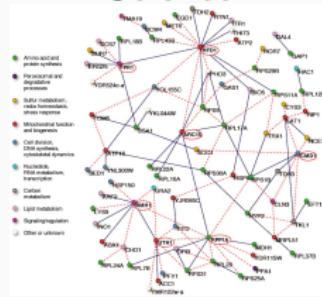


# What's in reach

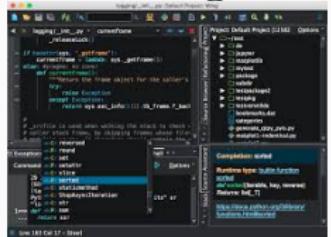
## Library learning+ Natural language



## Computer-Aided Science

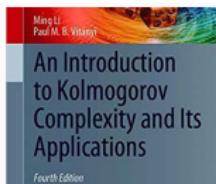


## Synthesis for software engineers

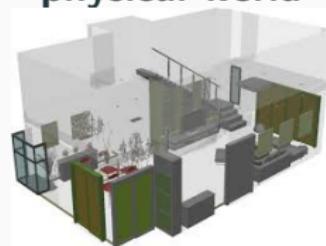


## Theory for program induction

$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$



## Modeling the physical world



These next steps are in reach.

More broadly: what's in the future of machine learning?

# What we want for the future of machine learning

Strong generalization

# What we want for the future of machine learning

Strong generalization

Bootstrapping, learning-to-learn, representation learning

# What we want for the future of machine learning

Strong generalization

Bootstrapping, learning-to-learn, representation learning

Discovering knowledge that humans can understand and build on

Josh Tenenbaum



Armando Solar-Lezama



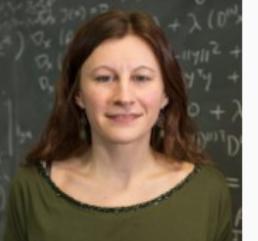
Evan Pu



Lucas Tian



Marta Kryven



Chris Yang



Ronald Alvarez



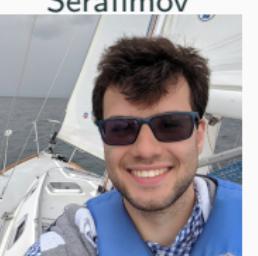
Max Nye



Cathy Wong



Kliment Serafimov



Eyal Dechter



Mathias Sable-Meyer



Lucas Morales



Tao Du



Felix Sosa



Sam Tenka

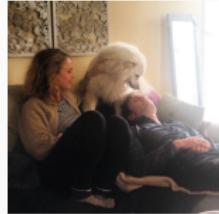


Julio Cortazar



thank you...

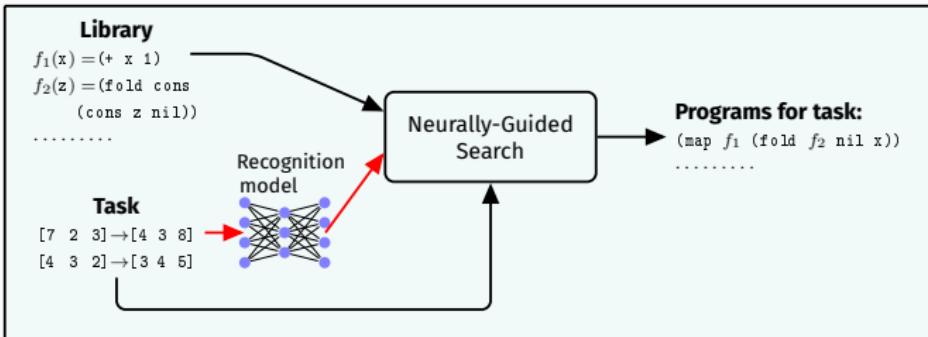
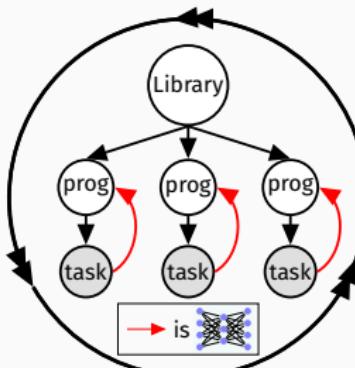
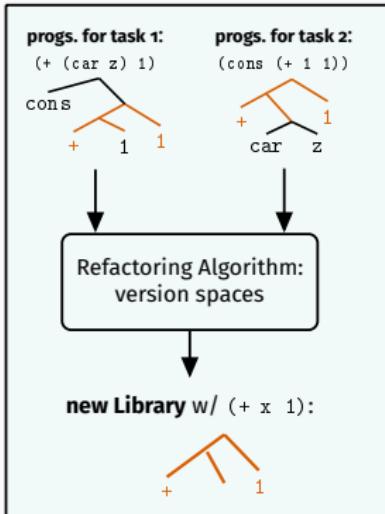
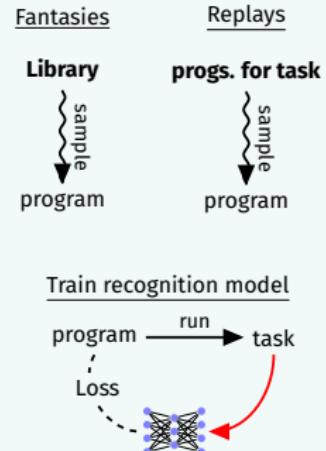
family



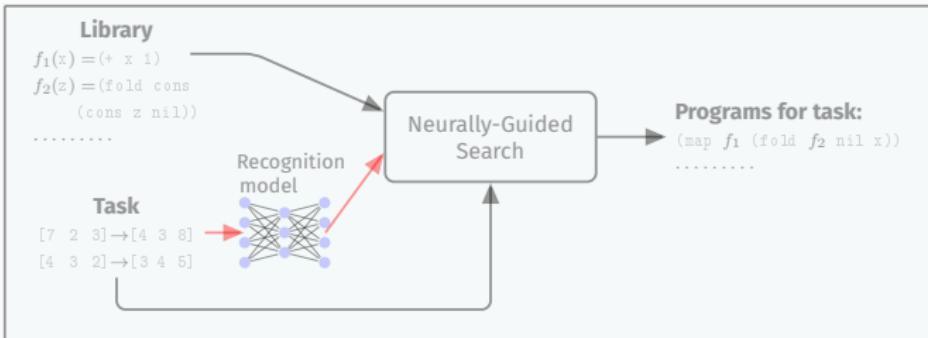
MIT Brain and Cognitive Sciences

twitch viewers

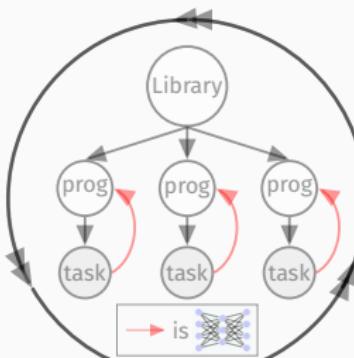
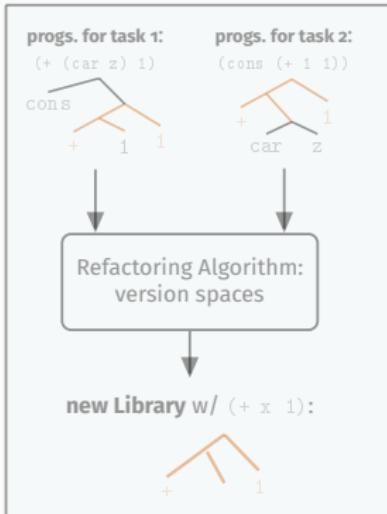
the end.

**WAKE****SLEEP: ABSTRACTION****SLEEP: DREAMING**

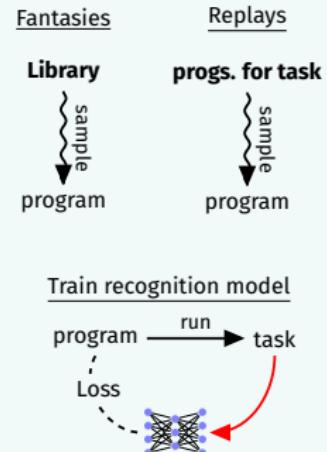
## WAKE



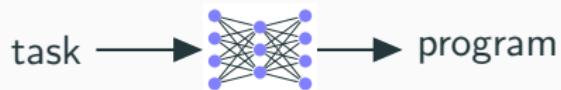
## SLEEP: ABSTRACTION



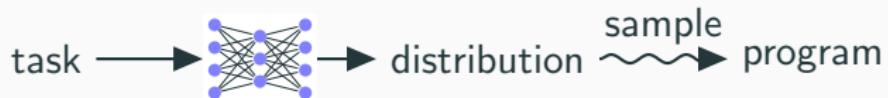
## SLEEP: DREAMING



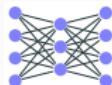
# Neural recognition model guides search



# Neural recognition model guides search



# Neural recognition model guides search

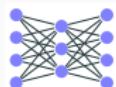


is a...

recurrent network (Devlin et al 2017)

unigram model (Menon et al 2013; Balog et al 2016)

# Neural recognition model guides search

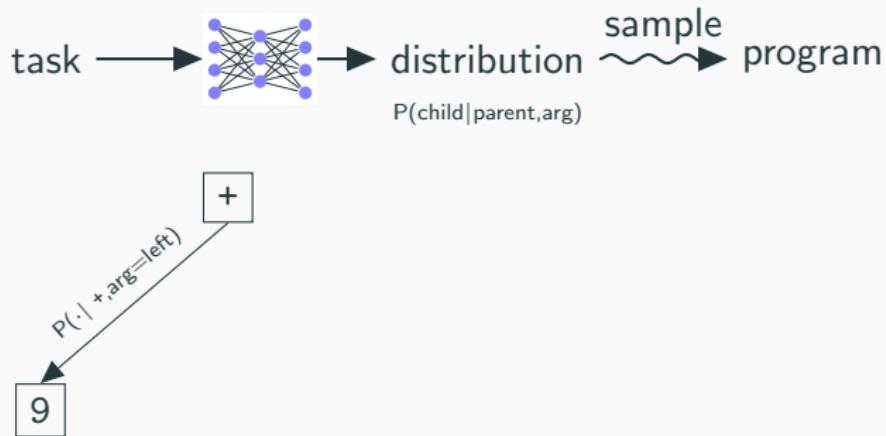


is a “**bigram**” model over syntax trees

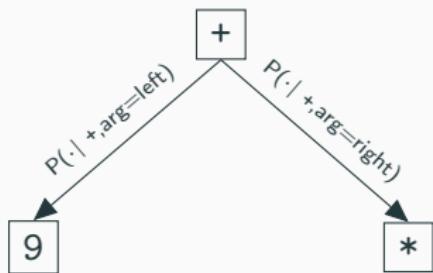
# Neural recognition model guides search



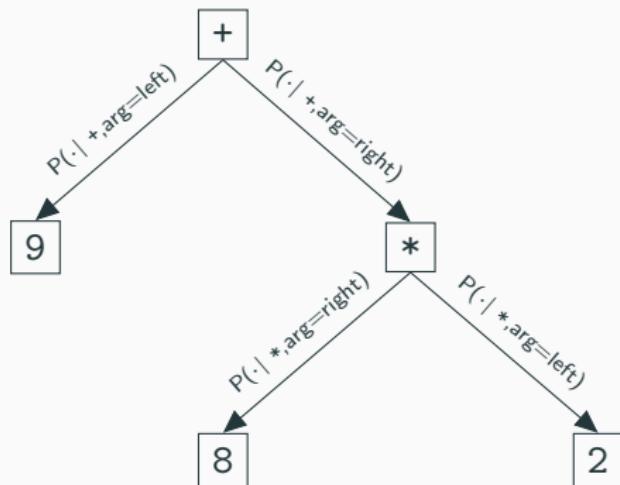
# Neural recognition model guides search



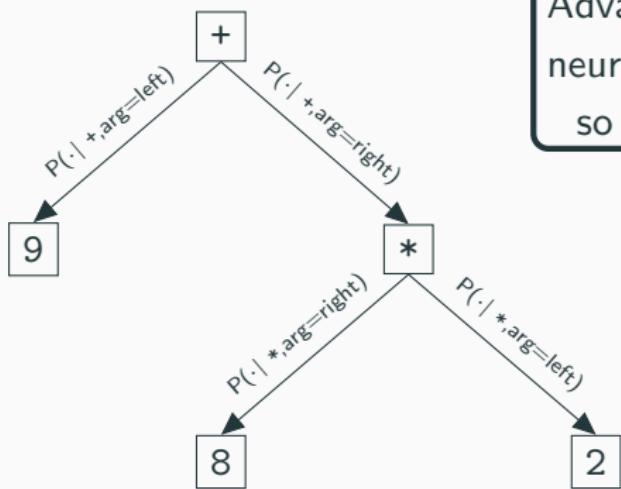
# Neural recognition model guides search



# Neural recognition model guides search

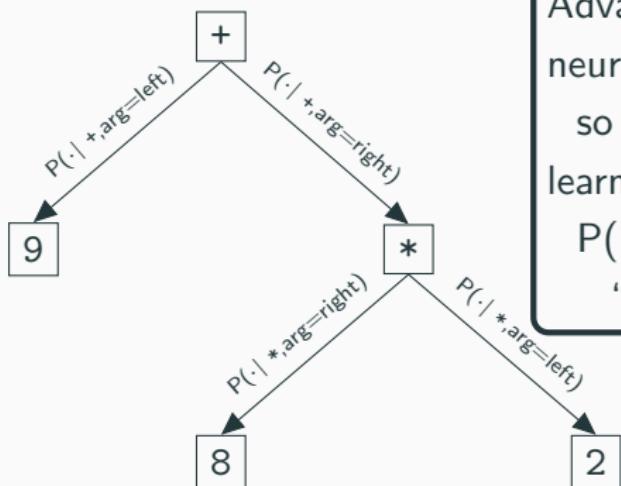


# Neural recognition model guides search



Advantages:  
neural net runs once per task,  
so CPU bottlenecks instead of GPU

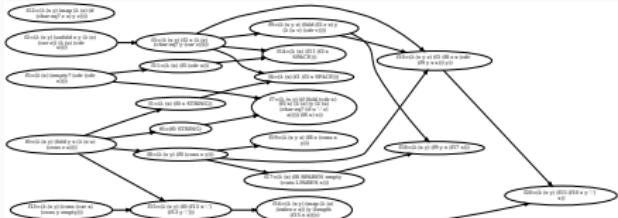
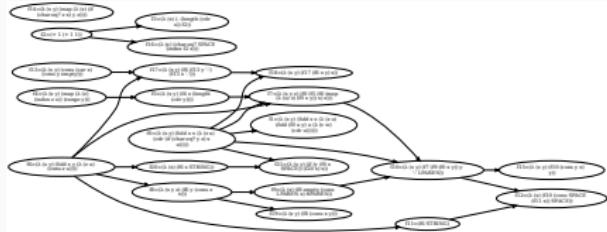
# Neural recognition model guides search



Advantages:  
neural net runs once per task,  
so CPU bottlenecks instead of GPU  
learns to break syntactic symmetries:  
 $P(1|*,\text{arg}=left)=0.0$   
“do not multiply by one”

## Library structure: Text Editing

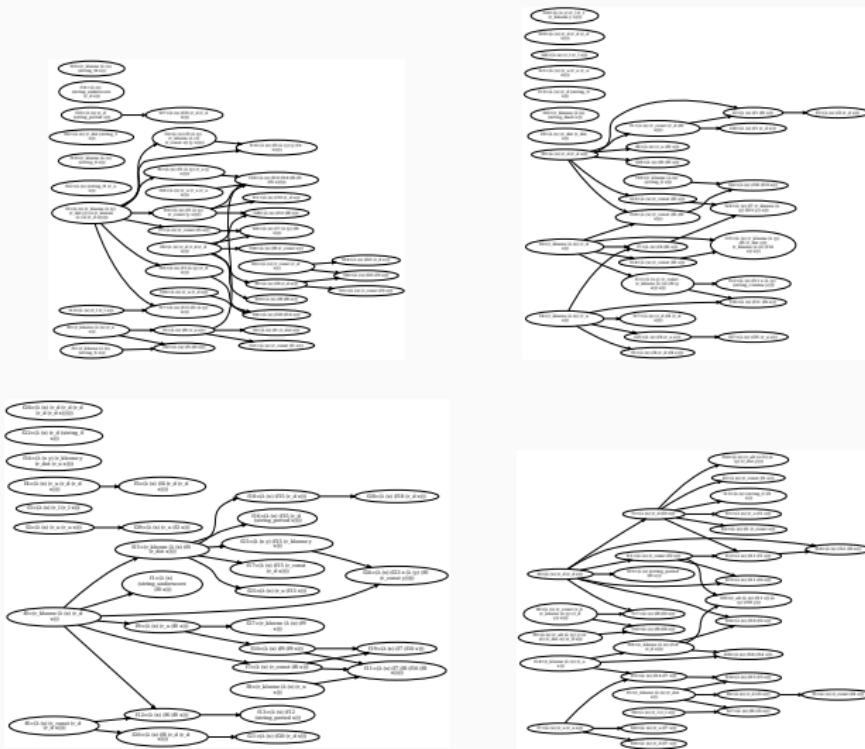
DreamCoder learns libraries for FlashFill-style text editing [Gulwani 2012]



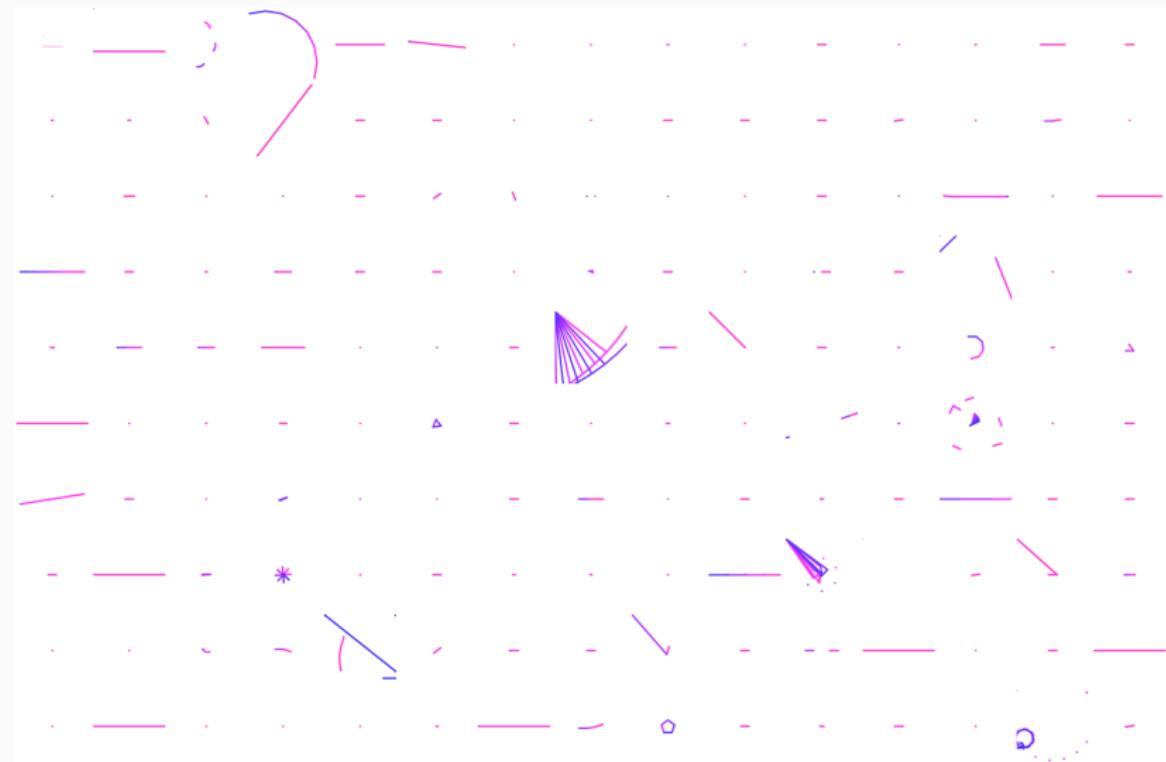
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

# 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

