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

Kevin Ellis

2021 Workshop on Artificial Scientific Discovery

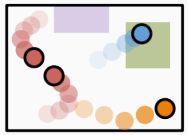
Cornell

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

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

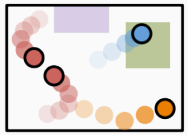
1. Scientific models as **generative programs**
2. Discovery as **program synthesis**
3. Higher-level knowledge

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

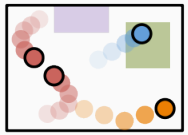
- *: large mass*
- *: medium mass*
- *: small mass*
- *: high friction*
- *: no friction*

Force parameters

Force b/w reds: attract

Ullman et al 2014:
Learning physics from dynamical scenes

Initial conditions



Property values

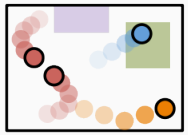
- *: large mass*
- *: medium mass*
- *: small mass*
- *: high friction*
- *: no friction*

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

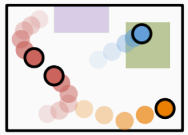
Force b/w reds: attract

```
(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

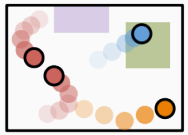
Force b/w reds: attract

```
(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*

```
(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 … )
```

Properties

Mass

```
(define (mass)
   (pair "mass" (uniform '(1 3 9))))
```

Friction

```
(define (friction)
   (pair "friction" '(uniform '(0 5 20))
```

Force classes

Pairwise: ① ↔ ②

```
(define (pairwise-force c1 c2)
 (let* ((a (uniform-draw '(-1 0 1))))
 (lambda (o1 o2)
  (let ((r (euc-dist o1 o2)))
   (/ (* a del(o1,col(o1)) del(o2,col
      (power r 2))))))
```

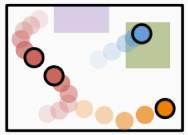
Global: ○→

```
(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 … )
```

Properties

Mass

```
(define (mass)
  (pair "mass" (uniform '(1 3 9))))
```

Friction

```
(define (friction)
  (pair "friction" '(uniform '(0 5 20))
```

Force classes

Pairwise: ① ↔ ②

```
(define (pairwise-force c1 c2)
 (let* ((a (uniform-draw '(-1 0 1))))
 (lambda (o1 o2)
  (let ((r (euc-dist o1 o2)))
   (/ (* a del(o1,col(o1)) del(o2,col
      (power r 2))))))
```

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 … ))
```

Newtonian Dynamics

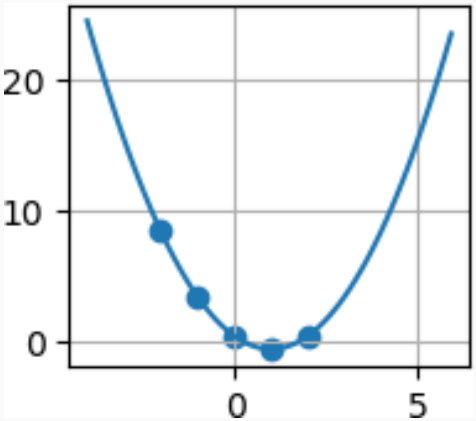
```
(define (run-dynamics entities forces
         init-cond steps dt)
  (if (= steps 0) '()
      (let* ((m (get-mass entities))
             (F (apply-forces forces entities))
             (a (/ F m))
             (new-cond (integrate init-cond
                         a dt noise)))
        (pair new-cond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

Ullman et al 2014:
Learning physics from dynamical scenes

Why programs?

Why programs?

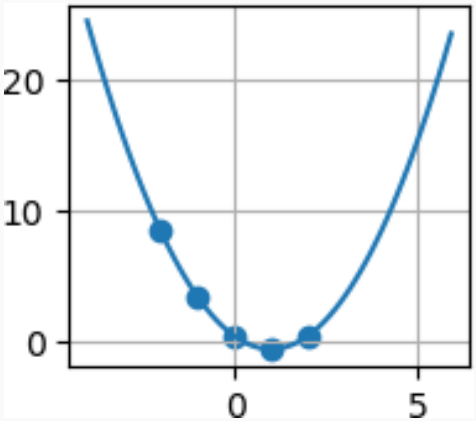
strong generalization




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

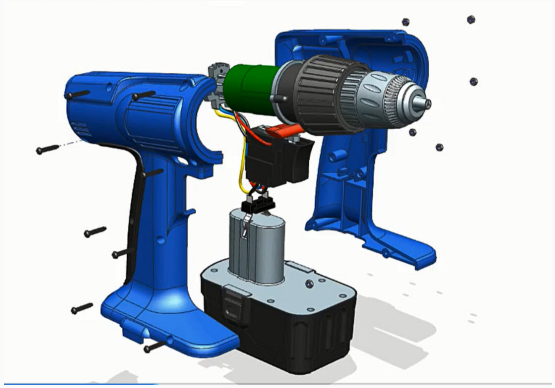
Why programs?

strong generalization

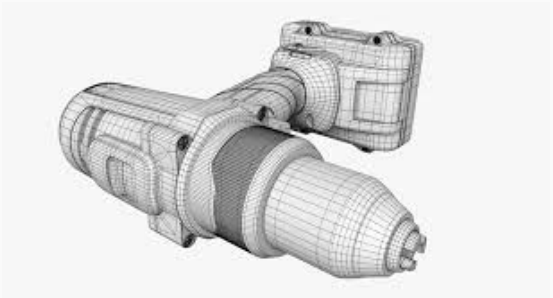



```
f(x)=(x-1)**2 - 0.5
```

interpretability

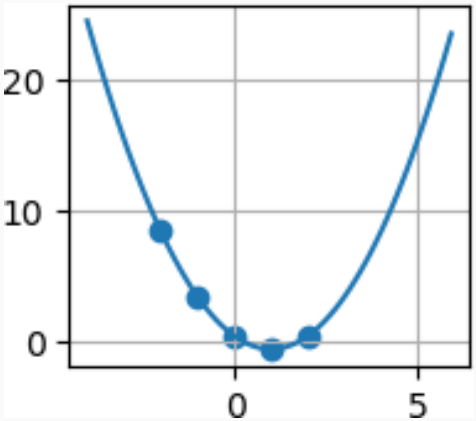


vs



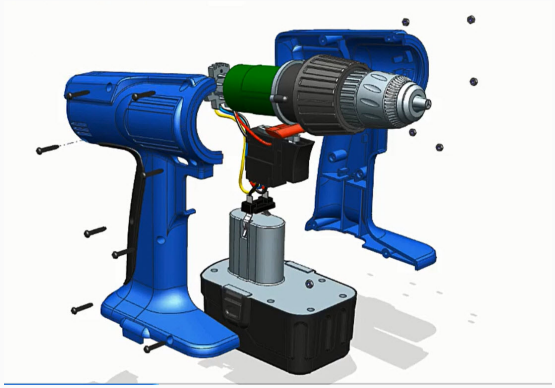
Why programs?

strong generalization

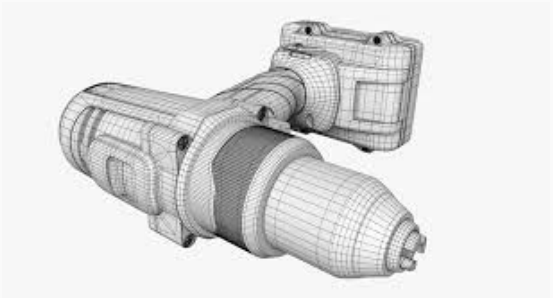



```
f(x)=(x-1)**2 - 0.5
```

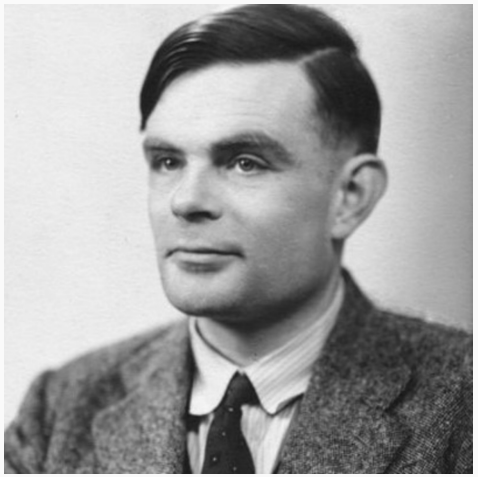
interpretability



vs

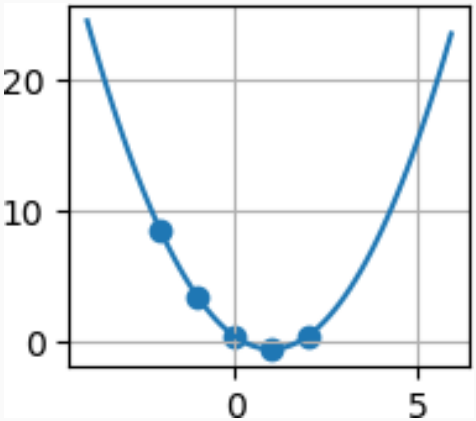


universal expressivity



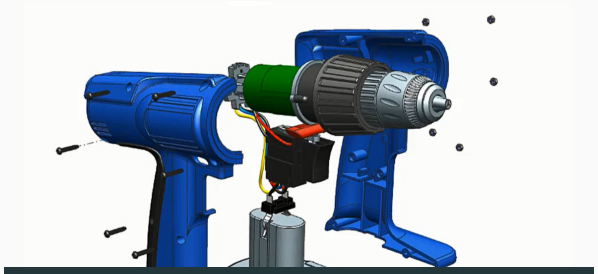
Why programs?

strong generalization



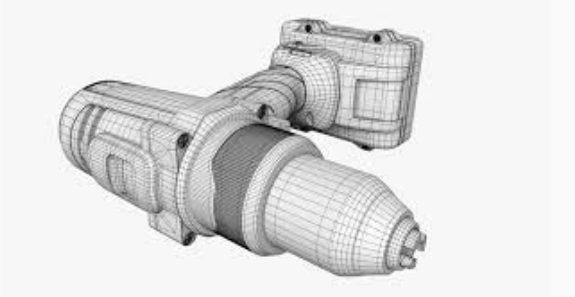

```
f(x)=(x-1)**2 - 0.5
```

interpretability

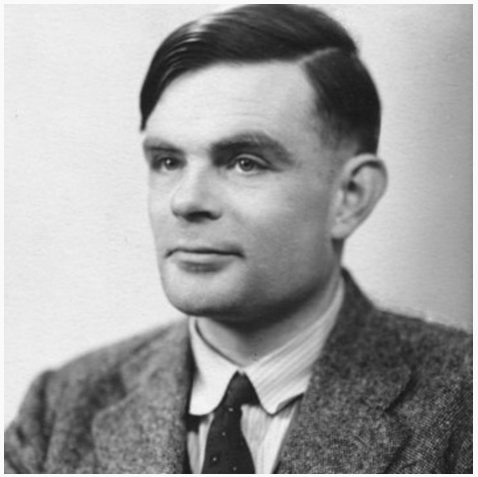


main challenge:
combinatorial search

vs



universal expressivity



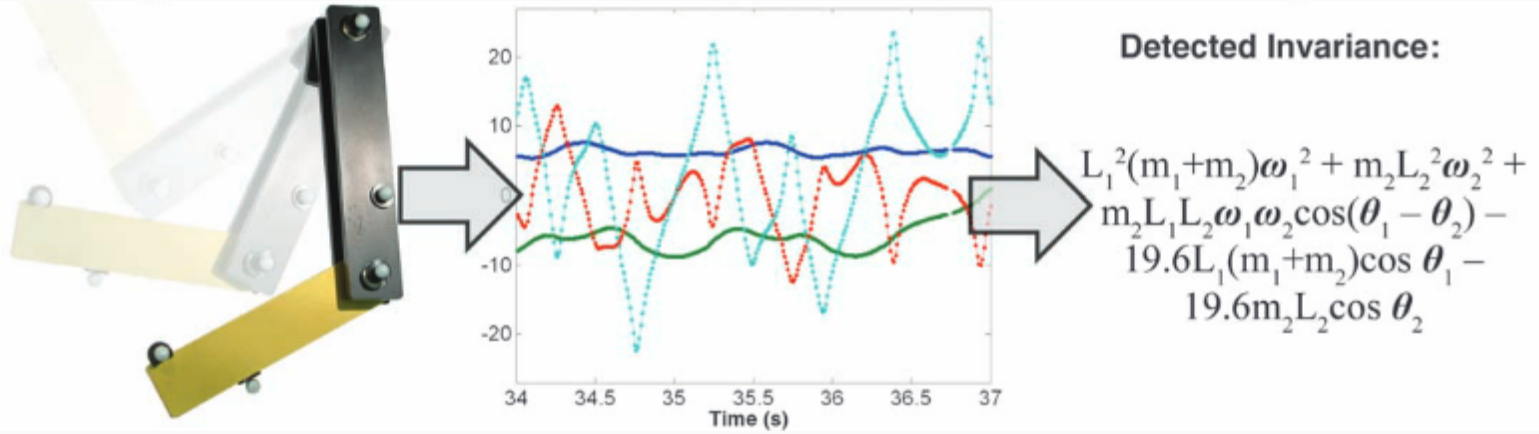
Higher-level Knowledge: Case study in linguistics

Higher-level Knowledge: General program discovery

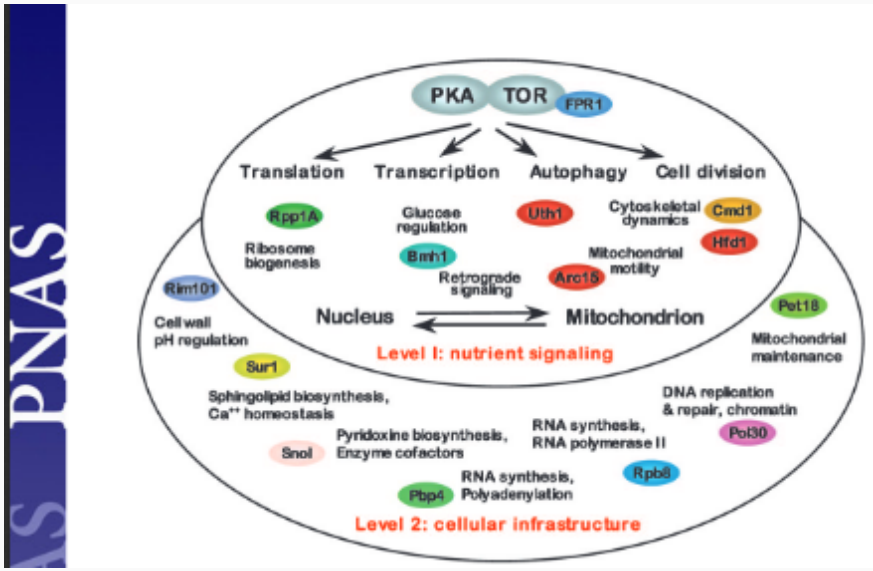
Higher-level Knowledge: Case study in linguistics

Higher-level Knowledge: General program discovery

Scientific discovery



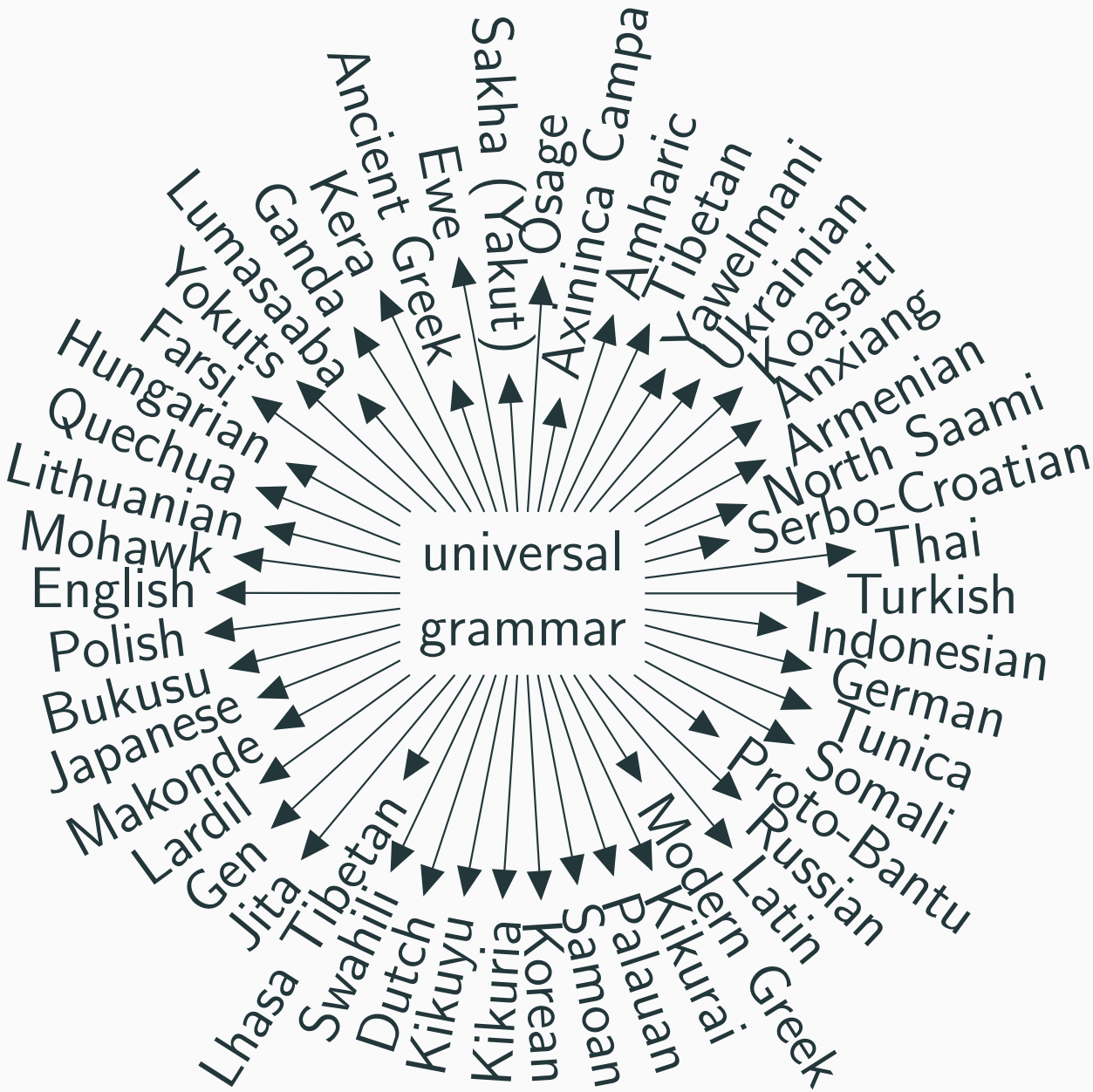
Schmidt & Lipson: "Distilling Free-Form Natural Laws from Experimental Data"



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

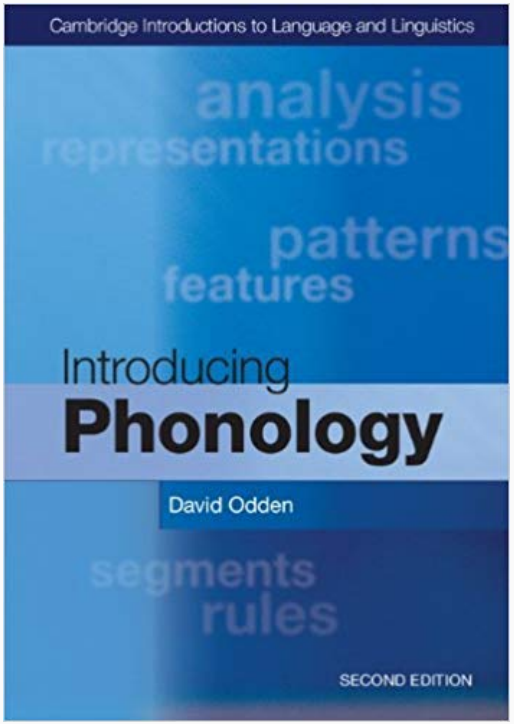
Serbo-Croatian:

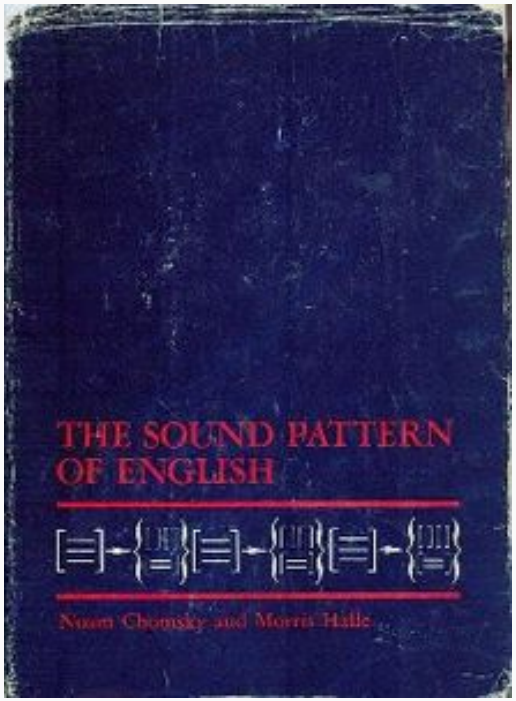
	masculine	feminine
"rich"	bogat	bogata
"mild"	blag	blaga
"green"	zelen	zelena
"clear"	**yasan**	yasna

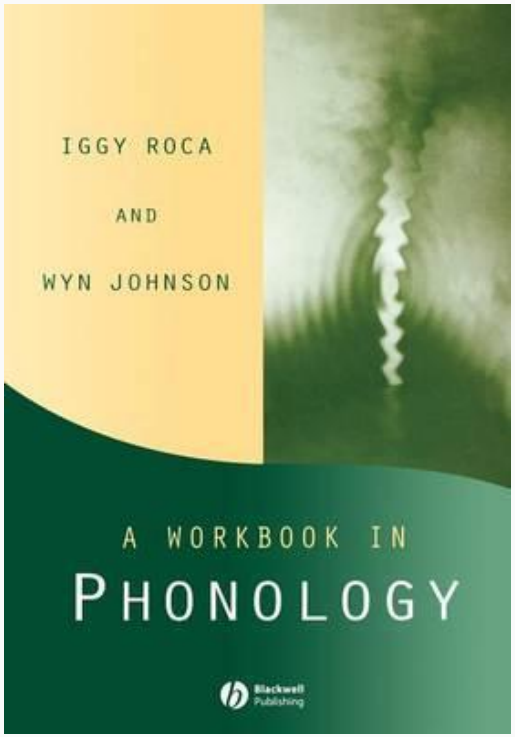
add "a" to stem to make feminine

insert "a" between two word-final consonants

$\varnothing \rightarrow$ a / C_C#


Cambridge Introductions to Language and Linguistics
analysis
representations
patterns
features
Introducing
Phonology
David Odden
segments
rules
SECOND EDITION


THE SOUND PATTERN OF ENGLISH
Noam Chomsky and Morris Halle


IGGY ROCA
AND
WYN JOHNSON
A WORKBOOK IN
PHONOLOGY
Blackwell Publishing

10 Sakha (Yakut)

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

Noun	*Plural*	*Associative*	
aya	ayalar	ayalɨɨn	'father'
paarta	paartalar	paartalɨɨn	'school desk'
tɨa	tɨalar	tɨalɨɨn	'forest'
kinige	kinigeler	kinigeliin	'book'
ǰie	ǰieler	ǰieliin	'house'
iye	iyeler	iyeliin	'mother'
kini	kiniler	kiniliin	'3rd person'
bie	bieler	bieliin	'mare'
oyo	oyolor	oyoluun	'child'
χopto	χoptolor	χoptoluun	'gull'
börö	börölör	börölüün	'wolf'
tɨal	tɨallar	tɨalɨɨn	'wind'
ɨal	ɨallar	ɨalɨɨn	'neighbor'
kuul	kuullar	kuulluun	'sack'
at	attar	attɨɨn	'horse'
balɨk	balɨktar	balɨktɨɨn	'fish'
ɨskaap	ɨskaaptar	ɨskaaptɨɨn	'cabinet'
oyus	oyustar	oyustuun	'bull'
kus	kustar	kustuun	'duck'
tünnük	tünnükter	tünnüktüün	'window'
sep	septer	septiin	'tool'
et	etter	ettiin	'meat'
örüs	örüster	örüstüün	'river'
tiis	tiister	tiistiin	'tooth'
soroχ	soroχtor	soroχtuun	'some person'
oχ	oχtor	oχtuun	'arrow'
oloppos	oloppostor	oloppostuun	'chair'
ötöχ	ötöχtör	ötöχtüün	'abandoned farm'
ubay	ubaydar	ubaydɨɨn	'elder brother'
saray	saraydar	saraydɨɨn	'barn'
tɨy	tɨydar	tɨydɨɨn	'foal'
atɨɨr	atɨɨrdar	atɨɨrdɨɨn	'stallion'
oyuur	oyuurdar	oyuurduun	'forest'
üčügey	üčügeyder	üčügeydiin	'good person'
eǰiy	eǰiyder	eǰiydiin	'elder sister'
tomtor	tomtordor	tomtorduun	'knob'
moyotoy	moyotoydor	moyotoyduun	'chipmunk'
kötör	kötördör	kötördüün	'bird'
bölköy	bölköydör	bölköydüün	'islet'
χatɨŋ	χatɨŋnar	χatɨŋnɨɨn	'birch'
aan	aannar	aannɨɨn	'door'
tɨŋ	tɨŋner	tɨŋnɨɨn	'squirrel'
sordoŋ	sordoŋnor	sordoŋnuun	'pike'
olom	olomnor	olomnuun	'ford'
oron	oronnor	oronnuun	'bed'
bödöŋ	bödöŋnör	bödöŋnüün	'strong one'

Noun	*Partitive*	*Comparative*	*Ablative*	
aya	ayata	ayataayar	ayattan	'father'
paarta	paartata	paartataayar	paartattan	'school desk'
tɨa	tɨata	tɨataayar	tɨattan	'forest'
kinige	kinigete	kinigeteeyer	kinigetten	'book'
ǰie	ǰiete	ǰieteeyer	ǰietten	'house'
iye	iyete	iyeteeyer	iyetten	'mother'
kini	kinite	kiniteeyer	kinitten	'3rd person'
bie	biete	bieteeyer	bietten	'mare'
oyo	oyoto	oyotooyor	oyotton	'child'
χopto	χoptoto	χoptotooyor	χoptotton	'gull'
börö	börötö	börötööyör	böröttön	'wolf'
tɨal	tɨalla	tɨallaayar	tɨaltan	'wind'
ɨal	ɨalla	ɨallaayar	ɨaltan	'neighbor'
kuul	kuulla	kuullaayar	kuultan	'sack'
moχsoyol	moχsoyollo	moχsoyollooyor	moχsoyolton	'falcon'
at	atta	attaayar	attan	'horse'
balɨk	balɨkta	balɨktaayar	balɨktan	'fish'
ɨskaap	ɨskaapta	ɨskaaptaayar	ɨskaaptan	'cabinet'
oyus	oyusta	oyustaayar	oyustan	'bull'
kus	kusta	kustaayar	kustan	'duck'
tünnük	tünnükte	tünnükteeyer	tünnükten	'window'

Turkic Sakha (Yakut)

observed data

	Singular	Plural
Bed	oron	oronnor
Mare	bie	bieler
Cabinet	ɨskaap	ɨskaaptar

138 total examples

Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)

grammar (unobserved)

SINGULAR→stem
PLURAL→stem+lar

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	ɨskaap	ɨskaaptar

138 total examples

Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

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 → d / [-lateral -tense]_
"l" becomes "d" next to "r", "t", but not "l"

r_2: C → [-voice] / [-voice]_
do not voice next to voiceless

r_3: V → [+rounded] / [+rounded] [-low]$_0$_
r_4: [+continuant -high] → [-rounded] / u C$_0$_
"harmonize" round vowels like "u", "o"

r_5: V → [-back -low] / [-back +vowel] [ ]$_0$_
"harmonize" vowels to be not at back of mouth

r_6: [-sonorant +voice] → [+nasal] / [+nasal]_
"nasalize" consonant next to a nasal, like "m"

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	ɨskaap	ɨskaaptar

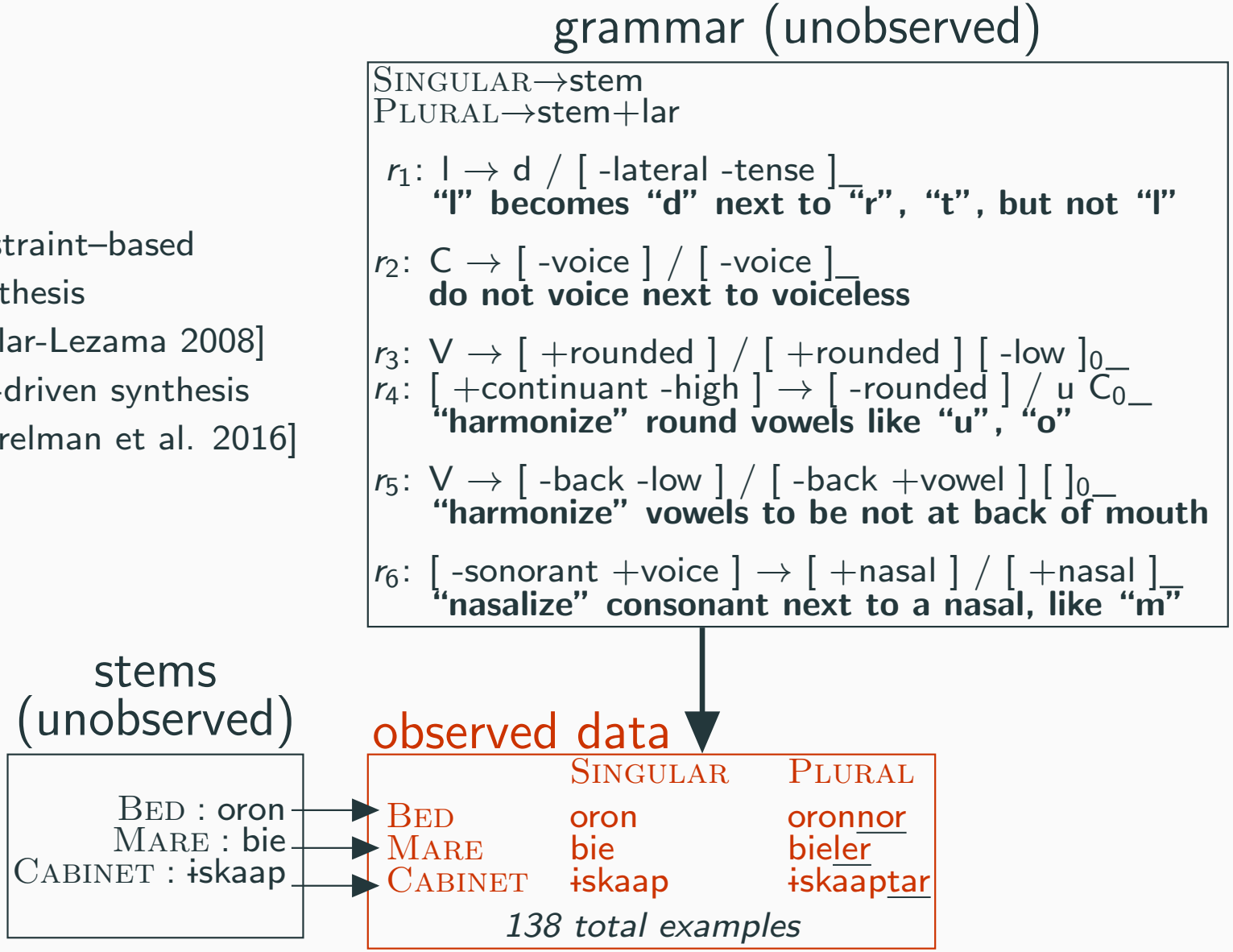
138 total examples

Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)

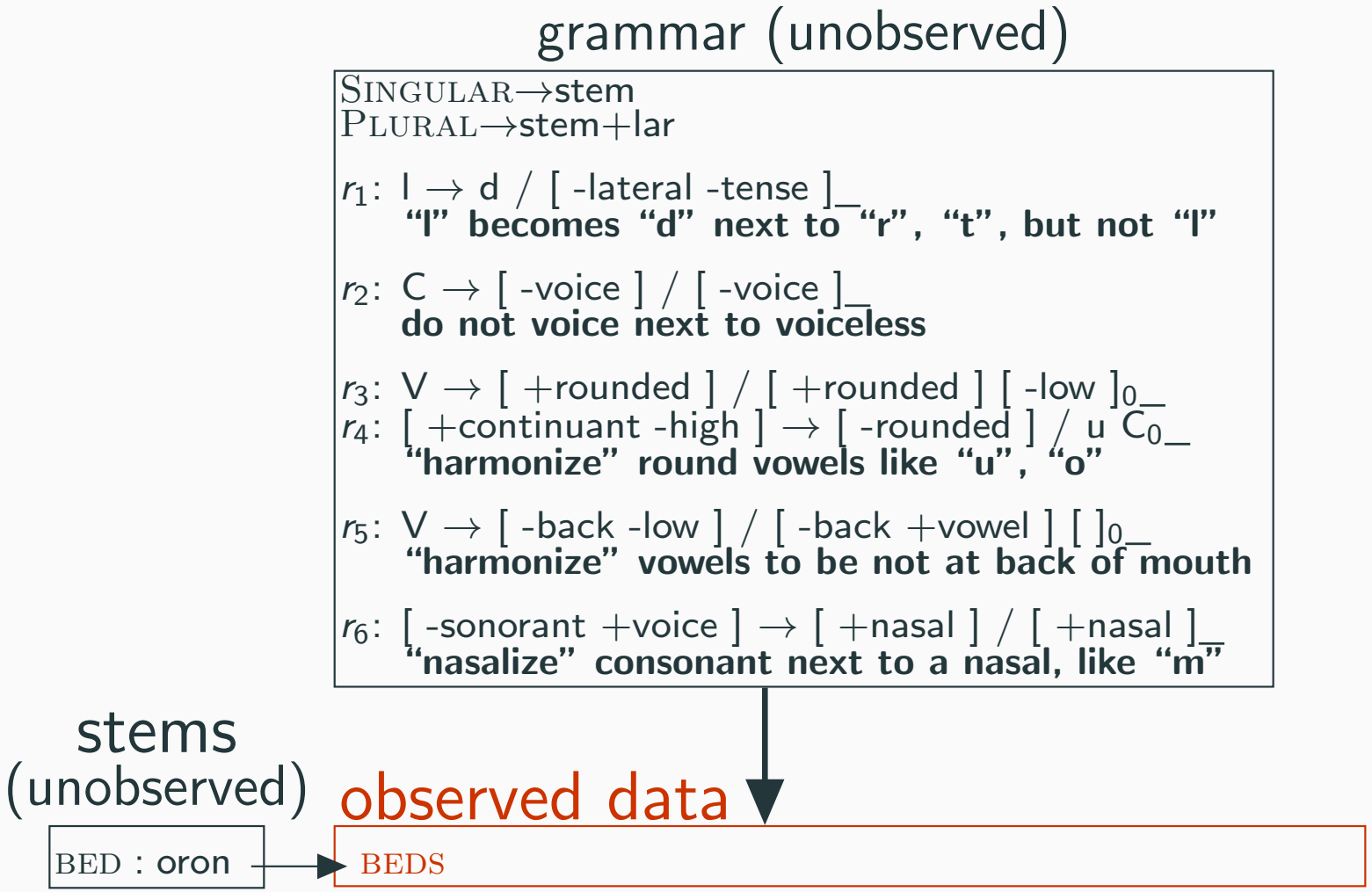
constraint–based synthesis [Solar-Lezama 2008]

test-driven synthesis [Perelman et al. 2016]



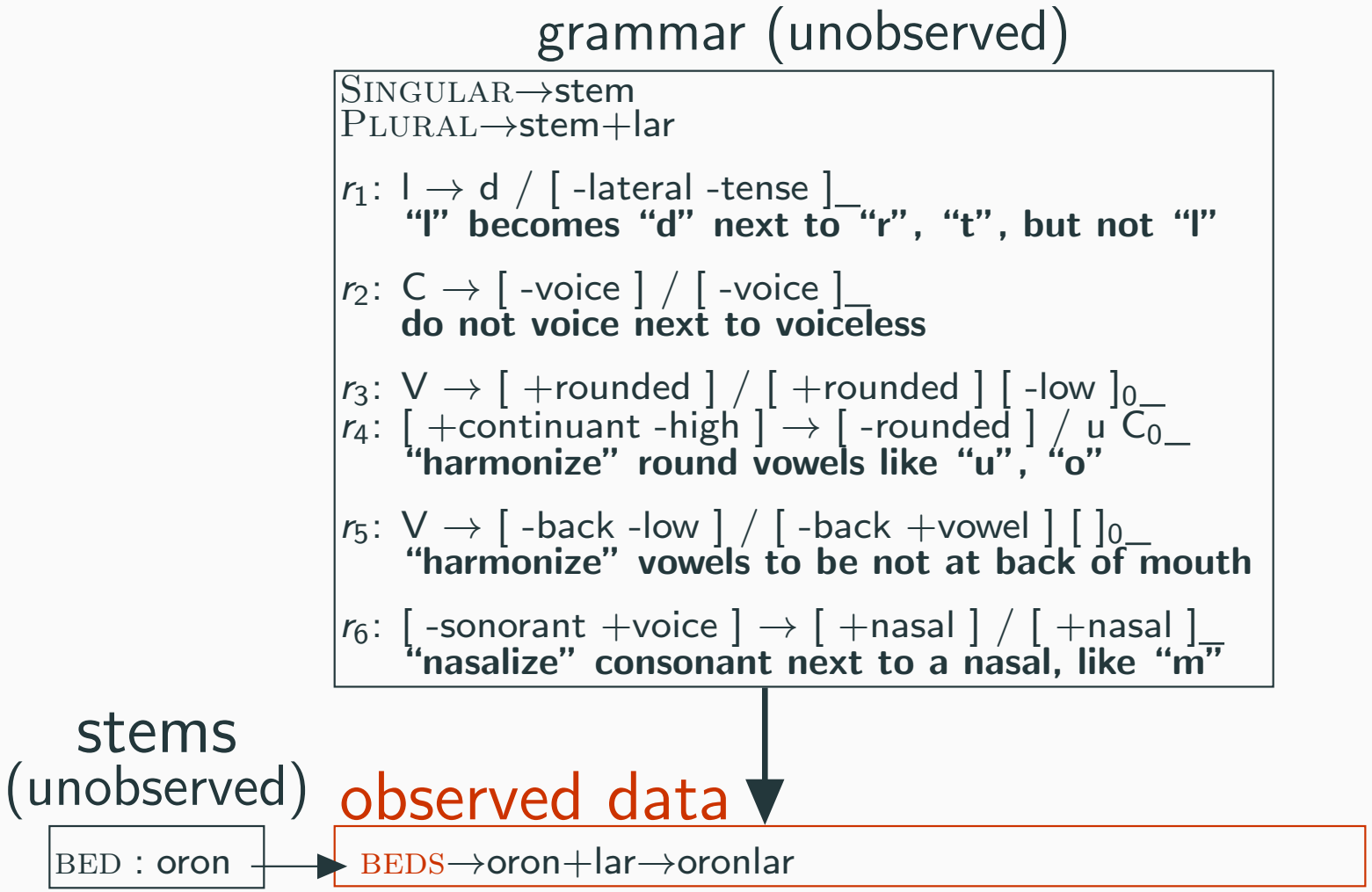
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)



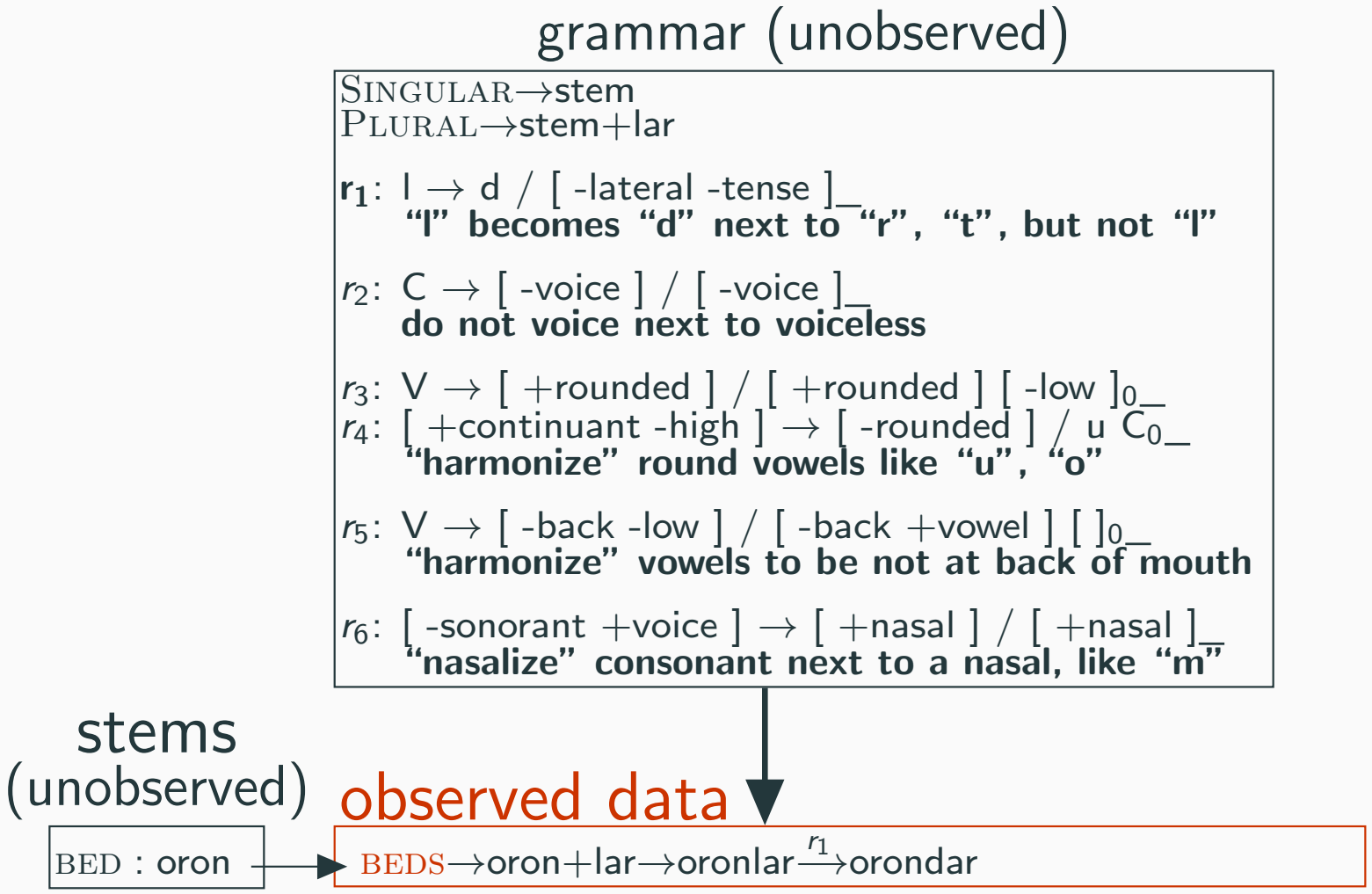
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)



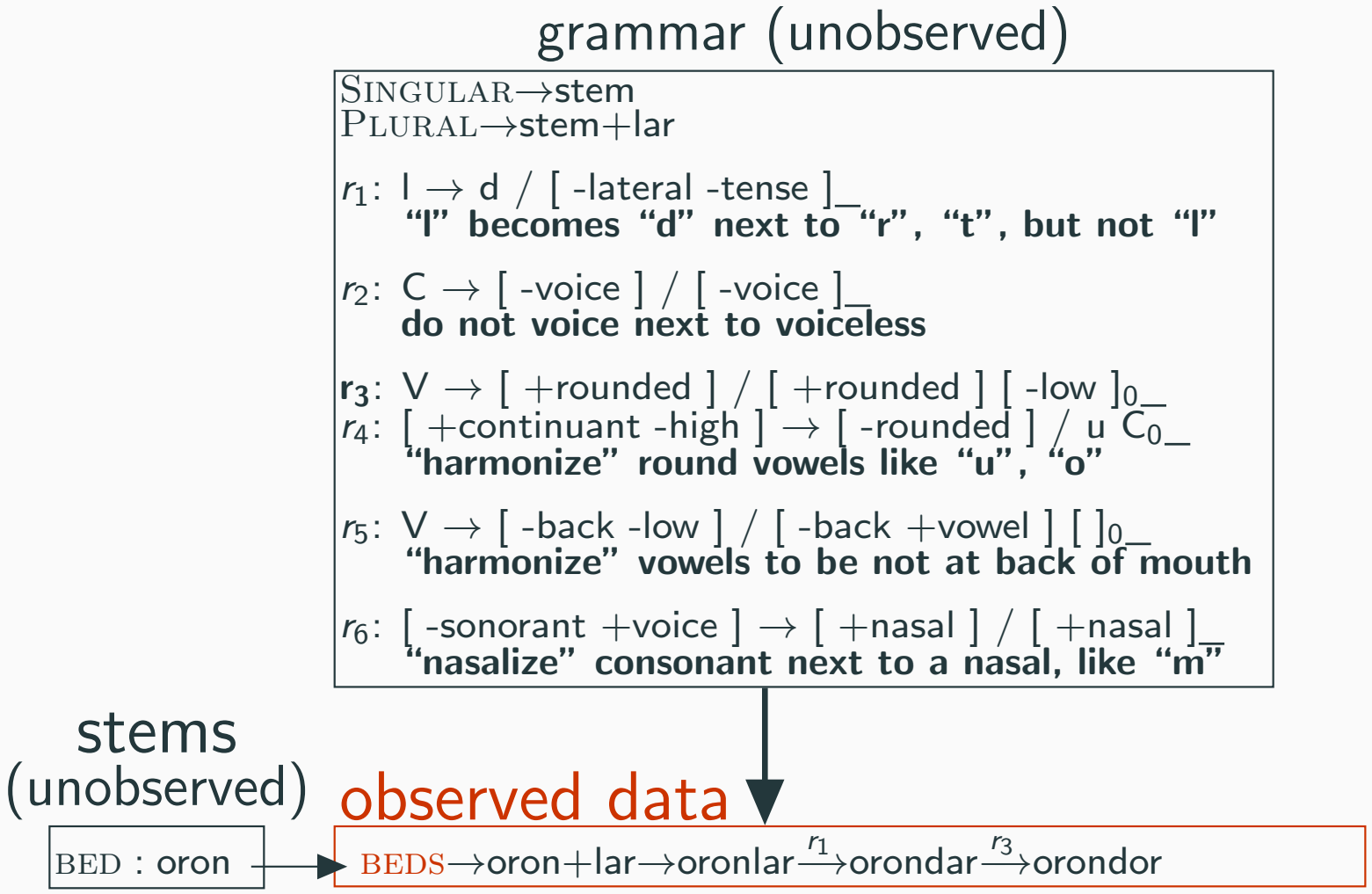
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)



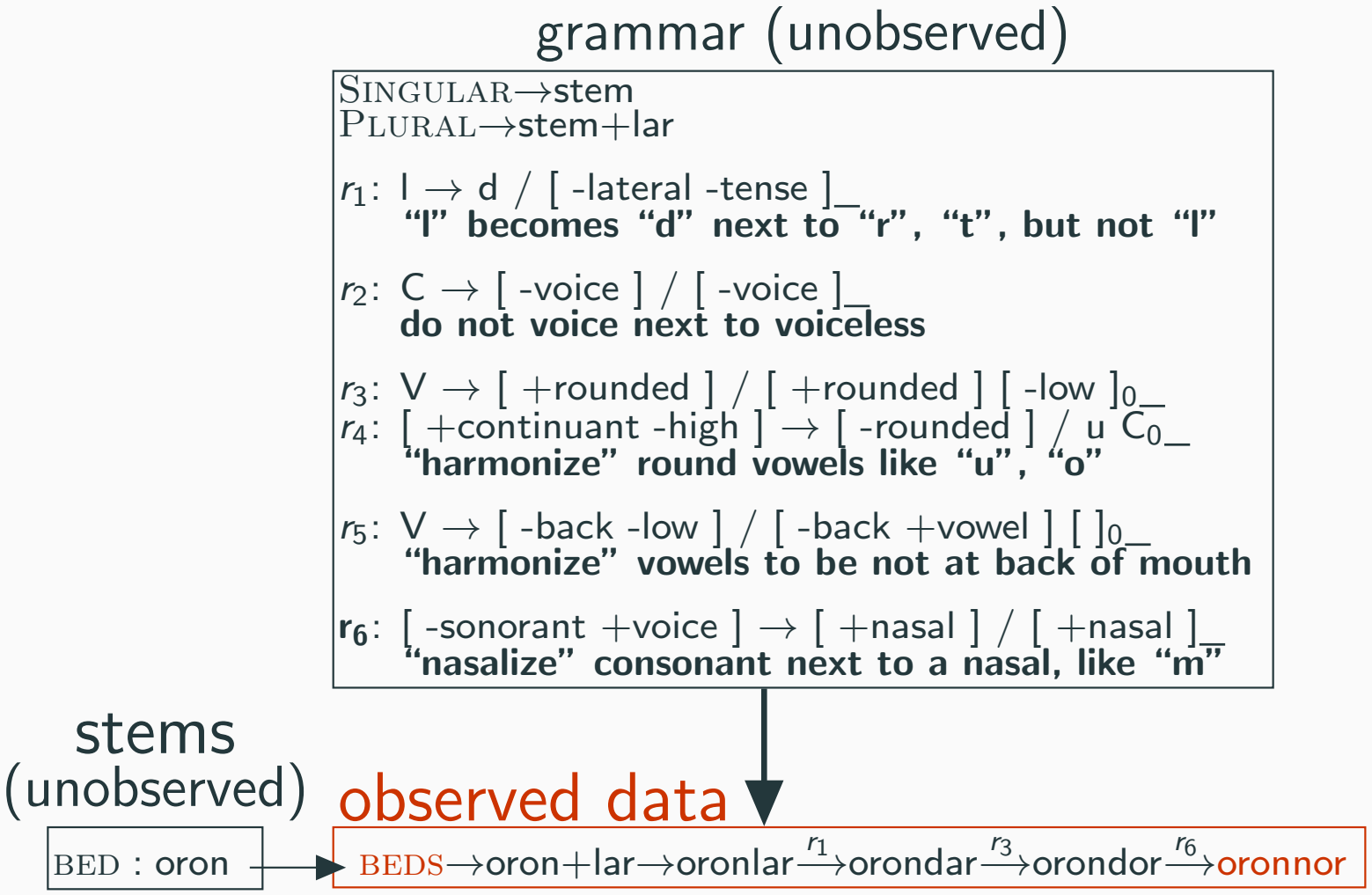
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)



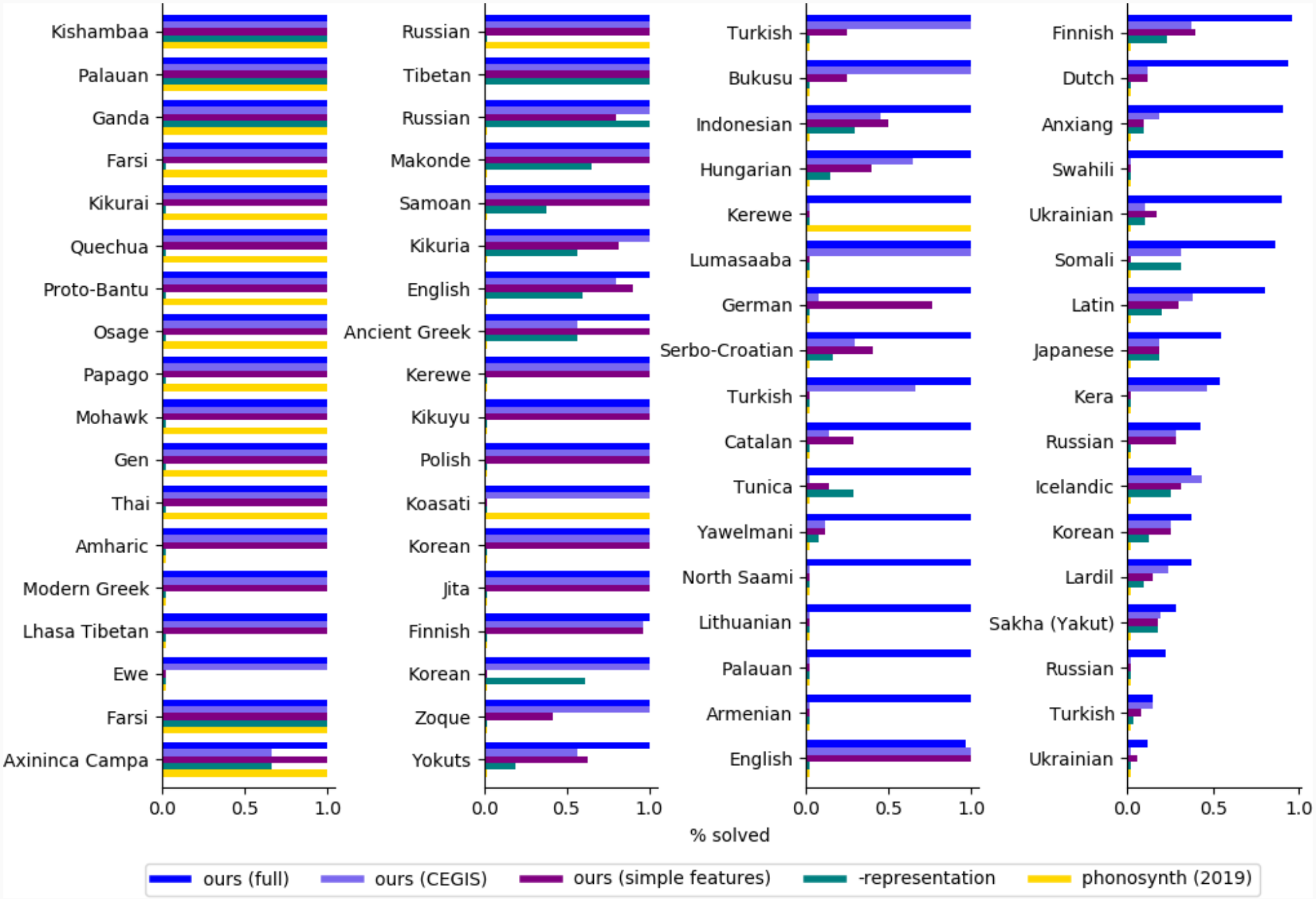
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Turkic Sakha (Yakut)

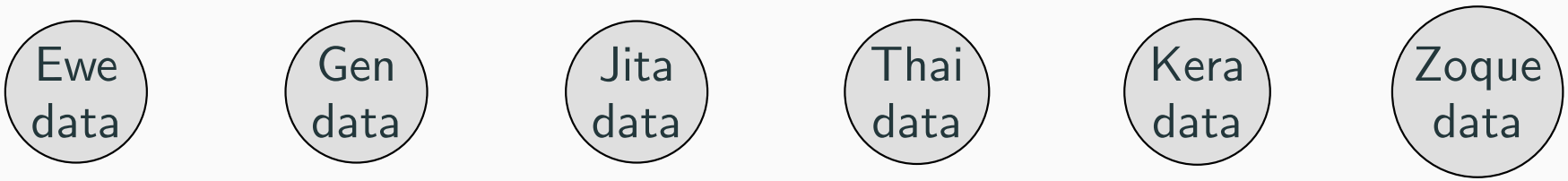


Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

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		


Kishambaa
Palauan
Ganda
Farsi
Kikurai
Quechua
Proto-Bantu
Osage
Papago
Mohawk
Gen
Thai
Amharic
Modern Greek
Lhasa Tibetan
Ewe
Farsi
Axininca Campa
Russian
Tibetan
Russian
Makonde
Samoan
Kikuria
English
Ancient Greek
Kerewe
Kikuyu
Polish
Koasati
Korean
Jita
Finnish
Korean
Zoque
Yokuts
Turkish
Bukusu
Indonesian
Hungarian
Kerewe
Lumasaaba
German
Serbo-Croatian
Turkish
Catalan
Tunica
Yawelmani
North Saami
Lithuanian
Palauan
Armenian
English
Finnish
Dutch
Anxiang
Swahili
Ukrainian
Somali
Latin
Japanese
Kera
Russian
Icelandic
Korean
Lardil
Sakha (Yakut)
Russian
Turkish
Ukrainian
0.0
0.5
1.0
% solved
ours (full)
ours (CEGIS)
ours (simple features)
-representation
phonosynth (2019)

Distilling higher-level knowledge

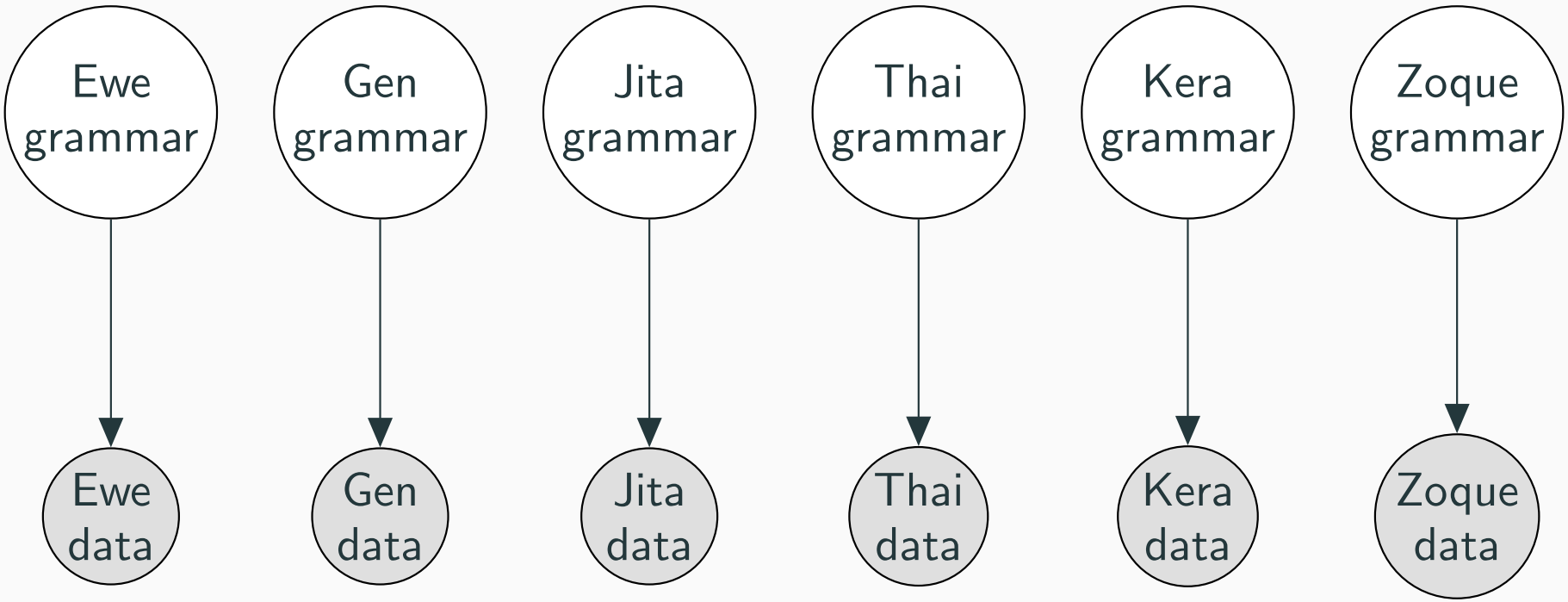


Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Distilling higher-level knowledge

dark: we know it

light: we wanna know it

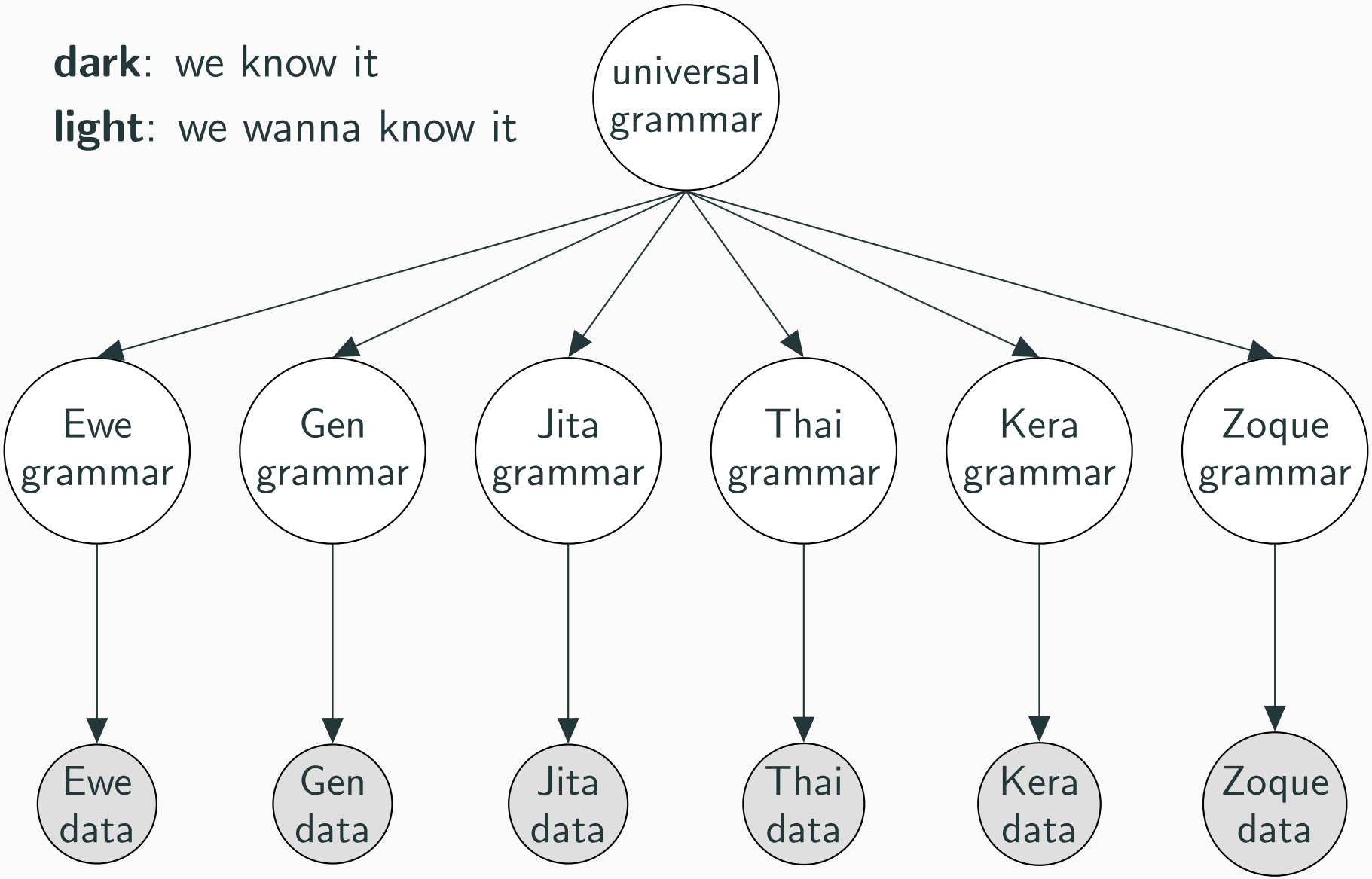


Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Distilling higher-level knowledge

dark: we know it

light: we wanna know it



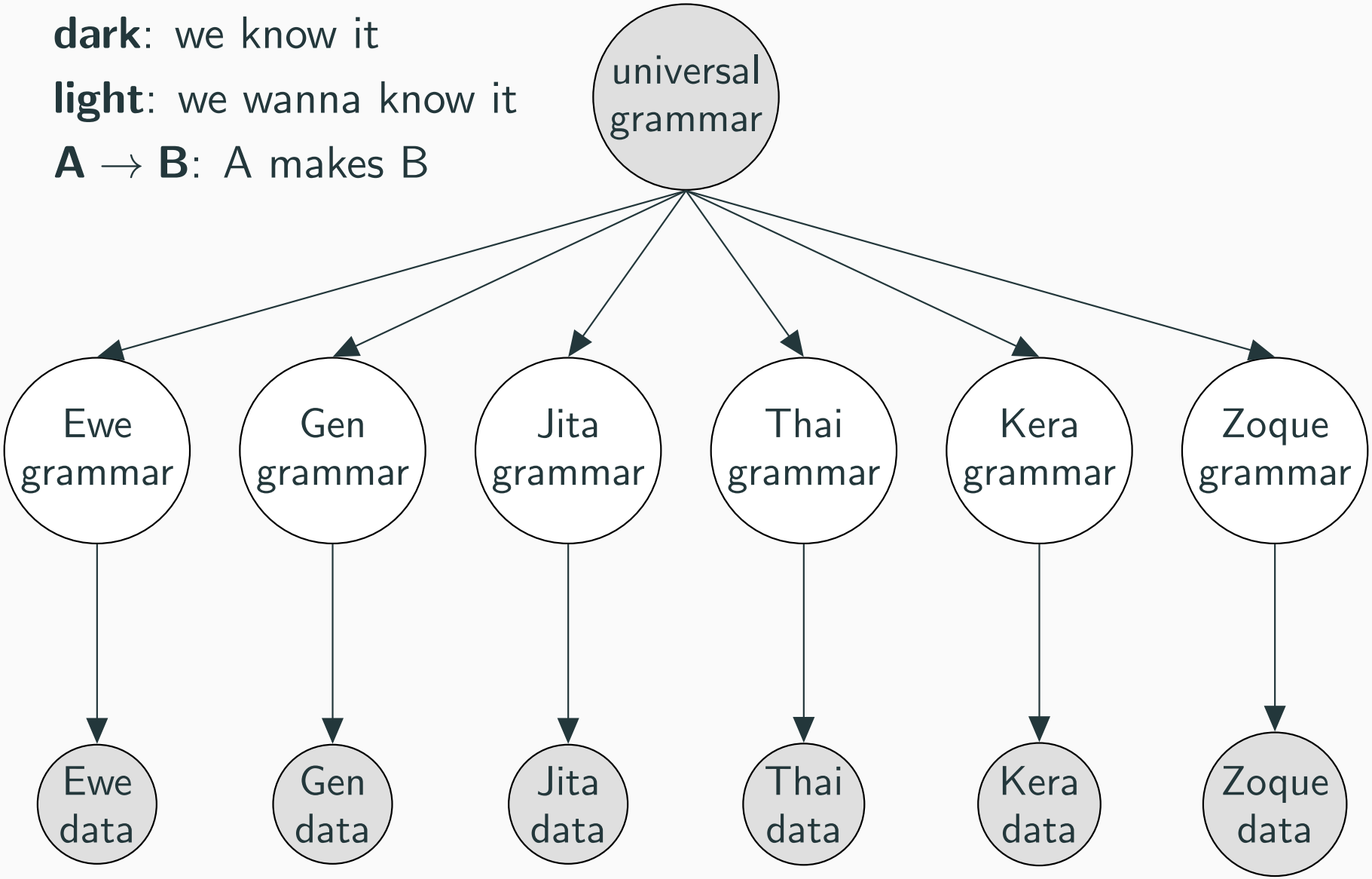
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Distilling higher-level knowledge

dark: we know it

light: we wanna know it

A $\rightarrow$ **B**: A makes B



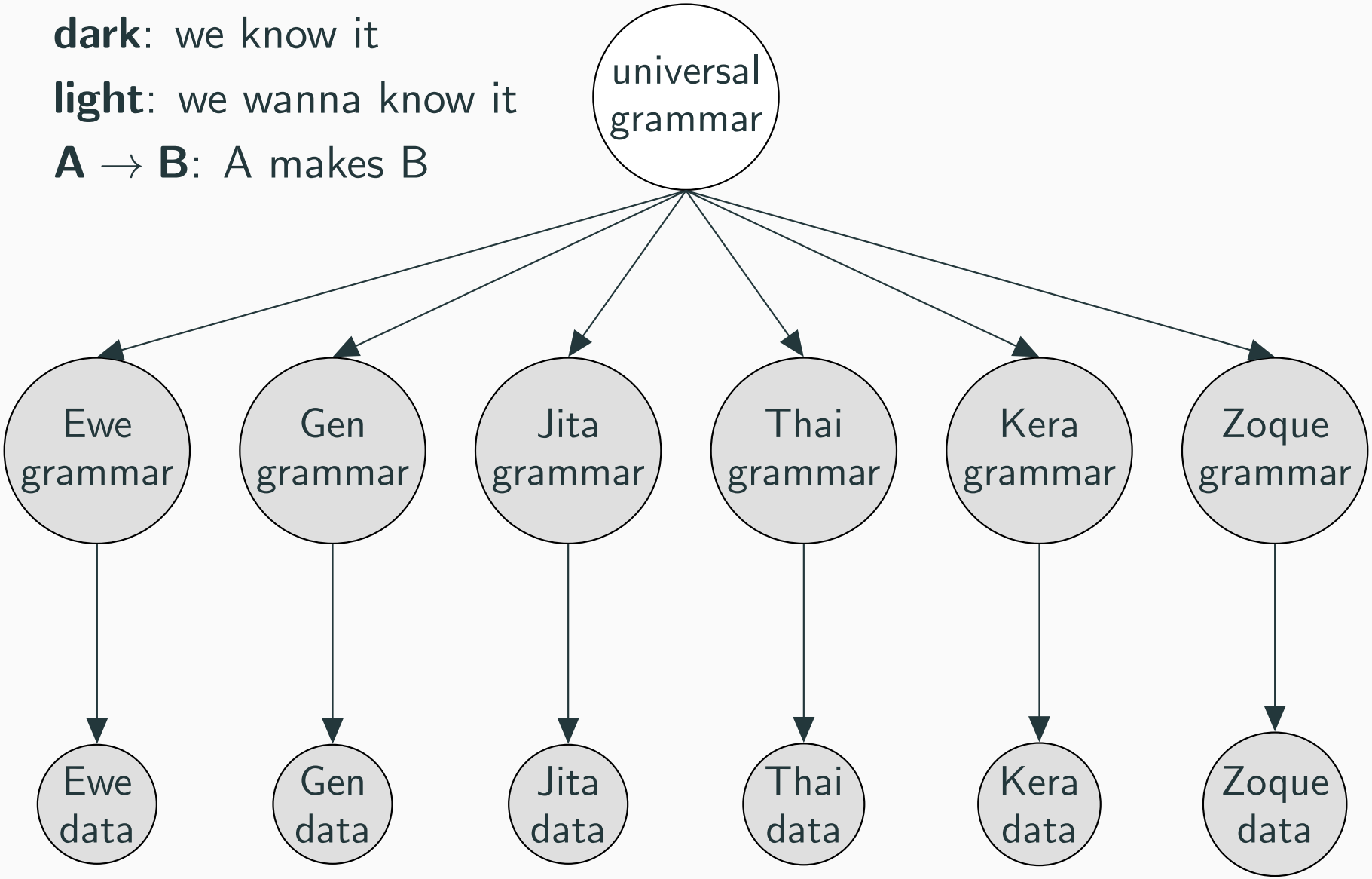
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Distilling higher-level knowledge

dark: we know it

light: we wanna know it

A $\rightarrow$ **B**: A makes B



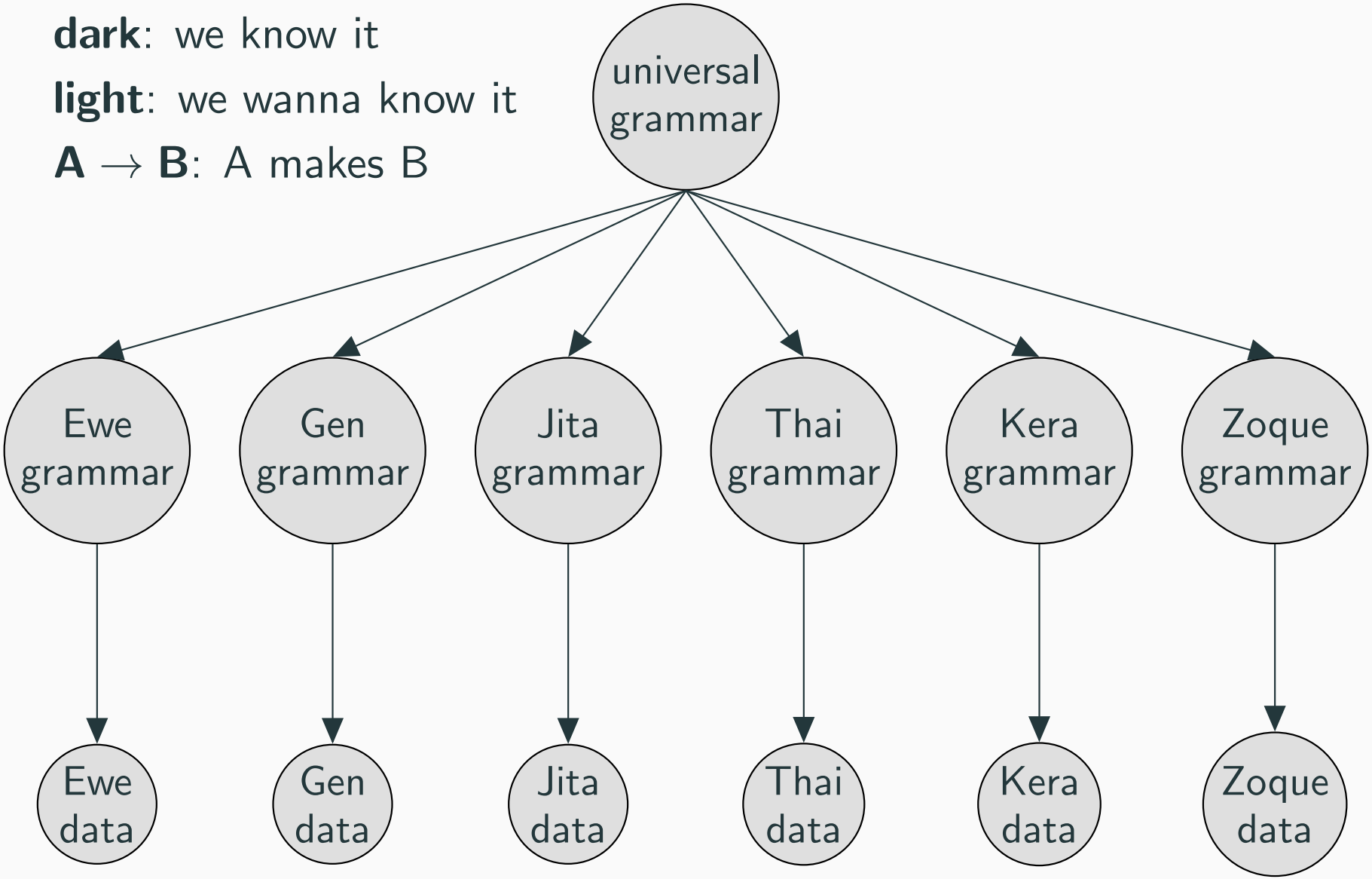
Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

Distilling higher-level knowledge

dark: we know it

light: we wanna know it

A $\rightarrow$ **B**: A makes B



Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2021.

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

Higher-level Knowledge: Case study in linguistics

Higher-level Knowledge: General program discovery

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

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]
...

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

Library learning

Initial Primitives

```
  :
 map
fold
 if
cons
  >
  :
```

Sample Problem: `sort list`

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

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

Library learning

Initial Primitives

```
map
fold
if
cons
>
```

Sample Problem: `sort list`

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

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

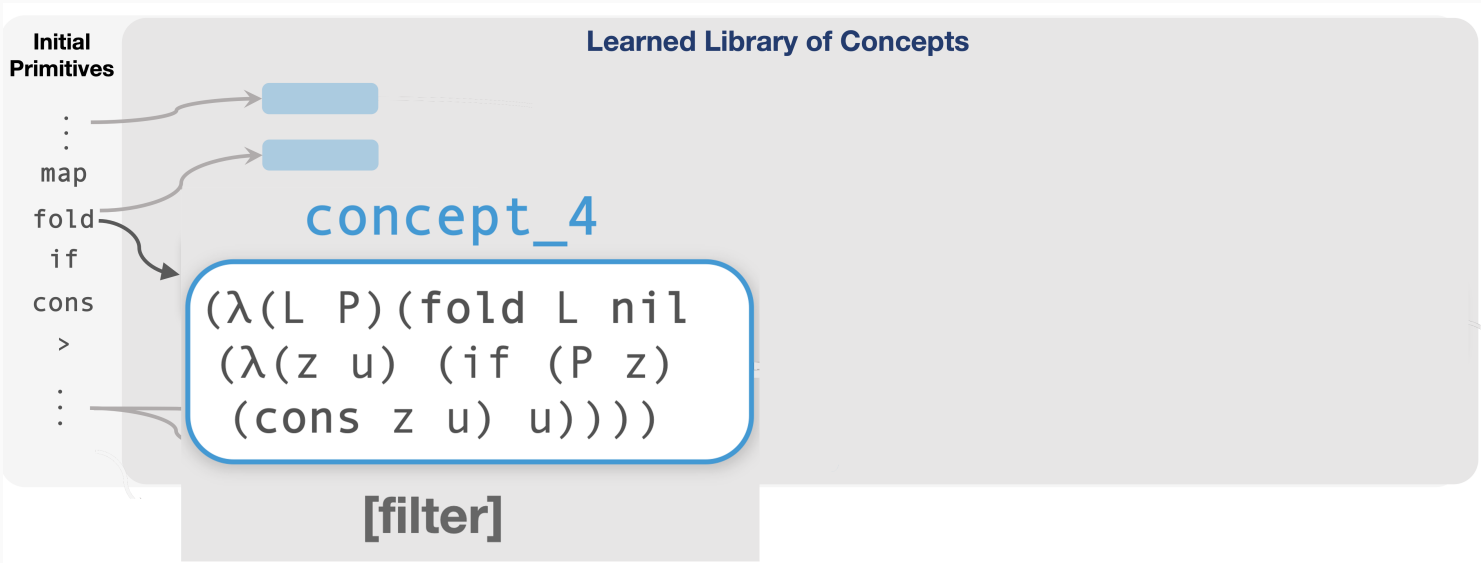
Library learning

Sample Problem: `sort list`

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

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

Library learning



Sample Problem: `sort list`

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

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

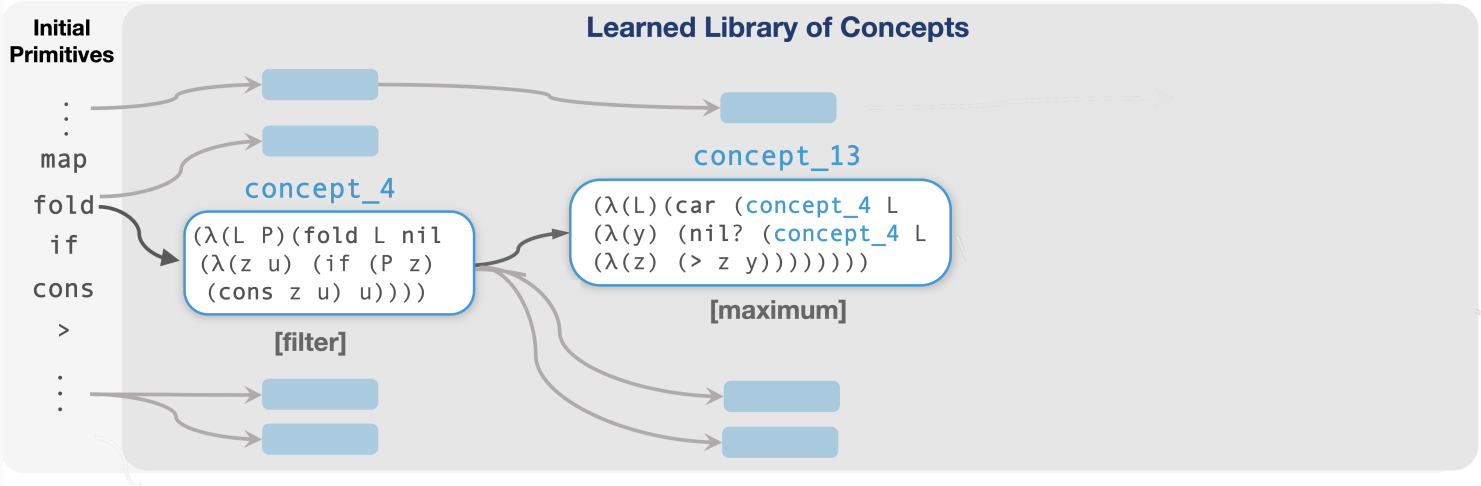
Library learning

Sample Problem: `sort list`

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

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

Library learning

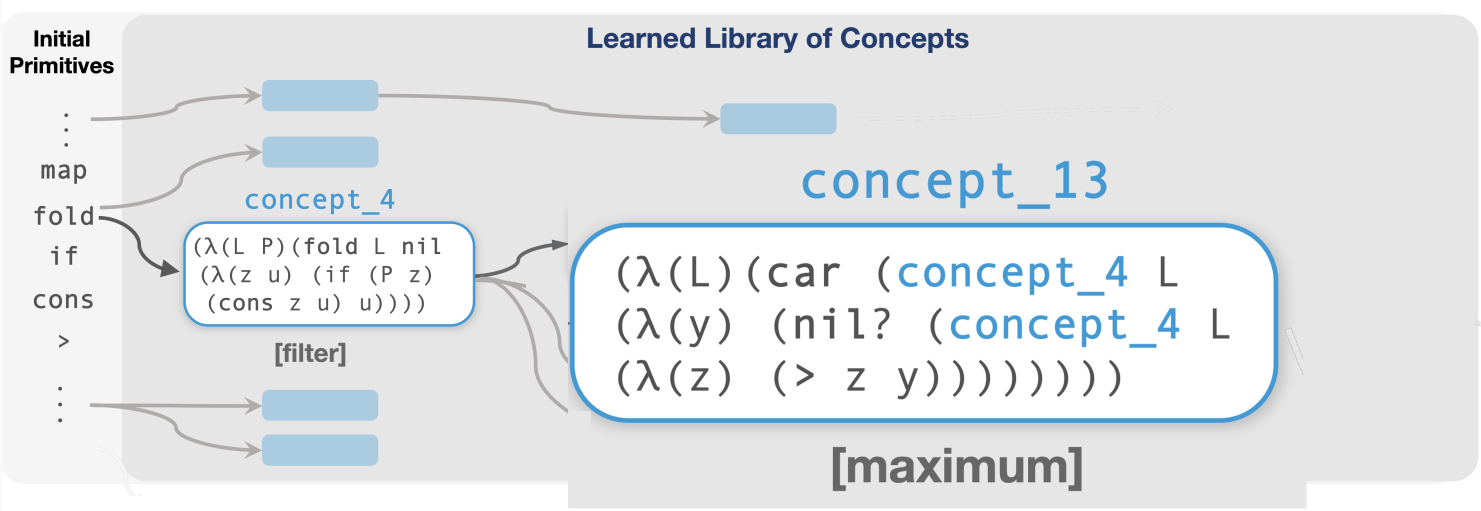


Sample Problem: `sort list`

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

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

Library learning

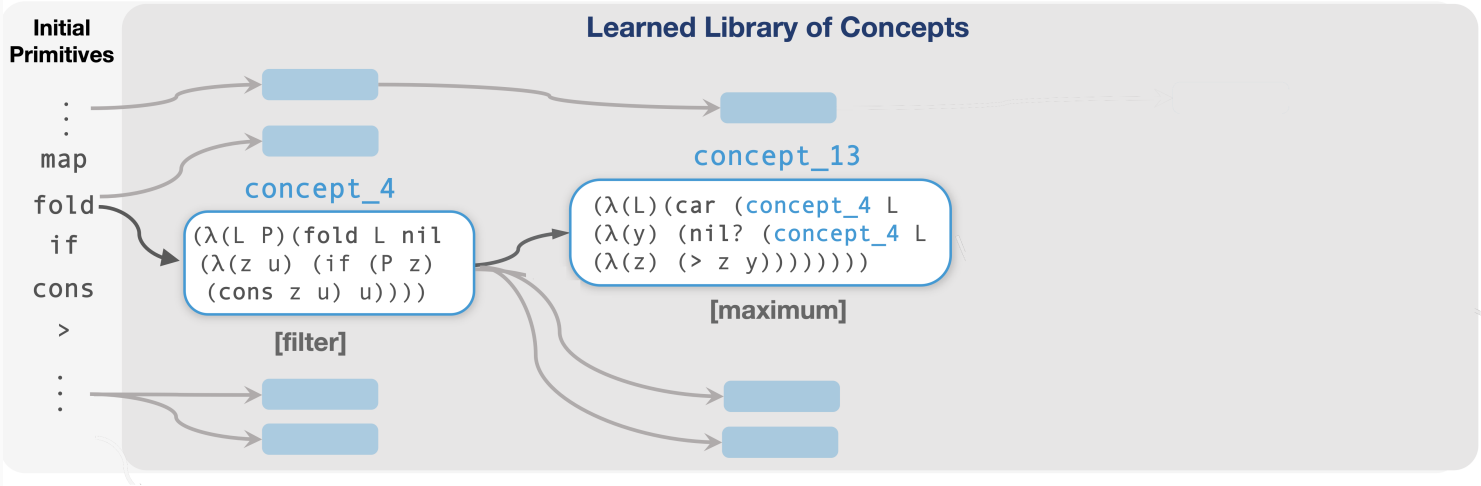


Sample Problem: `sort list`

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

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

Library learning

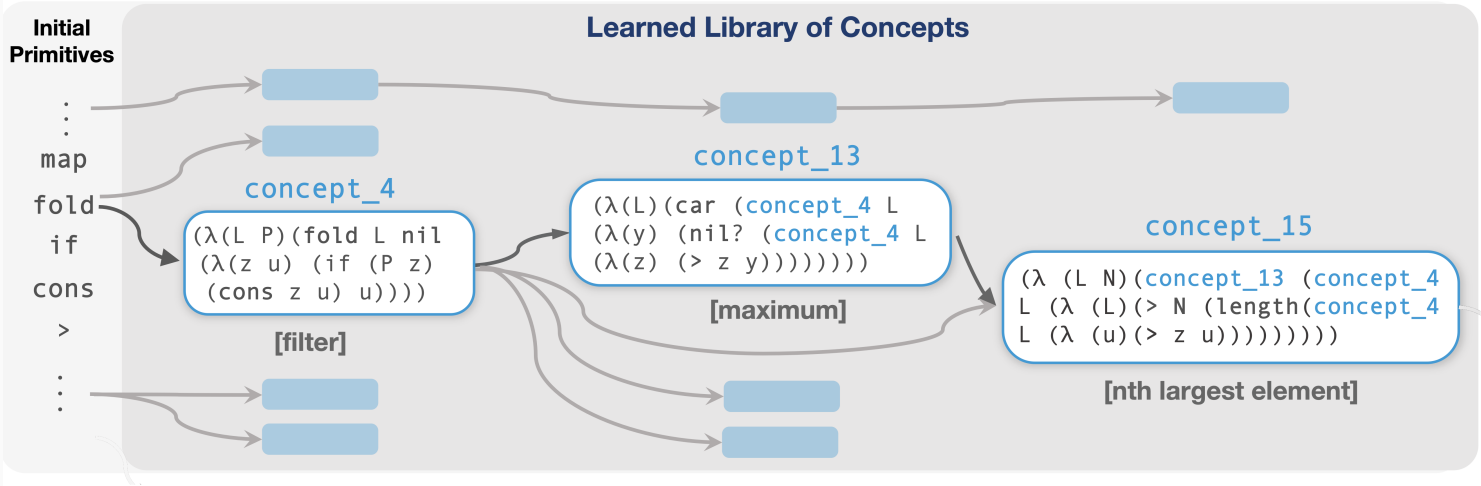


Sample Problem: `sort list`

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

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

Library learning

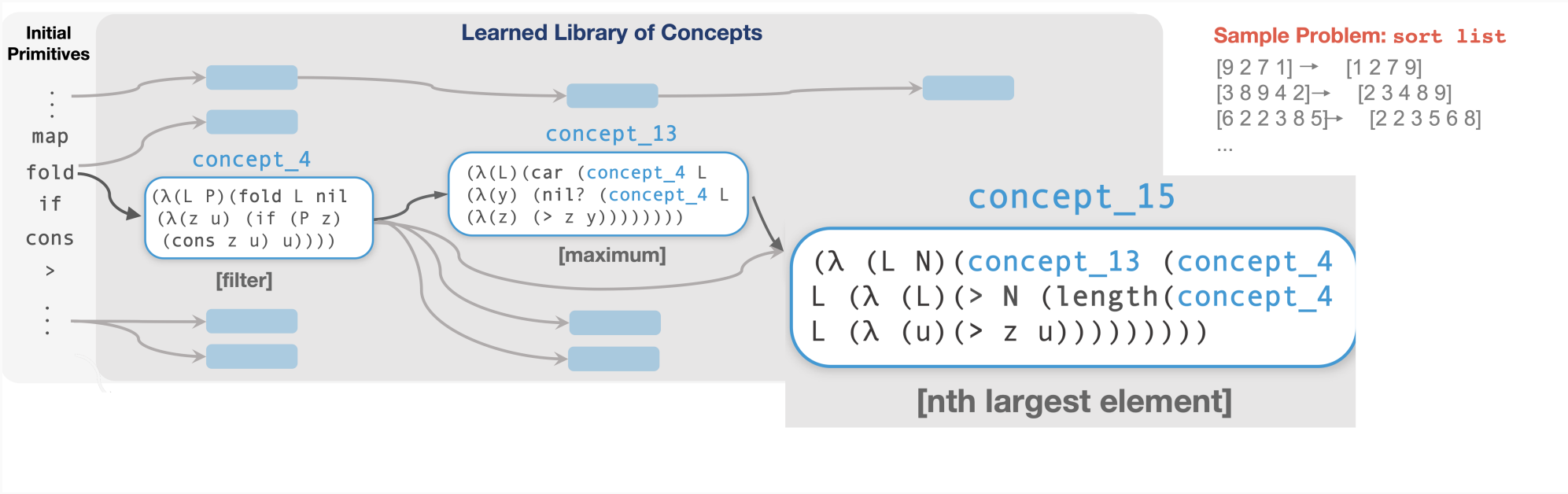


Sample Problem: `sort list`

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

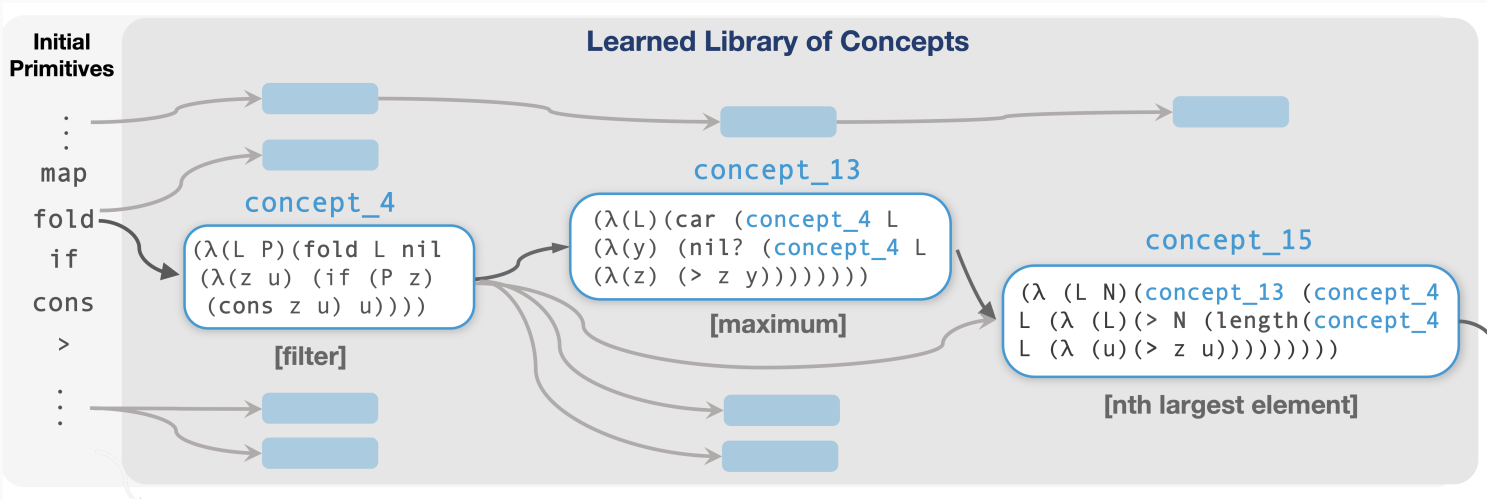
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

Library learning



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

Library learning



Sample Problem: `sort list`

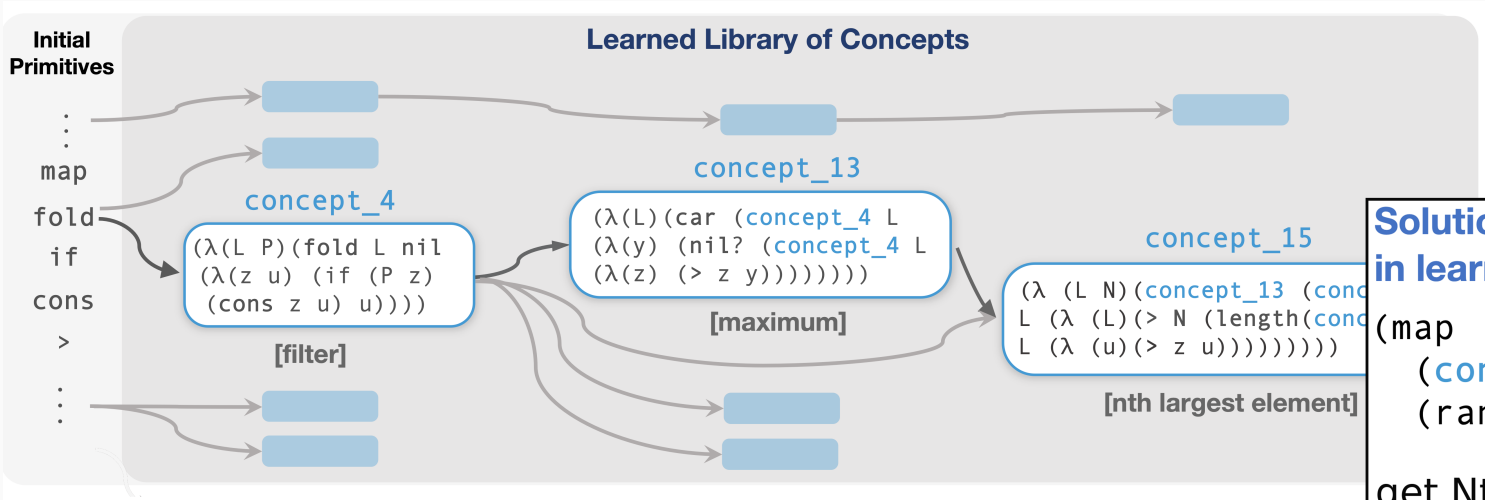
[9 2 7 1] → [1 2 7 9]
[3 8 9 4 2] → [2 3 4 8 9]
[6 2 2 3 8 5] → [2 2 3 5 6 8]
...

Solution to `sort list` discovered in learned language:

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

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

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]
...

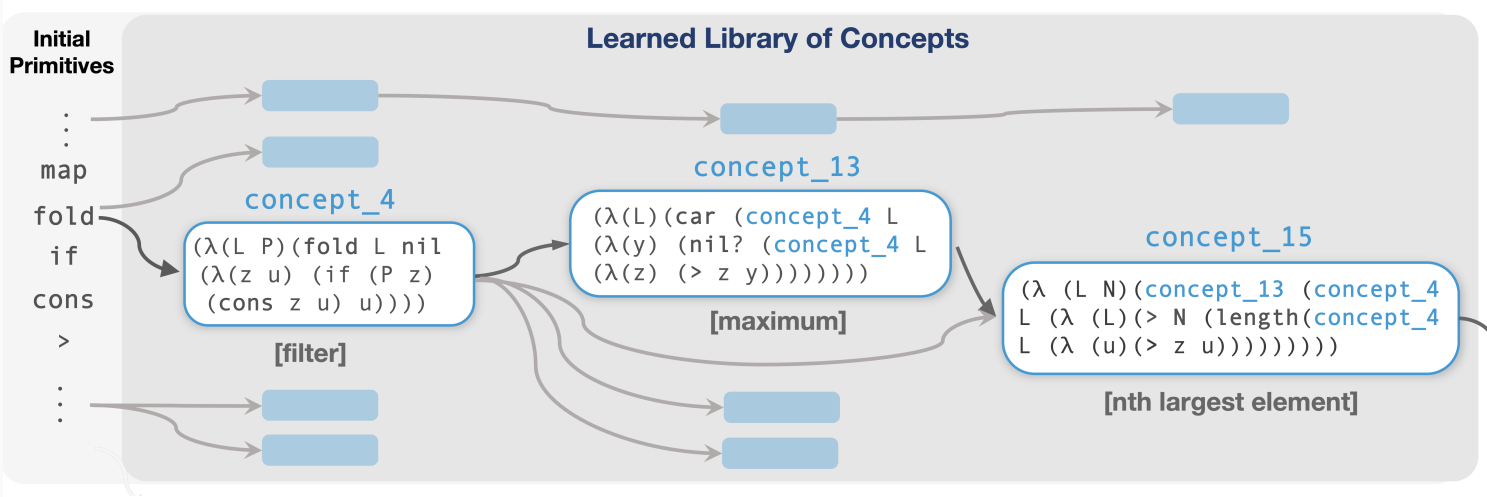
Solution to `sort list` discovered in learned language:

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

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

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

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)))
```

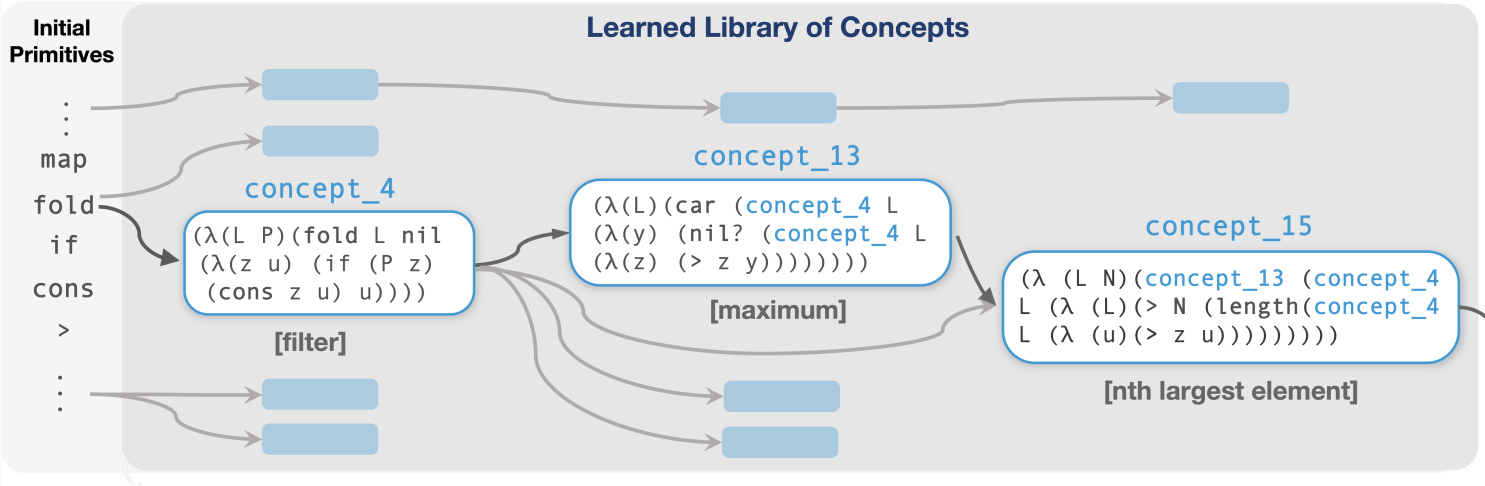
get Nth largest element, where N is 1, 2, 3, ...

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))))
```

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

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

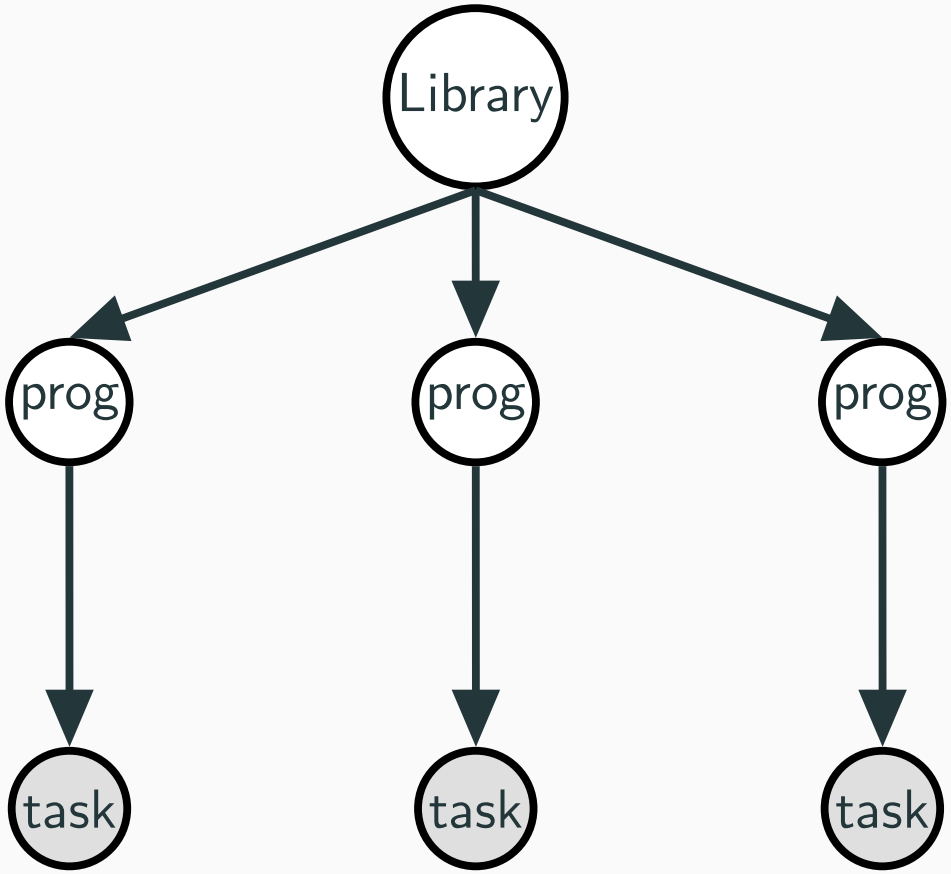
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

DreamCoder

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

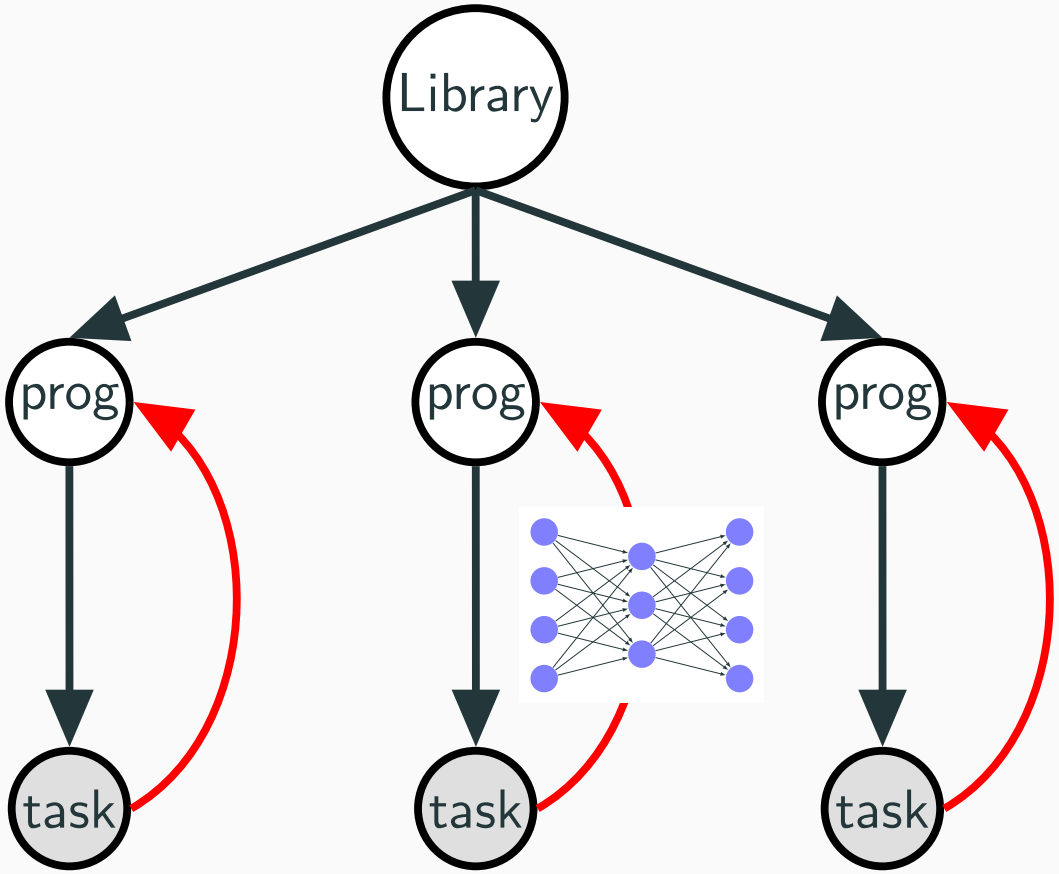
cf. Helmholtz machine, wake/sleep neural network training algorithms

Library learning as Bayesian inference



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

Library learning as neurally-guided Bayesian inference

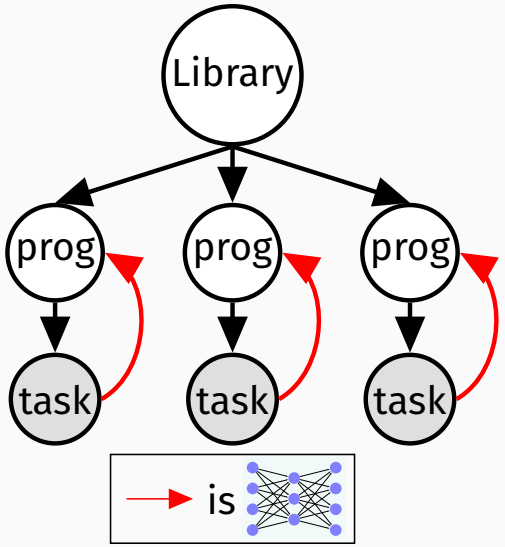


library learning via program analysis +

new neural inference network for program synthesis +

better program representation (Lisp+polymorphic types [Milner 1978])

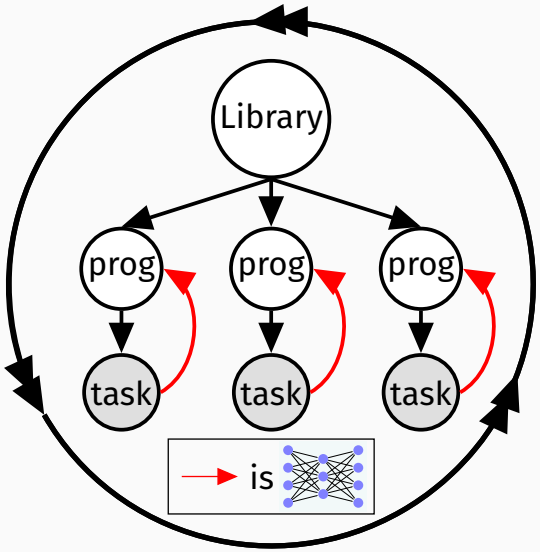
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

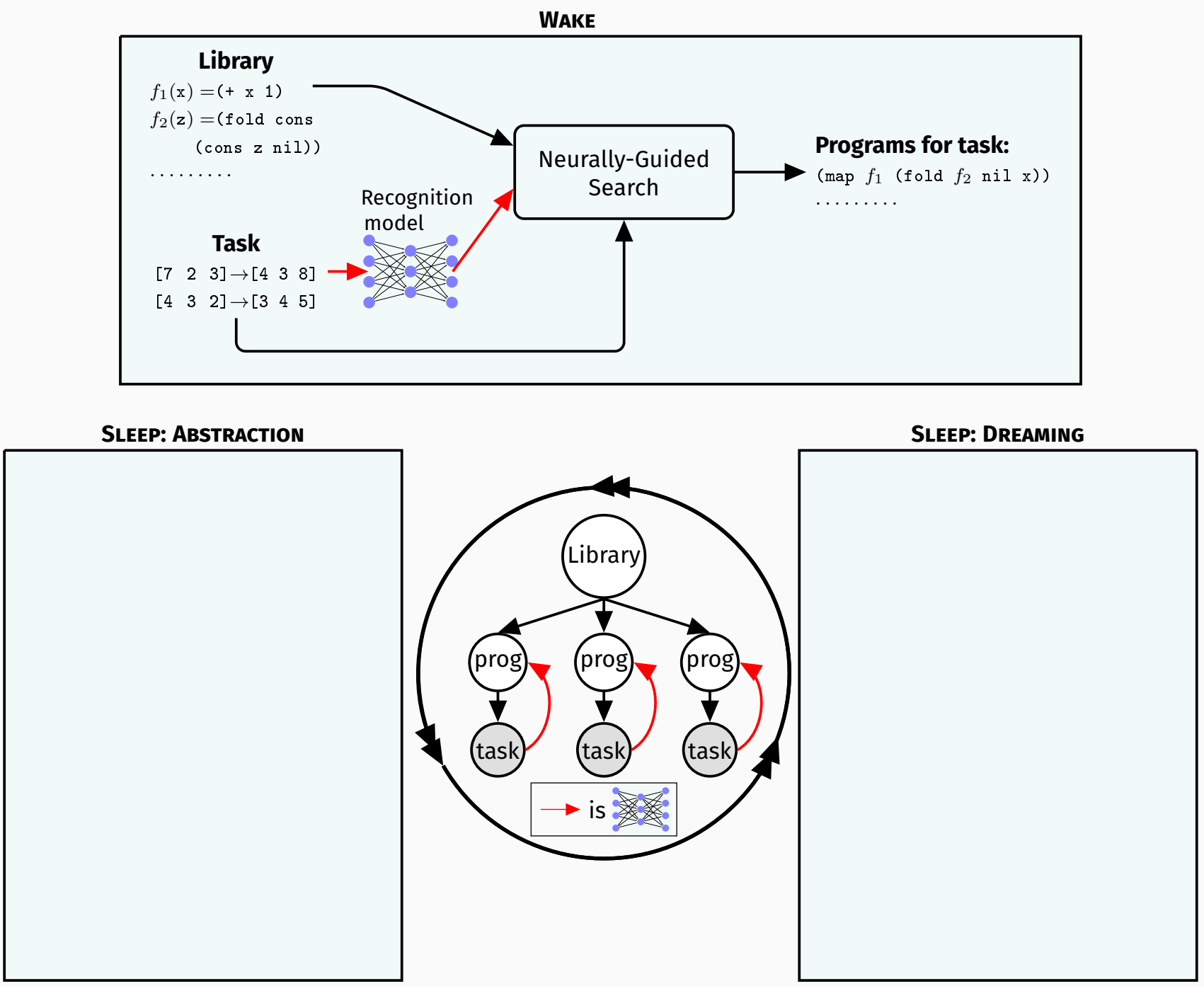

Library
prog
prog
prog
task
task
task
is

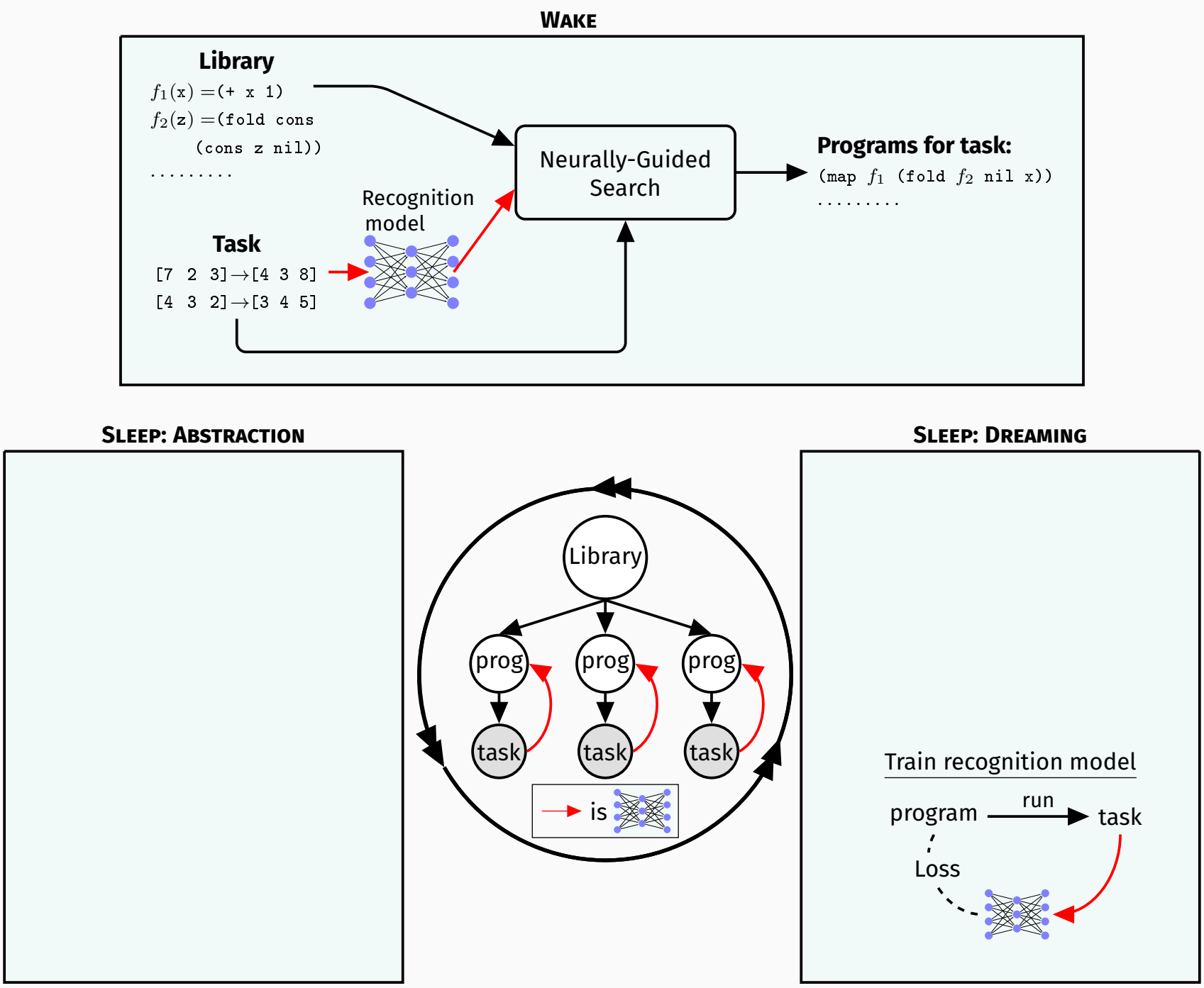
Wake

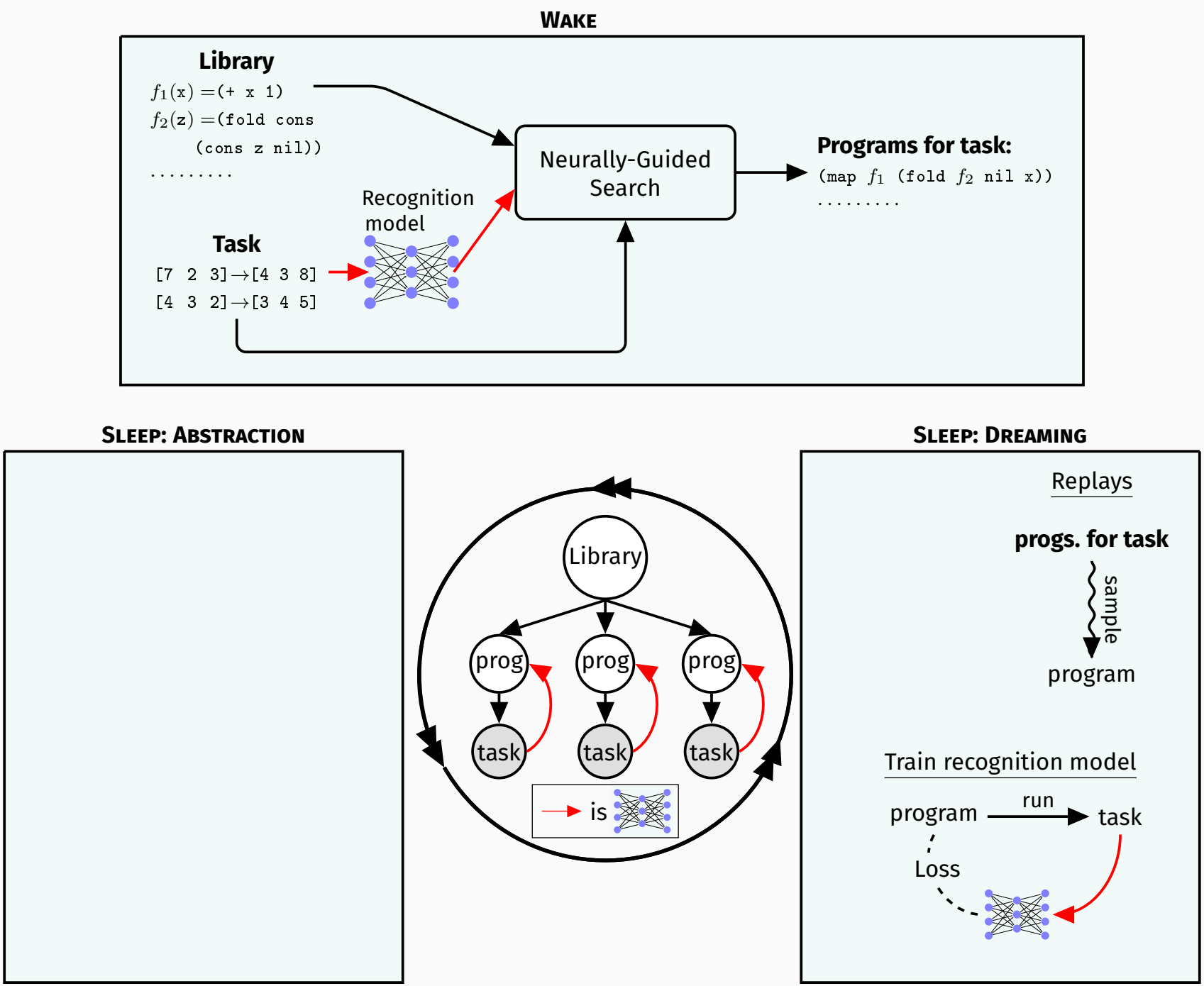
Sleep: Abstraction

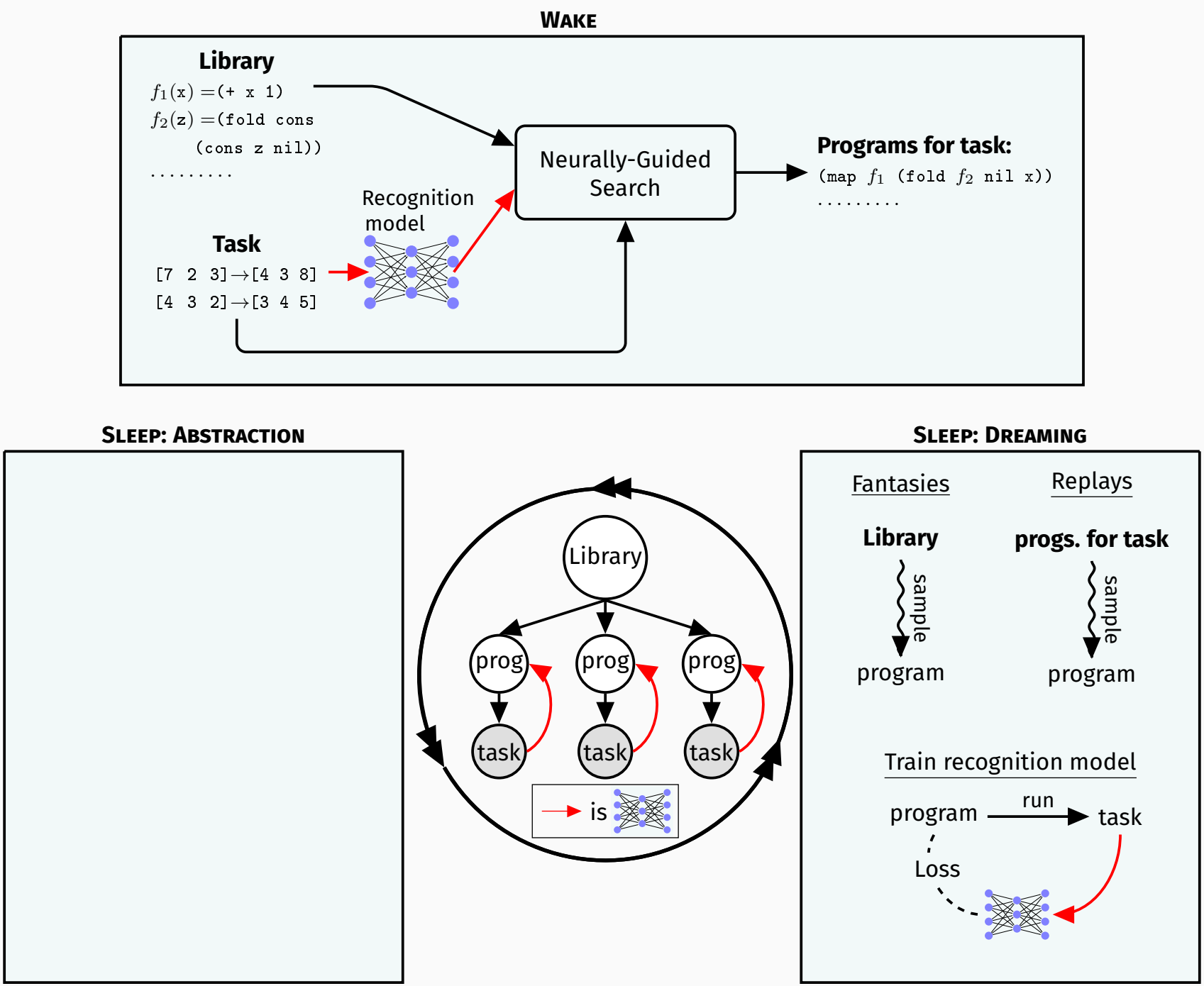
Sleep: Dreaming

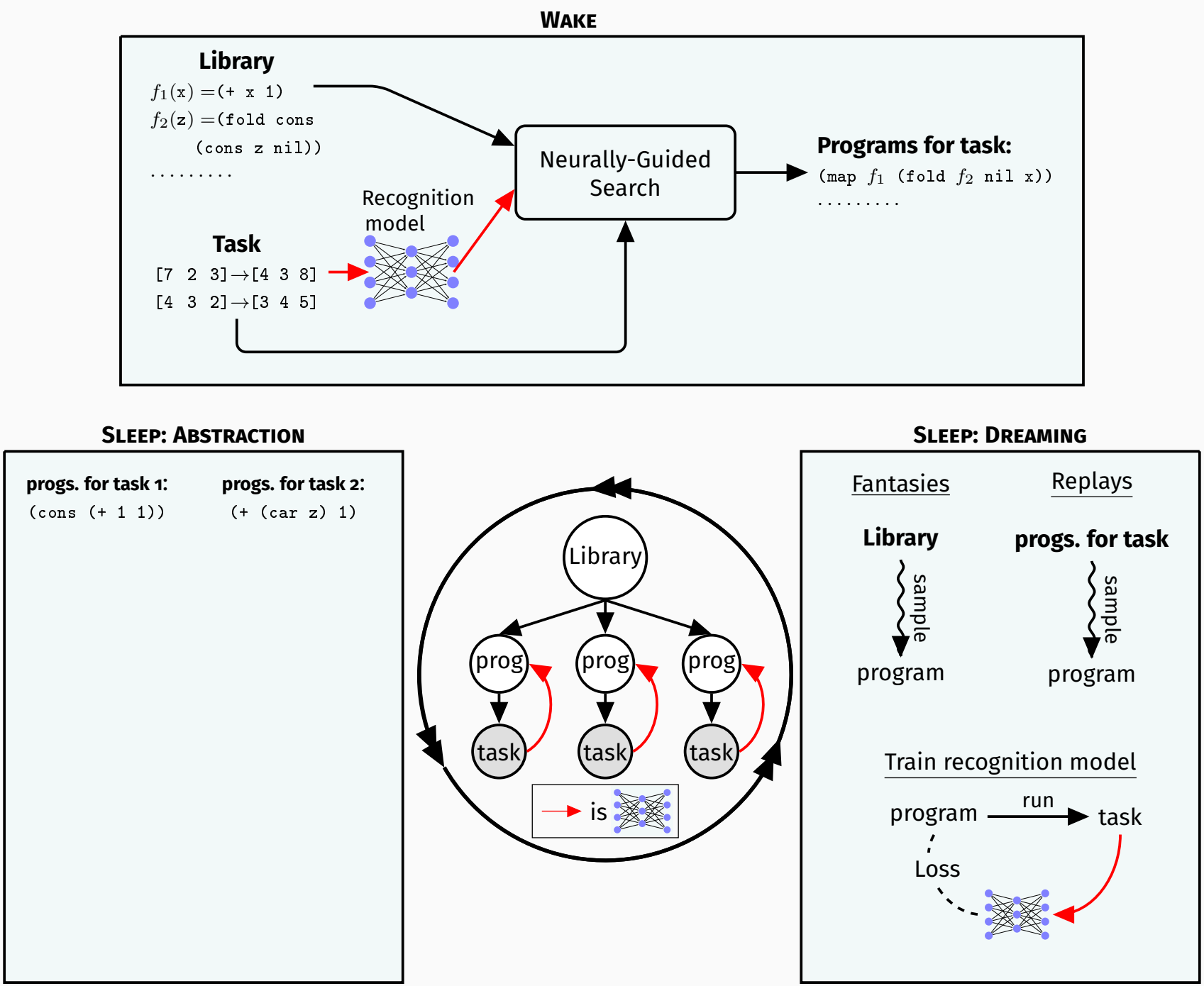


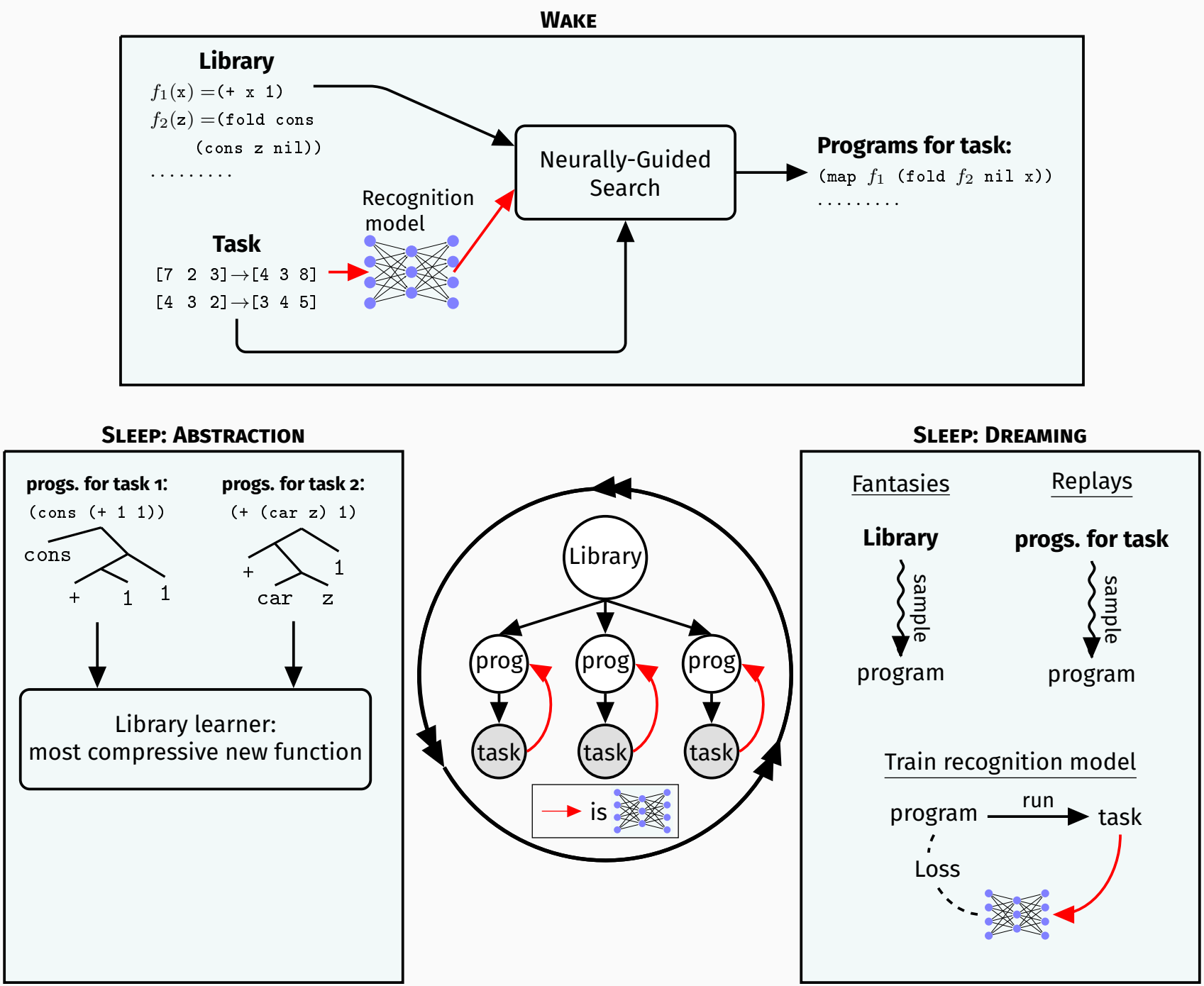

WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
SLEEP: DREAMING
Library
prog
prog
prog
task
task
task
is

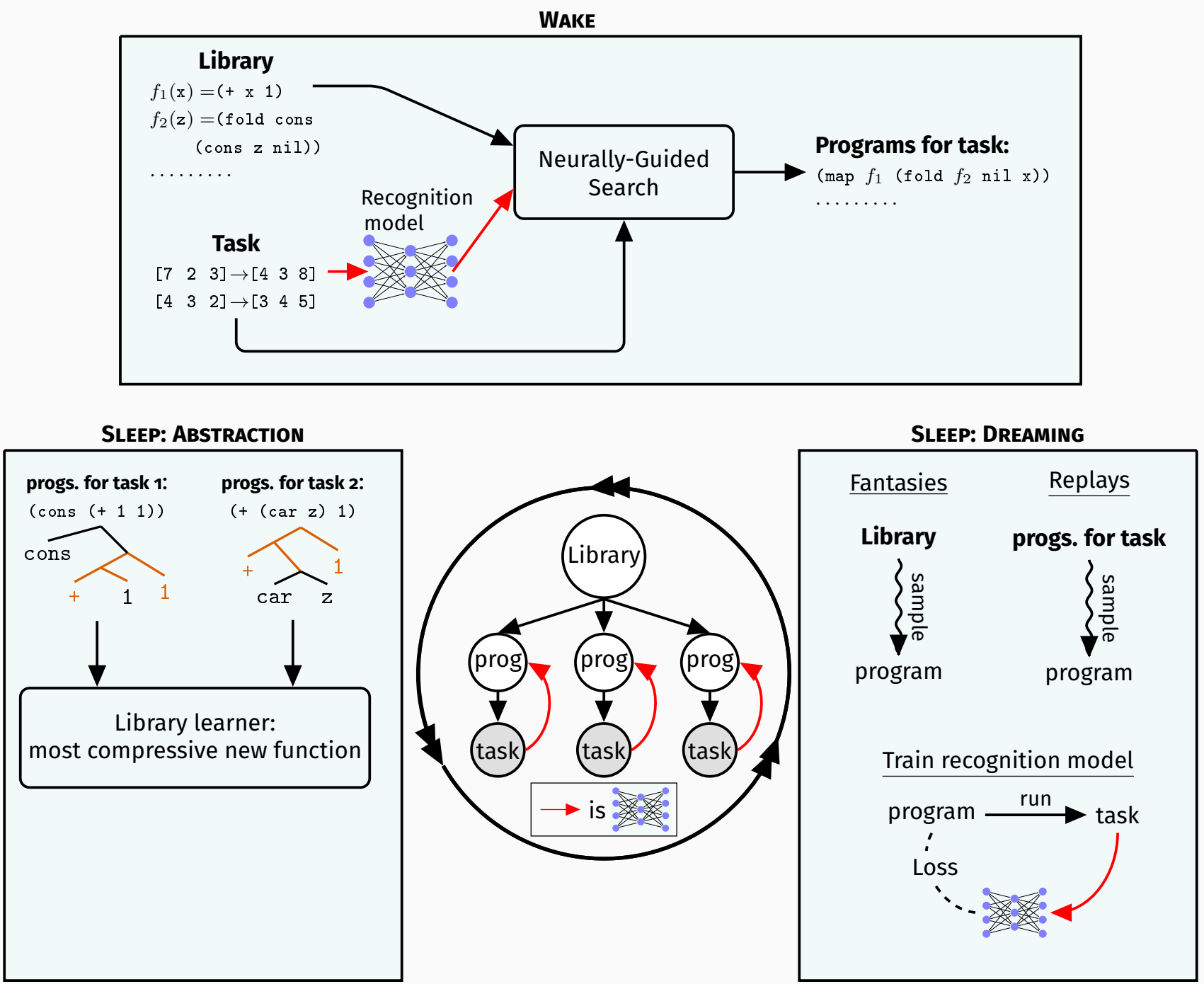

WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Train recognition model
program
run
task
Loss

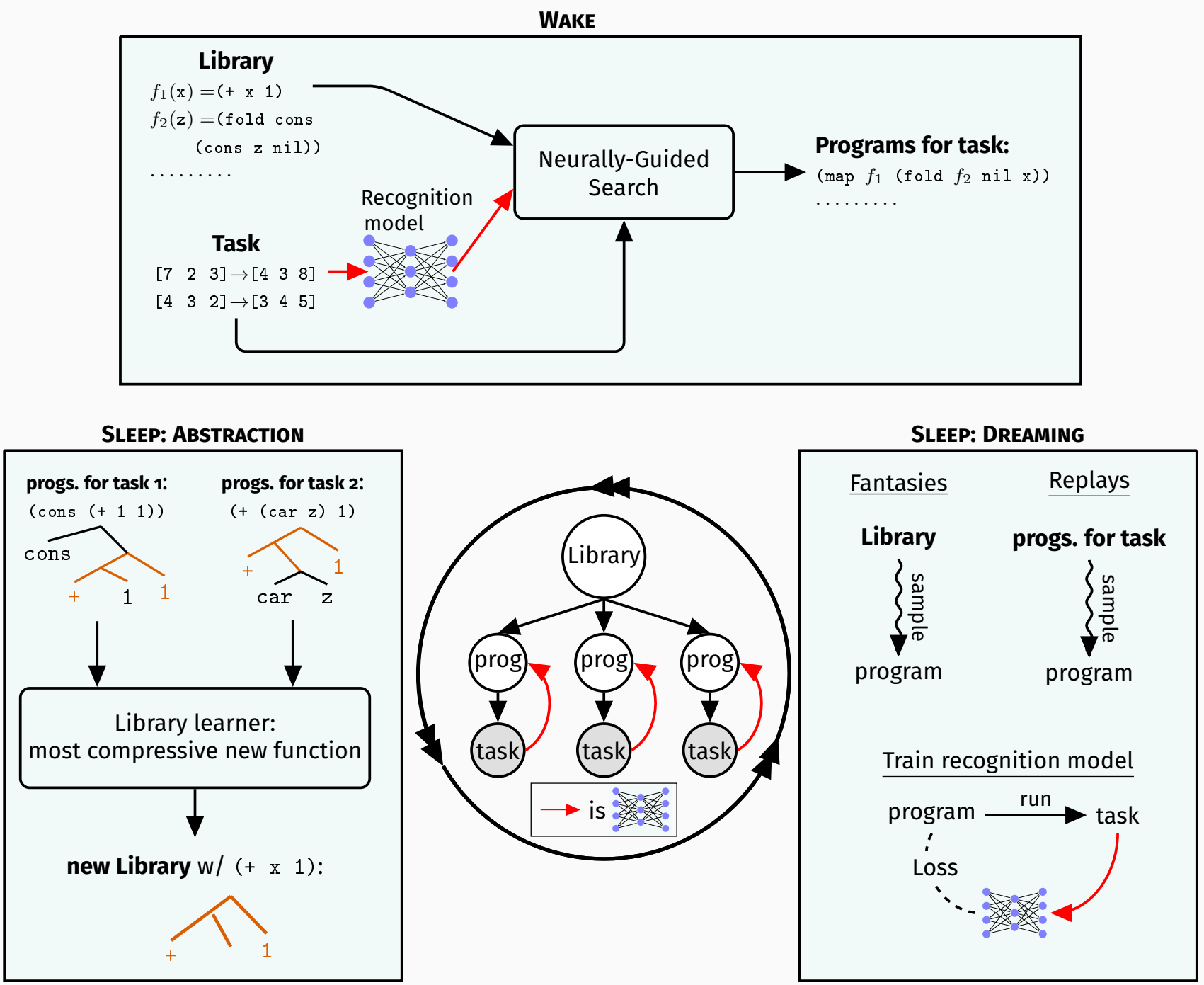

WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
SLEEP: DREAMING
Library
prog
prog
prog
task
task
task
is
Replays
progs. for task
sample
program
Train recognition model
program
run
task
Loss


WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Recognition
model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided
Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
SLEEP: DREAMING
Library
prog
prog
prog
task
task
task
is
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train recognition model
program
run
task
Loss


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]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(cons (+ 1 1))
progs. for task 2:
(+ (car z) 1)
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train recognition model
program
run
task
Loss


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]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(cons (+ 1 1))
cons
+ 1 1
progs. for task 2:
(+ (car z) 1)
+ 1
car z
Library learner:
most compressive new function
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train recognition model
program
run
task
Loss


WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Recognition model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(cons (+ 1 1))
cons
+ 1 1
progs. for task 2:
(+ (car z) 1)
+ 1
car z
Library learner:
most compressive new function
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train recognition model
program
run
task
Loss


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]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(cons (+ 1 1))
cons
+
1
1
progs. for task 2:
(+ (car z) 1)
+
1
car
z
Library learner:
most compressive new function
new Library w/ (+ x 1):
+
1
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
progs. for task
sample
sample
program
program
Train recognition model
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

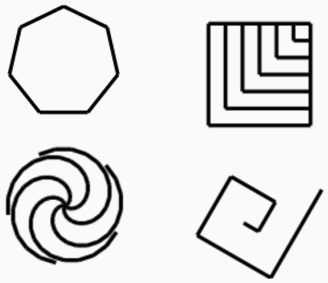
Phone numbers

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

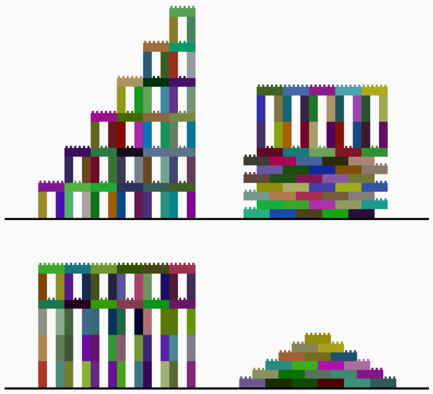
Currency

```
$100.25
$4.50
```

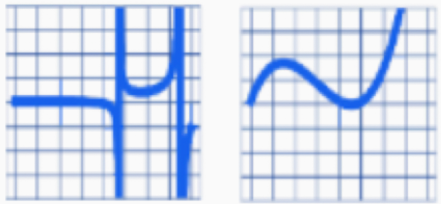
LOGO Graphics



Block Towers



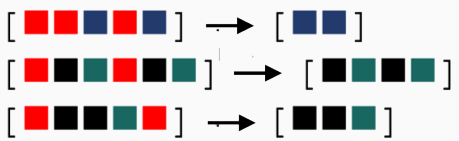
Symbolic Regression



$$y = f(x)$$

Recursive Programming

Filter Red



Physical Laws

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

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

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

DreamCoder Domains

List Processing

Sum List

```
[1 2 3] → 6
[4 6 8 1] → 17
```

Double

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

Text Editing

Abbreviate

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

Drop Last Three

```
shrdlu → shr
shakey → sha
```

Regexes

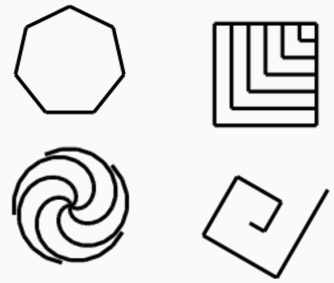
Phone numbers

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

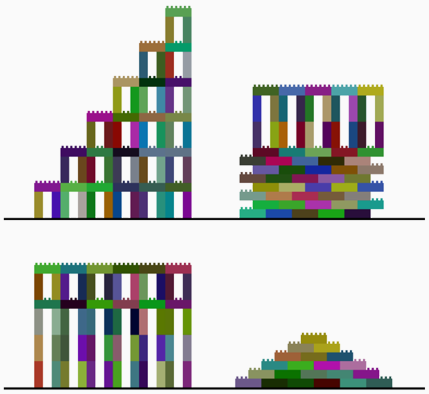
Currency

```
$100.25
$4.50
```

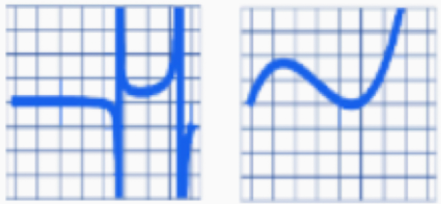
LOGO Graphics



Block Towers



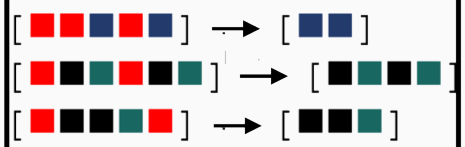
Symbolic Regression



$$y = f(x)$$

Recursive Programming

Filter Red



Physical Laws

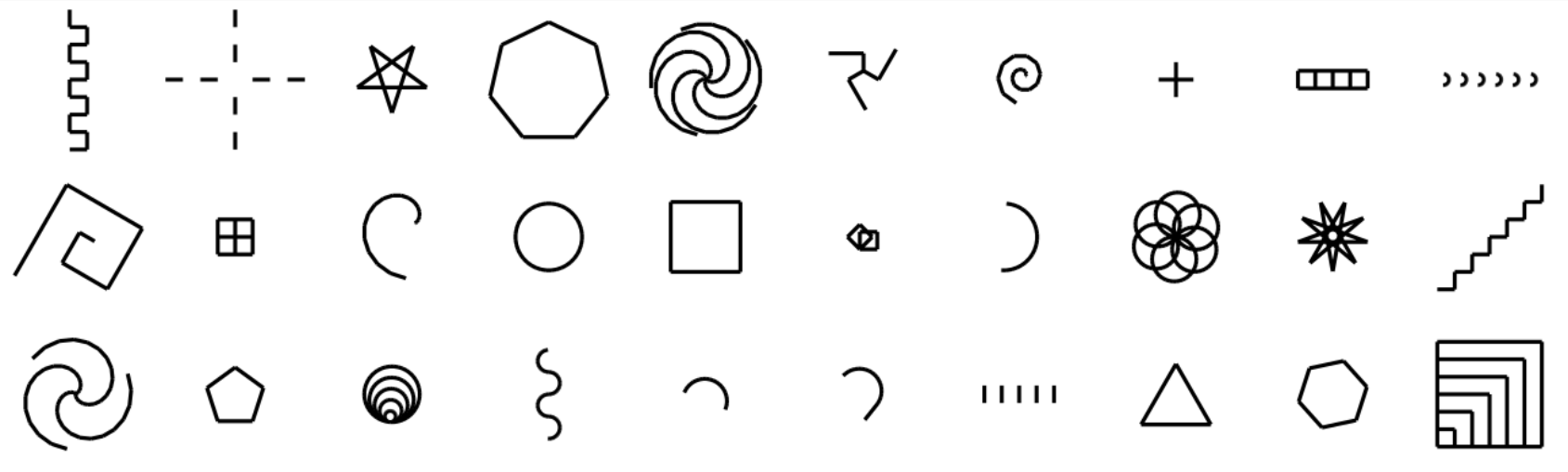
$$\vec{a} = \frac{1}{m}\sum_i \vec{F}_i$$

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

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

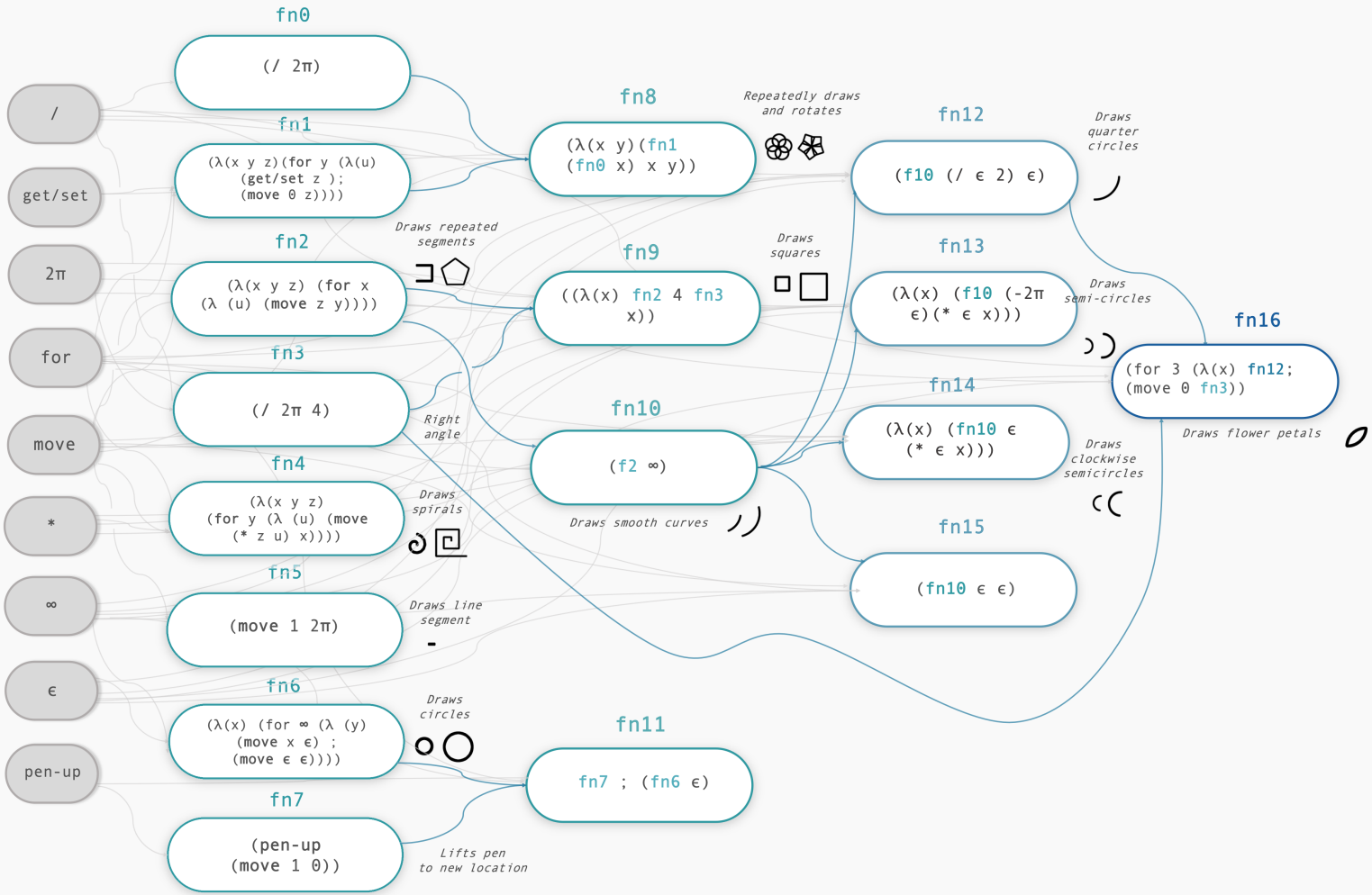
LOGO Turtle Graphics

30 out of 160 tasks



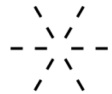
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

LOGO Turtle Graphics – learning an interpretable library



(fn8 5 (fn4 (* ϵ 2) ∞ ϵ))

(for 7 (λ (x) (fn9 x)))

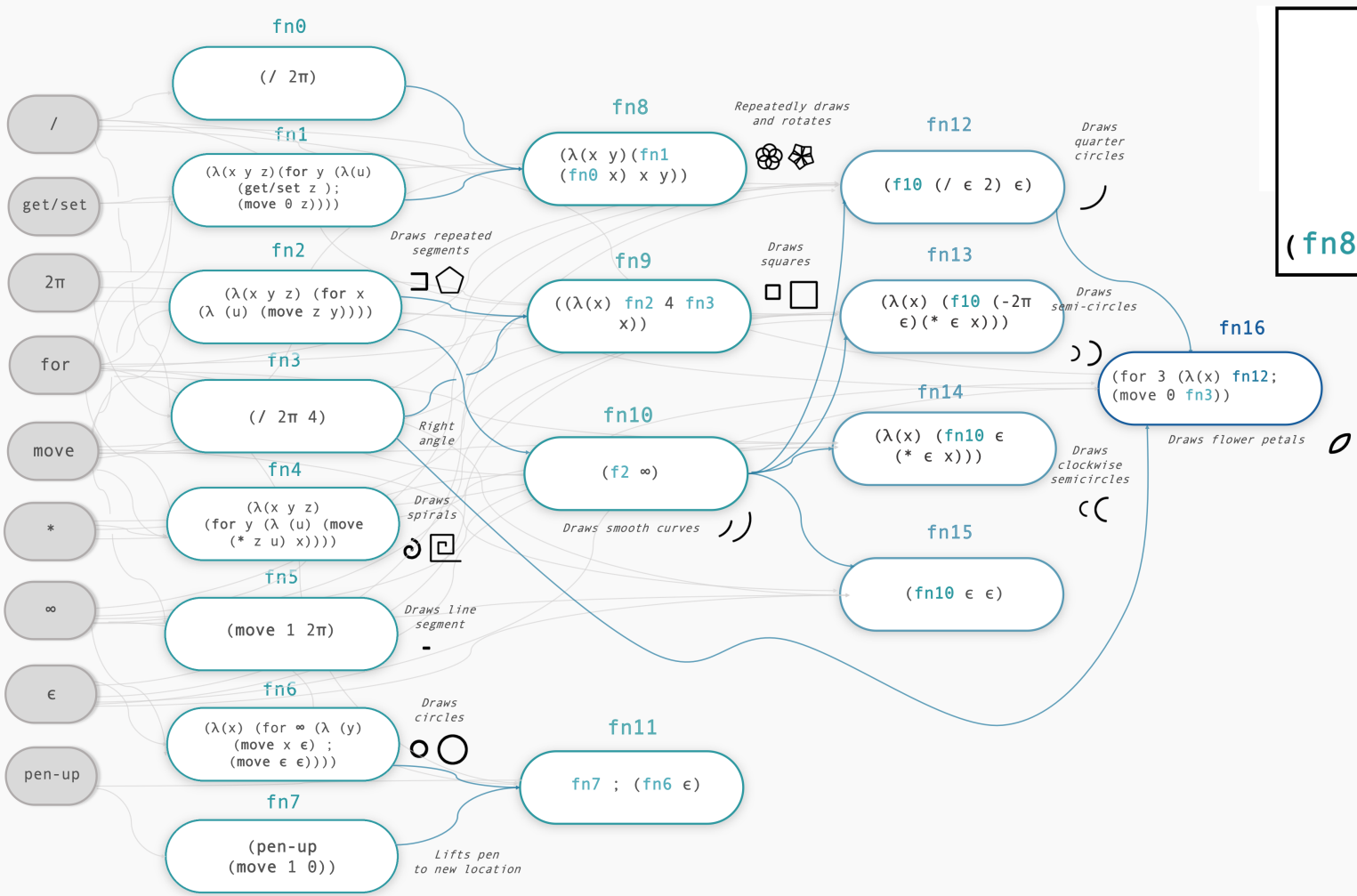


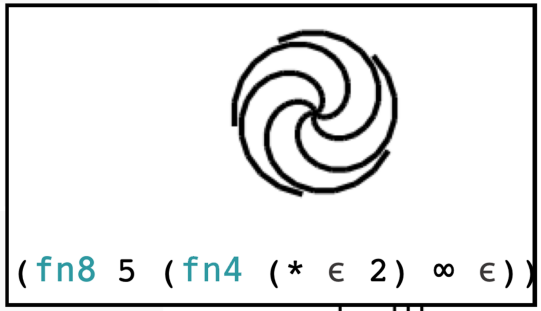
(fn8 6 (fn7 ; fn5 ; fn7 ; fn5))

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

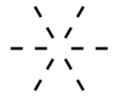
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

LOGO Turtle Graphics – learning an interpretable library





(for 7 (λ (x) (fn9 x)))

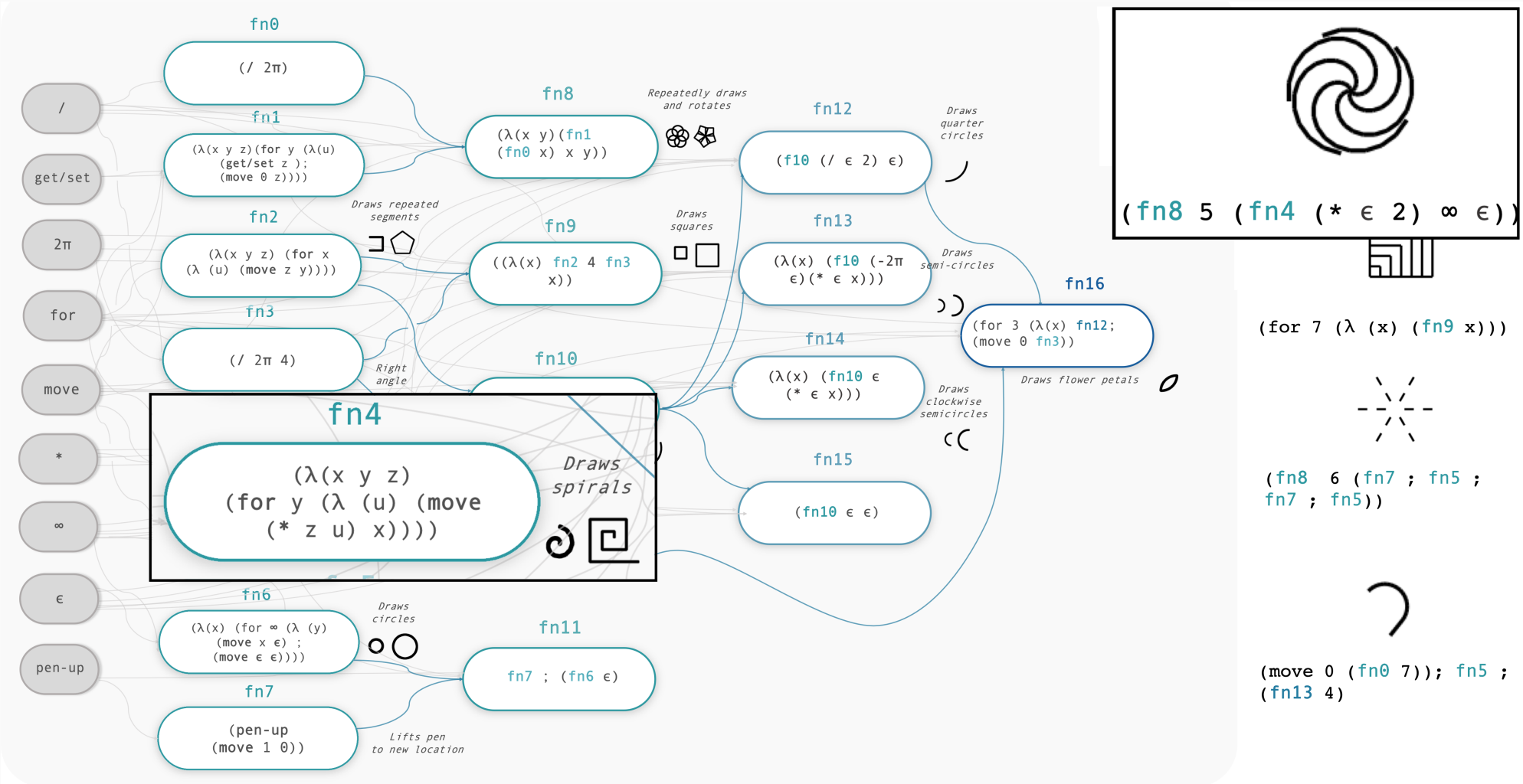


(fn8 6 (fn7 ; fn5 ; fn7 ; fn5))

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

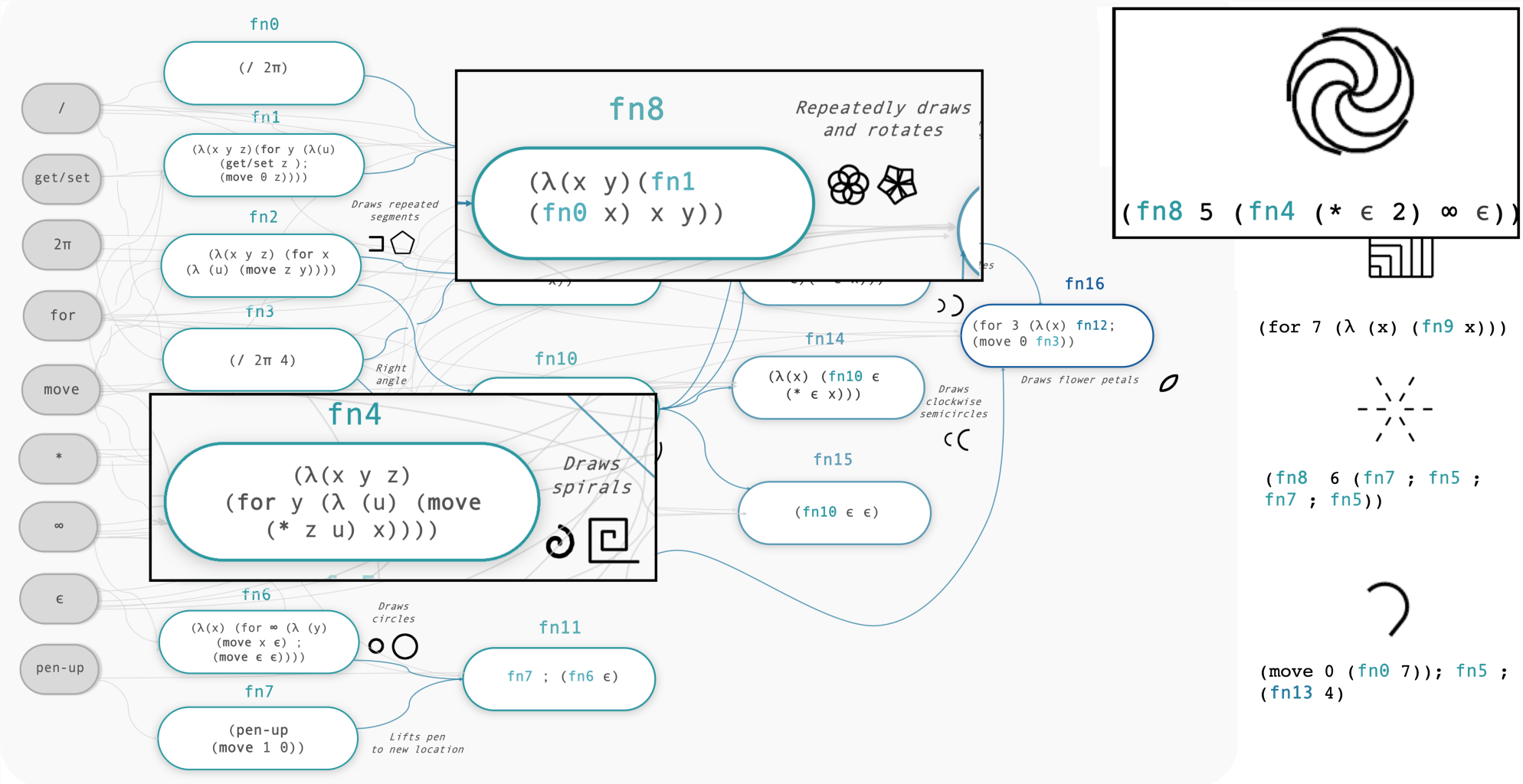
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

LOGO Turtle Graphics – learning an interpretable library



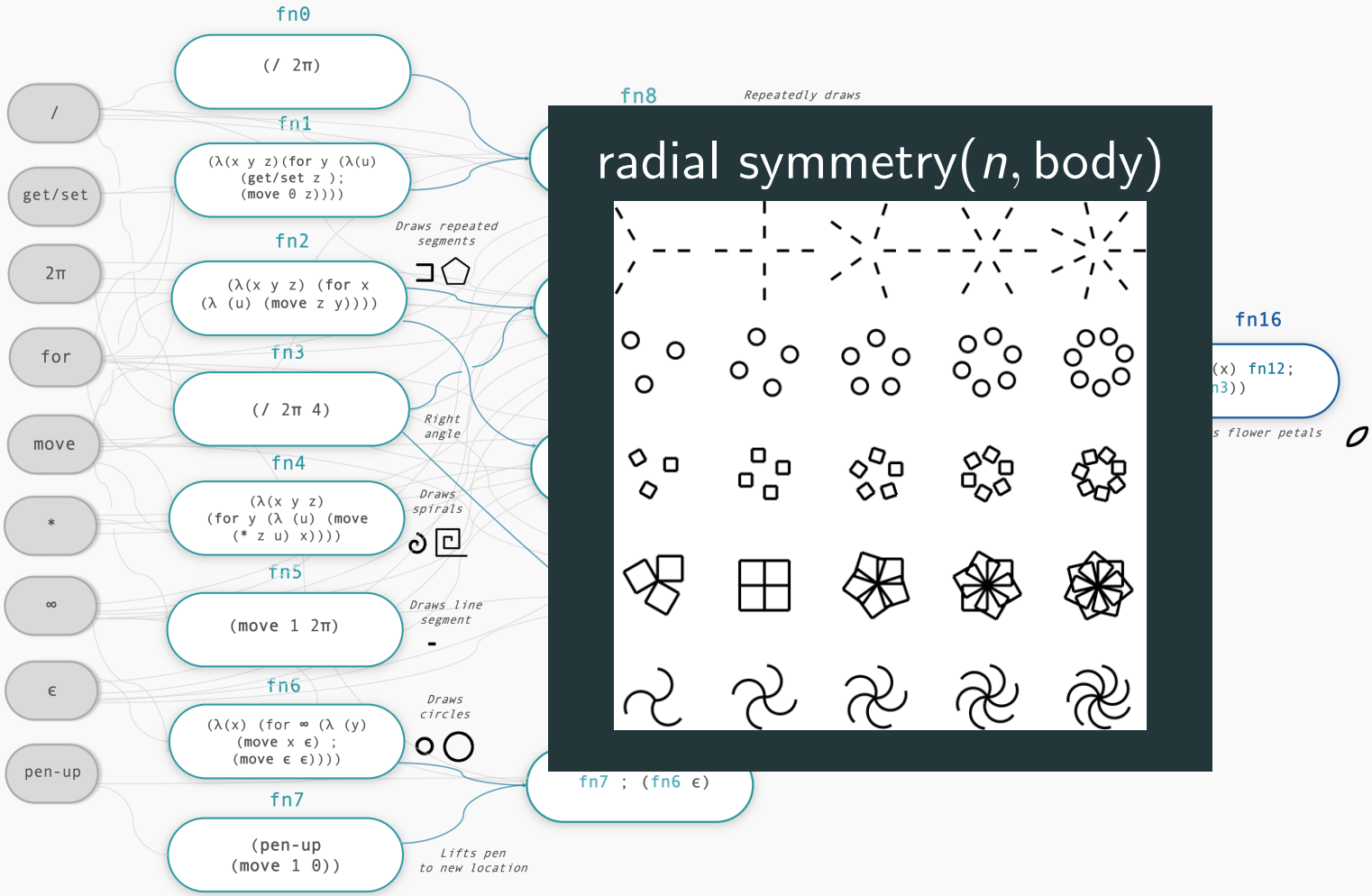
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

LOGO Turtle Graphics – learning an interpretable library

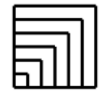


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

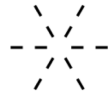
LOGO Turtle Graphics – learning an interpretable library



```
(fn8 5 (fn4 (* ϵ 2) ∞ ϵ))
```



```
(for 7 (λ (x) (fn9 x)))
```

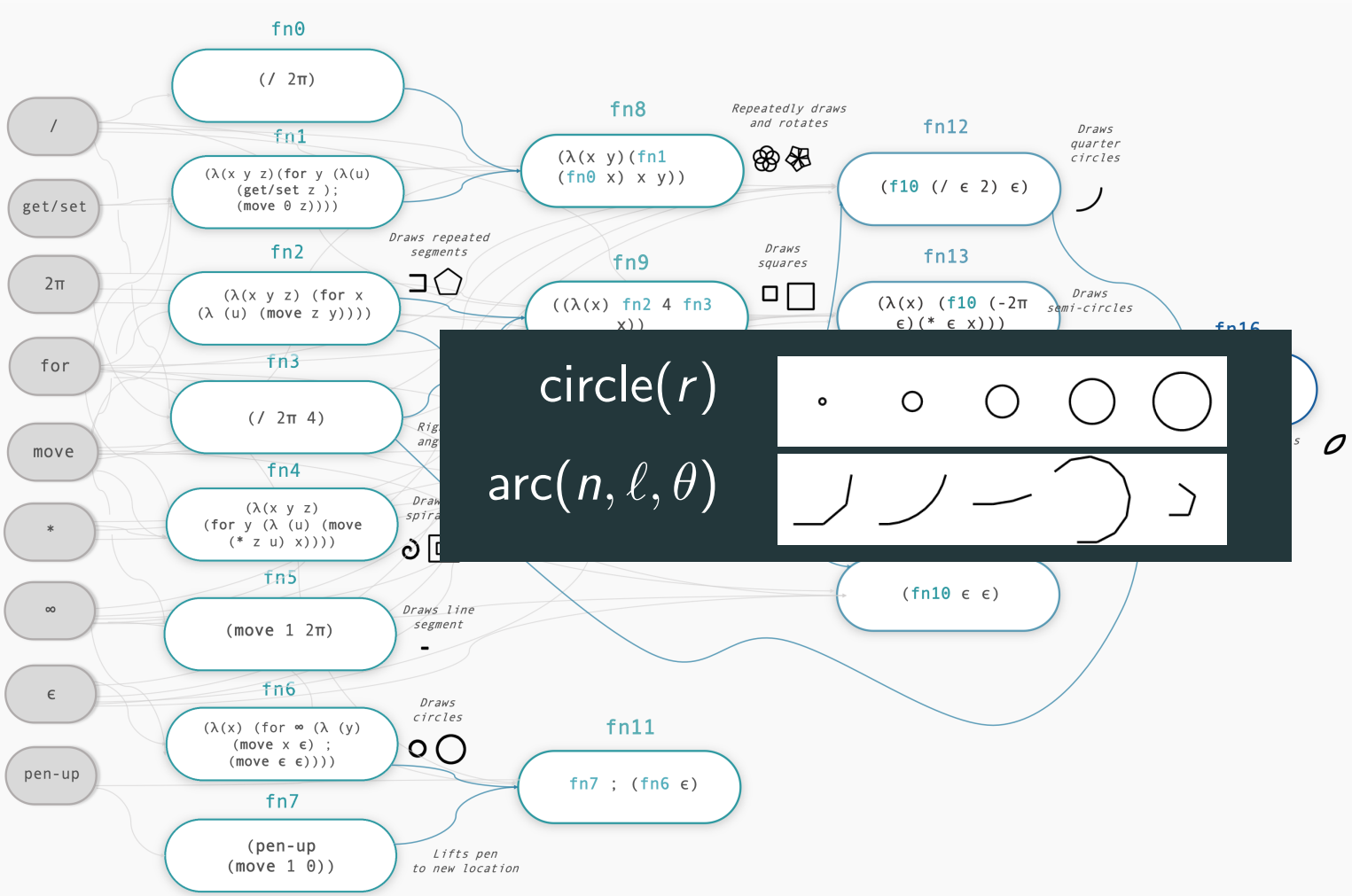


```
(fn8  6 (fn7 ; fn5 ;
fn7 ; fn5))
```

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

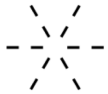
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

LOGO Turtle Graphics – learning an interpretable library



```
(fn8 5 (fn4 (* ϵ 2) ∞ ϵ))
```

```
(for 7 (λ (x) (fn9 x)))
```

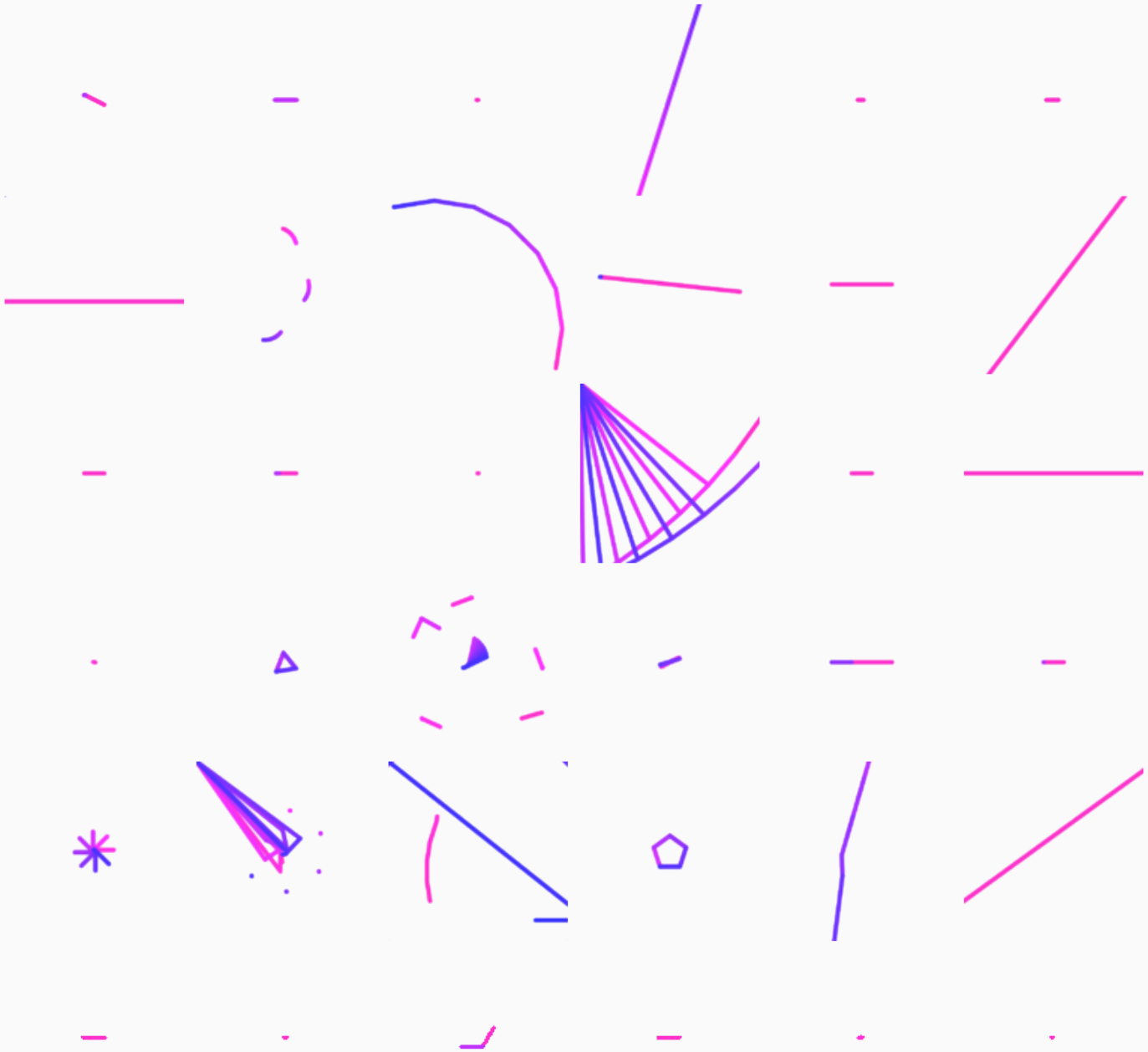


```
(fn8  6 (fn7 ; fn5 ;
fn7 ; fn5))
```

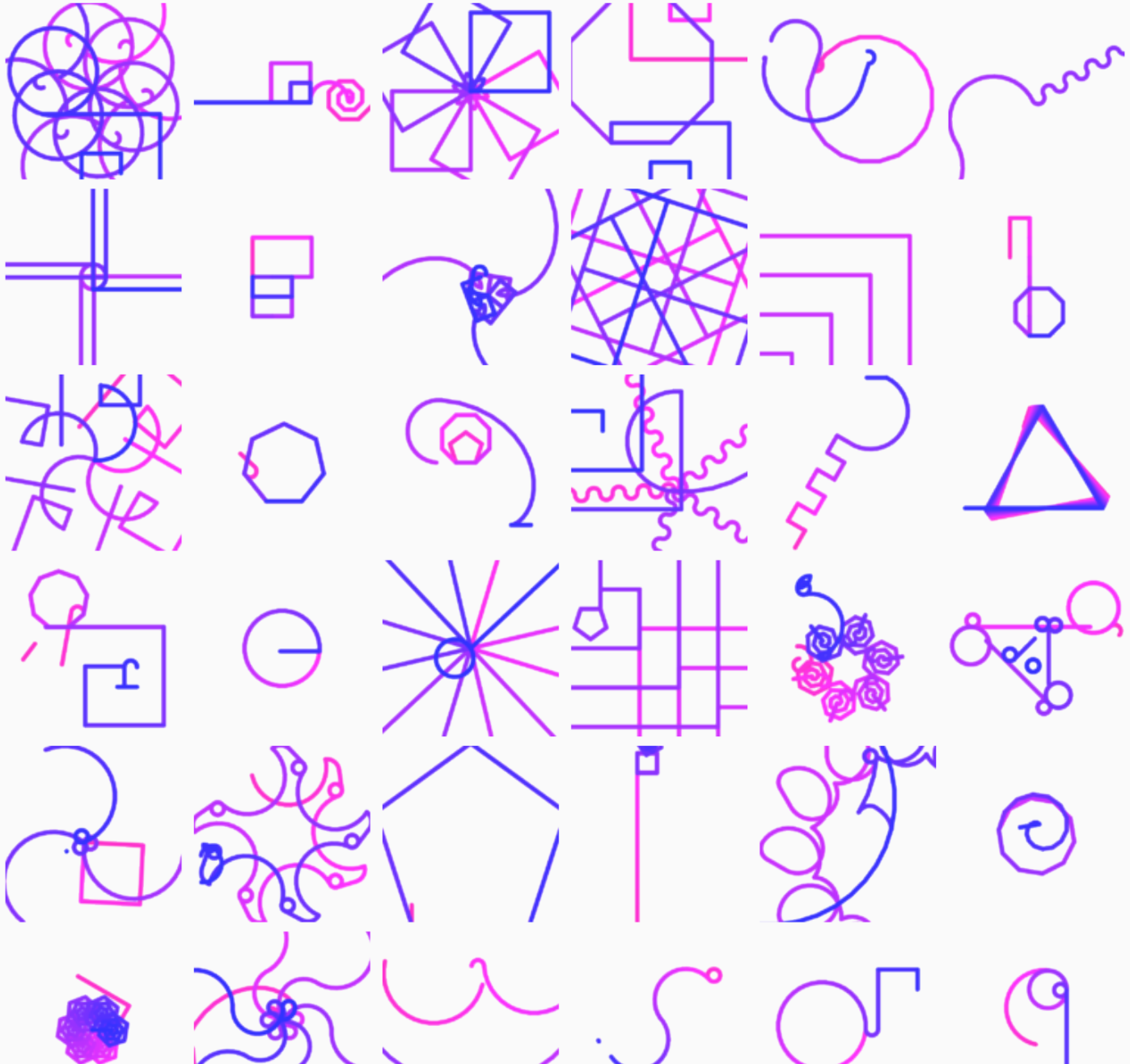
```
(move 0 (fn0 7)); fn5 ;
(fn13 4)
```

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

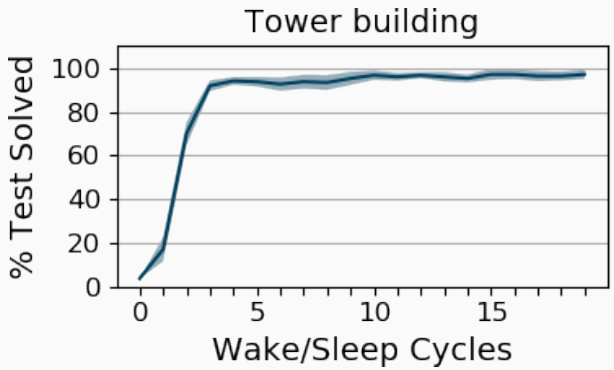
What does DreamCoder dream of? (before learning)



What does DreamCoder dream of? (after learning)

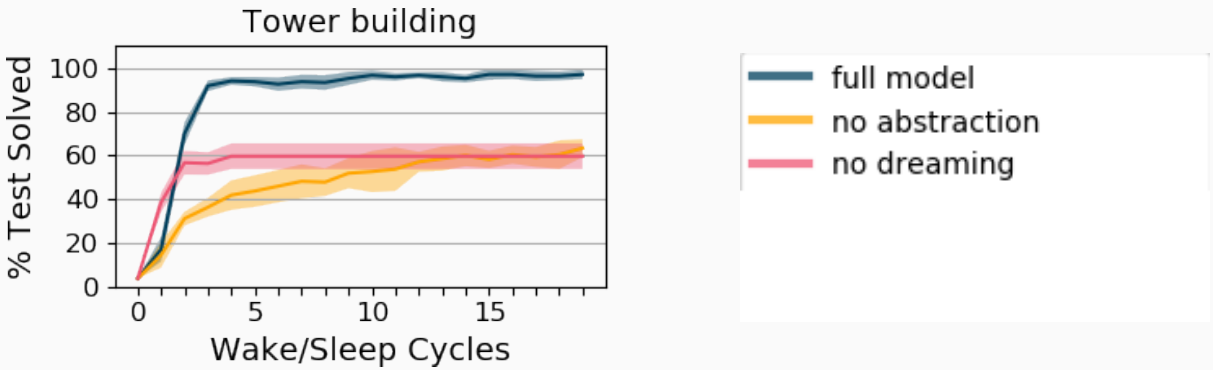


Learning dynamics



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

Learning dynamics



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

Synergy between Dreaming, Discovery, & Higher-level knowledge

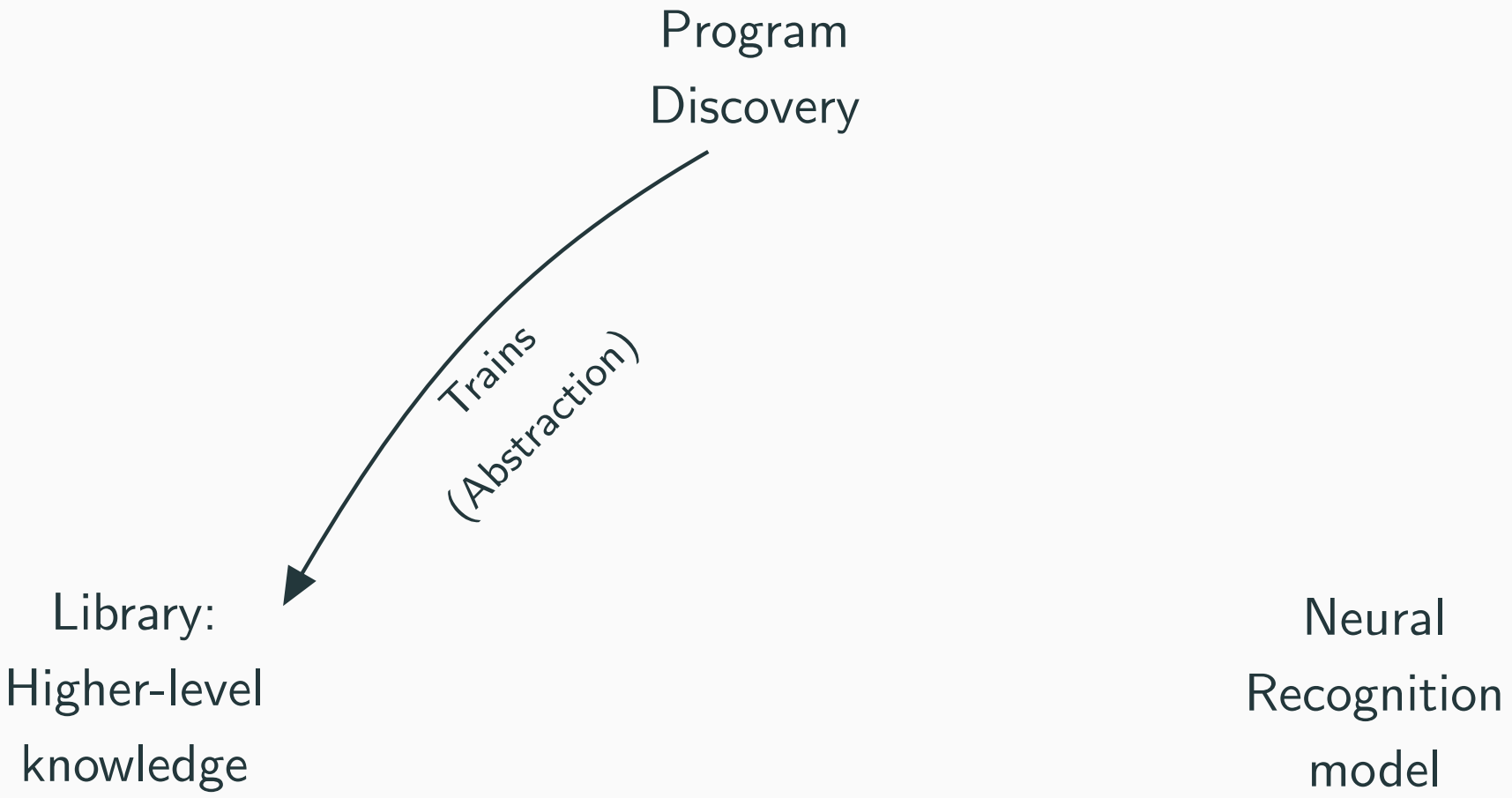
Program
Discovery

Library:
Higher-level
knowledge

Neural
Recognition
model

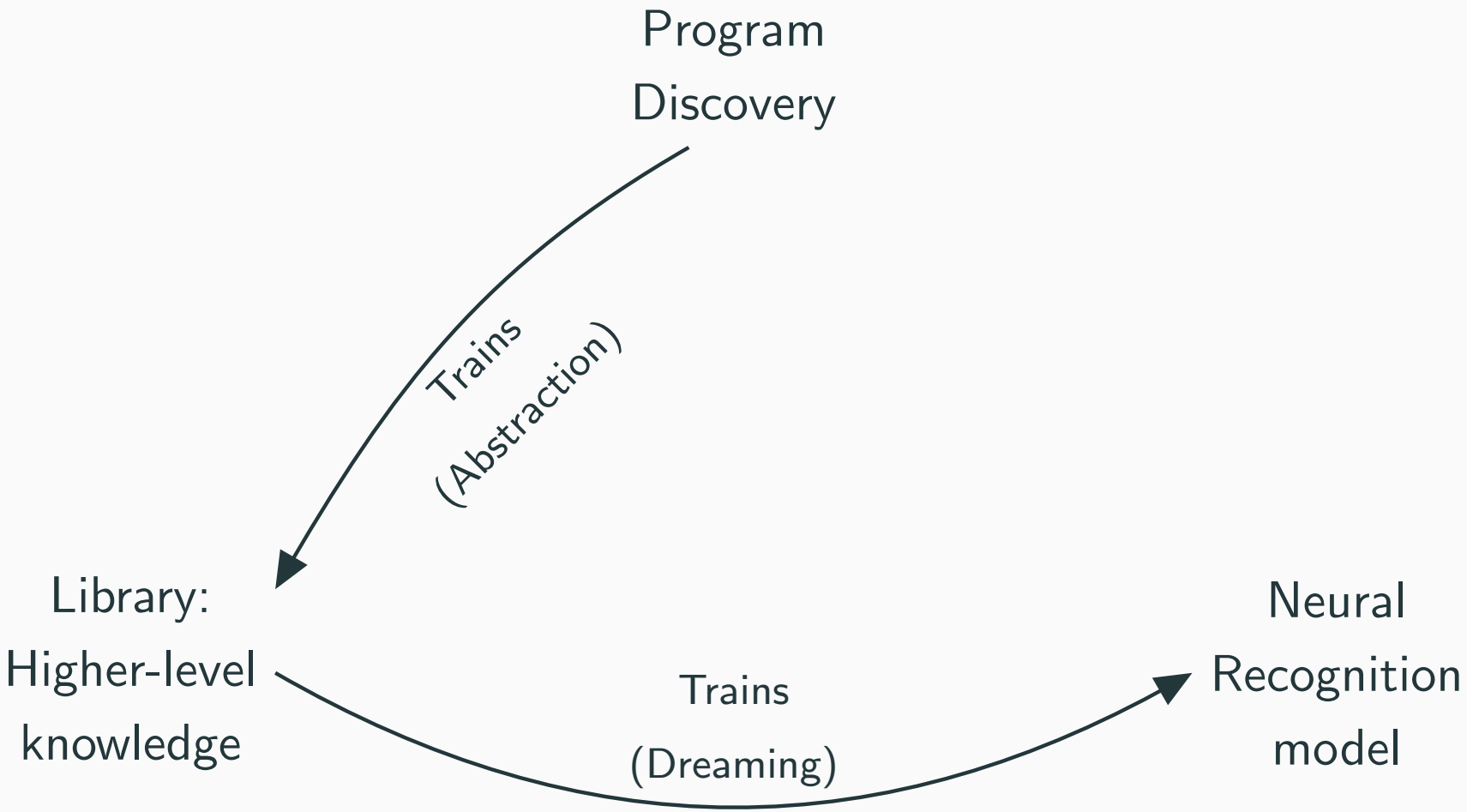
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

Synergy between Dreaming, Discovery, & Higher-level knowledge



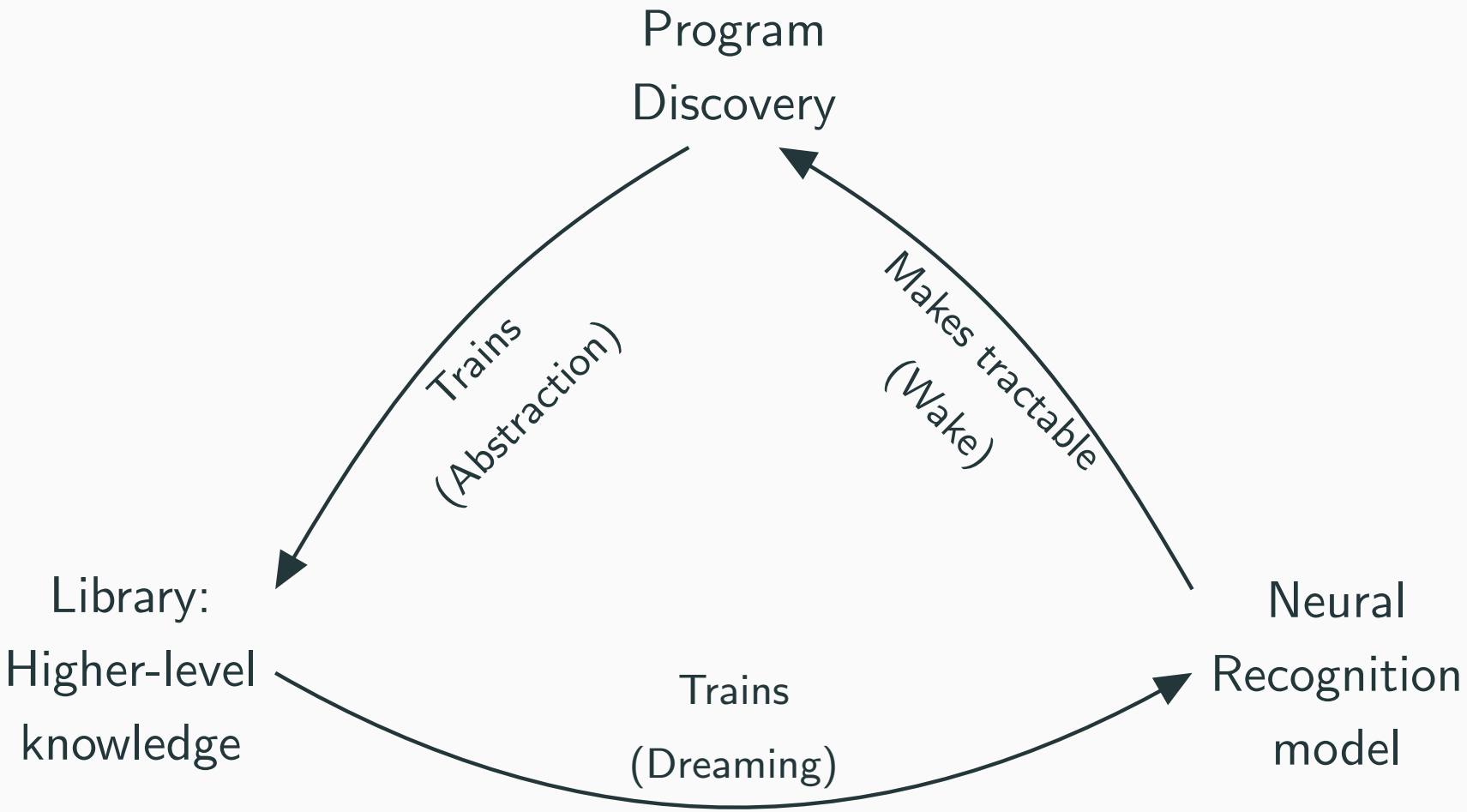
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

Synergy between Dreaming, Discovery, & Higher-level knowledge



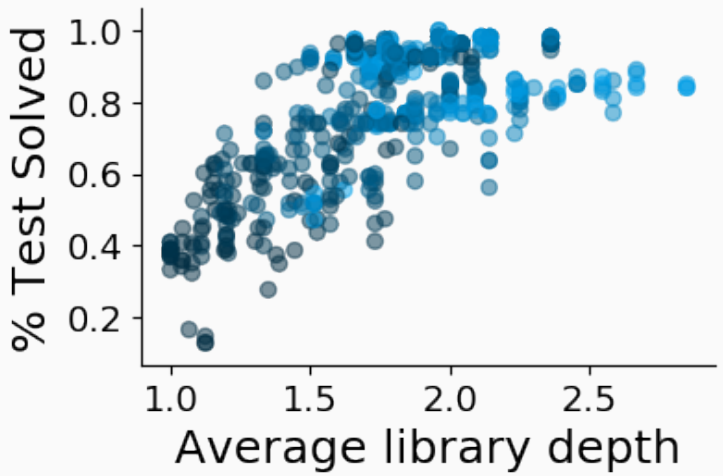
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

Synergy between Dreaming, Discovery, & Higher-level knowledge



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

Evidence for dreaming bootstrapping better libraries

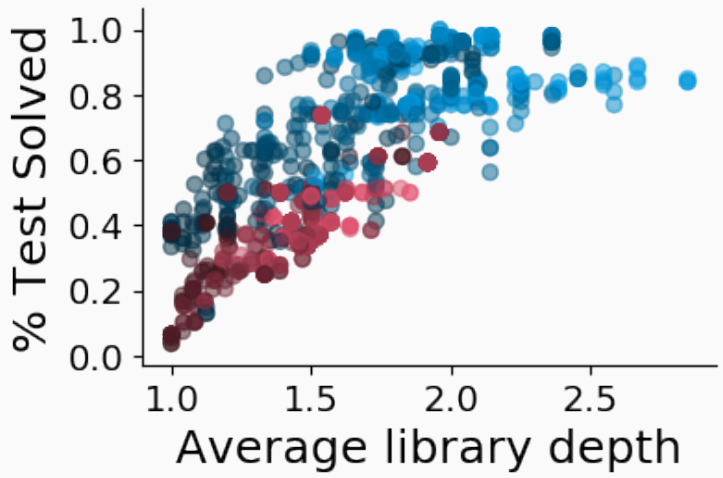


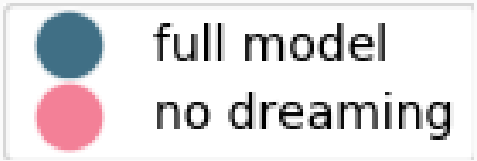
Darker: Early in learning

Brighter: Later in learning

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

Evidence for dreaming bootstrapping better libraries





Darker: Early in learning

Brighter: Later in learning

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

From learning libraries, to learning languages

From learning libraries, to learning languages

modern functional programming $\rightarrow$ physics

From learning libraries, to learning languages

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

Physics Formula Sheet

Mechanics

$x = x_0 + v_{x0}t + \frac{1}{2}a_x t^2$

$v = v_0 + at$

$v_x^2 - v_{x0}^2 = 2a(x - x_0)$

$\vec{a} = \frac{\Sigma \vec{F}}{m} = \frac{\vec{F}_{net}}{m}$

$|\vec{F}_{friction}| \leq \mu |\vec{F}_{Normal}|$

$\vec{p} = m\vec{v}$

$\Delta \vec{p} = \vec{F}\Delta t$

$KE = \frac{1}{2}mv^2$

$\Delta PE = mg\Delta y$

$\Delta E = W = Fd\cos\theta$

$a_c = \frac{v^2}{r}$

$\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$

$\omega = \omega_0 + \alpha t$

$T = \frac{2\pi}{\omega} = \frac{1}{f}$

$\nu = f\lambda$

$x = A\cos(2\pi f t)$

$\vec{\alpha} = \frac{\Sigma \vec{\tau}}{I} = \frac{\vec{\tau}_{net}}{I}$

$\vec{\tau} = r \times F$

$L = I\,\omega$

$\Delta L = \tau \Delta t$

$|\vec{F}_{spring}| = k|\vec{x}|$

$PE_{spring} = \frac{1}{2}kx^2$

$T_{spring} = 2\pi\sqrt{\frac{m}{k}}$

$T_{pendulum} = 2\pi\sqrt{\frac{\ell}{g}}$

$|\vec{F}_{gravity}| = G\frac{m_1 m_2}{r^2}$

$|\vec{F}_{gravity}| = m\vec{g}$

$PE_{gravity} = -G\frac{m_1 m_2}{r}$

$\rho = \frac{m}{V}$

$KE = \frac{1}{2}I\,\omega^2$

Electricity

$|\vec{F}_E| = k\left|\frac{q_1 q_2}{r^2}\right|$

$\Delta V = IR$

$R = \frac{\rho \ell}{A}$

$I = \frac{\Delta q}{\Delta t}$

$P = I\Delta V$

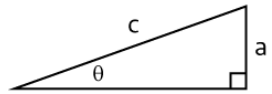
$R_{series} = R_1 + R_2 + \ldots + R_n$

$\frac{1}{R_{Parallel}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$

Geometry

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

Trigonometry



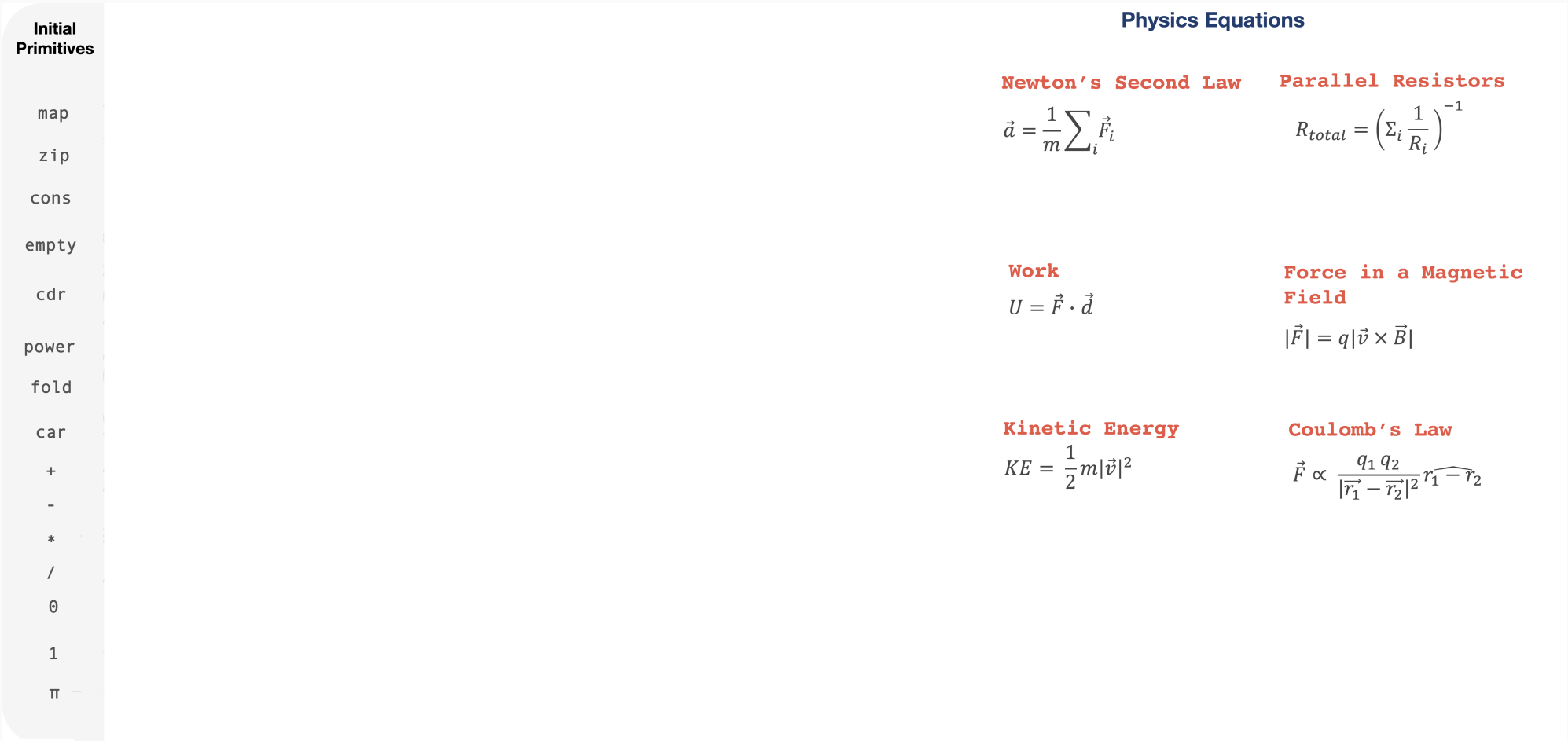
$c^2 = a^2 + b^2$

$\sin\theta = \frac{a}{c}$ $\cos\theta = \frac{b}{c}$ $\tan\theta = \frac{a}{b}$

Variables

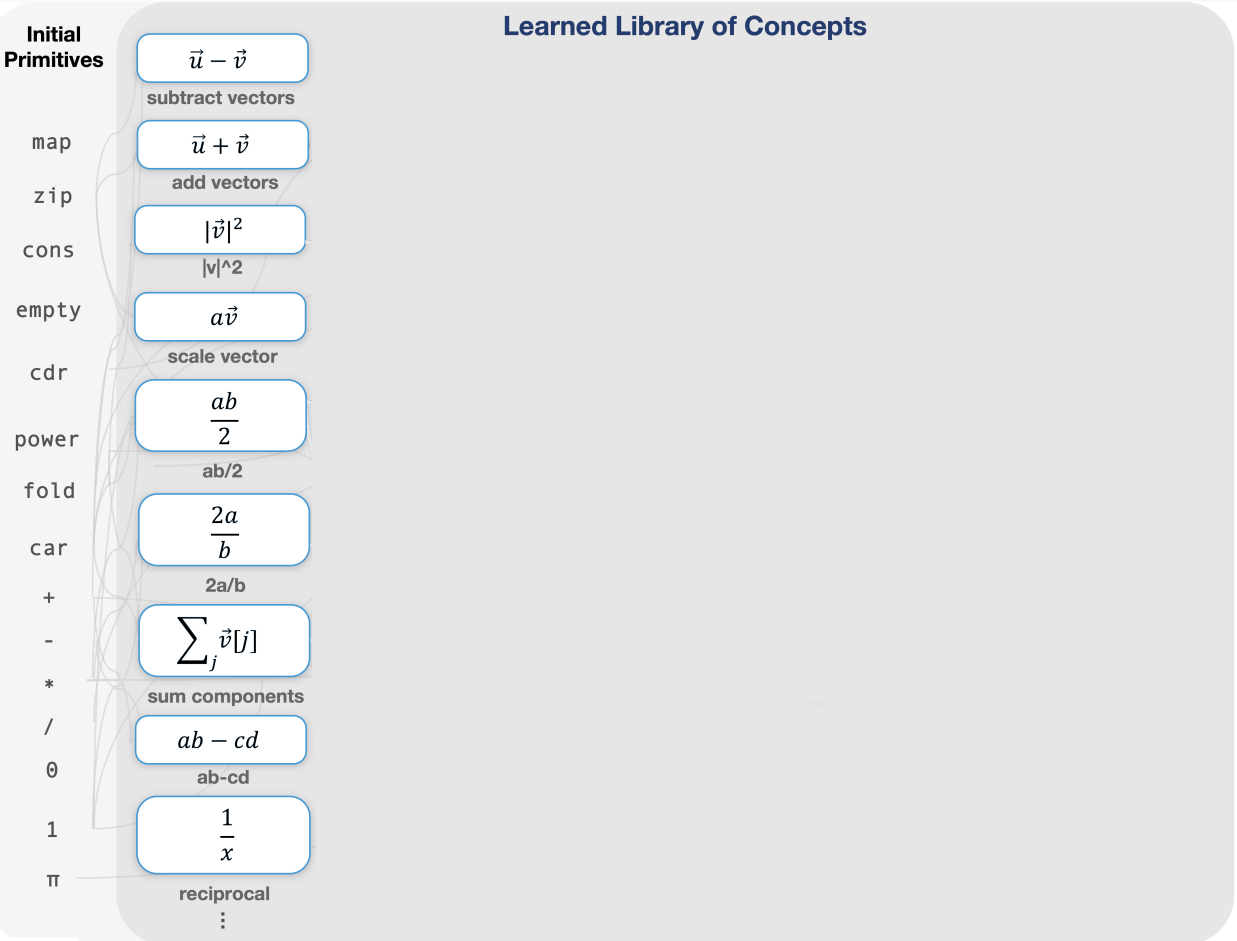
a = acceleration
A = amplitude
A = Area
b = base length
C = circumference
d = distance
E = energy
f = frequency
F = force
h = height
I = current
I = rotational inertia
KE = kinetic energy
k = spring constant
L = angular momentum
ℓ = length
m = mass
P = power
p = momentum
q = charge
r = radius
R = resistance
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
α = angular acceleration
λ = wavelength
μ = coefficient of friction

Growing languages for vector algebra and physics



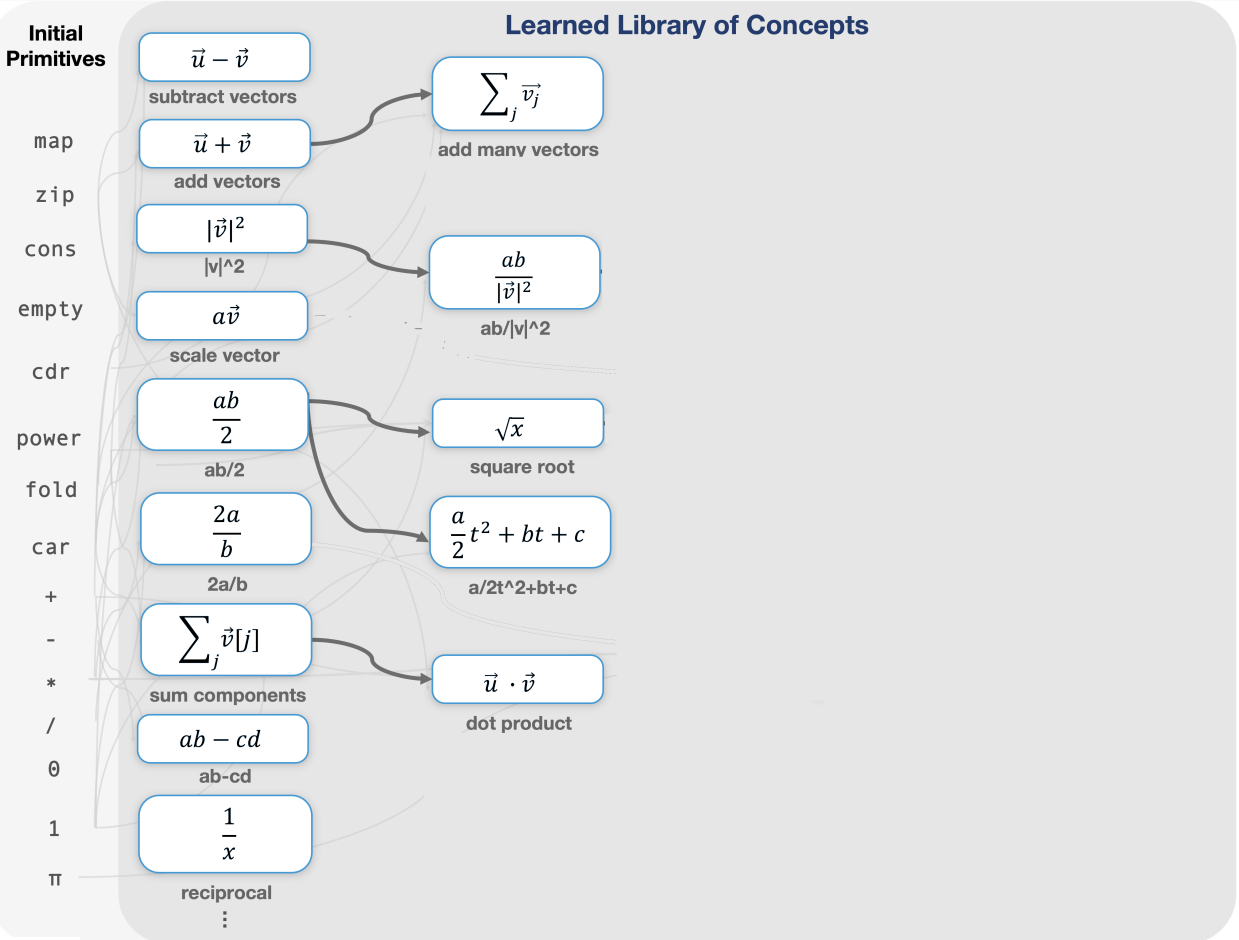
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

Growing languages for vector algebra and physics



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

Growing languages for vector algebra and physics



Physics Equations

Newton's Second Law

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

Parallel Resistors

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

Work

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

Force in a Magnetic Field

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

Kinetic Energy

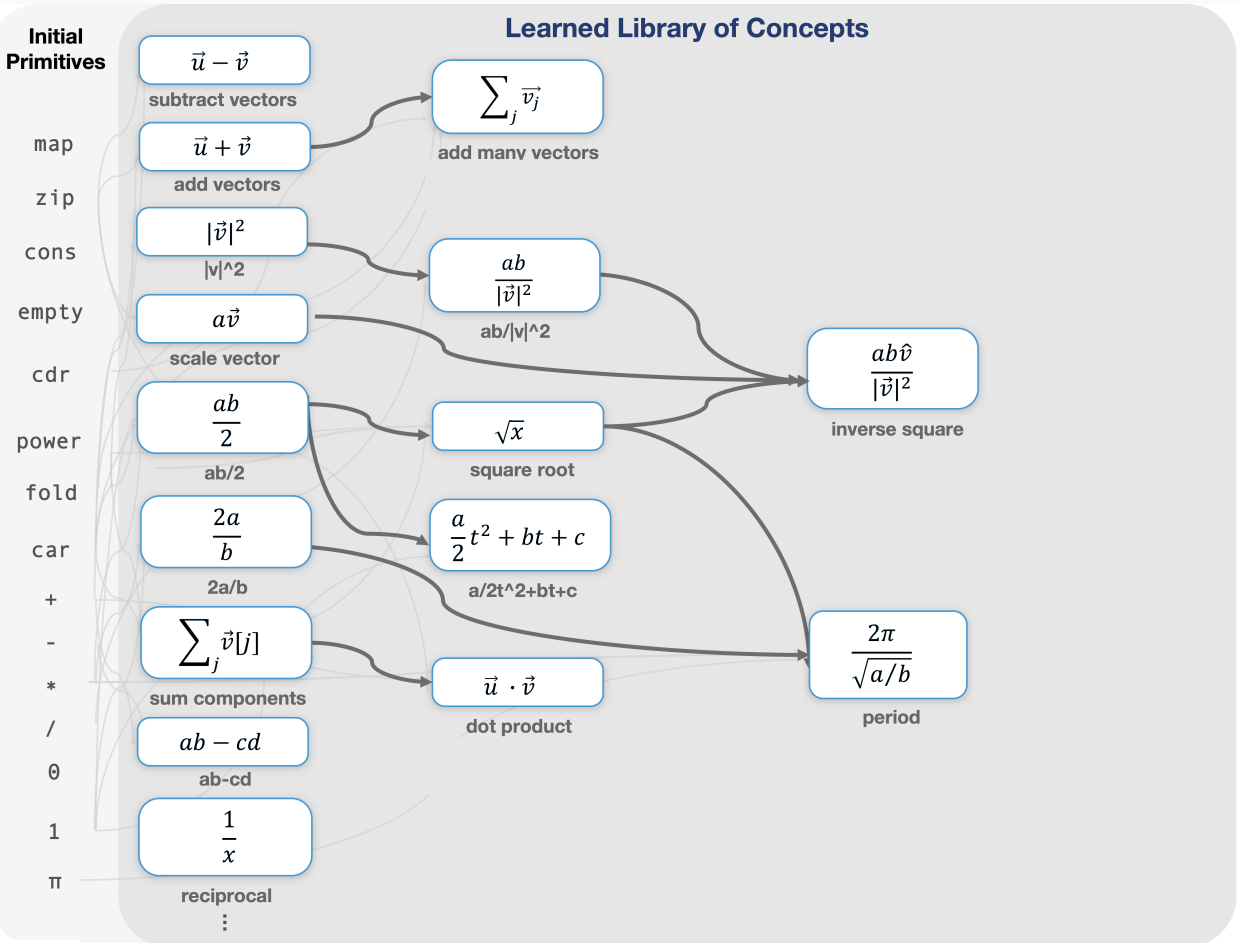
$$KE = \frac{1}{2}m|\vec{v}|^2$$

Coulomb's Law

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

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

Growing languages for vector algebra and physics



Physics Equations

Newton's Second Law

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

Parallel Resistors

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

Work

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

Force in a Magnetic Field

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

Kinetic Energy

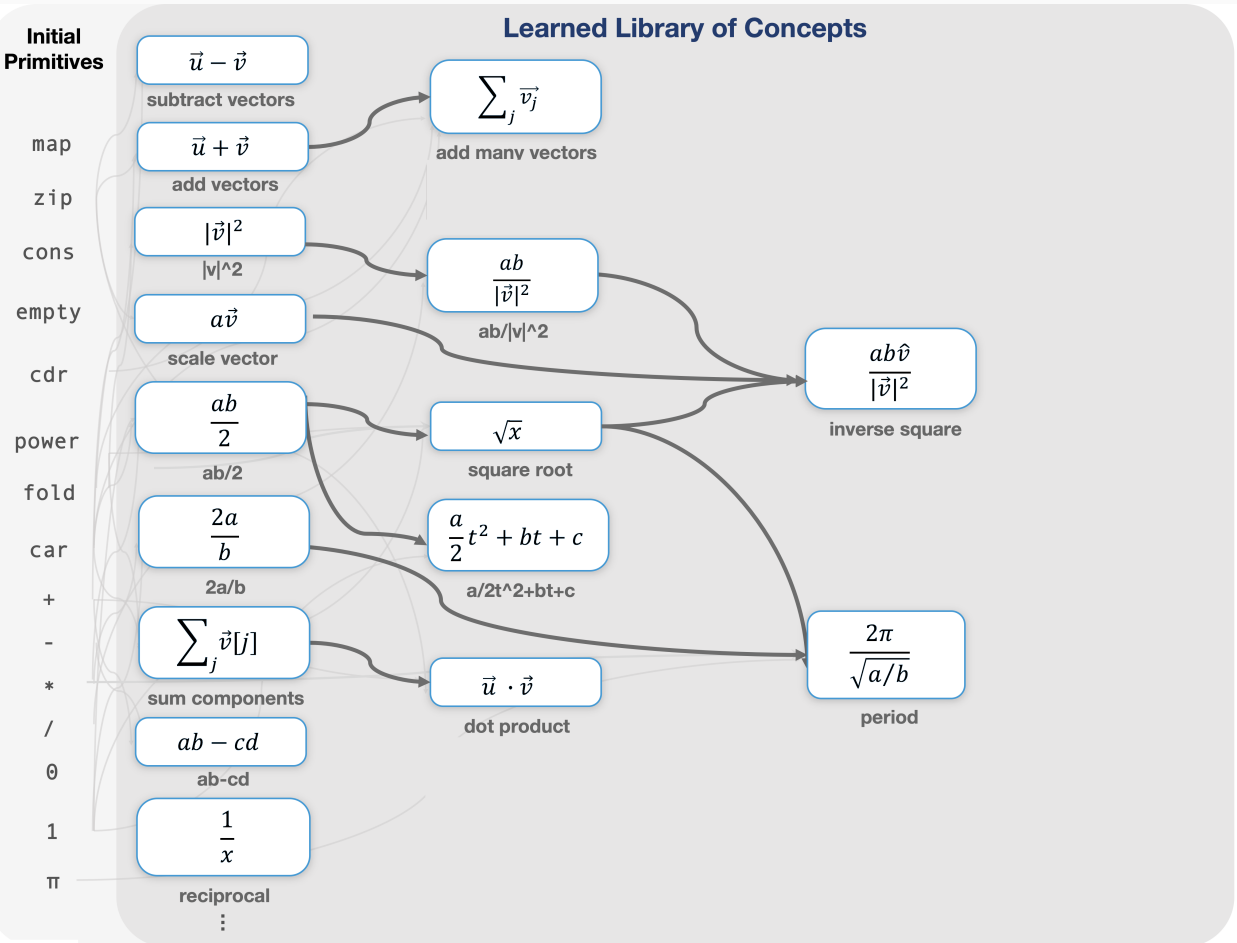
$$KE=\frac{1}{2}m|\vec{v}|^2$$

Coulomb's Law

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

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

Growing languages for vector algebra and physics



Physics Equations

Newton's Second Law

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

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

Parallel Resistors

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

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

Work

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

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

Force in a Magnetic Field

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

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

Kinetic Energy

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

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

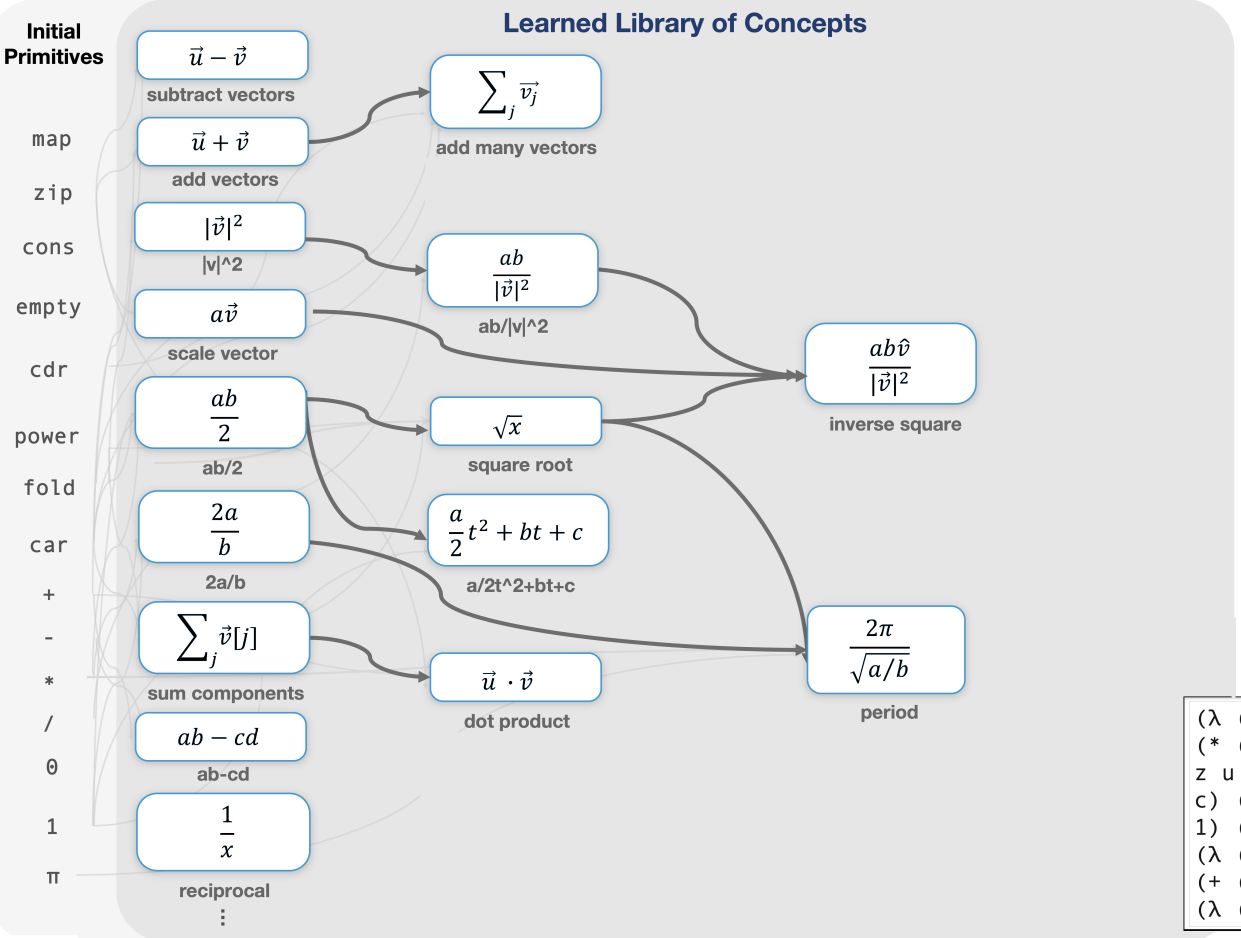
Coulomb's Law

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

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

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

Growing languages for vector algebra and physics



Physics Equations

Newton's Second Law

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

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

Parallel Resistors

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

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

Work

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

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

Force in a Magnetic Field

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

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

Kinetic Energy

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

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

Coulomb's Law

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

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

```
(λ (x y z u) (map (λ (v) (* (/
(* (power (/ (* x x) (fold (zip
z u (λ (w a) (- w a))) 0 (λ (b
c) (+ (* b b)    c)))) (/ (* 1
1) (+ 1 1))) y) (fold (zip z u
(λ (d e)  (- d e))) 0 (λ (f g)
(+ (* f f) g)))) v)) (zip z u
(λ (h i) (- h i)))))
```

Solution to `Coulomb's Law` if expressed in initial primitives

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

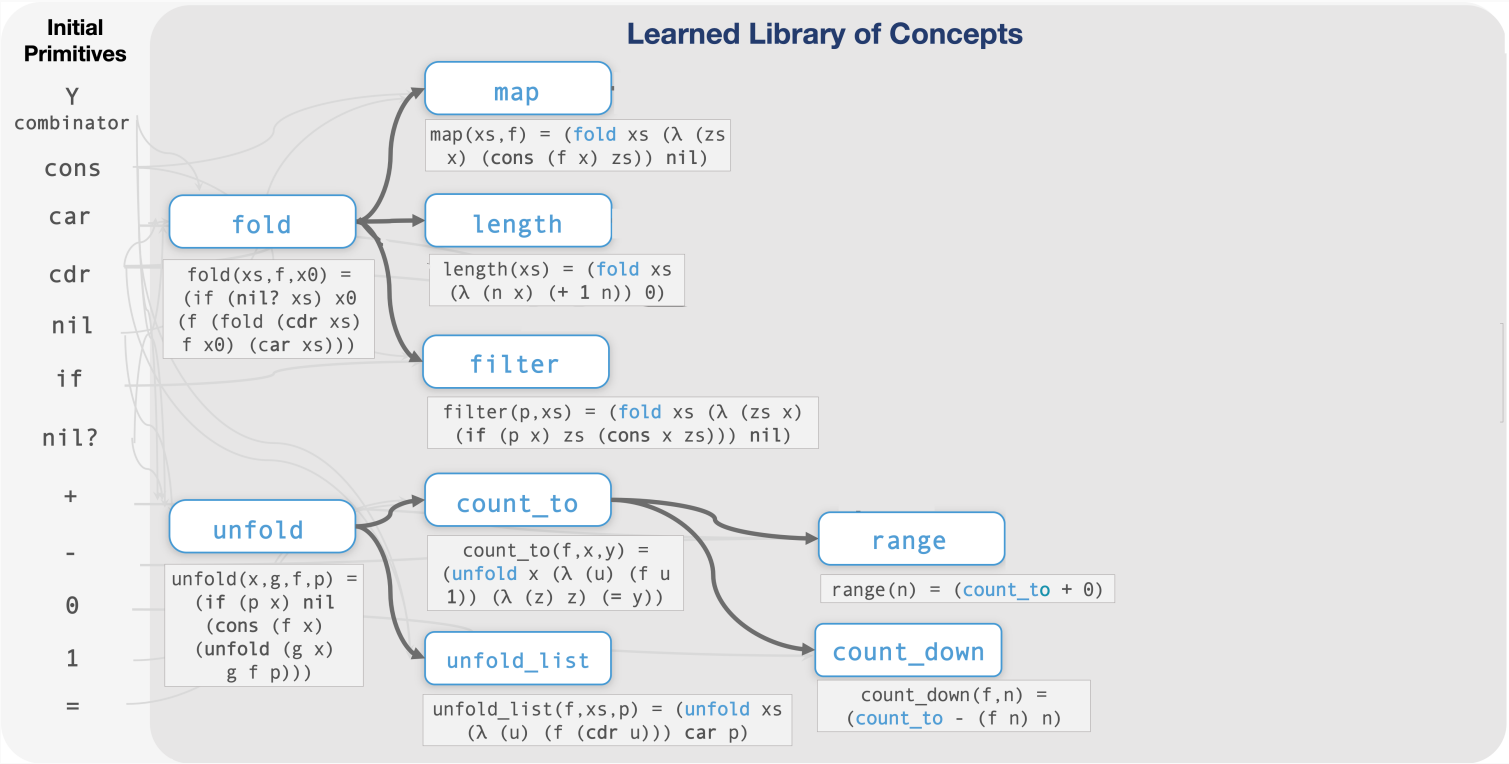
[1 8 2], [0 5 1] → [1 3 1]
[2 3 6], [1 2 4] → [1 1 2]

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

Growing a language for recursive programming

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

Growing a language for recursive programming



Recursive Programming Algorithms

Stutter

[■■] → [■■■■]
[■■■] → [■■■■■■]

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

Take every other

[■■■■] → [■■]
[■■■■■■] → [■■■]

```
(unfold_list cdr A nil?)
```

List lengths

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

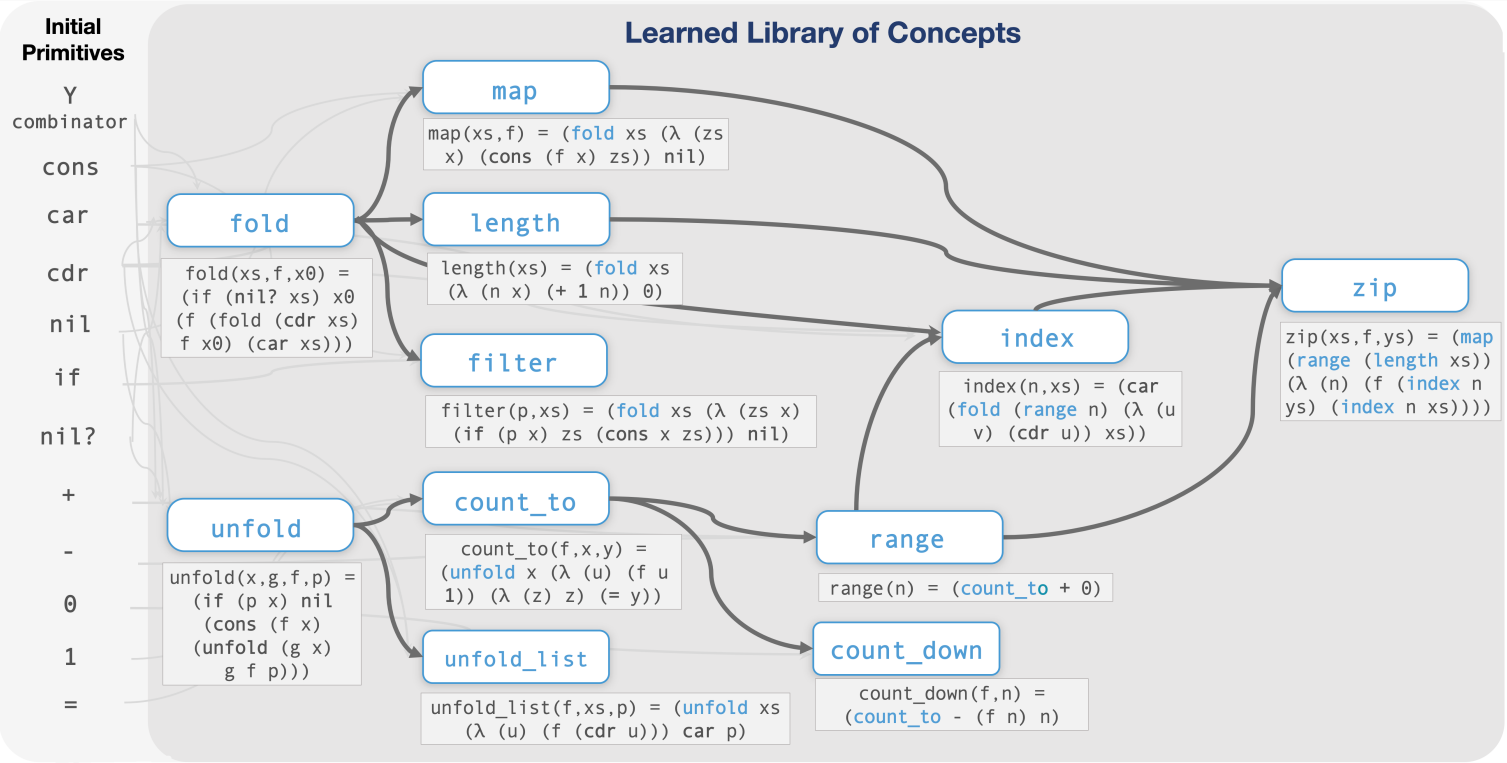
```
(map A length)
```

List differences

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

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

Growing a language for recursive programming



Recursive Programming Algorithms

Stutter

[■■] → [■■■■]
[■■■] → [■■■■■■]

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

Take every other

[■■■■] → [■■]
[■■■■■■] → [■■■]

```
(unfold_list cdr A nil?)
```

List lengths

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

```
(map A length)
```

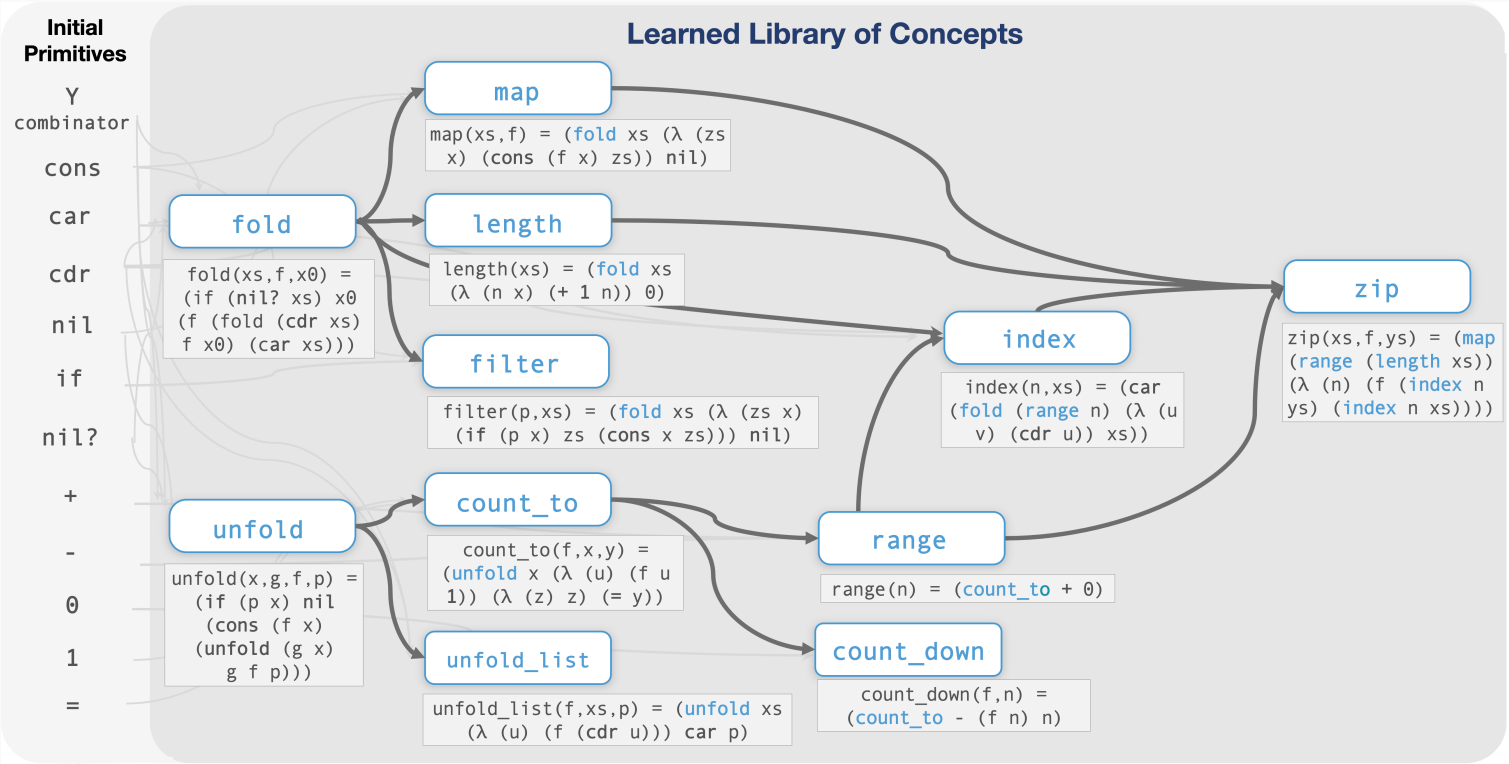
List differences

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

```
(zip A - B )
```

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

Growing a language for recursive programming



Recursive Programming Algorithms

Stutter

[■■] → [■■■■]
[■■■] → [■■■■■■]

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

Take every other

[■■■■] → [■■]
[■■■■■■] → [■■■]

```
(unfold_list cdr A nil?)
```

List lengths

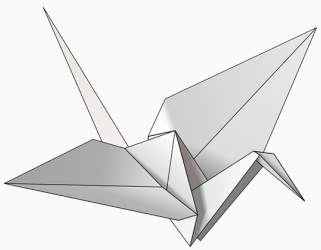
[[■■■], [■]] → [3 1]
[[■■], [], [■]] → [2 0 1]

```
(map A length)
```

List differences

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

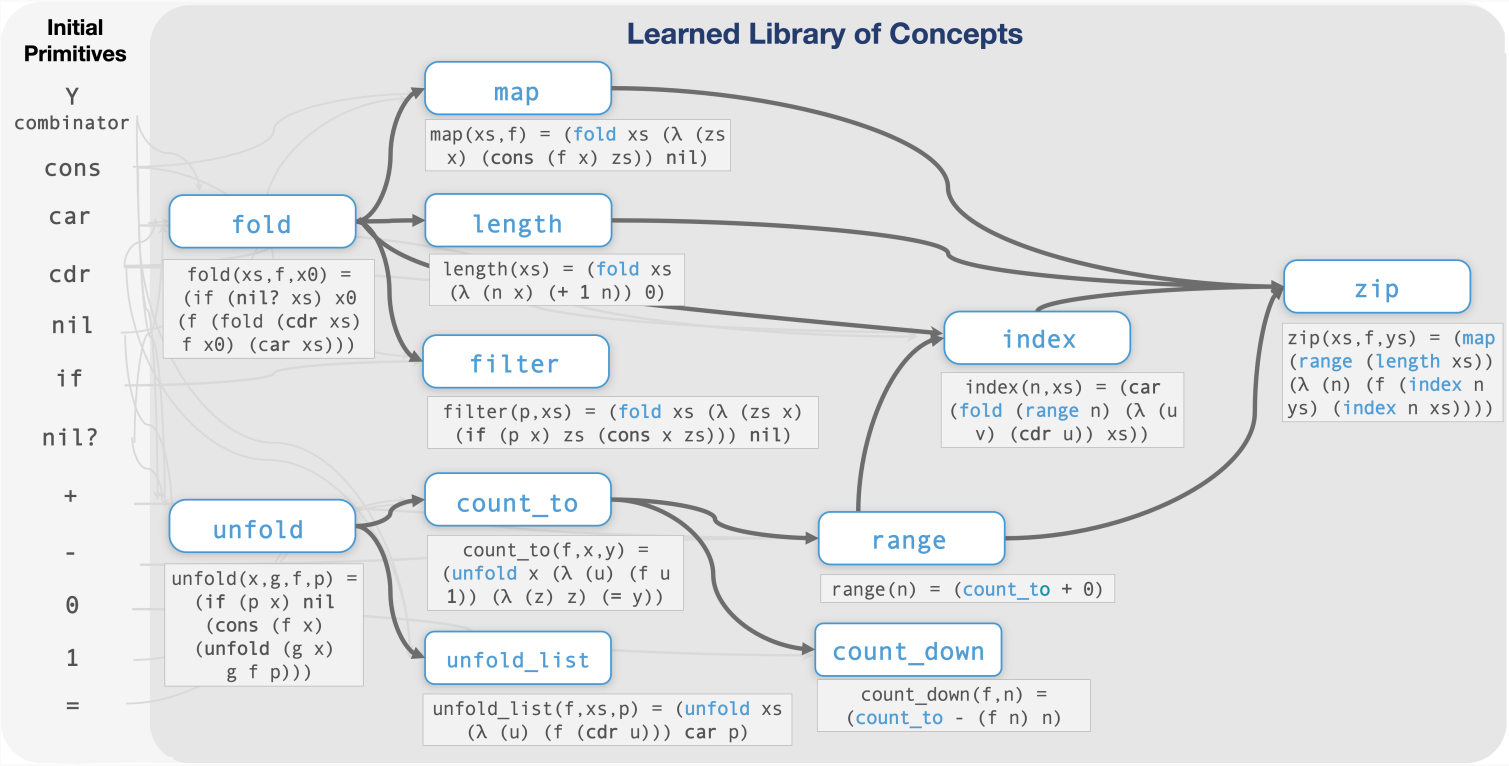
```
(zip A - B )
```



Origami Programming: Jeremy Gibbons, 2003

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

Growing a language for recursive programming



Recursive Programming Algorithms

Stutter

[■■] → [■■■■]
[■■■] → [■■■■■■]

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

Take every other

[■■■■] → [■■]
[■■■■■■] → [■■■]

```
(unfold_list cdr A nil?)
```

List lengths

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

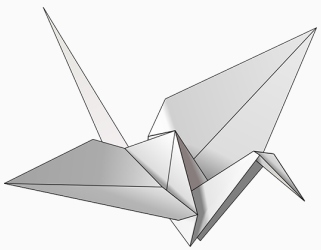
```
(map A length)
```

List differences

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

```
(zip A - B )
```

1 year of compute. 5 days on 64 CPUs.



Origami Programming: Jeremy Gibbons, 2003

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

Lessons

Programs as a knowledge representation for discovery problems

Lessons

Programs as a knowledge representation for discovery problems

Higher-level knowledge makes program synthesis tractable

Lessons

Programs as a knowledge representation for discovery problems

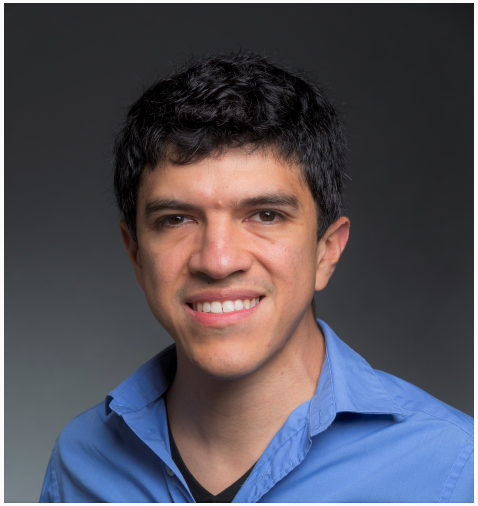
Higher-level knowledge makes program synthesis tractable

Higher-level knowledge makes these programs more interpretable

the end.

Josh Tenenbaum

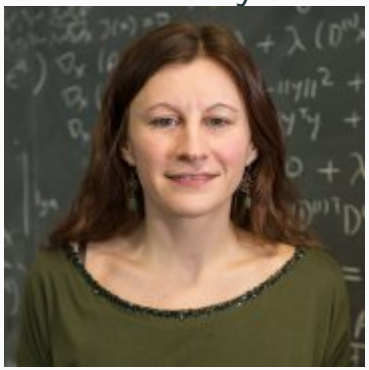
Armando Solar-Lezama



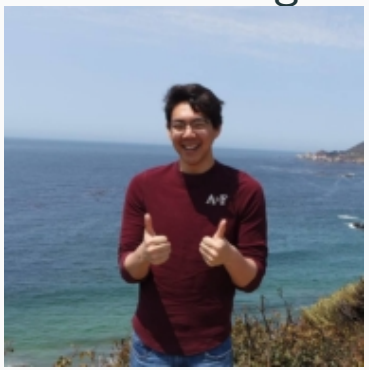
Evan Pu

Lucas Tian

Marta Kryven



Chris Yang



Ronald Alvarez

Max Nye

Cathy Wong

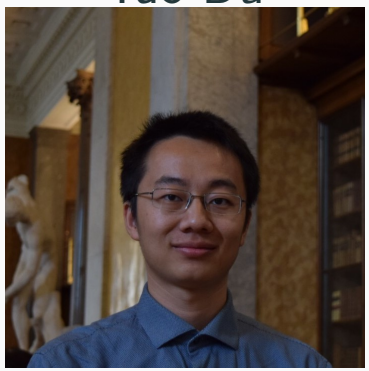
Kliment Serafimov

Eyal Dechter

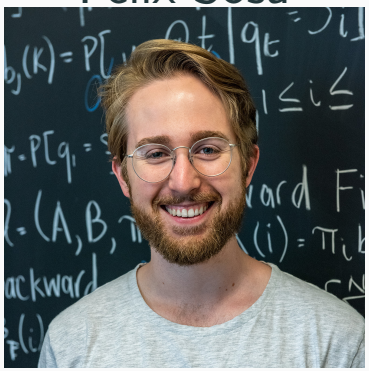
Mathias Sable-Meyer

Lucas Morales

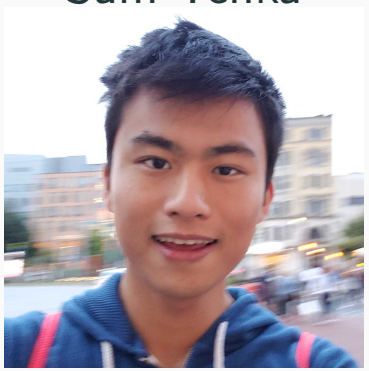
Tao Du



Felix Sosa



Sam Tenka



Julio Cortazar

thank you...

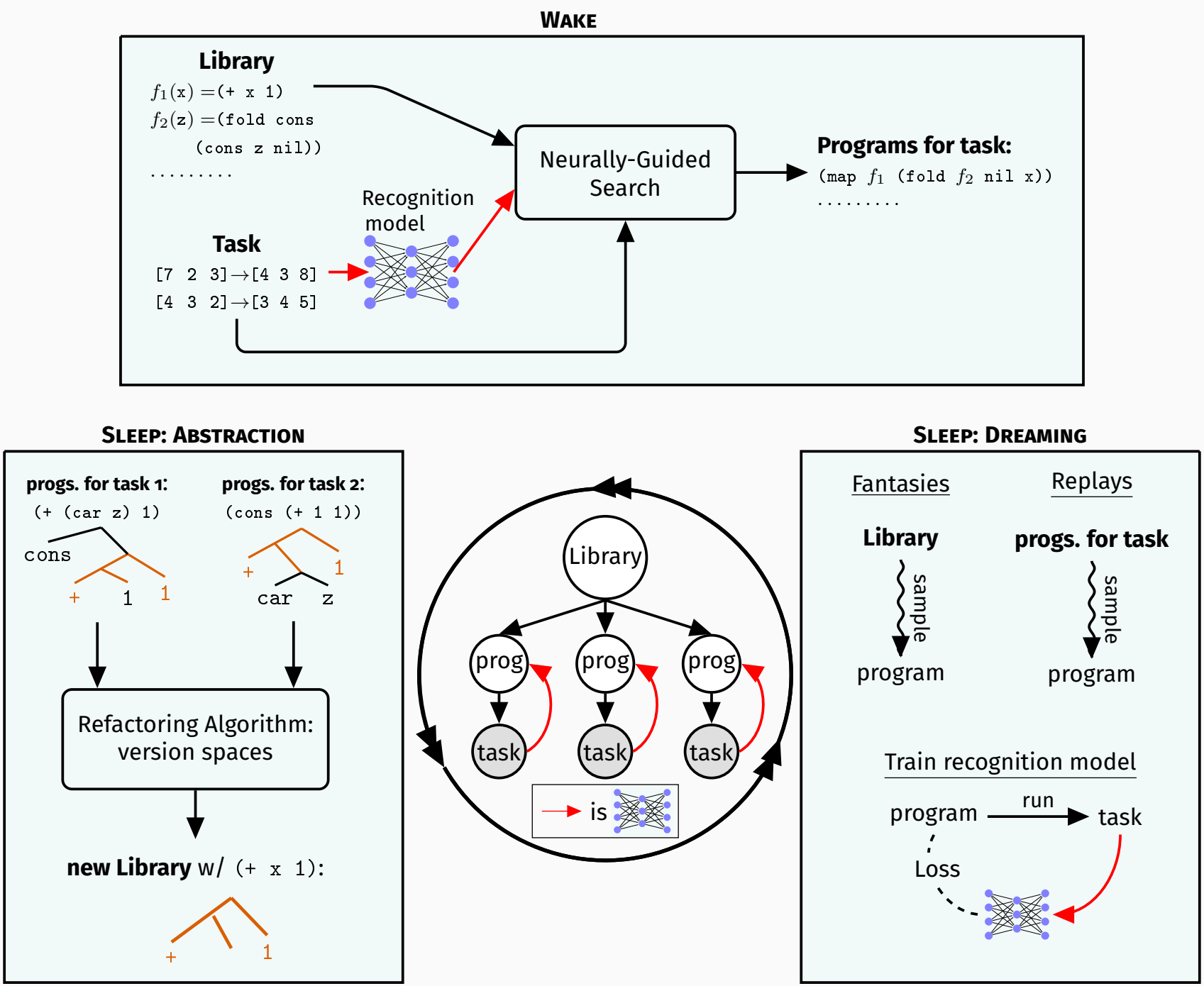
family

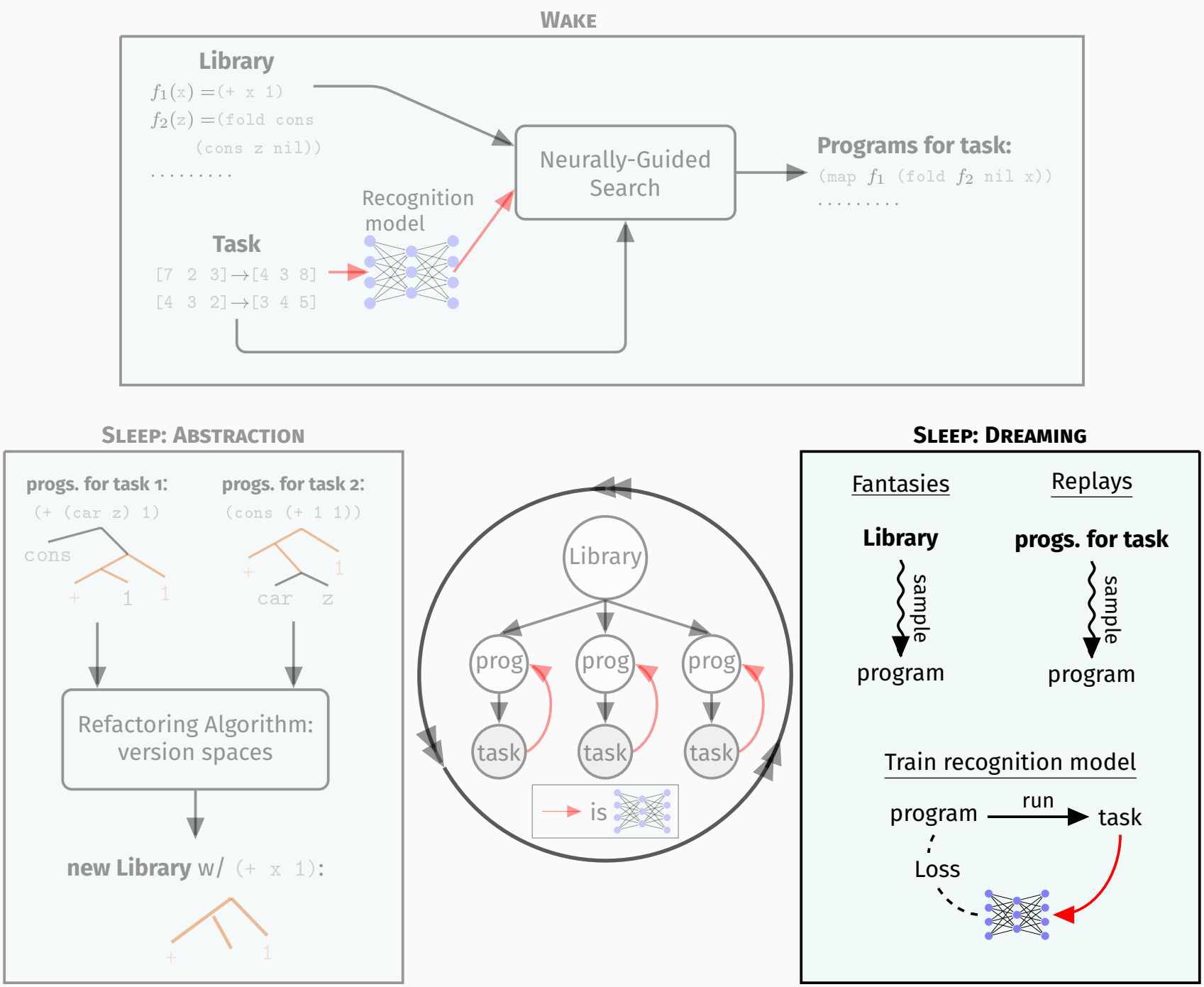



MIT Brain and Cognitive Sciences

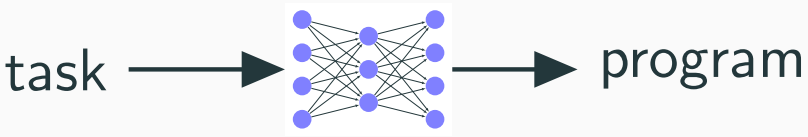
twitch viewers

the end.

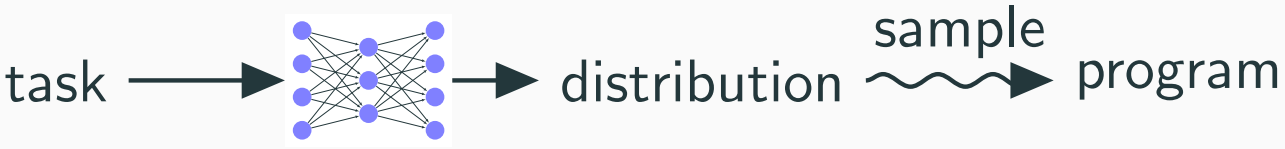

WAKE
Library
f_1(x) =(+ x 1)
f_2(z) =(fold cons
(cons z nil))
.....
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
Recognition model
Neurally-Guided Search
Programs for task:
(map f_1 (fold f_2 nil x))
.....
SLEEP: ABSTRACTION
progs. for task 1:
(+ (car z) 1)
cons
+
1
1
progs. for task 2:
(cons (+ 1 1))
+
1
car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


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

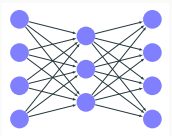
Neural recognition model guides search



Neural recognition model guides search



Neural recognition model guides search

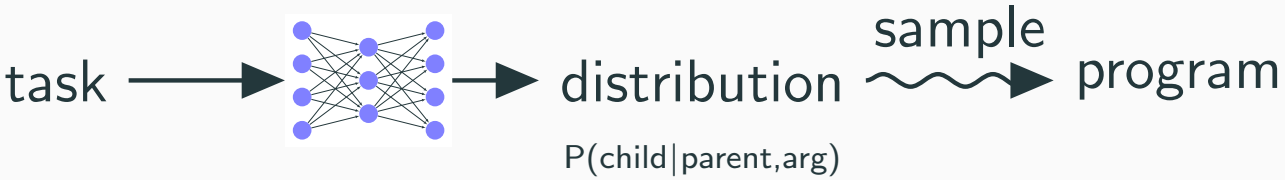


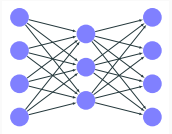
is a...

recurrent network (Devlin et al 2017)

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

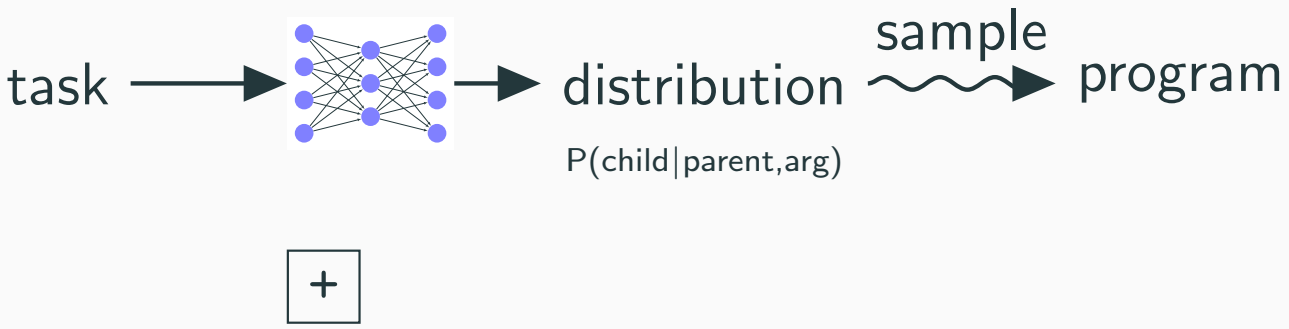
Neural recognition model guides search



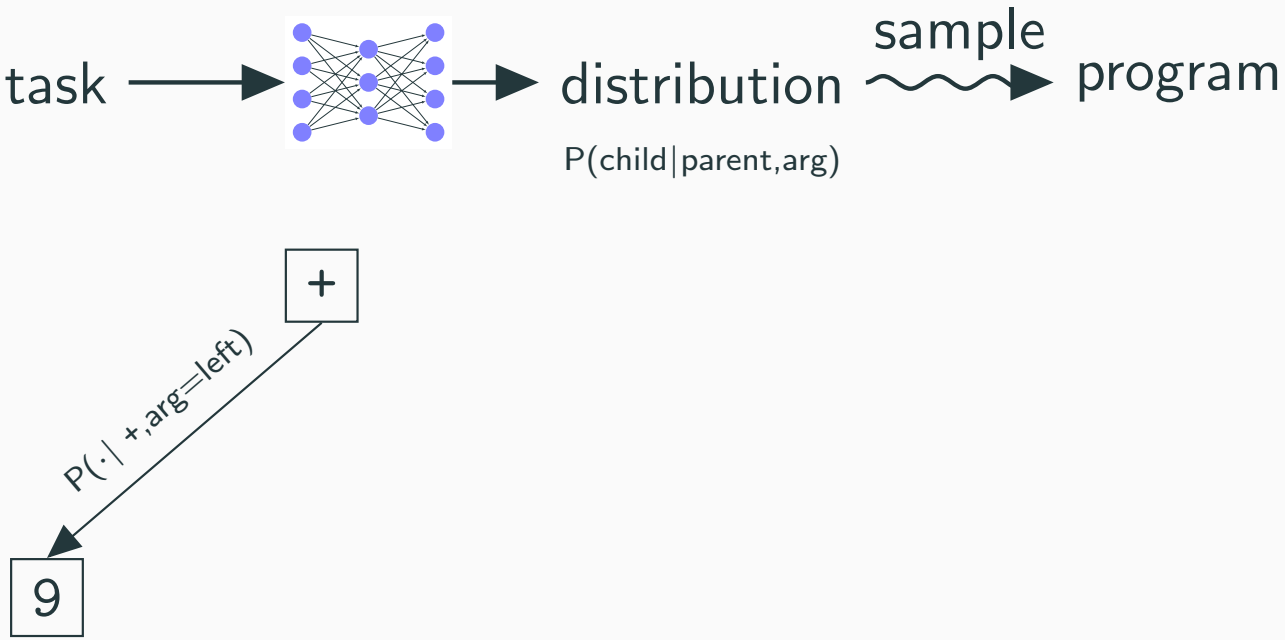


is a **"bigram" model over syntax trees**

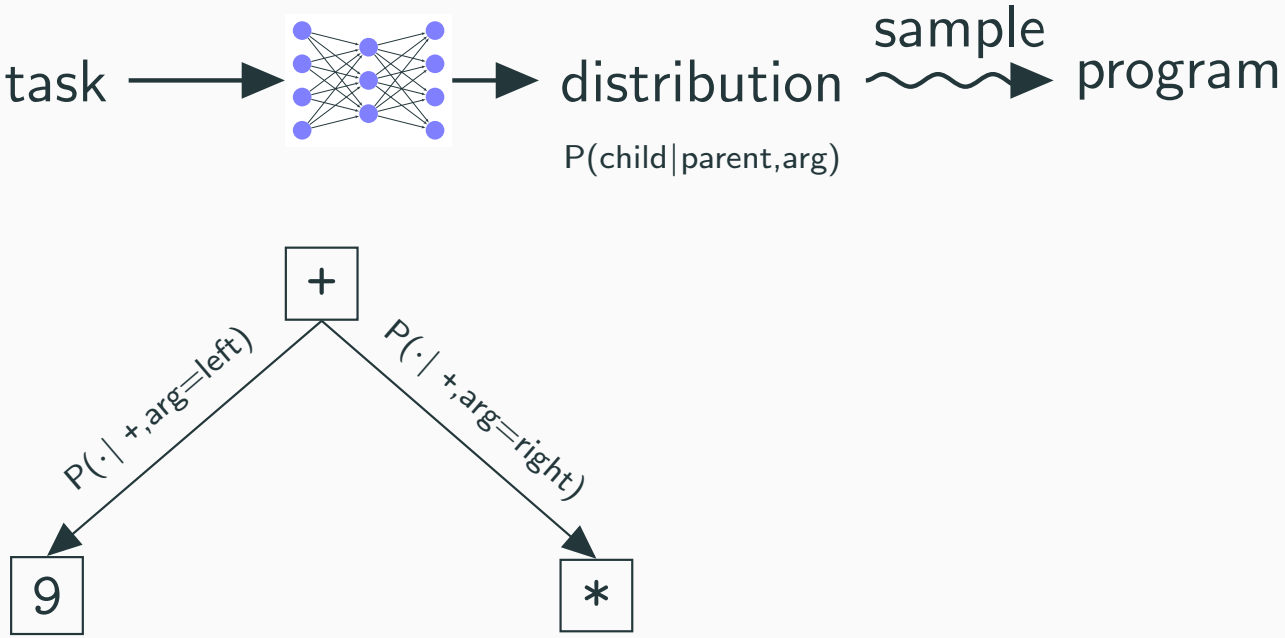
Neural recognition model guides search



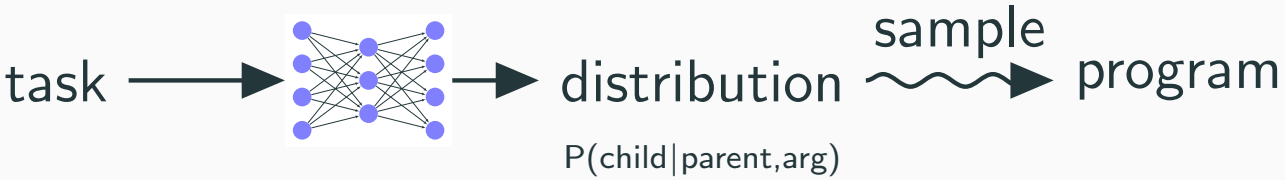
Neural recognition model guides search

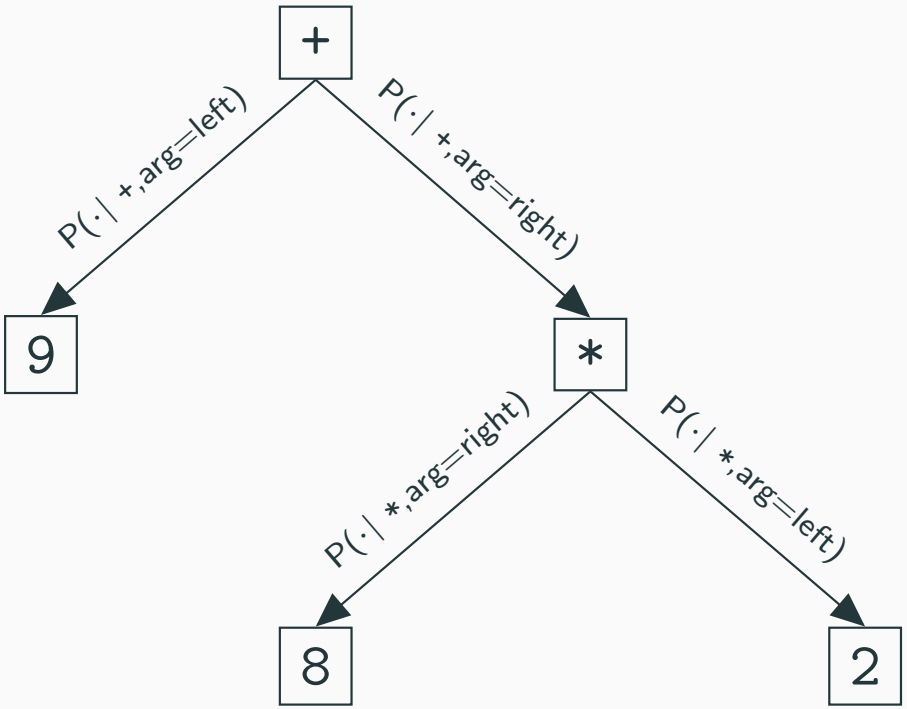


Neural recognition model guides search

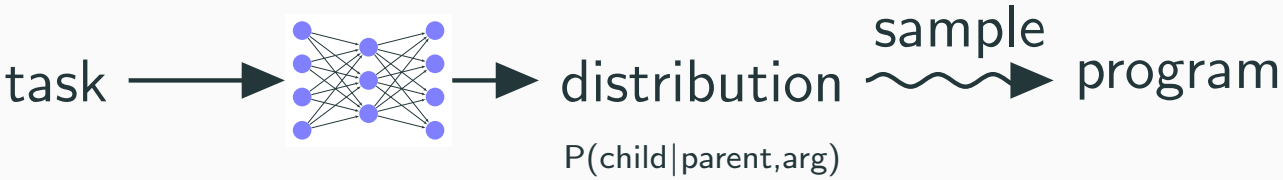


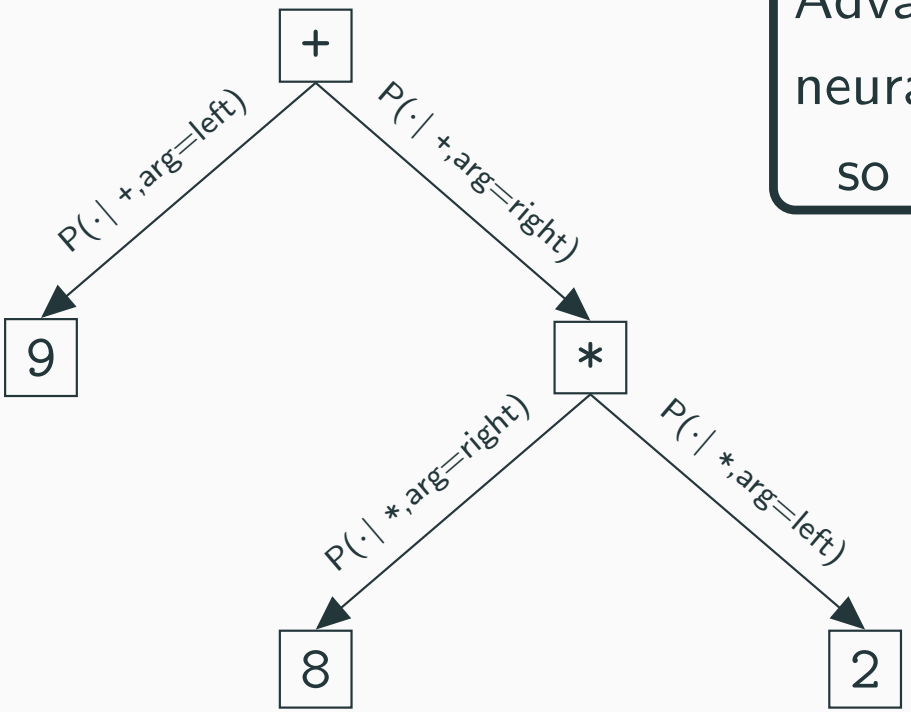
Neural recognition model guides search





Neural recognition model guides search



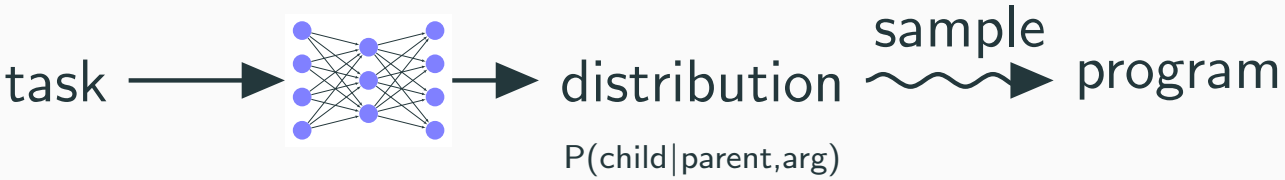


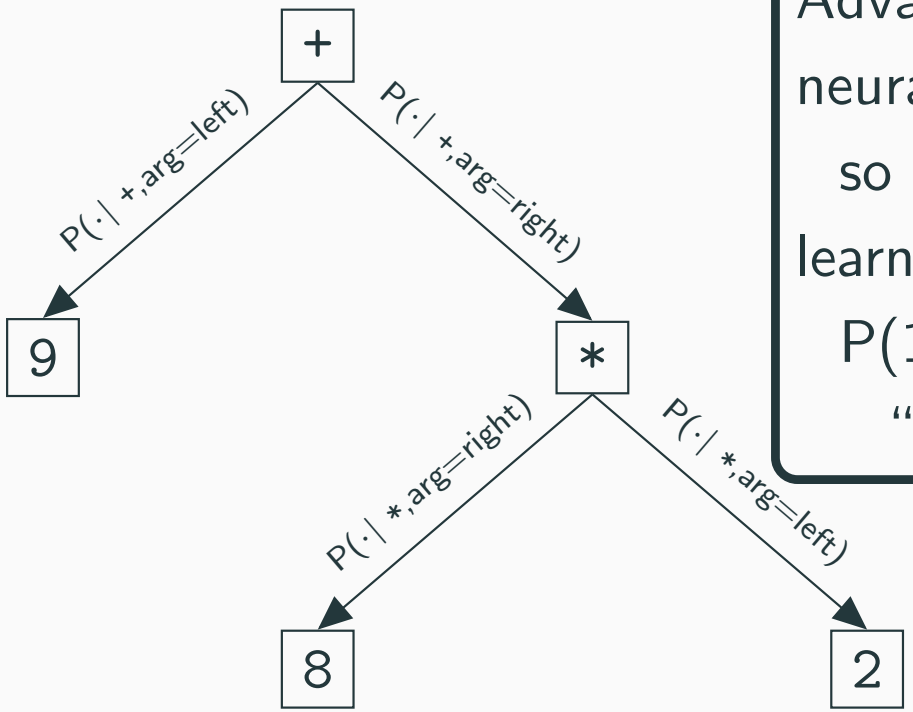
Advantages:

neural net runs once per task,

so CPU bottlenecks instead of GPU

Neural recognition model guides search





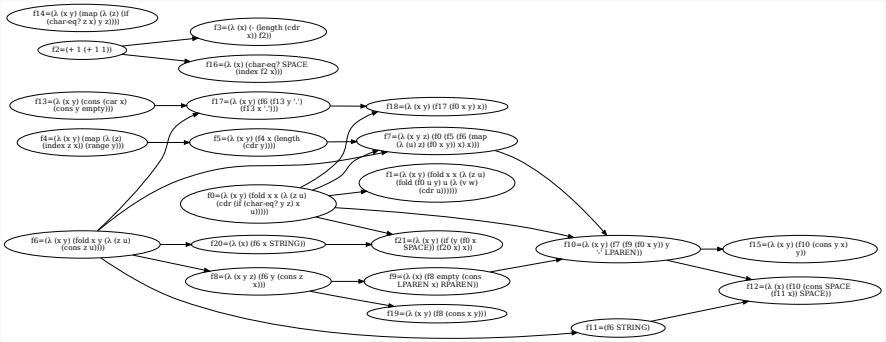
Advantages:

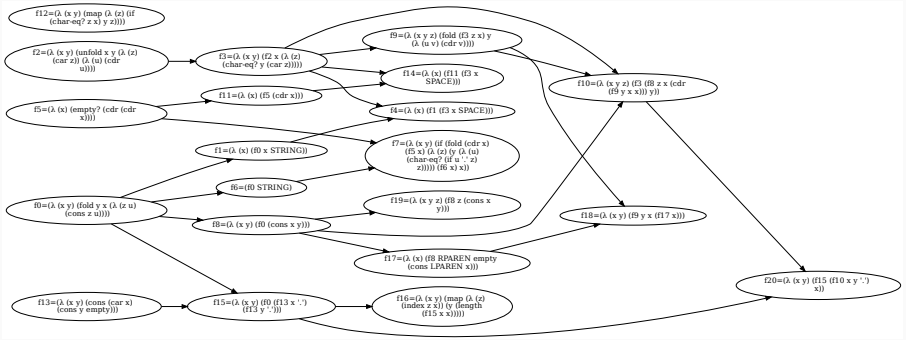
neural net runs once per task,
so CPU bottlenecks instead of GPU

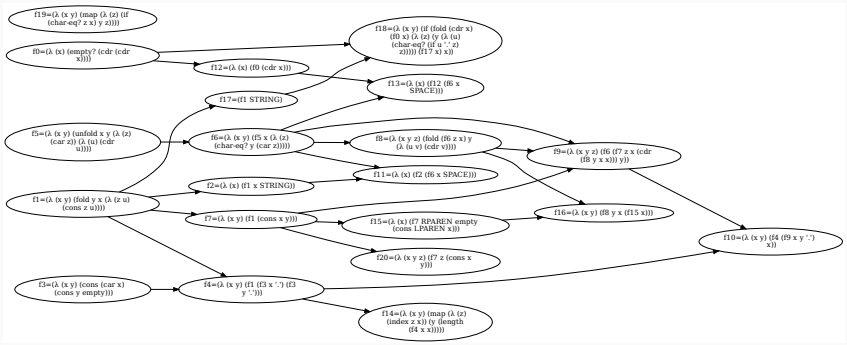
learns to break syntactic symmetries:
$P(1|*,\text{arg}=\text{left})=0.0$
"do not multiply by one"

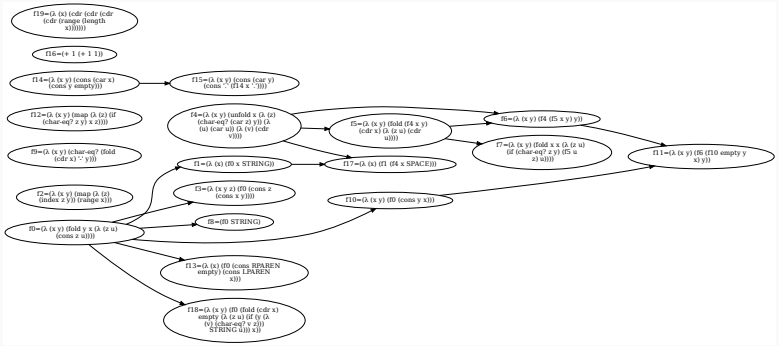
Library structure: Text Editing

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





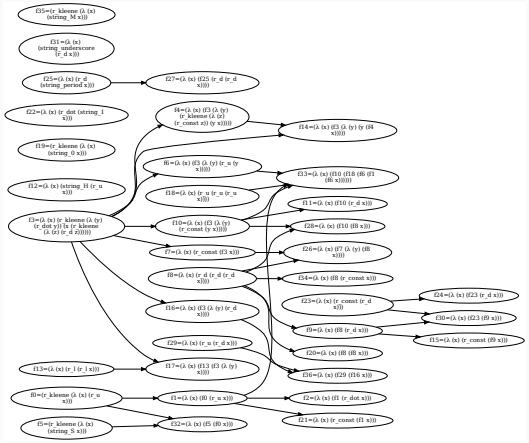


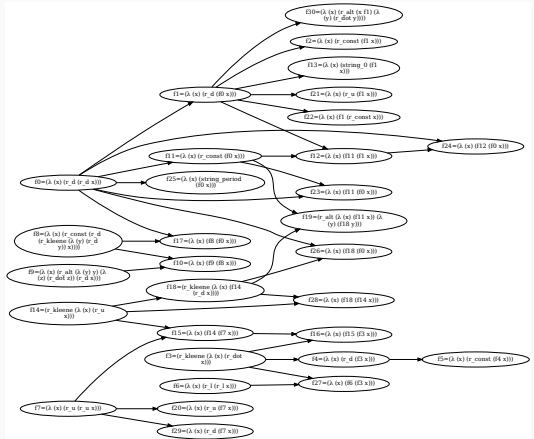


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. PLDI 2021.

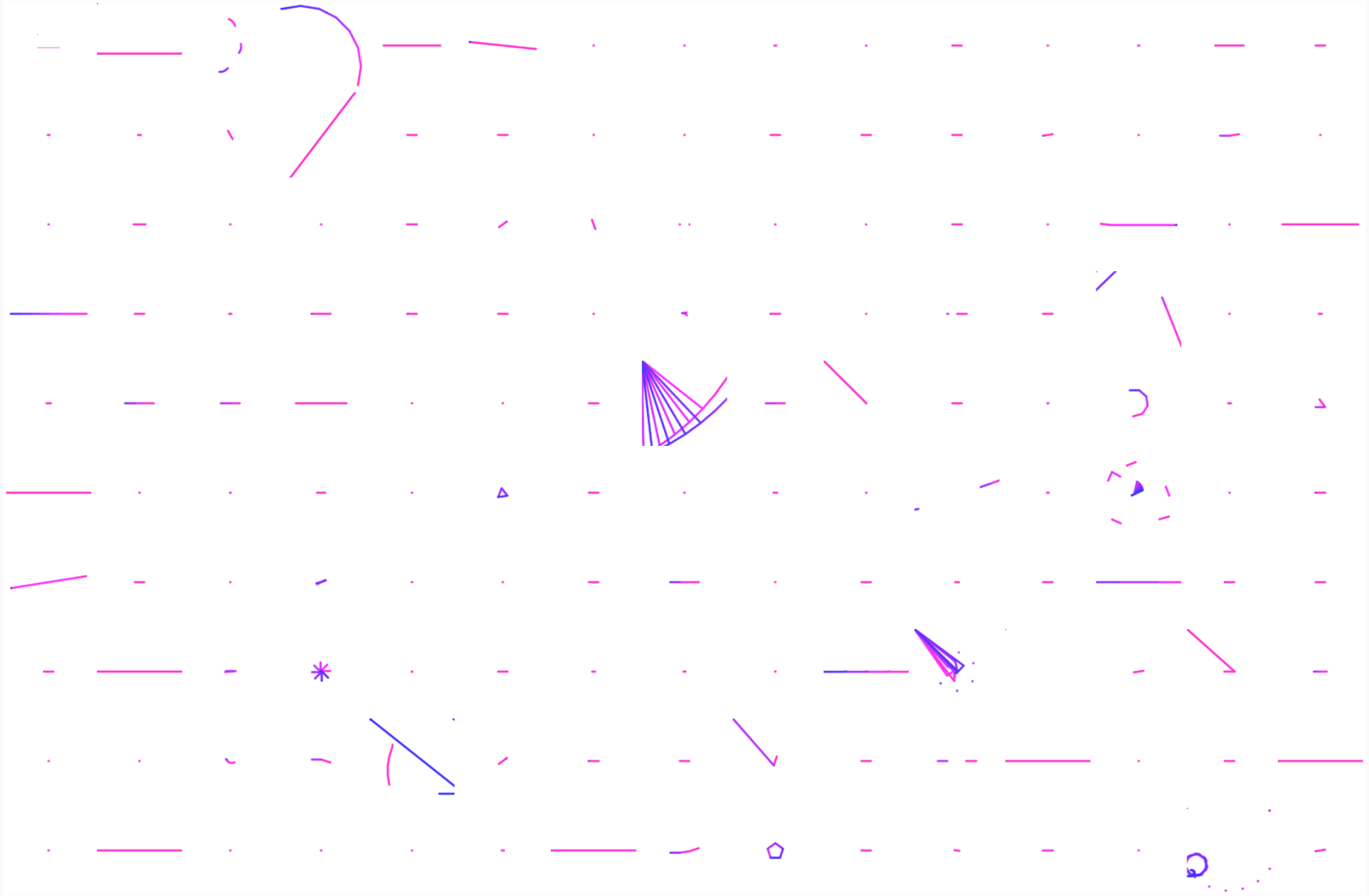
Library structure: Generating Text

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





150 random dreams before learning



150 random dreams after learning