Library Learning for Neurally-Guided Bayesian Program Induction

Kevin Ellis[1], Lucas Morales[1], Mathias Sablé-Meyer[2],
Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: ENS Paris-Saclay.

List processing:

[5 2 9] → [9 2 5]

[1 1 2 2] → [2 2 1 1]

[1 2 3 2] → [2 3 2 1]

Text editing:

P Kohli → Dr. Kohli

Sumit Gulwani → Dr. Gulwani

Danny Tarlow → Dr. Tarlow

Symbolic regression:

$ax^3 + bx^2 + cx + d$

$a/(x - b)$

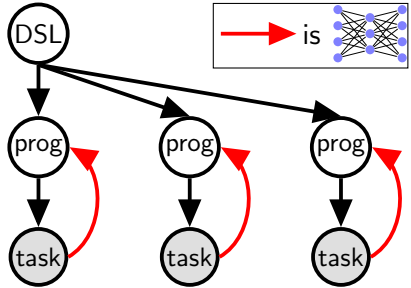
Explore/Compress/Compile (EC^2) learns to solve programming tasks like these by growing a library of code and training a neural net to search for programs written using the library

Library Learning

Tasks and Programs	DSL
`[7 2 3]→[7 3]` `[1 2 3 4]→[3 4]` `[4 3 2 1]→[4 3]` $f(\ell) = $`(f1 ℓ (λ (x) (> x 2)))` `[7 3]→False` `[3]→False` `[9 0 0]→True` `[0]→True` `[0 7 3]→True` $f(\ell) = $`(f3 ℓ 0)` `[2 7 8 1]→8` `[3 19 14]→19` $f(\ell) = $`(f2 ℓ)`	$f_0(\ell,$`r`$) = $`(foldr r ℓ cons)` ($f_0$: *Append lists* `r` *and* ℓ) $f_1(\ell,$`p`$) = $`(foldr ℓ nil (λ (x a) (if (p x) (cons x a) a)))` ($f_1$: *Higher-order filter function*) $f_2(\ell) = $`(foldr ℓ 0 (λ (x a) (if (> a x) a x)))` ($f_2$: *Maximum element in list* ℓ) $f_3(\ell,$`k`$) = $`(foldr ℓ (is-nil ℓ) (λ (x a) (if a a (= k x))))` ($f_2$: *Whether* ℓ *contains* `k`)

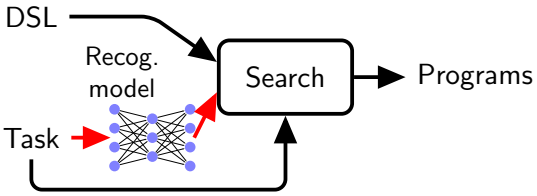
- Learned DSL primitives can call each other
- Rediscovers higher-order functions like `filter`

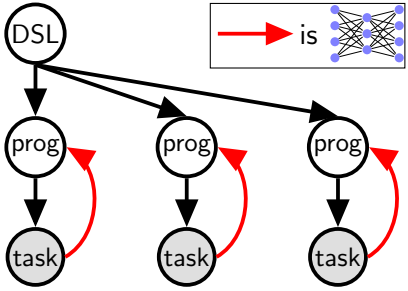
Explore/Compress/Compile as Amortized Bayesian Inference



Explore/Compress/Compile as Amortized Bayesian Inference

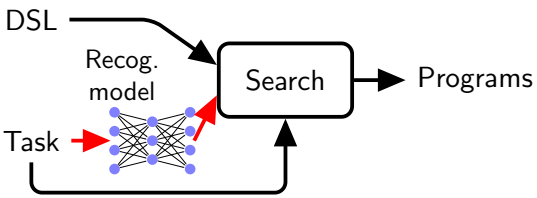
Explore: Infer programs, fixing DSL and neural net

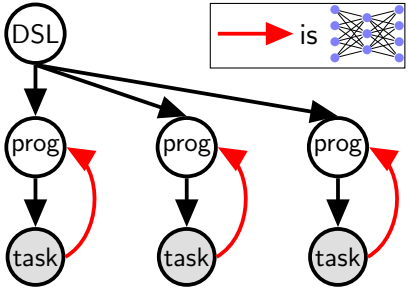




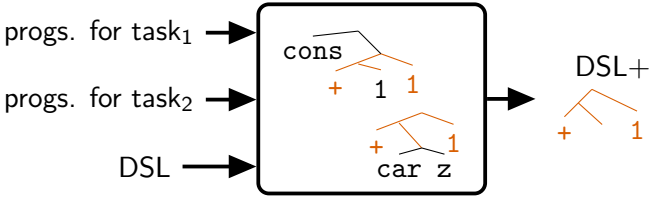
Explore/Compress/Compile as Amortized Bayesian Inference

Explore: Infer programs, fixing DSL and neural net



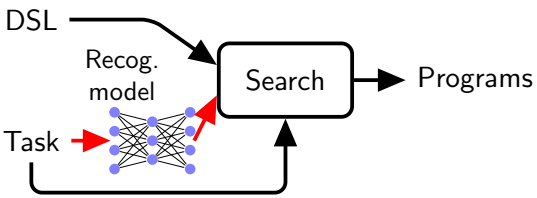


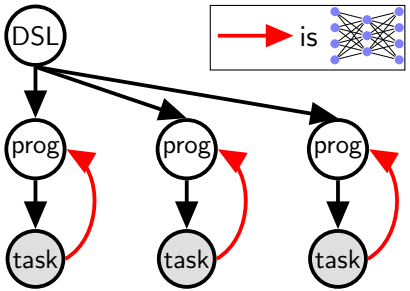
Compress: Update DSL, fixing programs



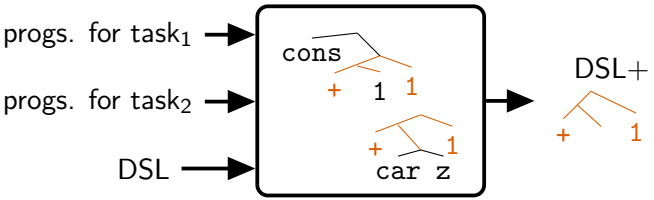
Explore/Compress/Compile as Amortized Bayesian Inference

Explore: Infer programs, fixing DSL and neural net

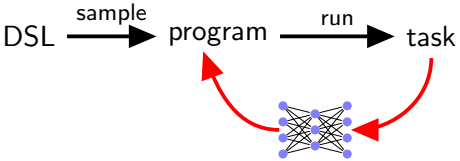




Compress: Update DSL, fixing programs

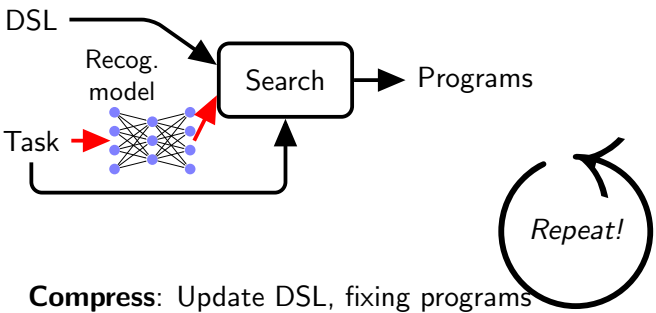


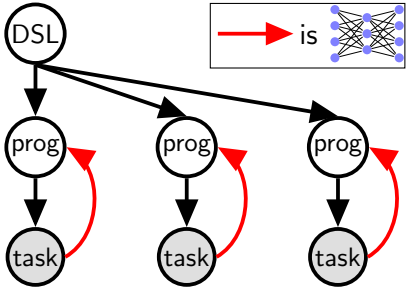
Compile: Train recognition model



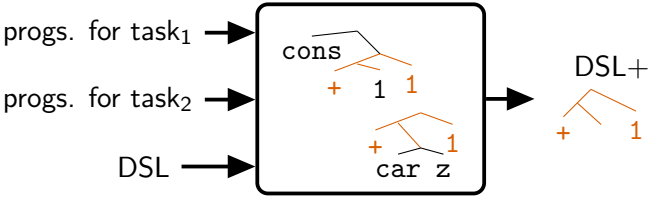
Explore/Compress/Compile as Amortized Bayesian Inference

Explore: Infer programs, fixing DSL and neural net

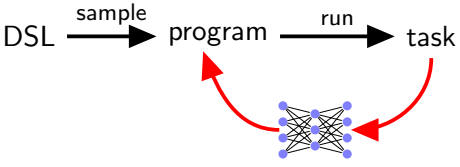




Compress: Update DSL, fixing programs



Compile: Train recognition model



Domain: List processing

Starts with: `foldr`, `unfold`, `if`, `map`, `length`, `index`, `=`, `+`, `-`, `0`, `1`, `cons`, `car`, `cdr`, `nil`, `is-nil`, `mod`, `*`, `>`, `is-square`, `is-prime`.
236 human-interpretable list processing tasks.
Discovers 38 new DSL primitives, including `filter`

Name	Input	Output
repeat-2	[7 0]	[7 0 7 0]
drop-3	[0 3 8 6 4]	[6 4]
rotate-2	[8 14 1 9]	[1 9 8 14]
count-head-in-tail	[1 2 1 1 3]	2
keep-mod-5	[5 9 14 6 3 0]	[5 0]
product	[7 1 6 2]	84

With functional programming "problem set" + 93 hours on 64 CPUs, rediscovers:
`map`, `foldr`, `unfold`, `range`, `length`, `index`, `zip`, +some arithmetic routines

Domain: Text Editing

In the style of FlashFill (Gulwani 2012). Starts with: `foldr`, `unfold`, `if`, `map`, `length`, `index`, `=`, `+`, `-`, `0`, `1`, `cons`, `car`, `cdr`, `nil`, `is-nil` , plus string & character constants.

Input	Output
+106 769-438	106.769.438
+83 973-831	83.973.831
Temple Anna H	TAH
Lara Gregori	LG

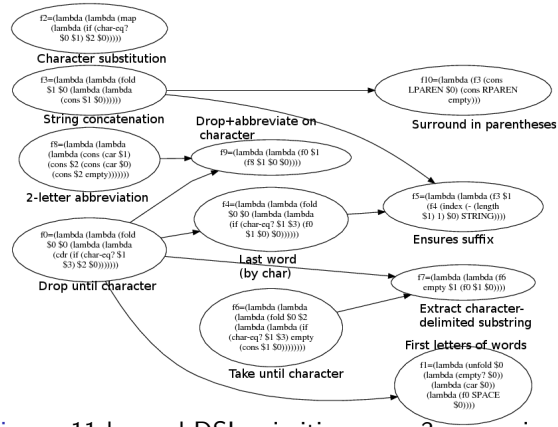


Figure: 11 learned DSL primitives over 3 successive iterations (3 columns). Learned primitives call each other (arrows).

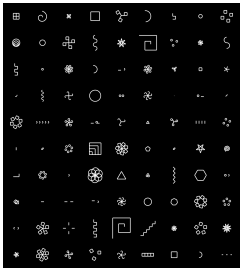
Domain: Symbolic regression from visual input

Starts with: `plus`, `times`, `divide`, `real-number`. Autograds through program
Bayesian likelihood P[data|prog] favors fewer continuous parameters

Programs & Tasks		DSL
		f_0(x) = (+ x real) f_1(x) = (f_0 (* real x)) f_2(x) = (f_1 (* x (f_0 x))) f_3(x) = (f_0 (* x (f_2 x))) f_4(x) = (f_0 (* x (f_3 x))) *(f_4: 4th order polynomial)* f_5(x) = (/ real (f_0 x)) *(f_5: rational function)*
f(x) = (f_1 x)	f(x) = (f_5 x)	
		
f(x) = (f_4 x)	f(x) = (f_3 x)	

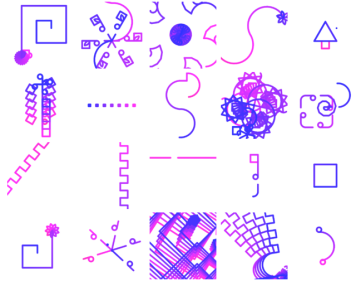
New domain: Generative graphics programs (Turtle/LOGO)

Training tasks



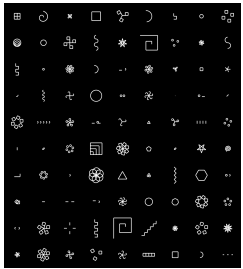
DSL samples before learning

DSL samples after learning



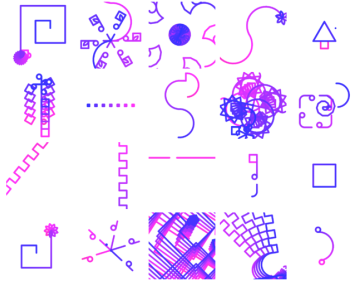
New domain: Generative graphics programs (Turtle/LOGO)

Training tasks



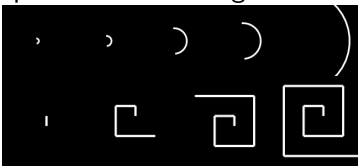
DSL samples before learning

DSL samples after learning



Learned DSL contains parametric drawing routines:

Semicircle:



Spiral:

Greek Spiral:

S-Curves:

Polygons & Stars:

Circles:

Learning to program: Poster AB #24

```
f2(p,f,n,x) = (if (p x) nil
                  (cons (f x) (f2 (n x))))
```
(f_2: *unfold*)

```
f3(i,l) = (if (= i 0) (car l)
              (f3 (f1 i) (cdr l))))
```
(f_3: *index*)

```
f4(f,l,x) = (if (empty? l) x
                (f (car l) (f4 (cdr l))))
```
(f_4: *fold*)

```
f5(f,l) = (if (empty? l) nil)
              (cons (f (car l)) (f5 (cdr l)))
```
(f_5: *map*)

Symbolic Regression

$f(x) = (f_1\ x)$

$f(x) = (f_6\ x)$

$f(x) = (f_4\ x)$

$f(x) = (f_3\ x)$

```
f0(x) = (+ x real)
f1(x) = (f0 (* real x))
f2(x) = (f1 (* x (f0 x)))
f3(x) = (f0 (* x (f2 x)))
f4(x) = (f0 (* x (f3 x)))
```
(f_4: *4th order polynomial*)

```
f5(x) = (/ real x)
f6(x) = (f5 (f0 x))
```
(f_6: *rational function*)