Building Machines that Discover Generalizable, Interpretable Knowledge

Kevin Ellis

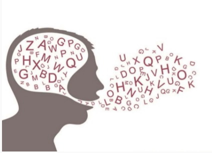
2020

MIT

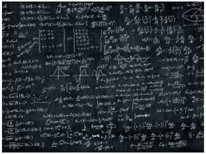
What computational problems are solved by intelligence?

an endless range of problems

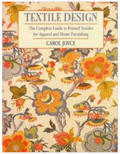
language



science

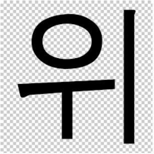


design



using new devices

writing new characters



coding

```
(MEMBER
 (LAMBDA (X L)
  (COND  ((NULL L) NIL)
         ((EQ X (FIRST L)) T)
         (T (MEMBER X (REST L))))))
```

Allen, Anatomy of Lisp, 1975

engineering

play

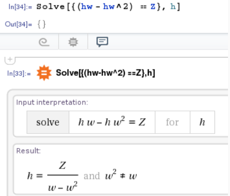
What computational frameworks can contribute to this picture?

Three AI traditions

What computational frameworks can contribute to this picture?

Three AI traditions

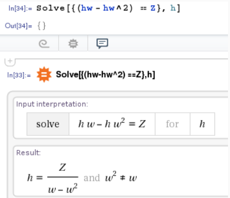
Symbolic



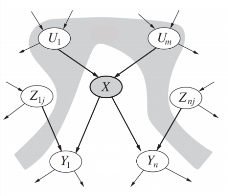
What computational frameworks can contribute to this picture?

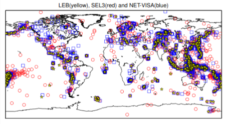
Three AI traditions

Symbolic



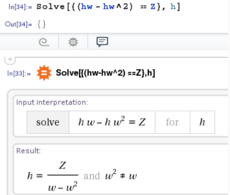
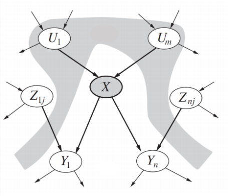
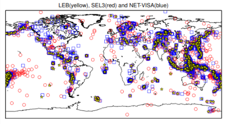
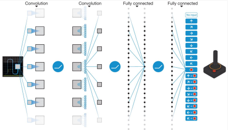
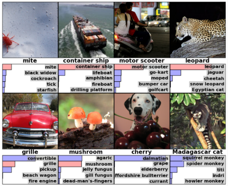
Probabilistic





What computational frameworks can contribute to this picture?

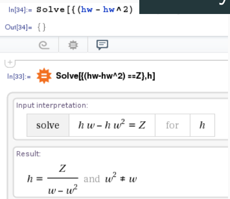
Three AI traditions

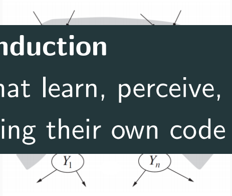
Symbolic	Probabilistic	Neural
 	 	 

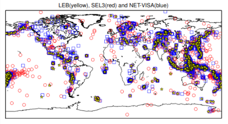
What computational frameworks can contribute to this picture?

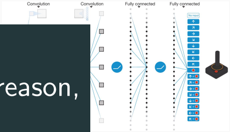
Three AI traditions

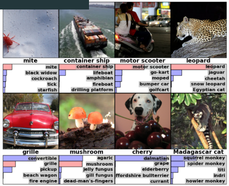
Symbolic	Probabilistic	Neural











Program induction

machines that learn, perceive, and reason,

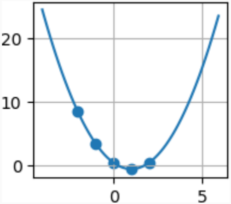
by writing their own code

Why program induction?

Why program induction?

strong generalization

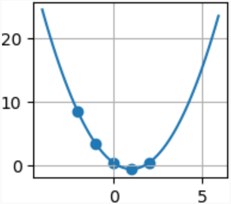
+data efficiency



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

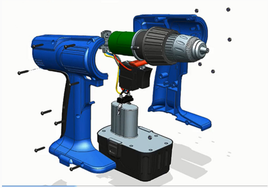
Why program induction?

strong generalization
+data efficiency

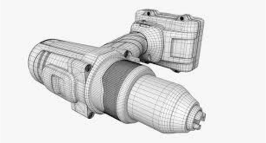


`f(x)=(x-1)**2 - 0.5`

interpretability

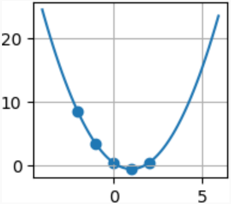


vs



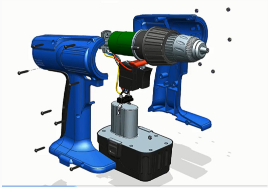
Why program induction?

strong generalization
+data efficiency

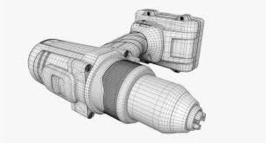


`f(x)=(x-1)**2 - 0.5`

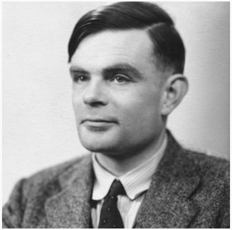
interpretability



vs

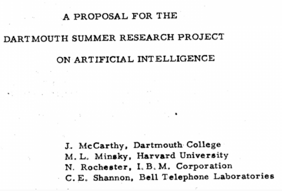


universal expressivity



Why didn't this old idea work?

Program induction goes back to the 1956 Dartmouth Workshop that founded the field of AI



A PROPOSAL FOR THE

DARTMOUTH SUMMER RESEARCH PROJECT

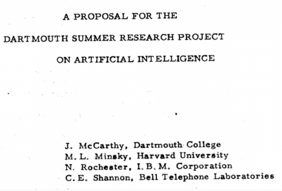
ON ARTIFICIAL INTELLIGENCE

J. McCarthy, Dartmouth College
M. L. Minsky, Harvard University
N. Rochester, I. B. M. Corporation
C. E. Shannon, Bell Telephone Laboratories

John McCarthy, Ray Solomonoff

Why didn't this old idea work?

Program induction goes back to the 1956 Dartmouth Workshop that founded the field of AI



A PROPOSAL FOR THE

DARTMOUTH SUMMER RESEARCH PROJECT

ON ARTIFICIAL INTELLIGENCE

J. McCarthy, Dartmouth College
M. L. Minsky, Harvard University
N. Rochester, I. B. M. Corporation
C. E. Shannon, Bell Telephone Laboratories

John McCarthy, Ray Solomonoff

main obstacle: combinatorial search is hard

Why try again?

better toolkits: neural+probabilistic+symbolic, and knowing how to combine them

Why try again?

better toolkits: neural+probabilistic+symbolic, and knowing how to combine them

maturing **program synthesis** techniques

Why try again?

better toolkits: neural+probabilistic+symbolic, and knowing how to combine them

maturing **program synthesis** techniques

better compute+parallel algorithms

A lesson from the AI winter

We need an on-ramp of practical, tractable problems:

semantic parsing [Liang et al. 2011; Zettlemoyer et al. 2007]

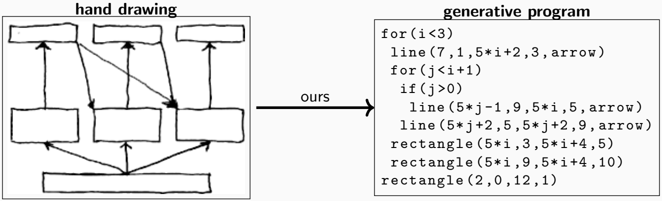
programming by examples [Gulwani 2011]

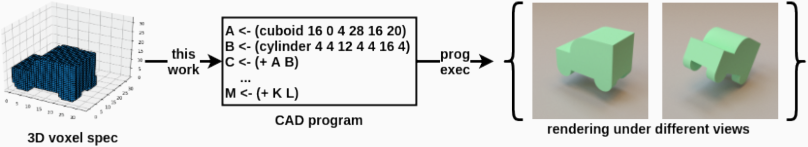
computer-aided-programming [Solar-Lezama 2008]

inverse procedural modeling [Kulkarni et al. 2015]

Perception, Synthesizing models, Learning-to-Learn

Theme #1: high-level visual understanding, pixels→programs

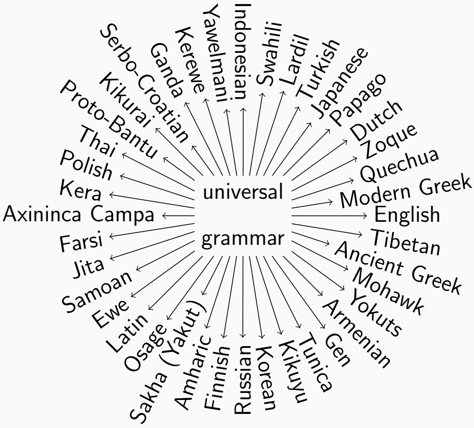




Perception, Synthesizing models, Learning-to-Learn

Theme #1: high-level visual understanding, pixels→programs

Theme #2: synthesizing human-understandable models



Perception, Synthesizing models, Learning-to-Learn

Theme #1: high-level visual understanding, pixels→programs

Theme #2: Synthesizing human-understandable models

Theme #3: learning to synthesize programs

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`

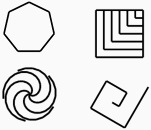
`(650) 555-2368`

Currency

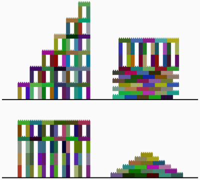
`$100.25`

`$4.50`

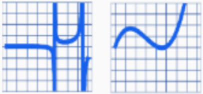
LOGO Graphics



Block Towers



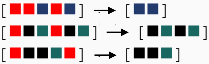
Symbolic Regression



$y = f(x)$

Recursive Programming

Filter Red



Physical Laws

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

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

Program Induction and perception

model discovery

learning to learn

Vision is more than knowing what is where

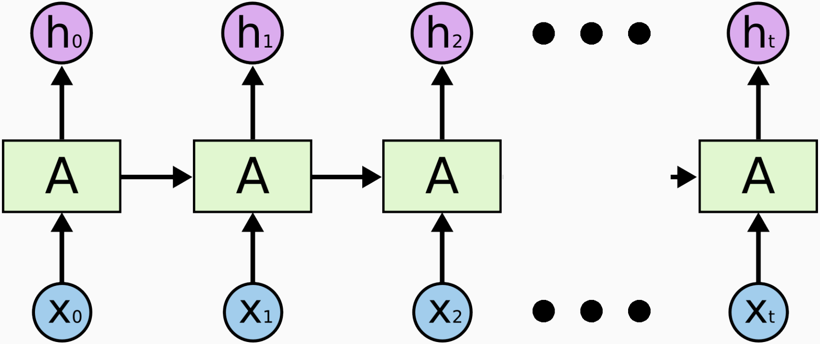
Can you visually extrapolate this aqueduct?

Vision is more than knowing what is where

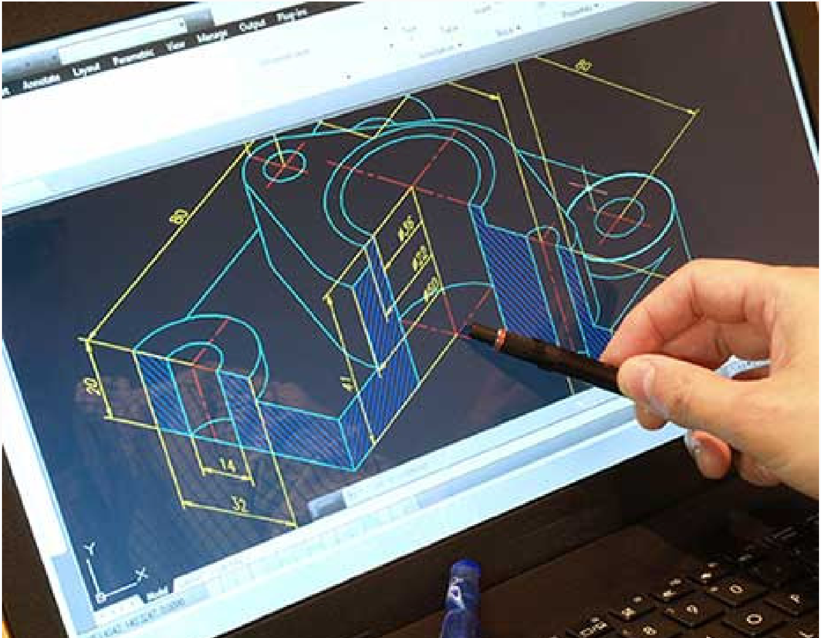
Can you visually extrapolate this aqueduct?

Vision is more than knowing what is where

Can you infer what goes in the ellipses?

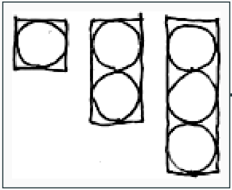


Vision is more than knowing what is where



Learning to infer graphics programs from hand-drawn images

hand drawing



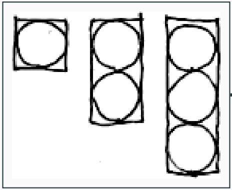
machine-generated program

```
for (0 <= i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (0 <= j < i + 1)
  circle(3*i+1,-2*j+5)
```

Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

Learning to infer graphics programs from hand-drawn images

hand drawing



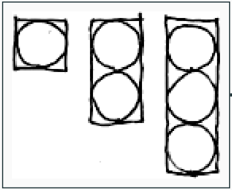
machine-generated program

```
for (0 <= i < 3 + 1)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (0 <= j < i + 1)
  circle(3*i+1,-2*j+5)
```

Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

Learning to infer graphics programs from hand-drawn images

hand drawing



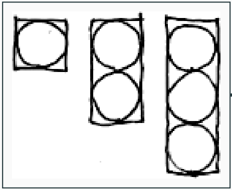
machine-generated program

```
for (0 <= i < 3 + 1)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (0 <= j < i + 1)
  circle(3*i+1,-2*j+5)
```

Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

Learning to infer graphics programs from hand-drawn images

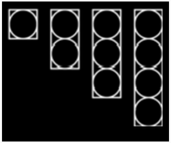
hand drawing



machine-generated program

```
for (0 <= i < 3 + 1)
  rectangle(3*i,-2*i+4,
            3*i+2,6)
  for (0 <= j < i + 1)
    circle(3*i+1,-2*j+5)
```

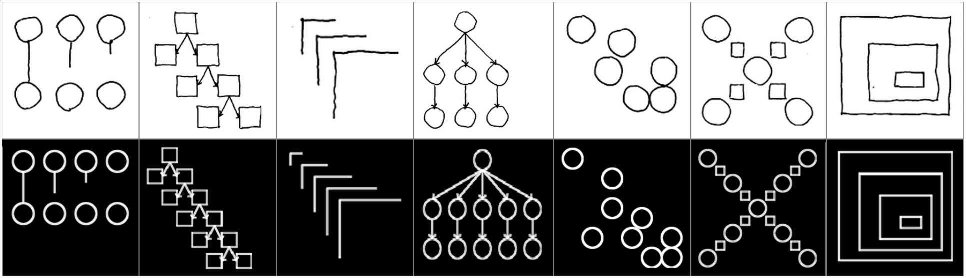
autogenerated extrapolation



one-shot generalization / extrapolation

Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

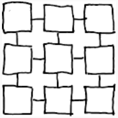
Extrapolation from a single image



Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

How to infer graphics programs from hand-drawn images

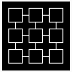
Image (Observed)



render

Drawing Commands (Latent)

```
line, line,
rectangle,
line, ...
```



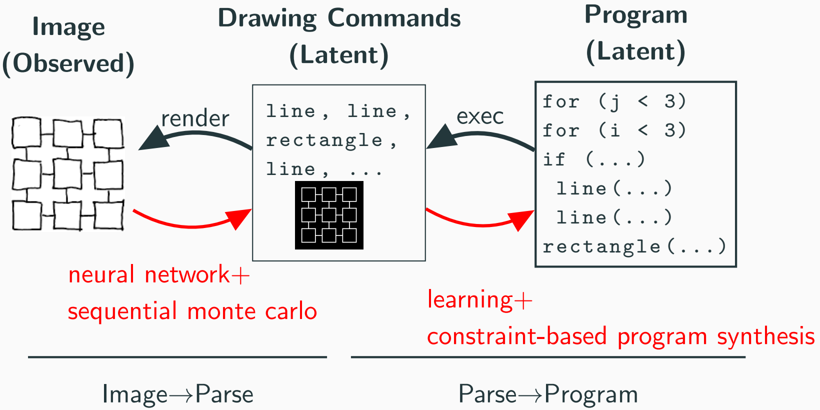
exec

Program (Latent)

```
for (j < 3)
for (i < 3)
if (...)
 line(...)
 line(...)
rectangle(...)
```

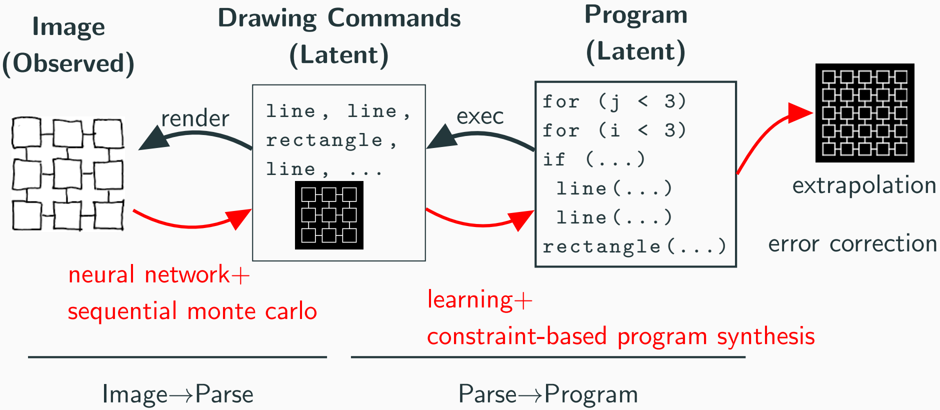
Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

How to infer graphics programs from hand-drawn images



Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

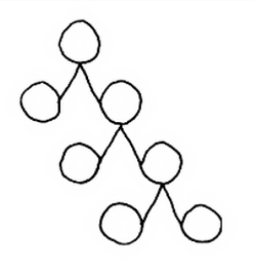
How to infer graphics programs from hand-drawn images



Ellis, Ritchie, Solar-Lezama, Tenenbaum. NeurIPS 2018.

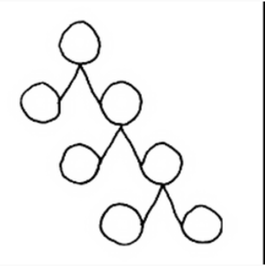
Top-down influences on perception

drawing

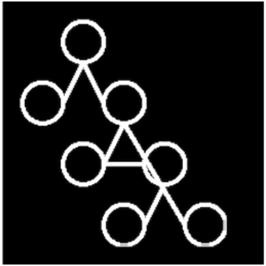


Top-down influences on perception

drawing

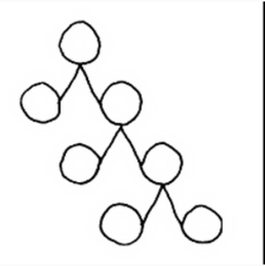


bottom-up neural net

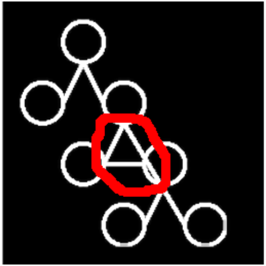


Top-down influences on perception

drawing

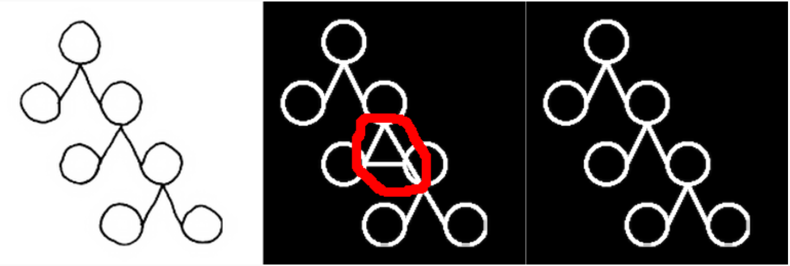


bottom-up neural net



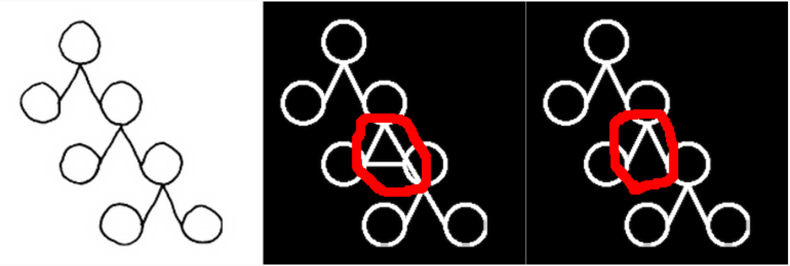
Top-down influences on perception

drawing | bottom-up neural net | w/ top-down program bias



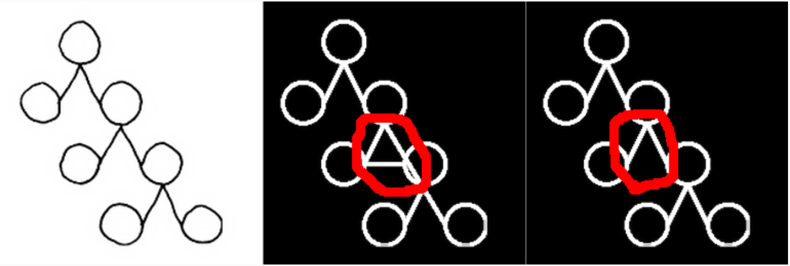
Top-down influences on perception

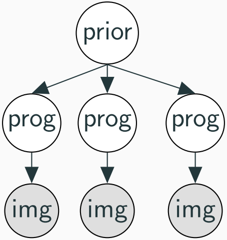
drawing | bottom-up neural net | w/ top-down program bias



Top-down influences on perception

drawing | bottom-up neural net | w/ top-down program bias



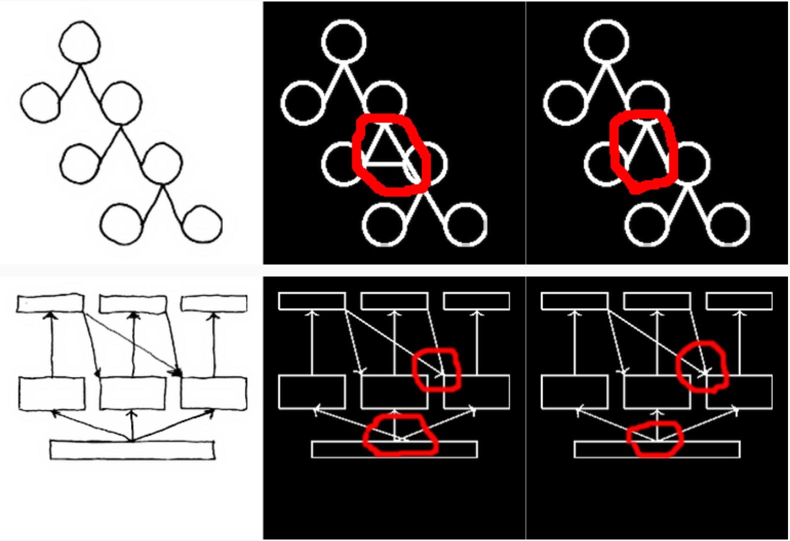


predicted program =

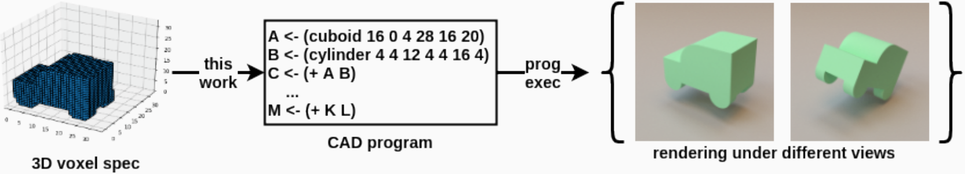
$\arg\max_{\text{progs}} \mathbb{P}\left[\text{img}|\text{prog}\right] \mathbb{P}\left[\text{prog}|\text{prior}\right]$

Top-down influences on perception

drawing | bottom-up neural net | w/ top-down program bias

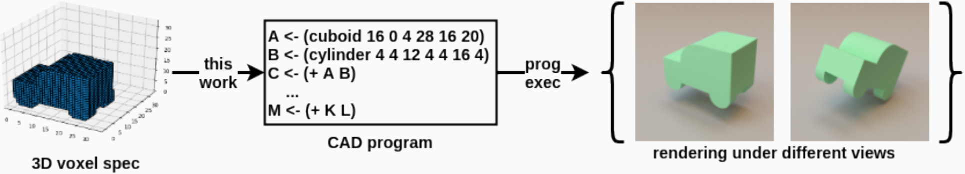


3D program induction



Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.
*equal contribution

3D program induction



Challenge: combinatorial search!

Branching factor: > 1.3 million per line of code, ≈ 20 lines of code

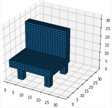
search space size: $(1.3 \text{ million})^{20} \approx 10^{122}$ programs

Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.

*equal contribution

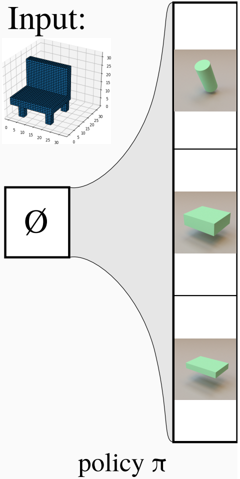
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]

Input:

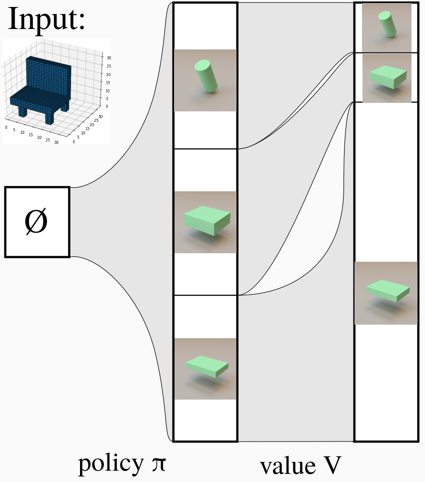


Ø

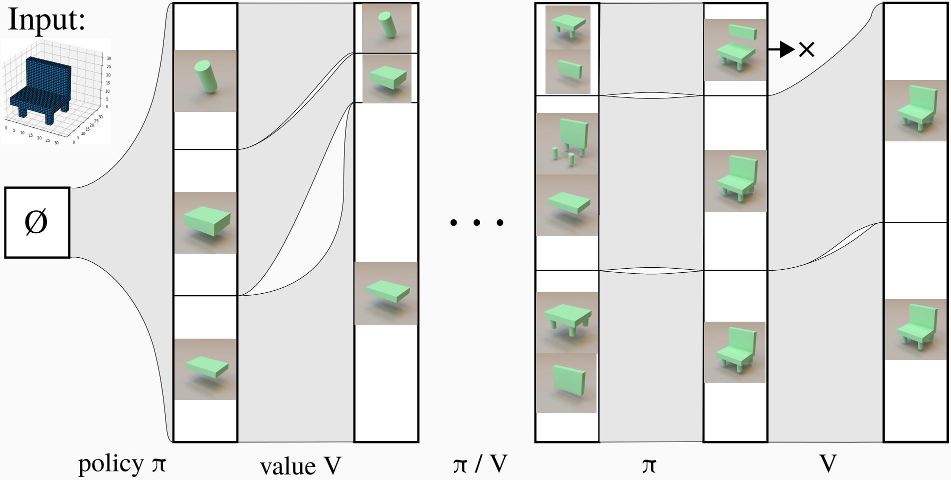
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]



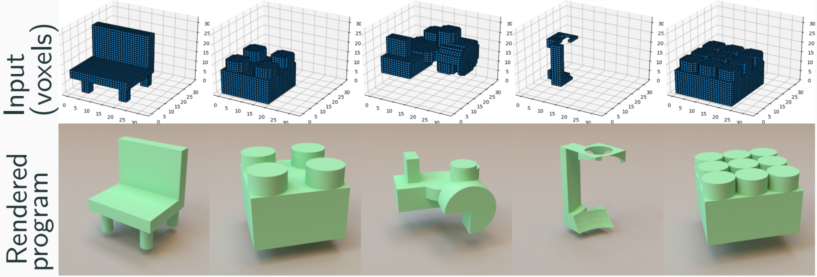
Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]



Solution: stochastic **tree search** + learn **policy** that writes code + learn **value** function that assesses execution of program so far; analogous to **AlphaGo** [Silver et al. 2016]

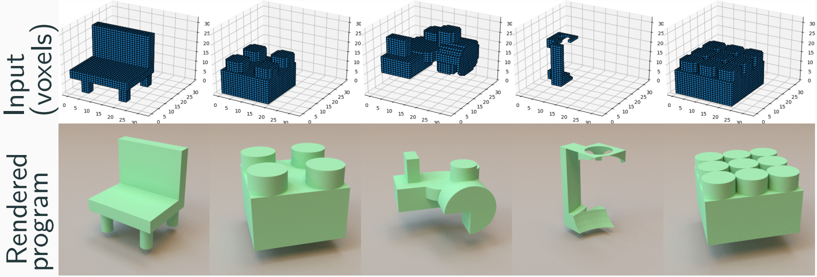


3D program induction



Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.
*equal contribution

3D program induction



same architecture learns to synthesize text editing programs (FlashFill, Gulwani 2012)

Ellis*, Nye*, Pu*, Sosa*, Tenenbaum, Solar-Lezama. NeurIPS 2019.
*equal contribution

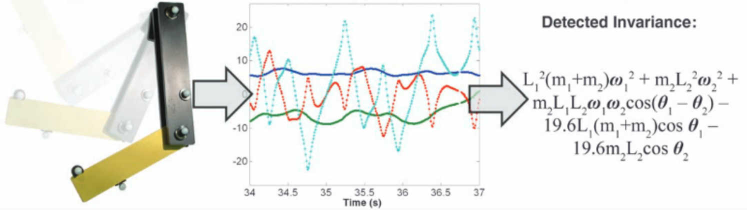
Lessons

The inductive bias from a programming language gives extrapolation, or strong generalization

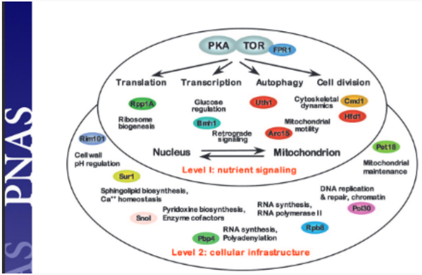
Combine the best of different techniques: neural nets for perception and pattern recognition, symbols for reasoning, Bayesian methods for uncertainty

Program Induction and perception
model discovery
learning to learn

Scientific discovery



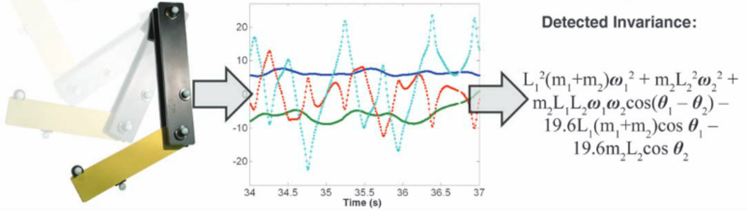
Schmidt & Lipson: "Distilling Free-Form Natural Laws from Experimental Data"



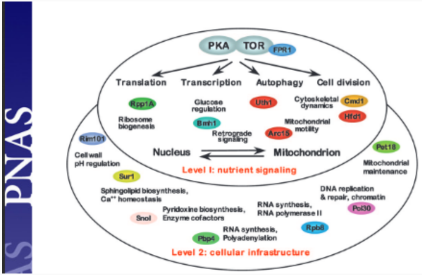
Lezon et al. 2006

inferring genetic interaction networks

Scientific discovery



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



THE APPLICATION OF ALGORITHMIC PROBABILITY TO PROBLEMS IN ARTIFICIAL INTELLIGENCE

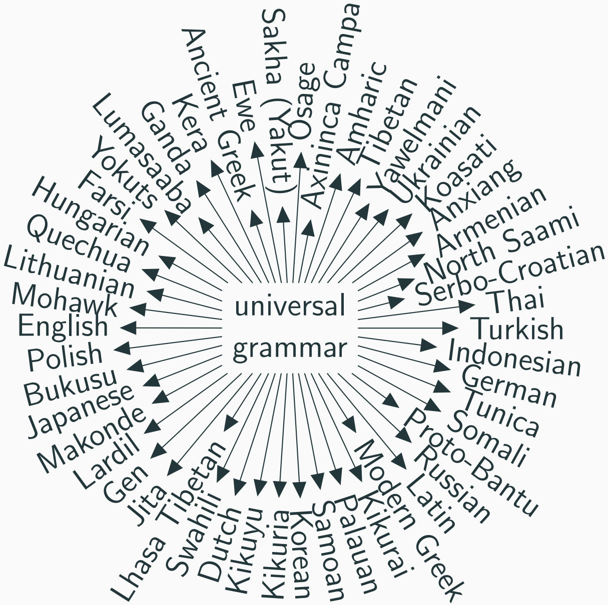
Ray J. Solomonoff 1986

This operation corresponds to Kepler's laws summarizing and compressing Tycho Brahe's empirical data on planetary motion. Algorithmic Complexity theory has this ability to synthesize, to find general laws in masses of unorganized and partially organized knowledge. It is in this area that its greatest value for A.I. lies.

Lezon et al. 2006

inferring genetic interaction networks

Discovering human-understandable models of language



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

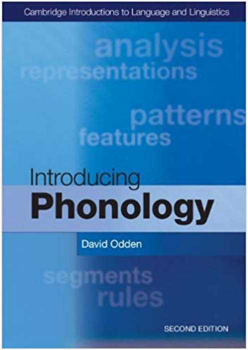
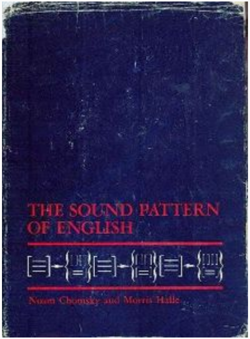
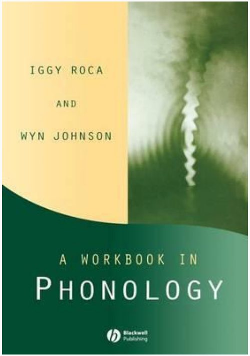
Serbo-Croatian:

	masculine	feminine	stem (unobserved)
"rich"	bogat	bogata	bogat
"mild"	blag	blaga	blag
"green"	zelen	zelena	zelen
"clear"	**yasan**	yasna	yasn

add "a" to stem to make feminine

insert "a" between two word-final consonants

$\varnothing \rightarrow \text{a} \;/\; \text{C\_C\#}$


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*	
aɣa	aɣalar	aɣalɨɨn	‘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’
oɣo	oɣolor	oɣoluun	‘child’
χopto	χoptolor	χoptoluun	‘gull’
börö	börölör	börölüün	‘wolf’
tɨal	tɨallar	tɨallɨɨn	‘wind’
ɨal	ɨallar	ɨallɨɨn	‘neighbor’
kuul	kuullar	kuulluun	‘sack’
at	attar	attɨɨn	‘horse’
balɨk	balɨktar	balɨktɨɨn	‘fish’
ɨskaap	ɨskaaptar	ɨskaaptɨɨn	‘cabinet’
oɣus	oɣustar	oɣustuun	‘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’
eǰiiy	eǰiiyder	eǰiiydiin	‘elder sister’
tomtor	tomtordor	tomtorduun	‘knob’
moɣotoy	moɣotoydor	moɣotoyduun	‘chipmunk’
kötör	kötördör	kötördüün	‘bird’
bölköy	bölköydör	bölköydüün	‘islet’
χatɨŋ	χatɨŋnar	χatɨŋnɨɨn	‘birch’
aan	aannar	aannɨɨn	‘door’
tiiŋ	tiiŋner	tiiŋniin	‘squirrel’
sordoŋ	sordoŋnor	sordoŋnuun	‘pike’
olom	olomnor	olomnuun	‘ford’
oron	oronnor	oronnuun	‘bed’
bödöŋ	bödöŋnör	bödöŋnüün	‘strong one’

Noun	*Partitive*	*Comparative*	*Ablative*	
aɣa	aɣata	aɣataaɣar	aɣattan	‘father’
paarta	paartata	paartataaɣar	paartattan	‘school desk’
tɨa	tɨata	tɨataaɣar	tɨattan	‘forest’
kinige	kinigete	kinigeteeɣer	kinigetten	‘book’
ǰie	ǰiete	ǰieteeɣer	ǰietten	‘house’
iye	iyete	iyeteeɣer	iyetten	‘mother’
kini	kinite	kiniteeɣer	kinitten	‘3rd person’
bie	biete	bieteeɣer	bietten	‘mare’
oɣo	oɣoto	oɣotooɣor	oɣotton	‘child’
χopto	χoptoto	χoptotooɣor	χoptotton	‘gull’
börö	börötö	börötööɣör	böröttön	‘wolf’
tɨal	tɨalla	tɨallaaɣar	tɨaltan	‘wind’
ɨal	ɨalla	ɨallaaɣar	ɨaltan	‘neighbor’
kuul	kuulla	kuullaaɣar	kuultan	‘sack’
moχsoɣol	moχsoɣollo	moχsoɣollooɣor	moχsoɣolton	‘falcon’
at	atta	attaaɣar	attan	‘horse’
balɨk	balɨkta	balɨktaaɣar	balɨktan	‘fish’
ɨskaap	ɨskaapta	ɨskaaptaaɣar	ɨskaaptan	‘cabinet’
oɣus	oɣusta	oɣustaaɣar	oɣustan	‘bull’
kus	kusta	kustaaɣar	kustan	‘duck’
tünnük	tünnükte	tünnükteeɣer	tünnükten	‘window’

Turkic Sakha (Yakut)

observed data

	Singular	Plural
Bed	oron	oron<u>nor</u>
Mare	bie	bie<u>ler</u>
Cabinet	ɨskaap	ɨskaap<u>tar</u>

138 total examples

Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2020.

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. 2020.

Turkic Sakha (Yakut)

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] $[\text{ -low }]_0$_
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. 2020.

Turkic Sakha (Yakut)

grammar (unobserved)

SINGULAR→stem
PLURAL→stem+lar

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

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

r_3: V $\rightarrow$ [+rounded] / [+rounded] [-low $]_0$_
r_4: [+continuant -high] $\rightarrow$ [-rounded] / u C_0_
“harmonize” round vowels like “u”, “o”

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

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

stems (unobserved)

BED : oron
MARE : bie
CABINET : ɨskaap

observed data

	SINGULAR	PLURAL
BED	oron	oronnor
MARE	bie	bieler
CABINET	ɨskaap	ɨskaaptar

138 total examples

Ellis, Albright, Solar-Lezama, Tenenbaum, O’Donnell. 2020.

Turkic Sakha (Yakut)

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"

stems (unobserved)

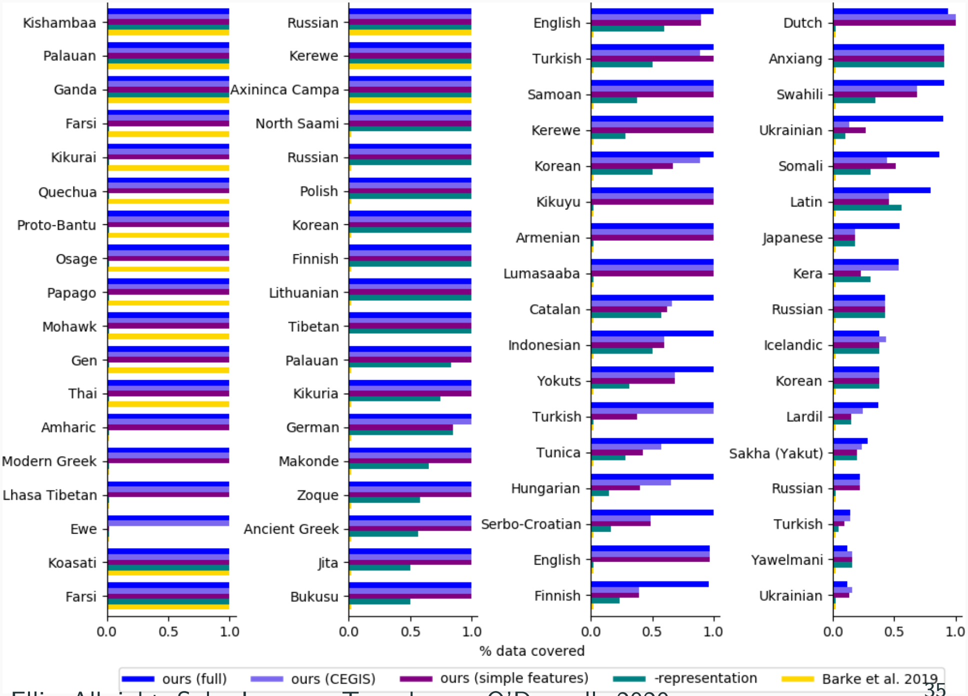
CABINET : ɨskaap
BED : oron
MARE : bie
RIVER : örus

observed data

CABINETS→ɨskaap+lar→ɨskaaplar $\xrightarrow{r_1}$ ɨskaapdar $\xrightarrow{r_2}$ ɨskaaptar
BEDS→oron+lar→oronlar $\xrightarrow{r_1}$ orondar $\xrightarrow{r_3}$ orondor $\xrightarrow{r_6}$ oronnor
MARES→bie+lar→bielar $\xrightarrow{r_5}$ bieler
RIVER (ASSOC)→örus+lɨɨn→öruslɨɨn $\xrightarrow{r_1}$ örusdɨɨn $\xrightarrow{r_2}$
örustɨɨn $\xrightarrow{r_3}$ örustuun $\xrightarrow{r_5}$ [örüstüün]

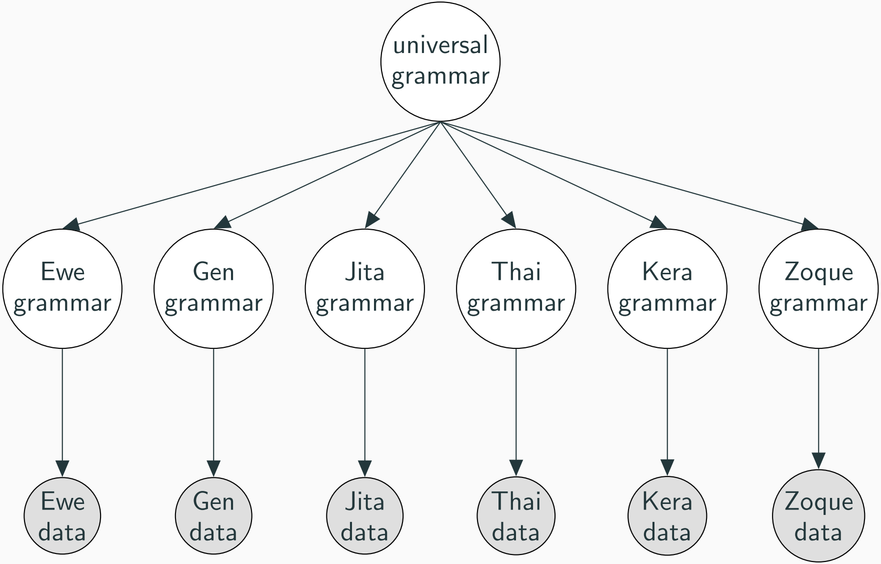
Kishambaa	Russian	English	Dutch
Palauan	Kerewe	Turkish	Anxiang
Ganda	Axininca Campa	Samoan	Swahili
Farsi	North Saami	Kerewe	Ukrainian
Kikurai	Russian	Korean	Somali
Quechua	Polish	Kikuyu	Latin
Proto-Bantu	Korean	Armenian	Japanese
Osage	Finnish	Lumasaaba	Kera
Papago	Lithuanian	Catalan	Russian
Mohawk	Tibetan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

Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2020.



Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2020.

Distilling higher-level knowledge



Ellis, Albright, Solar-Lezama, Tenenbaum, O'Donnell. 2020.

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

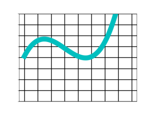
Goal: acquire domain-specific knowledge needed to induce a class of programs

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

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)



→ Learned program inducer → program: $f(x) = 0.3x^3 + 1.1x^2 - 2x + 0.6$

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]
...

Ellis, Morales, Sable-Meyer, Solar-Lezama, Tenenbaum. NeurIPS 2018.
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

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. 2020.

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. 2020.

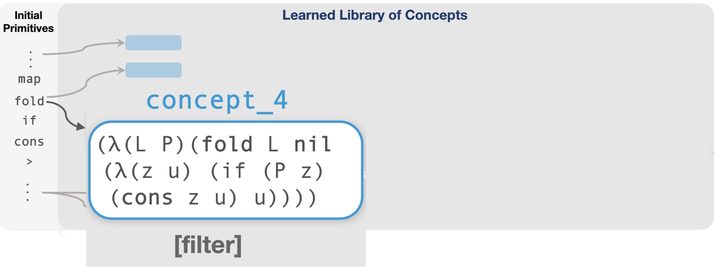
Library learning

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. 2020.

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. 2020.

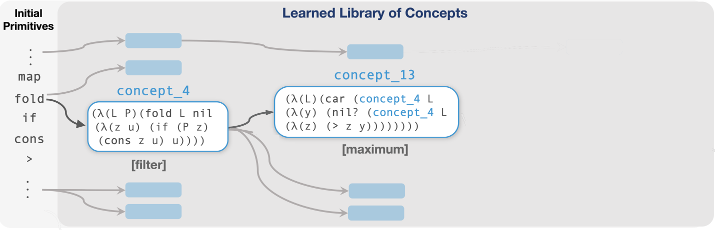
Library learning

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. 2020.

Library learning

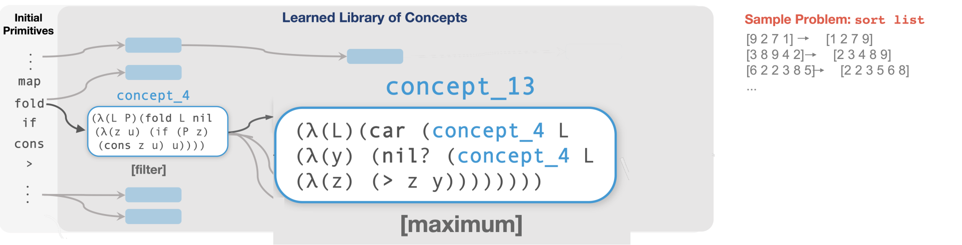


Sample Problem: `sort list`

[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]
...

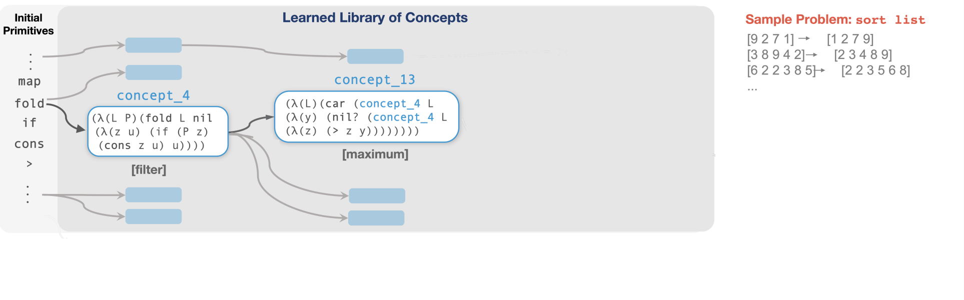
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library learning



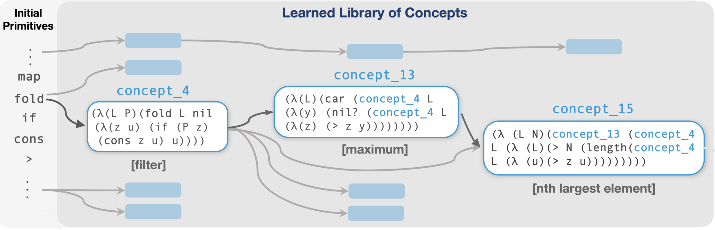
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library learning

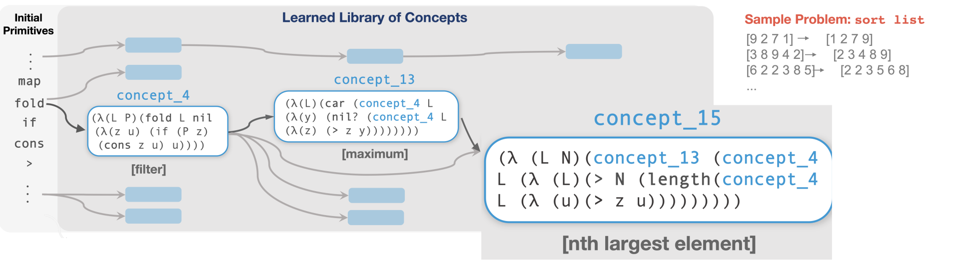


Sample Problem: `sort list`

[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]
...

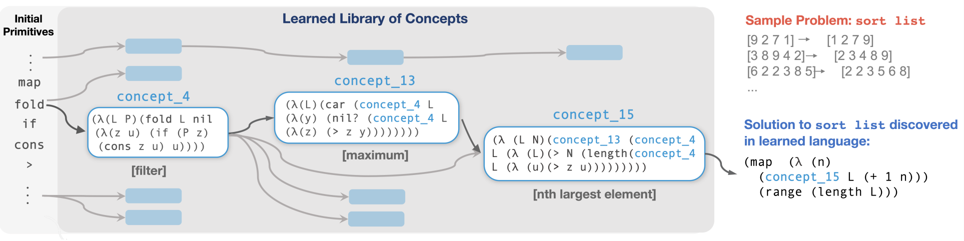
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library learning



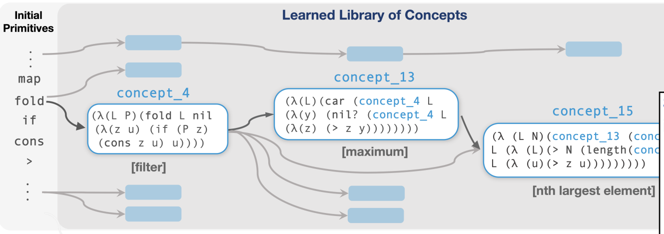
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

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]
...

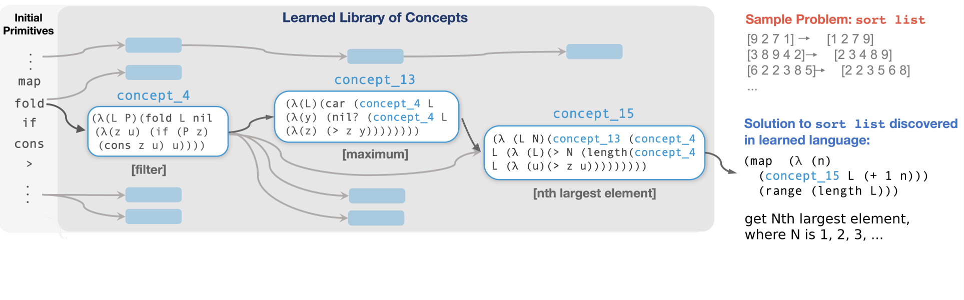
Solution to `sort list` discovered in learned language:

```
(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. 2020.

Library learning

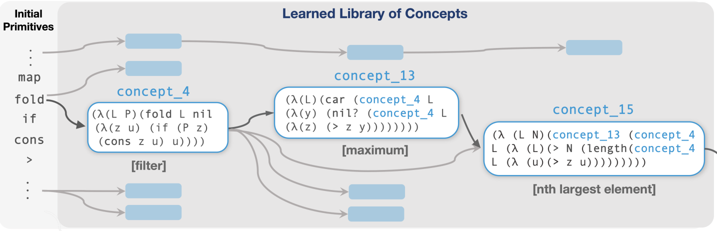


Solution rewritten in initial primitives:

```
(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))))
```

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

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 $\approx 10^{78}$ years

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

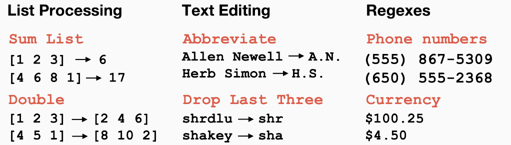
DreamCoder

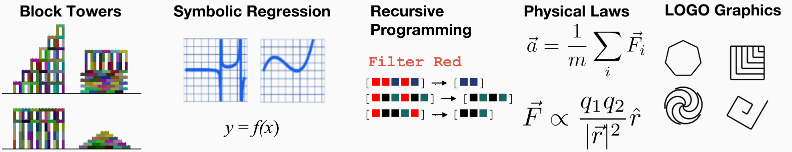
- **Wake:** Solve problems by writing programs
- **Sleep:** Improve library and neural recognition model:
    - **Abstraction sleep:** Improve library
    - **Dream sleep:** Improve neural recognition model

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

DreamCoder

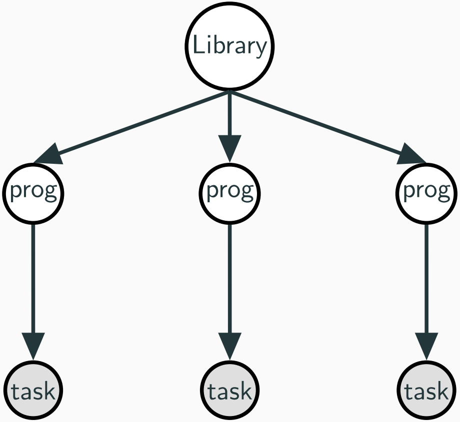
- **Wake:** Solve problems by writing programs
- **Sleep:** Improve library and neural recognition model:
  - **Abstraction sleep:** Improve library
  - **Dream sleep:** Improve neural recognition model





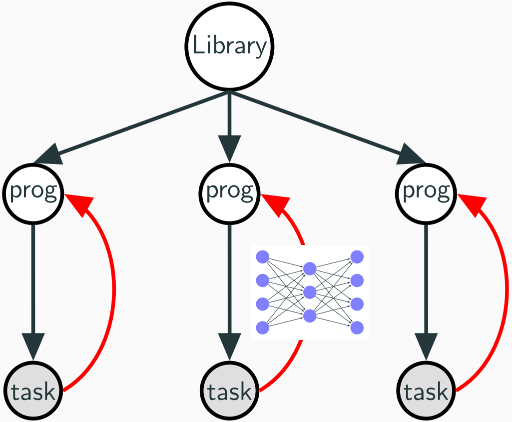
cf. Helmholtz machine, wake/sleep neural network training algorithms

Library learning as Bayesian inference



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

Library learning as neurally-guided Bayesian inference

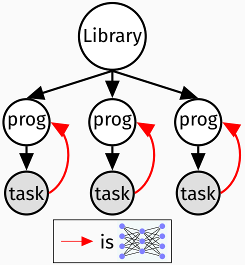


library learning via program analysis +

new neural inference network for program synthesis +

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

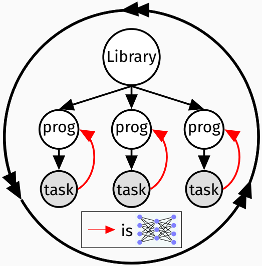
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.


Library
prog
prog
prog
task
task
task
is

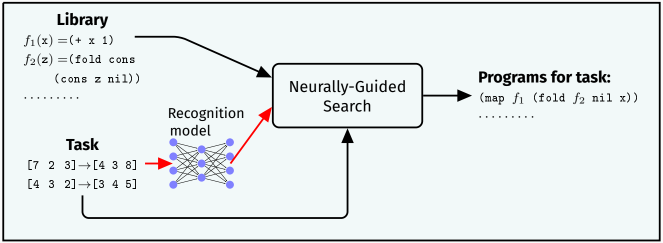
Wake

Sleep: Abstraction

Sleep: Dreaming

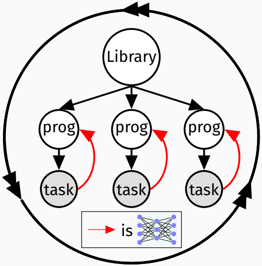


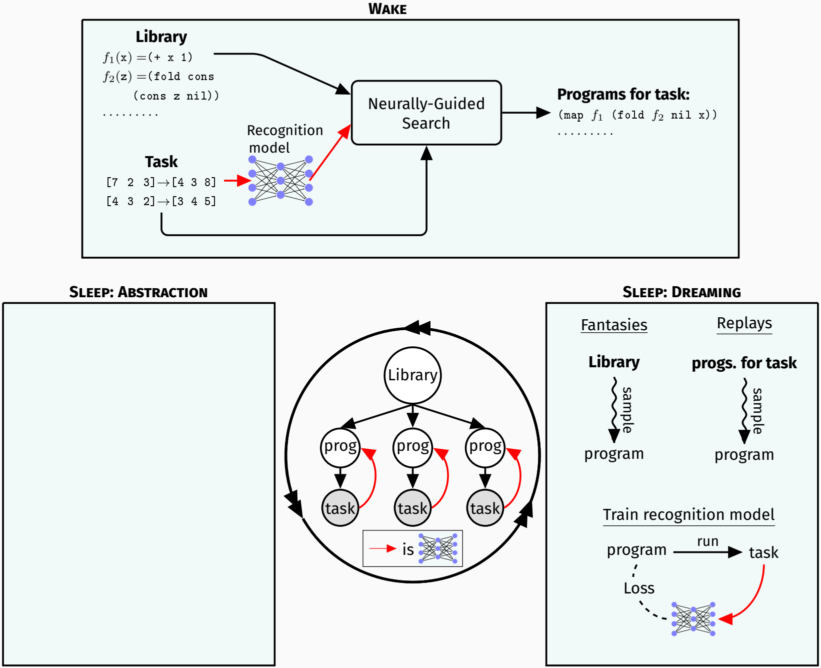
Wake



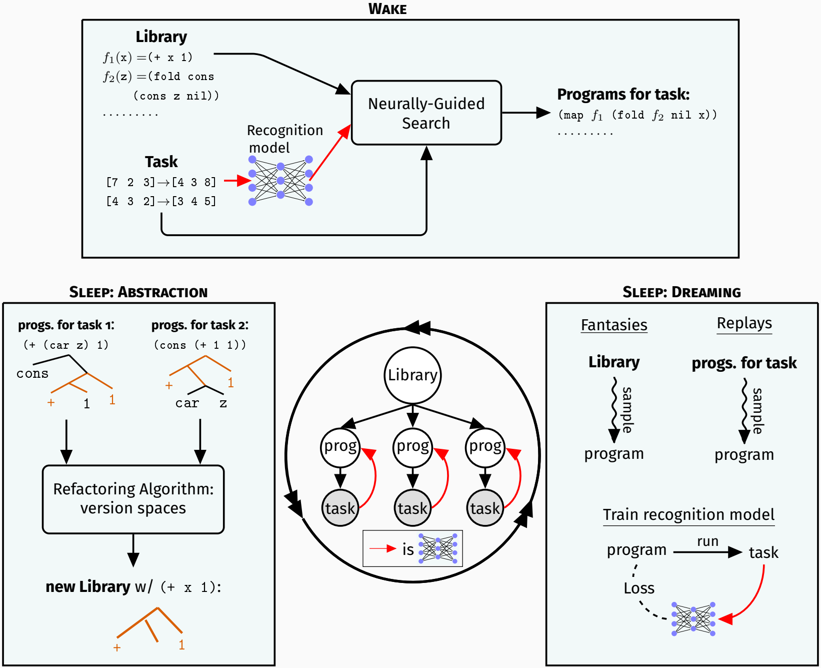
Sleep: Abstraction

Sleep: Dreaming





WAKE
Library
f1(x) =(+ x 1)
f2(z) =(fold cons
(cons z nil))
....
Neurally-Guided
Search
Programs for task:
(map f1 (fold f2 nil x))
....
Recognition
model
Task
[7 2 3]→[4 3 8]
[4 3 2]→[3 4 5]
SLEEP: ABSTRACTION
Library
prog
prog
prog
task
task
task
is
SLEEP: DREAMING
Fantasies
Replays
Library
sample
program
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))
....
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
sample
program
progs. for task
sample
program
Train recognition model
program
run
task
Loss

Abstraction Sleep: Growing the library via refactoring

$5 + 5$

Abstraction Sleep: Growing the library via refactoring

$5 + 5$

`(+ 5 5)`

Abstraction Sleep: Growing the library via refactoring

$5 + 5$

`(+ 5 5)`

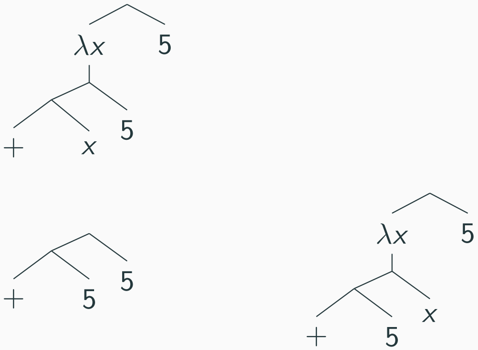
Abstraction Sleep: Growing the library via refactoring

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

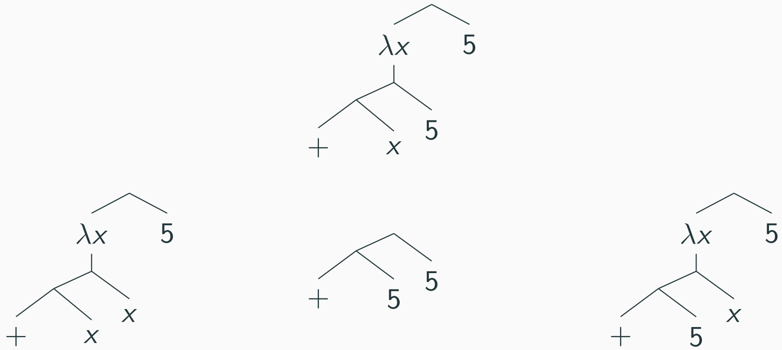
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring



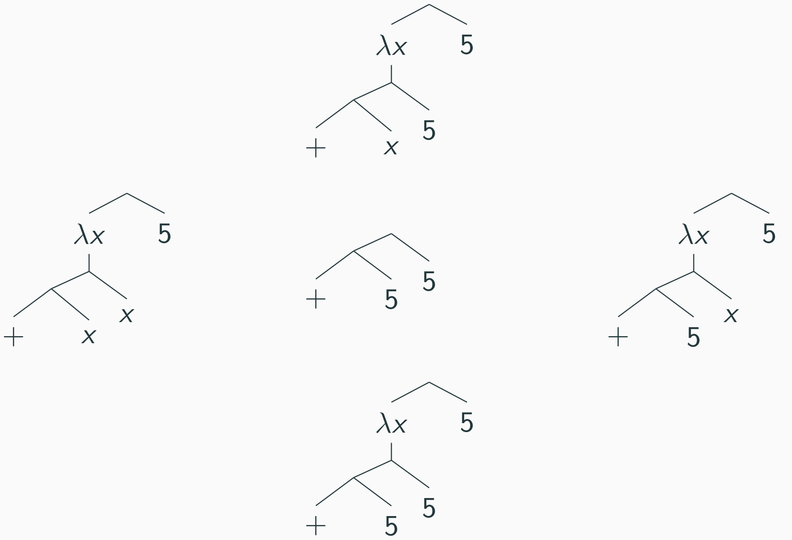
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring



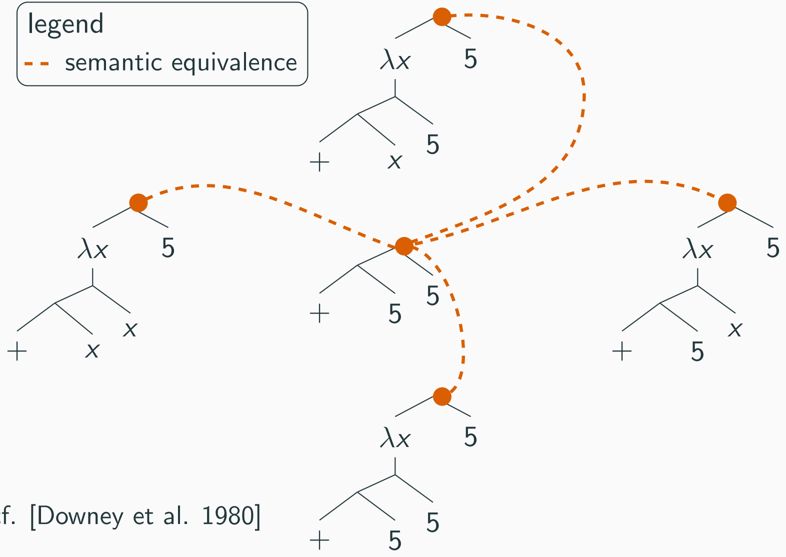
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring



cf. [Downey et al. 1980]

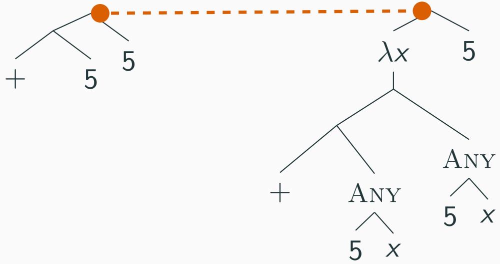
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

legend

semantic equivalence

ANY nondeterministic choice



cf. Downey et al. 1980

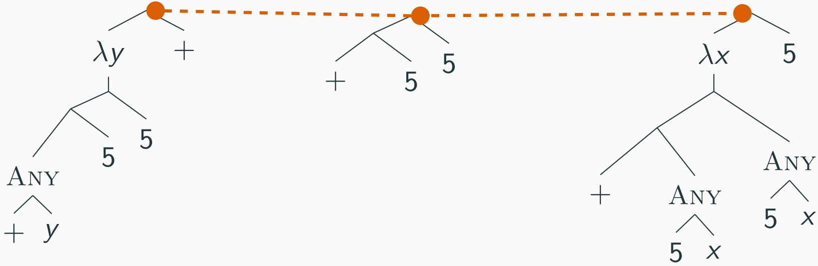
Gulwani 2012

Liang et al. 2010

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

legend
- - - - semantic equivalence
ANY nondeterministic choice

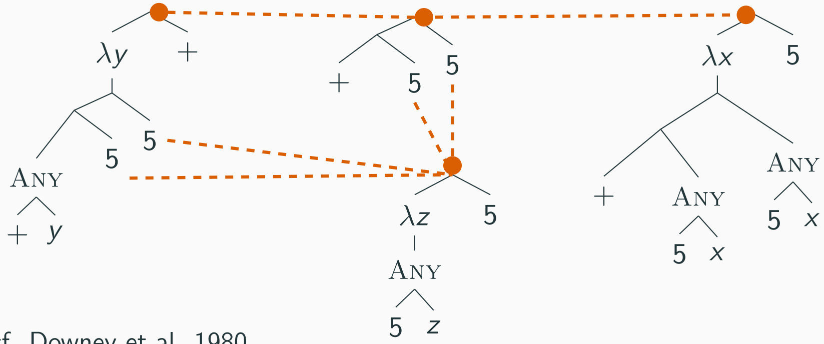


cf. Downey et al. 1980
Gulwani 2012
Liang et al. 2010

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

legend
semantic equivalence
ANY nondeterministic choice

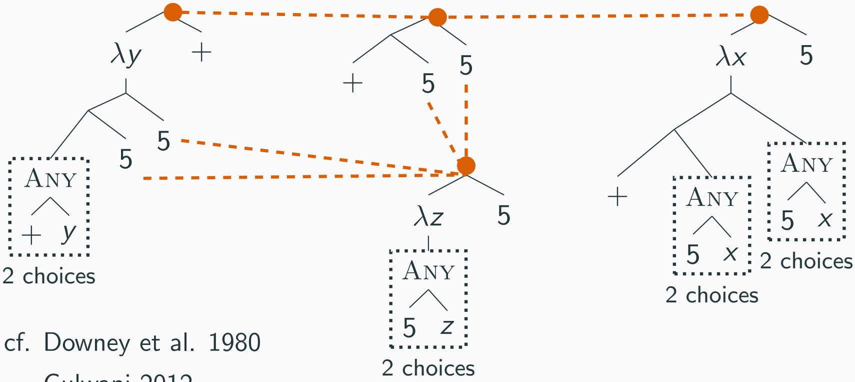


cf. Downey et al. 1980
Gulwani 2012
Liang et al. 2010

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

legend
semantic equivalence
ANY nondeterministic choice



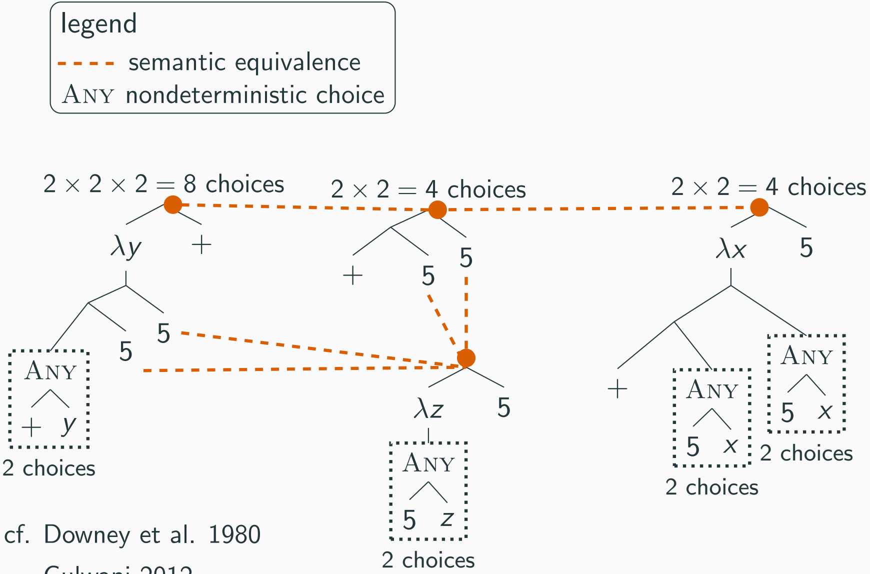
cf. Downey et al. 1980

Gulwani 2012

Liang et al. 2010

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

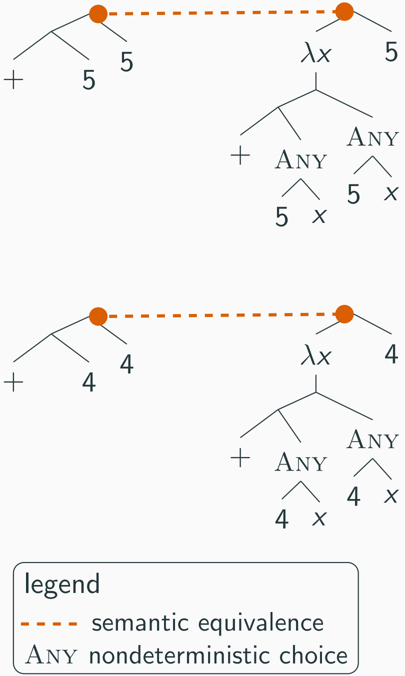


cf. Downey et al. 1980

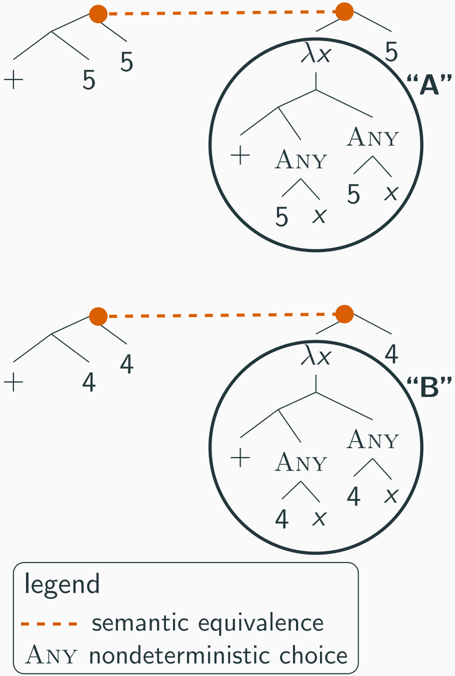
Gulwani 2012

Liang et al. 2010

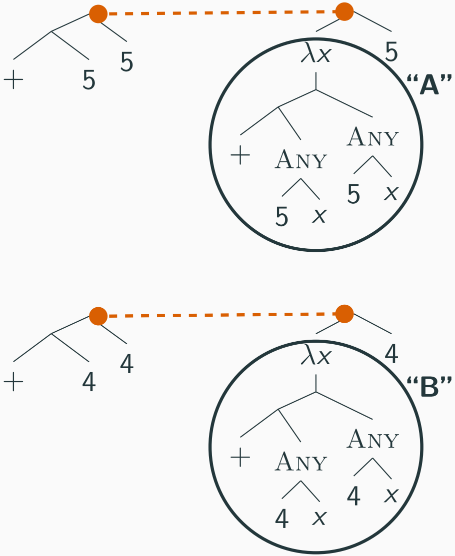
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.



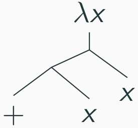
Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.



Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.



A intersect B:

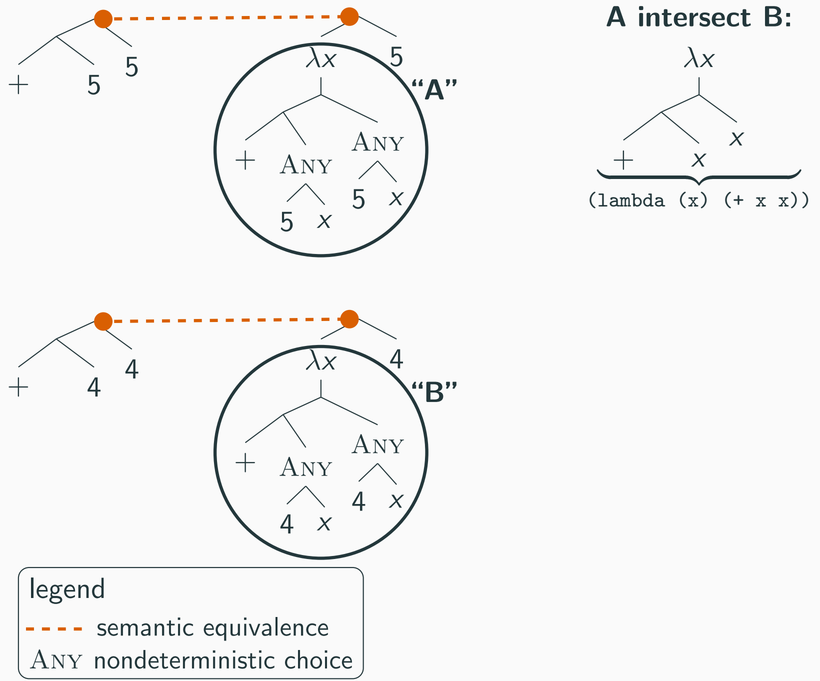


legend

semantic equivalence

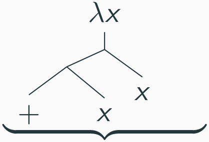
Any nondeterministic choice

Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.



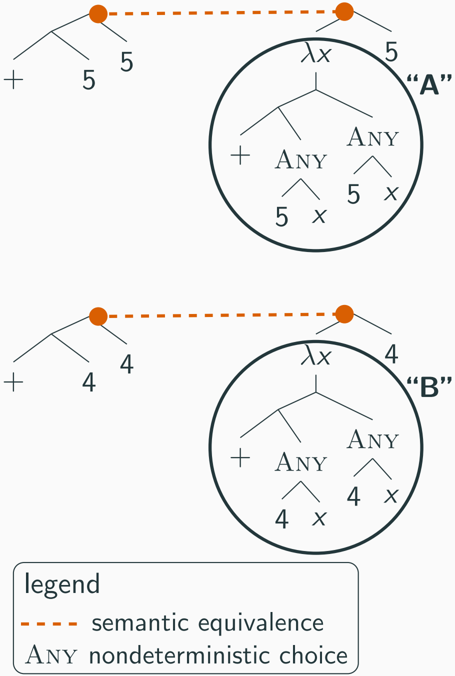
Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.

A intersect B:

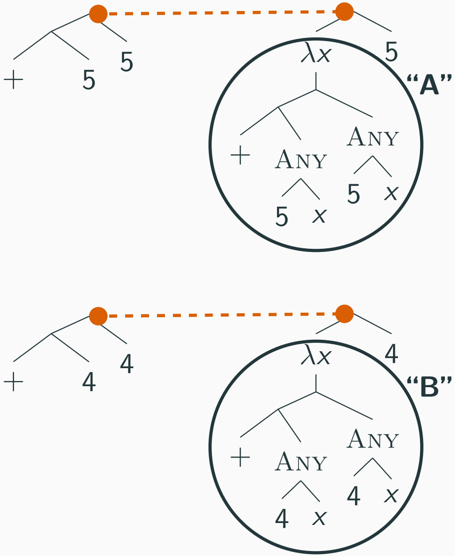


`(lambda (x) (+ x x))`

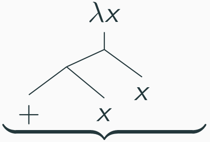
= `double`



Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.



A intersect B:



`(lambda (x) (+ x x))`

$=$ `double`

w/o double	w/ double
`(+ 5 5)`	`(double 5)`
`(+ 4 4)`	`(double 4)`
`(+ 3 3)`	`(double 3)`

...

legend
semantic equivalence
Any nondeterministic choice

Ellis, Wong, ..., Solar-Lezama, Tenenbaum. 2020.

Abstraction Sleep: Growing the library via refactoring

Task: [1 2 3]→[2 4 6]
[4 3 4]→[8 6 8]

Wake: program search

```
(Y (λ (r l) (if (nil? l) nil
 (cons (+ (car l) (car l))
       (r (cdr l))))))
```

Task: [1 2 3]→[0 1 2]
[4 3 4]→[3 2 3]

Wake: program search

```
(Y (λ (r l) (if (nil? l) nil
 (cons (- (car l) 1)
       (r (cdr l))))))
```

Abstraction Sleep: Growing the library via refactoring

Task: [1 2 3]→[2 4 6]
[4 3 4]→[8 6 8]

Wake: program search

```
(Y (λ (r l) (if (nil?  l) nil
  (cons (+ (car l) (car l))
        (r (cdr l))))))
```

refactor
($\mathbf{10^{14}}$ refactorings)

```
((λ (f) (Y (λ (r l) (if (nil?  l)
              nil
              (cons (f (car l))
               (r (cdr l)))))))
 (λ (z) (+ z z)))
```

Task: [1 2 3]→[0 1 2]
[4 3 4]→[3 2 3]

Wake: program search

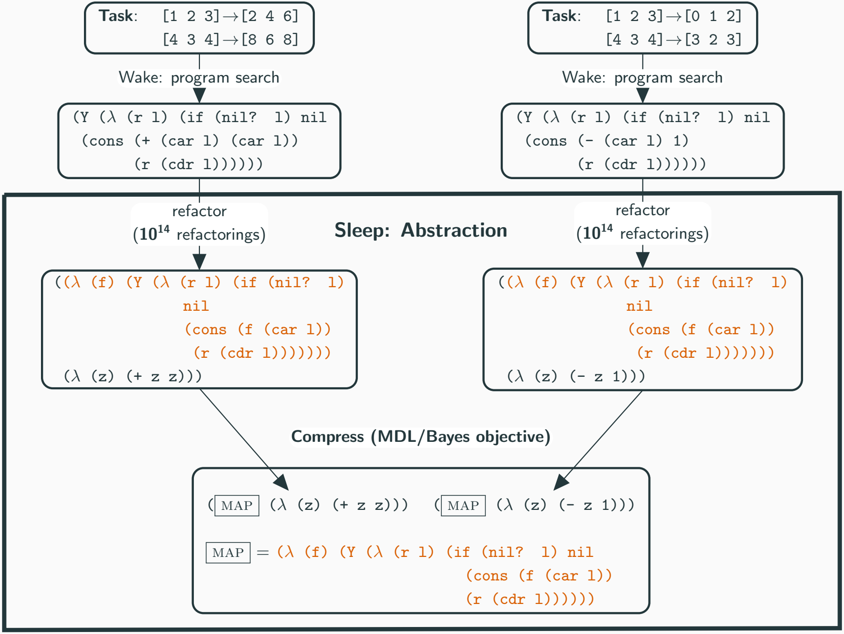
```
(Y (λ (r l) (if (nil?  l) nil
  (cons (- (car l) 1)
        (r (cdr l))))))
```

refactor
($\mathbf{10^{14}}$ refactorings)

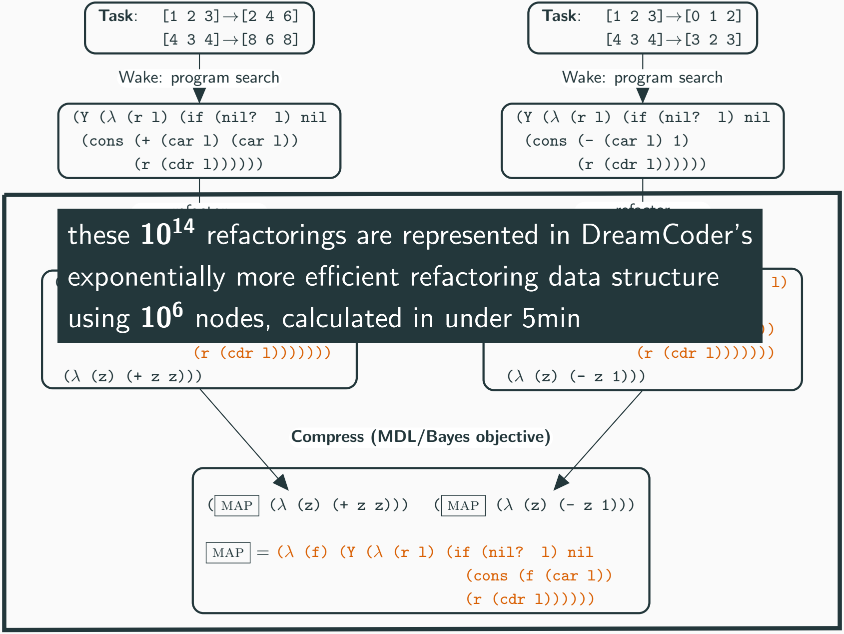
```
((λ (f) (Y (λ (r l) (if (nil?  l)
              nil
              (cons (f (car l))
               (r (cdr l)))))))
 (λ (z) (- z 1)))
```

Sleep: Abstraction

Abstraction Sleep: Growing the library via refactoring



Abstraction Sleep: Growing the library via refactoring



these 10^{14} refactorings are represented in DreamCoder's exponentially more efficient refactoring data structure using 10^{6} nodes, calculated in under 5min

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

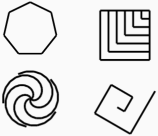
Phone numbers

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

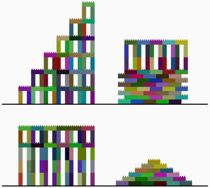
Currency

`$100.25`
`$4.50`

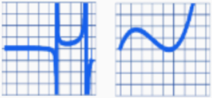
LOGO Graphics



Block Towers



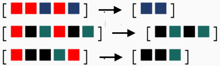
Symbolic Regression



$y = f(x)$

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. 2020.

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

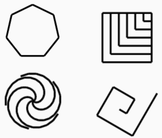
Phone numbers

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

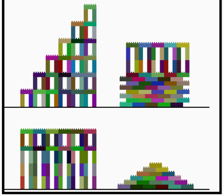
Currency

$100.25
$4.50

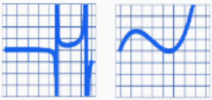
LOGO Graphics



Block Towers



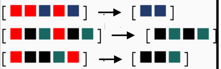
Symbolic Regression



$y = f(x)$

Recursive Programming

Filter Red



Physical Laws

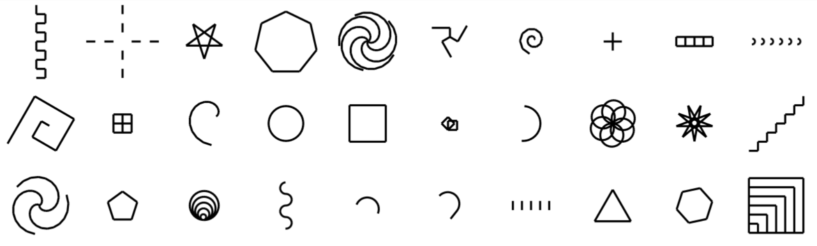
$$\vec{a} = \frac{1}{m} \sum_i \vec{F}_i$$

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

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

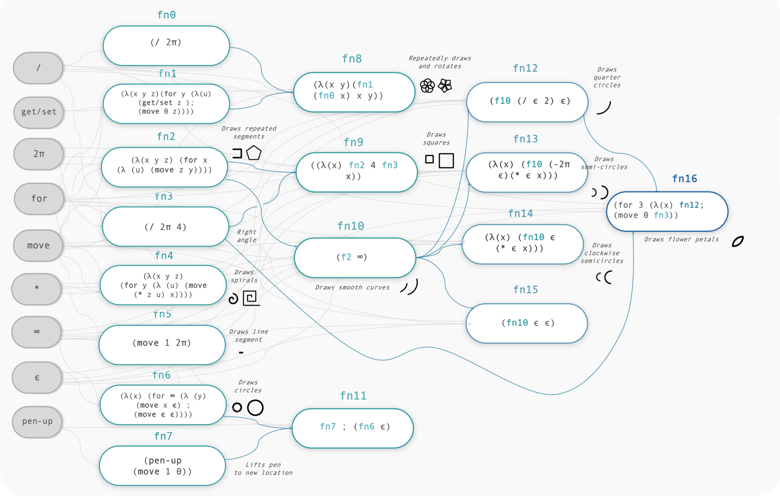
LOGO Turtle Graphics

30 out of 160 tasks



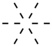
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

LOGO Turtle Graphics – learning an interpretable library



(fn8 5 (fn4 (* ϵ 2) ∞ ϵ))

(for 7 (λ (x) (fn9 x)))

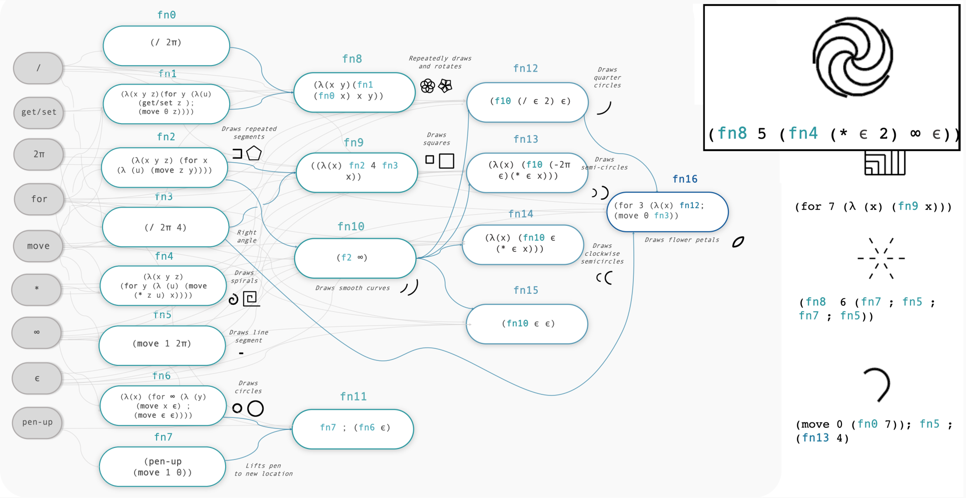


(fn8 6 (fn7 ; fn5 ; fn7 ; fn5))

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

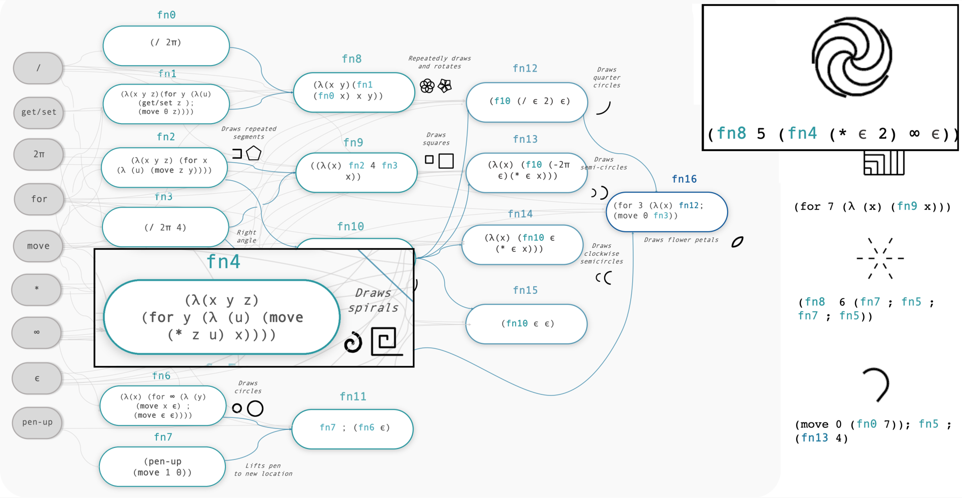
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

LOGO Turtle Graphics – learning an interpretable library



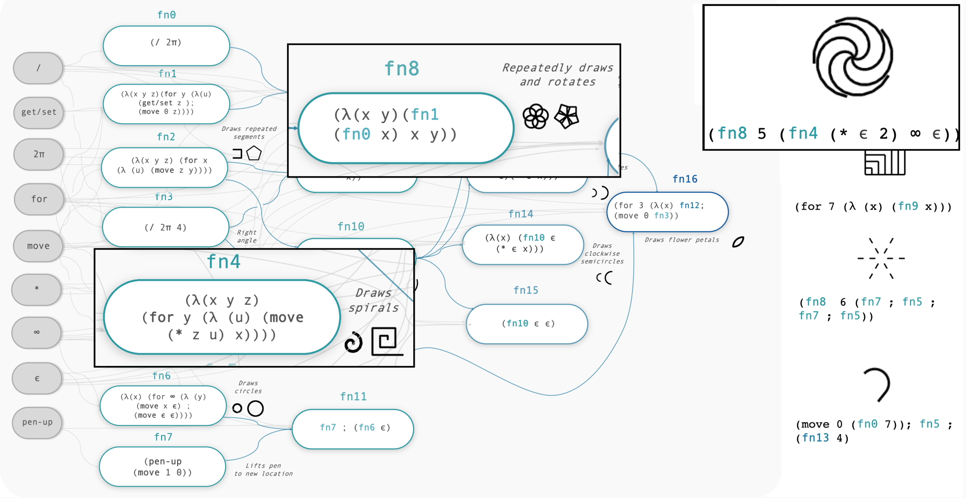
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

LOGO Turtle Graphics – learning an interpretable library



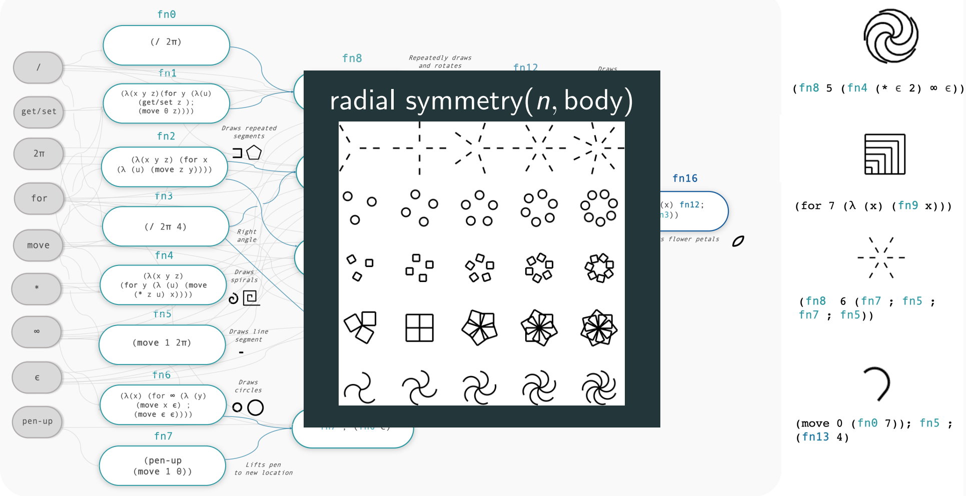
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

LOGO Turtle Graphics – learning an interpretable library

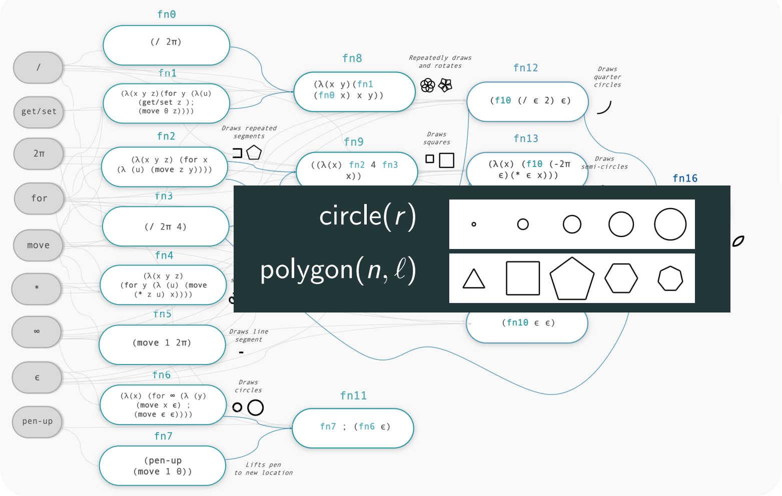


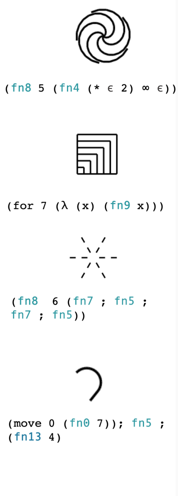
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

LOGO Turtle Graphics – learning an interpretable library



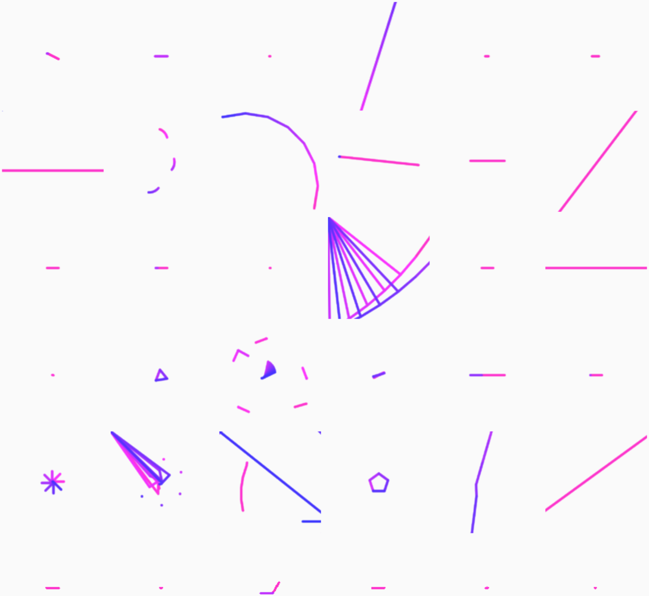
LOGO Turtle Graphics – learning an interpretable library



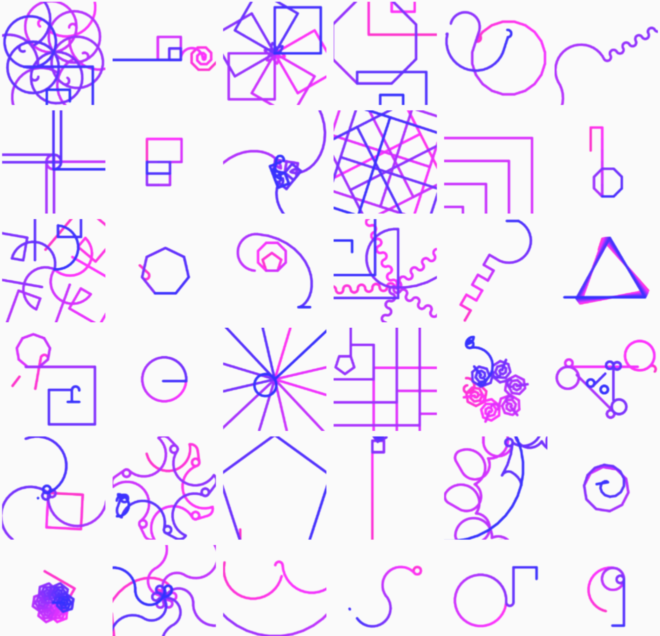


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

What does DreamCoder dream of? (before learning)

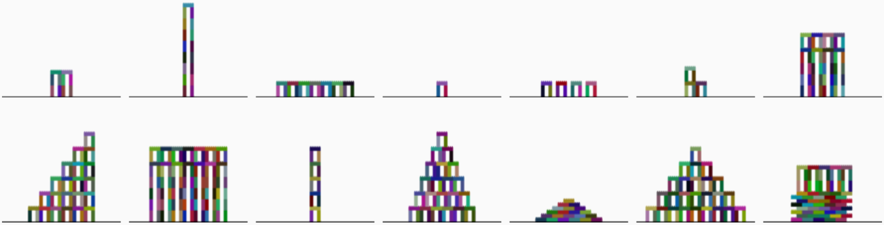


What does DreamCoder dream of? (after learning)



Planning to build towers

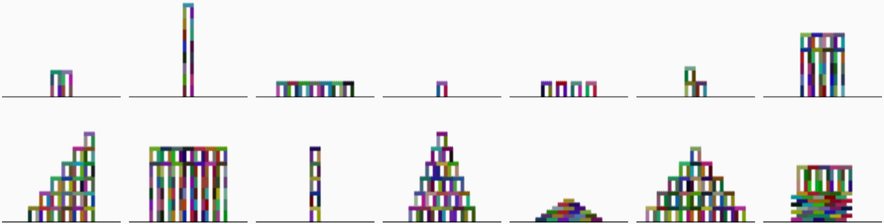
example tasks (112 total)



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Planning to build towers

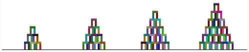
example tasks (112 total)



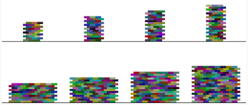
learned library routines ($\approx$ 20 total)

arch(h)

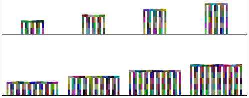
pyramid(h)



wall(w, h)

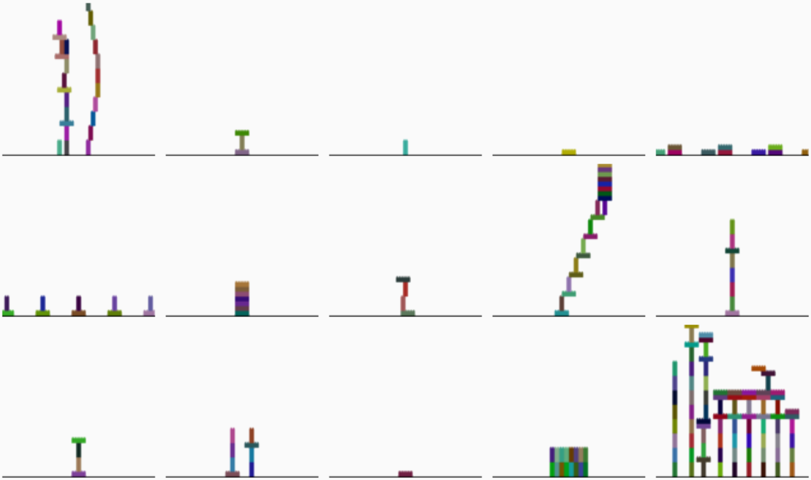


bridge(w, h)

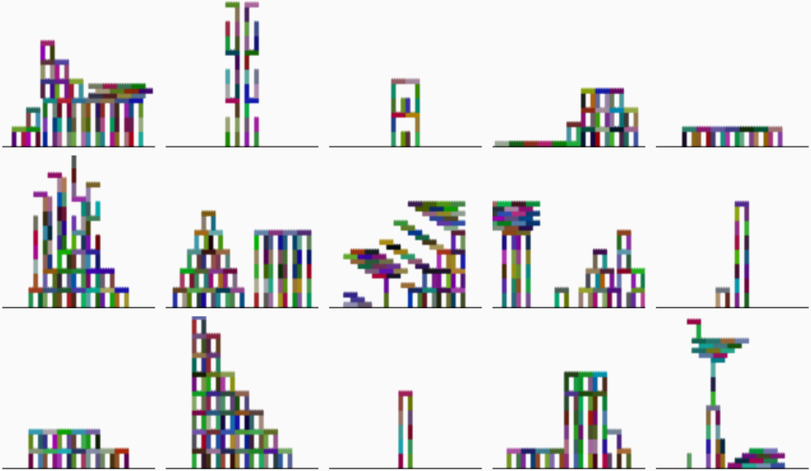


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

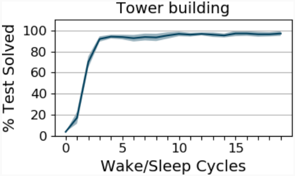
Dreams before learning



Dreams after learning

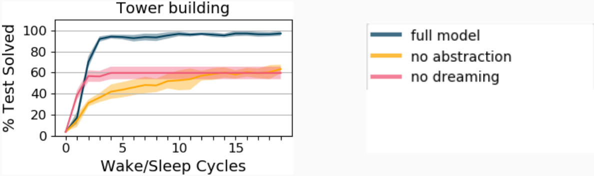


Learning dynamics



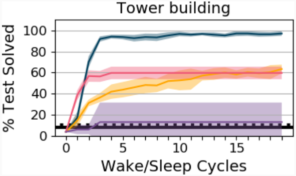
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

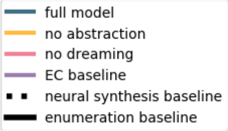
Learning dynamics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Learning dynamics

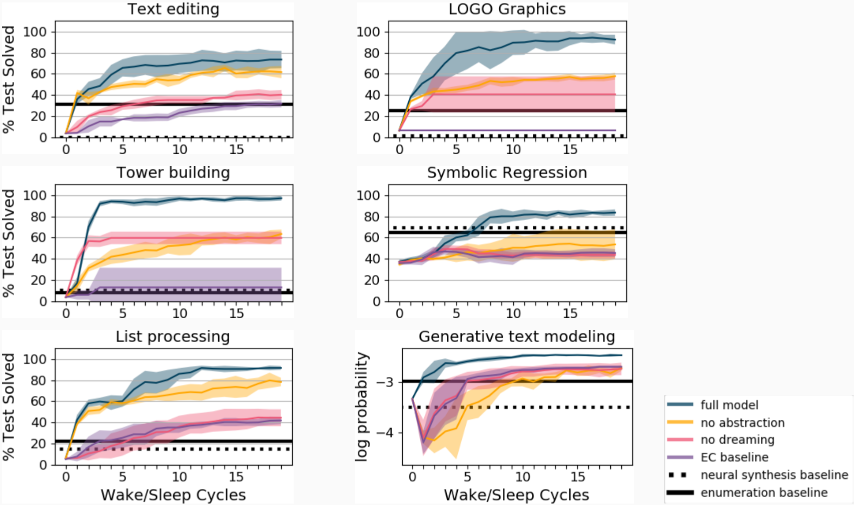




baselines: Exploration-Compression, EC [Dechter et al. 2013]
neural program synthesis, RobustFill [Devlin et al. 2017]
24 hours of brute-force enumeration

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Learning dynamics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Synergy between dreaming and library learning

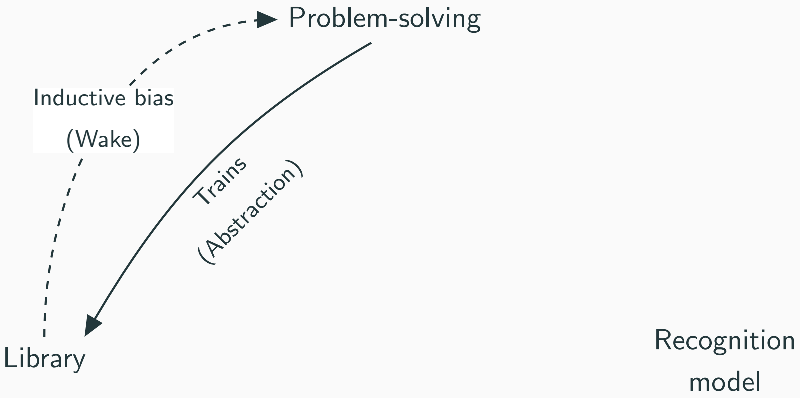
Problem-solving

Library

Recognition
model

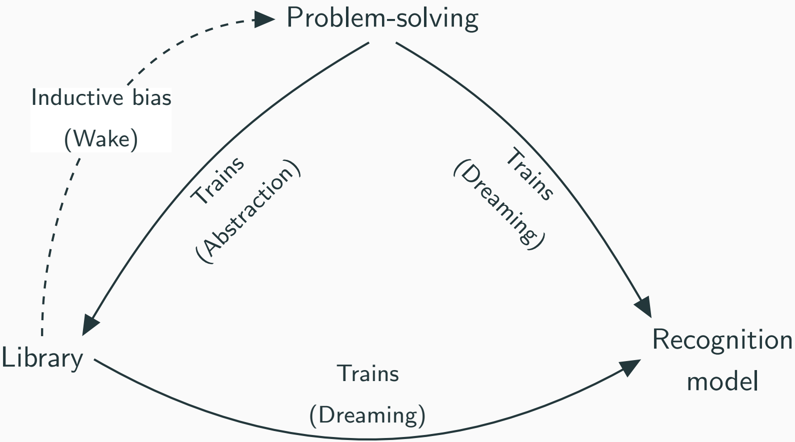
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Synergy between dreaming and library learning



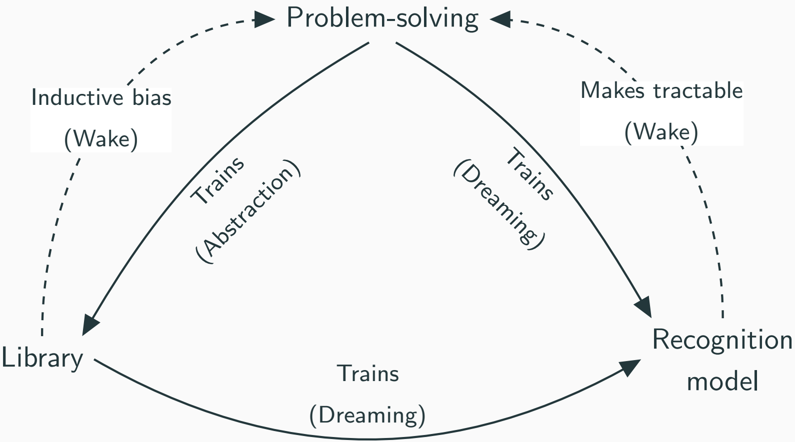
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Synergy between dreaming and library learning



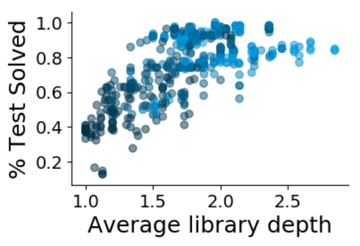
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Synergy between dreaming and library learning



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Evidence for dreaming bootstrapping better libraries

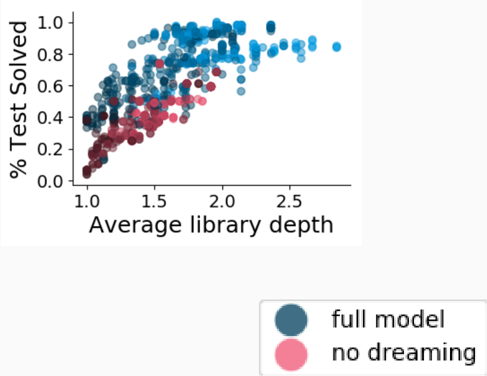


Darker: Early in learning

Brighter: Later in learning

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Evidence for dreaming bootstrapping better libraries

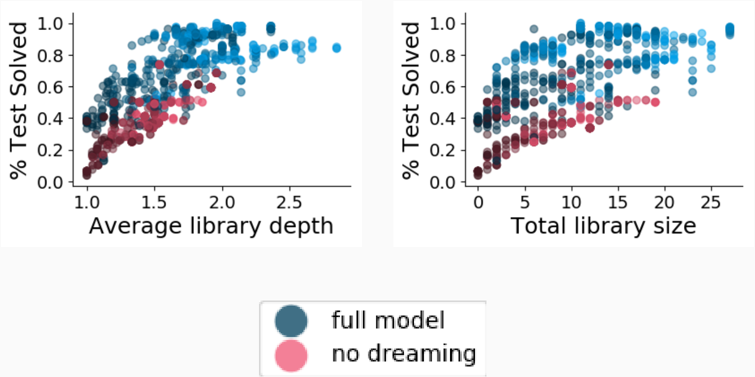


Darker: Early in learning

Brighter: Later in learning

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Evidence for dreaming bootstrapping better libraries



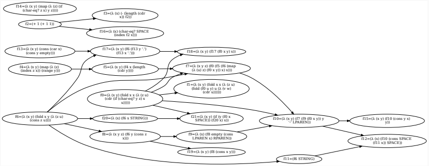
Darker: Early in learning

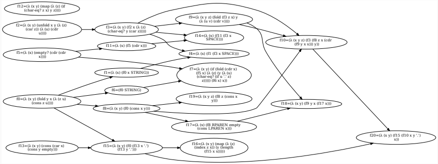
Brighter: Later in learning

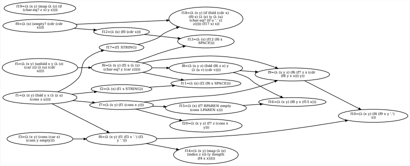
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

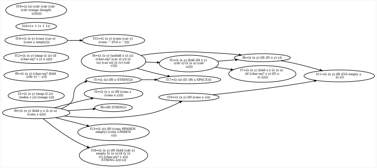
Library structure: Text Editing

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





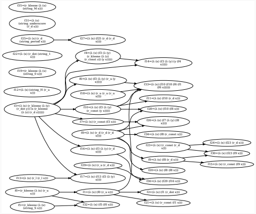


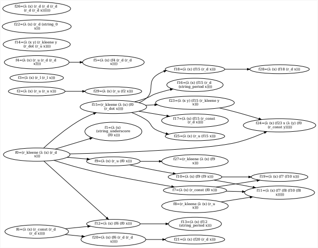


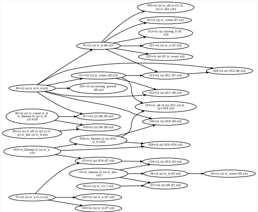
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Library structure: Generating Text

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







From learning libraries,
to learning languages

From learning libraries, to learning languages

functional programming $\rightarrow$ physics

From learning libraries, to learning languages

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

Physics Formula Sheet

Mechanics

$x = x_0 + v_{x0}t + \frac{1}{2}a_x t^2$	$a_c = \frac{v^2}{r}$	$\lvert\vec{F}_{spring}\rvert = k\lvert\vec{x}\rvert$
$v = v_0 + at$	$\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$	$PE_{spring} = \frac{1}{2}kx^2$
$v_x^2 - v_{x0}^2 = 2a(x - x_0)$	$\omega = \omega_0 + \alpha t$	$T_{spring} = 2\pi\sqrt{\frac{m}{k}}$
$\vec{a} = \frac{\Sigma\vec{F}}{m} = \frac{\vec{F}_{net}}{m}$	$T = \frac{2\pi}{\omega} = \frac{1}{f}$	$T_{pendulum} = 2\pi\sqrt{\frac{\ell}{g}}$
$\lvert\vec{F}_{friction}\rvert \leq \mu\lvert\vec{F}_{Normal}\rvert$	$v = f\lambda$	
$\vec{p} = m\vec{v}$	$x = A\cos(2\pi f t)$	$\lvert\vec{F}_{gravity}\rvert = G\frac{m_1 m_2}{r^2}$
$\Delta\vec{p} = \vec{F}\Delta t$	$\vec{\alpha} = \frac{\Sigma\vec{\tau}}{I} = \frac{\vec{\tau}_{net}}{I}$	$\lvert\vec{F}_{gravity}\rvert = m\vec{g}$
$KE = \frac{1}{2}mv^2$	$\vec{\tau} = r \times F$	$PE_{gravity} = -G\frac{m_1 m_2}{r}$
$\Delta PE = mg\Delta y$	$L = I\omega$	$\rho = \frac{m}{V}$
$\Delta E = W = Fd\cos\theta$	$\Delta L = \tau\Delta t$	$KE = \frac{1}{2}I\omega^2$

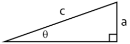
Electricity

$\lvert\vec{F}_E\rvert = k\left\lvert\frac{q_1 q_2}{r^2}\right\rvert$	$\Delta V = IR$	$R = \frac{\rho\ell}{A}$
$I = \frac{\Delta q}{\Delta t}$	$P = I\Delta V$	
$R_{series} = R_1 + R_2 + ... + R_n$	$\frac{1}{R_{Parallel}} = \frac{1}{R_1} + \frac{1}{R_2} + ... + \frac{1}{R_n}$	

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

Initial Primitives

map
zip
cons
empty
cdr
power
fold
car
+
-
*
/
0
1
π

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

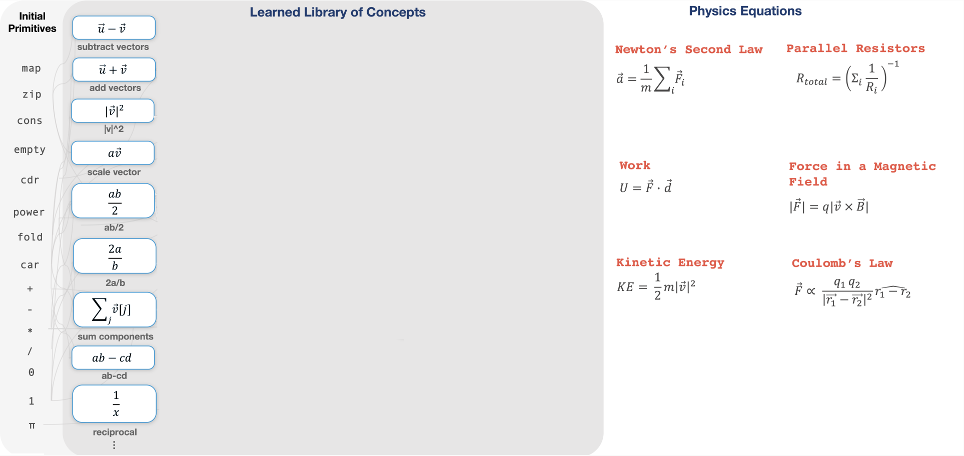
$KE = \frac{1}{2}m|\vec{v}|^2$

Coulomb's Law

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

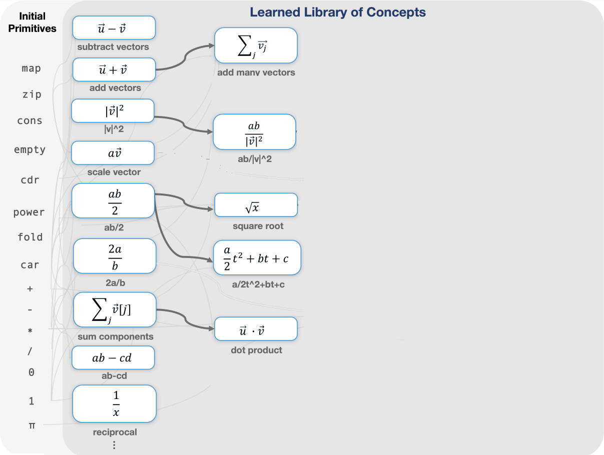
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing languages for vector algebra and physics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

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

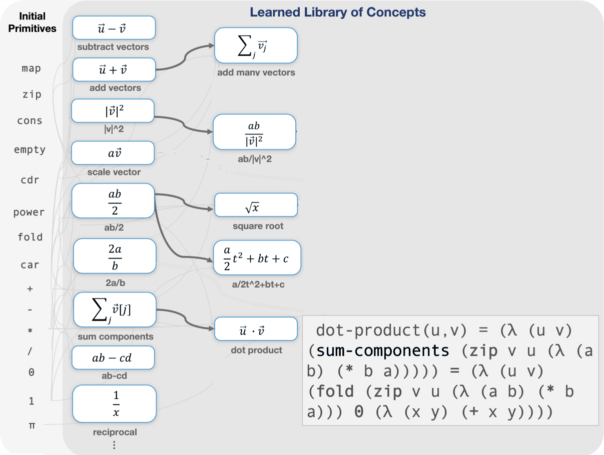
$$KE = \frac{1}{2}m|\vec{v}|^2$$

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. 2020.

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

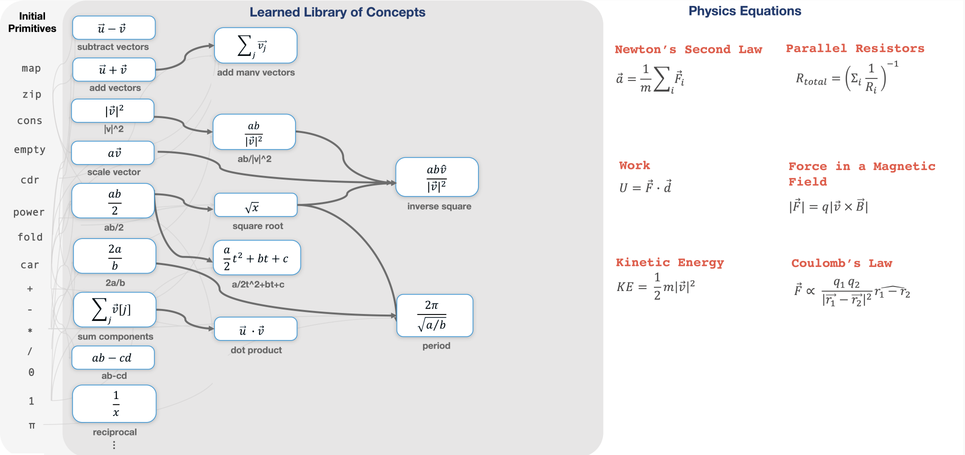
$$KE = \frac{1}{2}m|\vec{v}|^2$$

Coulomb's Law

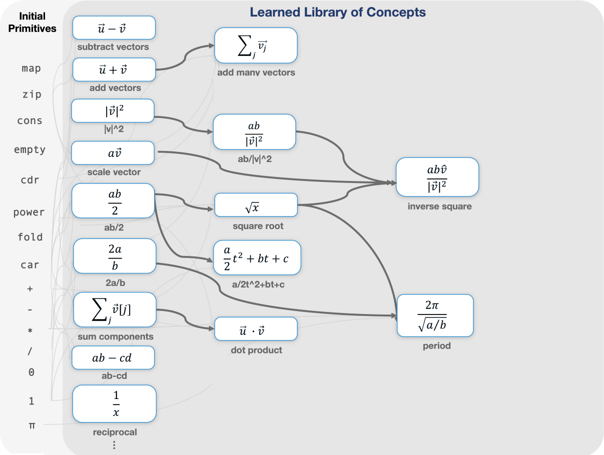
$$\vec{F} \propto \frac{q_1 q_2}{|\vec{r_1} - \vec{r_2}|^2}\widehat{r_1 - r_2}$$

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing languages for vector algebra and physics



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))
```

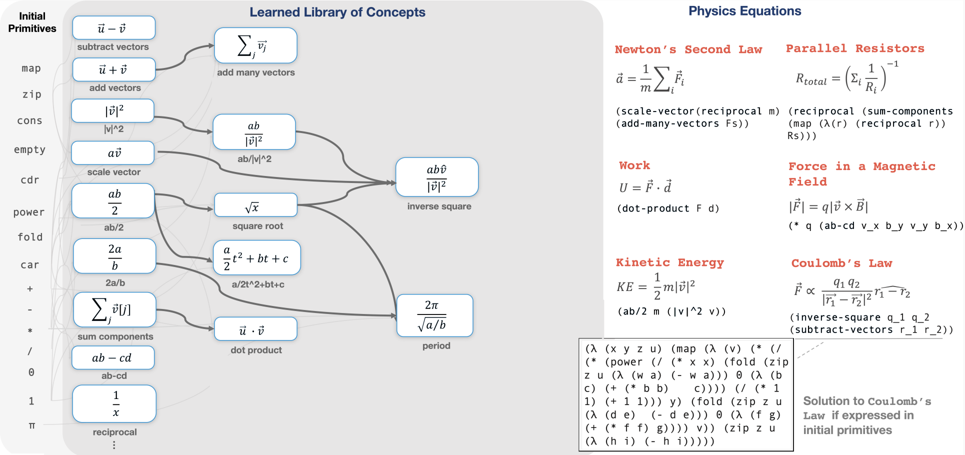
Coulomb's Law

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

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

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing languages for vector algebra and physics



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing a language for recursive programming

Initial Primitives

Y combinator

cons

car

cdr

nil

if

nil?

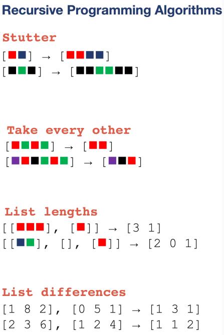
+

-

0

1

=

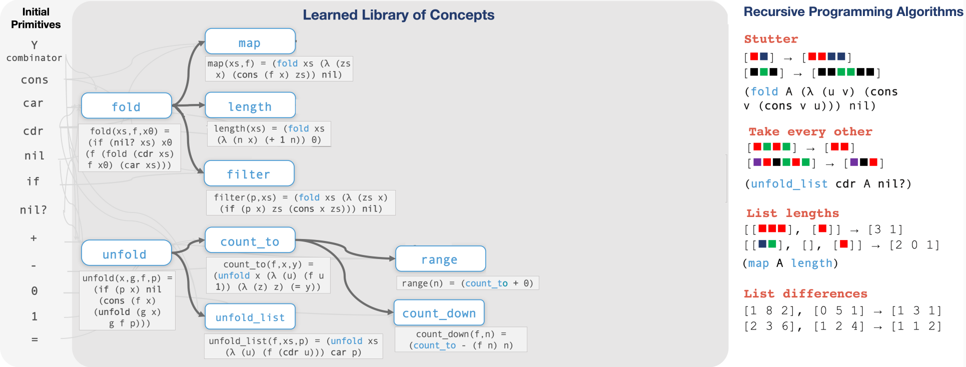


Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing a language for recursive programming

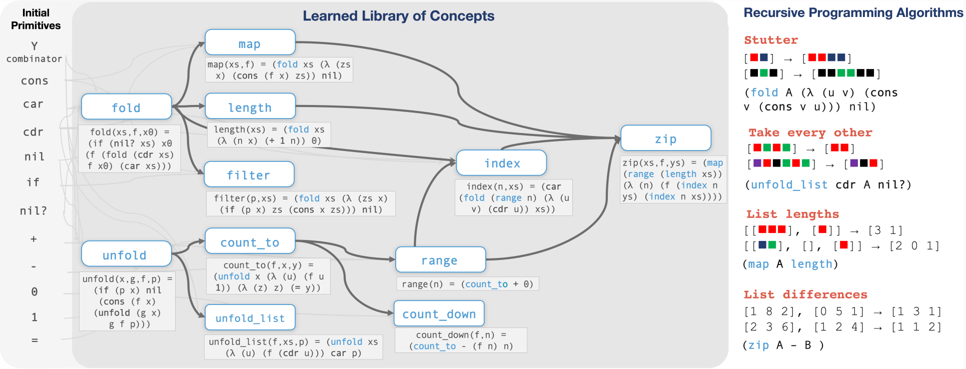
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing a language for recursive programming



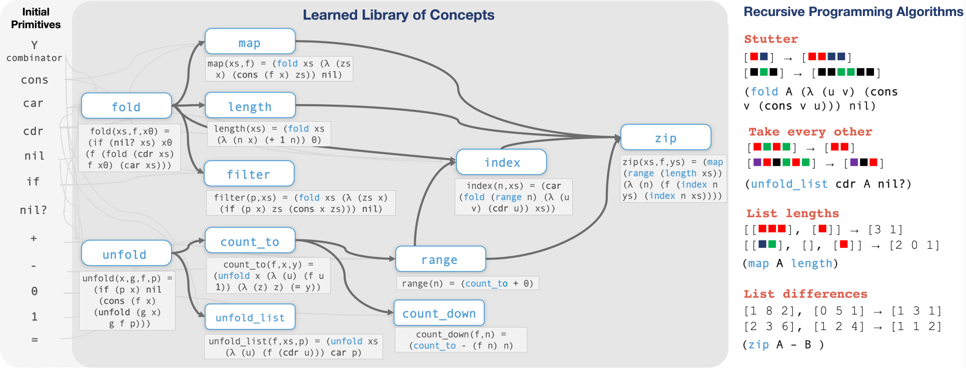
Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

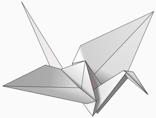
Growing a language for recursive programming



Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing a language for recursive programming

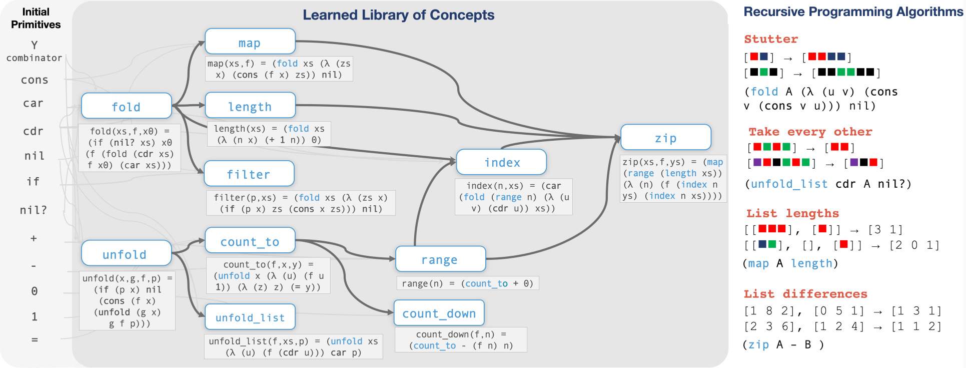




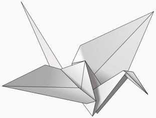
Origami Programming: Jeremy Gibbons, 2003

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

Growing a language for recursive programming



1 year of compute. 5 days on 64 CPUs.



Origami Programming: Jeremy Gibbons, 2003

Ellis, Wong, Nye, ..., Solar-Lezama, Tenenbaum. 2020.

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

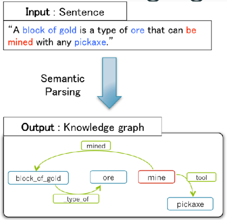
the future

What we've got now:
a toolkit for program induction,
addressing combinatorial program search via learning, integrating symbolic, probabilistic, and neural techniques

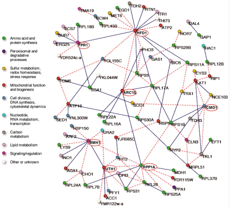
Where will this toolkit take us?

What’s in reach

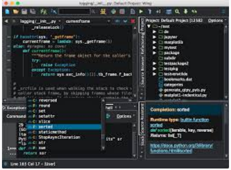
Library learning+
Natural language



Computer-Aided
Science

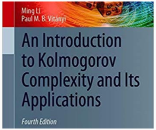


Synthesis for
software engineers

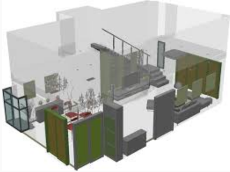


Theory for
program induction

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

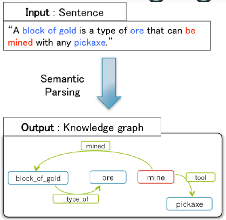


Modeling the
physical world

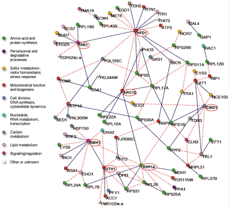


What's in reach

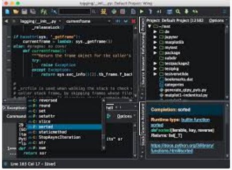
Library learning+ Natural language



Computer-Aided Science

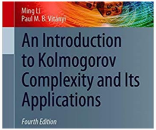


Synthesis for software engineers

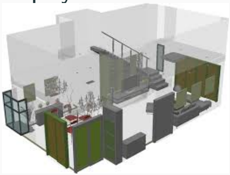


Theory for program induction

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

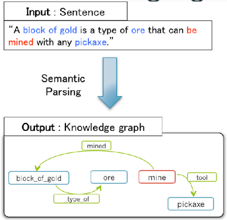


Modeling the physical world

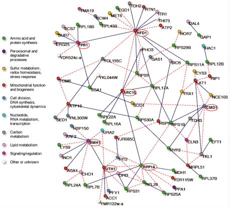


What's in reach

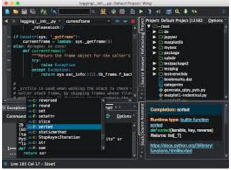
Library learning+ Natural language



Computer-Aided Science

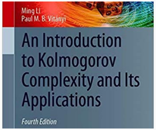


Synthesis for software engineers



Theory for program induction

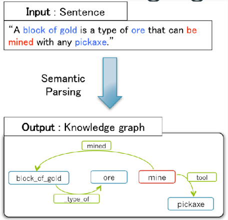
$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$



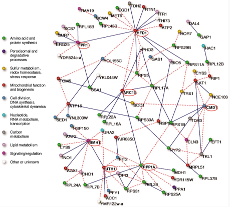
Modeling the physical world

What's in reach

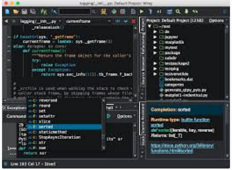
Library learning+ Natural language



Computer-Aided Science

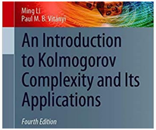


Synthesis for software engineers



Theory for program induction

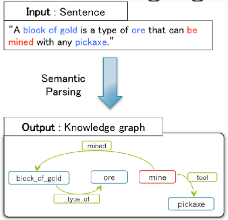
$$P_M(x) = \sum_{i=1}^{\infty} 2^{-|s_i(x)|}$$



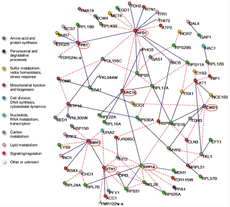
Modeling the physical world

What's in reach

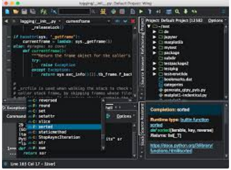
Library learning+ Natural language



Computer-Aided Science

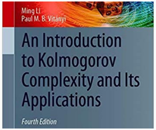


Synthesis for software engineers

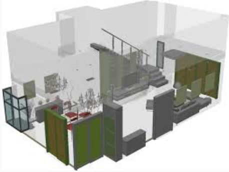


Theory for program induction

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

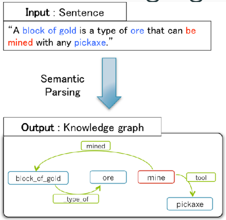


Modeling the physical world

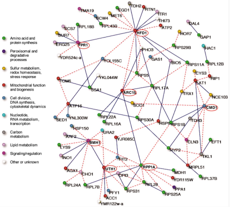


What's in reach

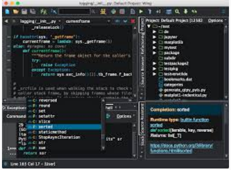
Library learning+
Natural language



Computer-Aided
Science

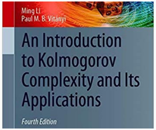


Synthesis for
software engineers

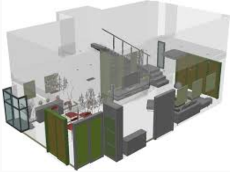


Theory for
program induction

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



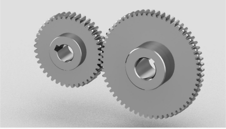
**Modeling the
physical world**



Within reach: Modeling the physical world

hinge

gear



doorknob

Within reach: Modeling the physical world

Built in

Learned Library

The world

2D geometry

symmetry

...

3D geometry

extrude

...

Rigid body

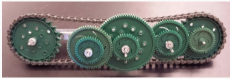
dynamics

...

Programming

for loop

...



Within reach: Modeling the physical world

Built in

2D geometry

symmetry

...

3D geometry

extrude

...

Rigid body

dynamics

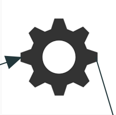
...

Programming

for loop

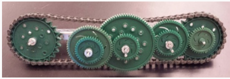
...

Learned Library



`gear(n,`ℓ`,`r`)`

The world



Within reach: Modeling the physical world

Built in | Learned Library | The world

2D geometry

symmetry

...

3D geometry

extrude

...

Rigid body

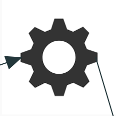
dynamics

...

Programming

for loop

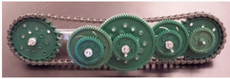
...



`gear(n,`ℓ`,`r`)`

`geartrain(K,`$\vec{n}$`,`$\vec{\ell}$`,`$\vec{r}$`)`

$\vdots$



Within reach: Modeling the physical world

Built in

2D geometry

symmetry

...

3D geometry

extrude

...

Rigid body

dynamics

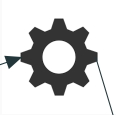
...

Programming

for loop

...

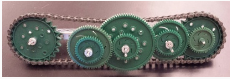
Learned Library



gear(n,ℓ,r)

geartrain(K,$\vec{n}$,$\vec{\ell}$,$\vec{r}$)

The world



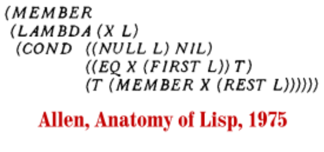
What's far off, but worthwhile?

What kinds of future machines could learn all this?

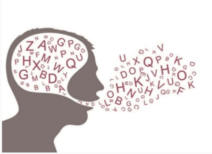
using new devices

play

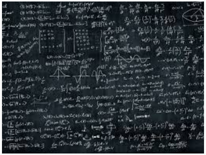
coding



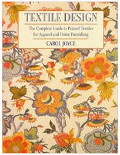
language



science



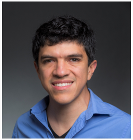
design



Collaborators

Josh Tenenbaum

Armando Solar-Lezama



Tim O'Donnell

Adam Albright

Max Nye

Cathy Wong

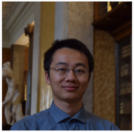
Yewen Pu

Dan Ritchie

Mathias Sable-Meyer

Lucas Morales

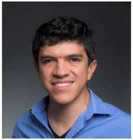
Tao Du



Collaborators

Josh Tenenbaum

Armando Solar-Lezama



Tim O'Donnell

Adam Albright

Max Nye

Cathy Wong

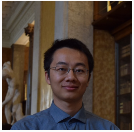
Yewen Pu

Dan Ritchie

Mathias Sable-Meyer

Lucas Morales

Tao Du



thank you