Toward Natural Language Supervision

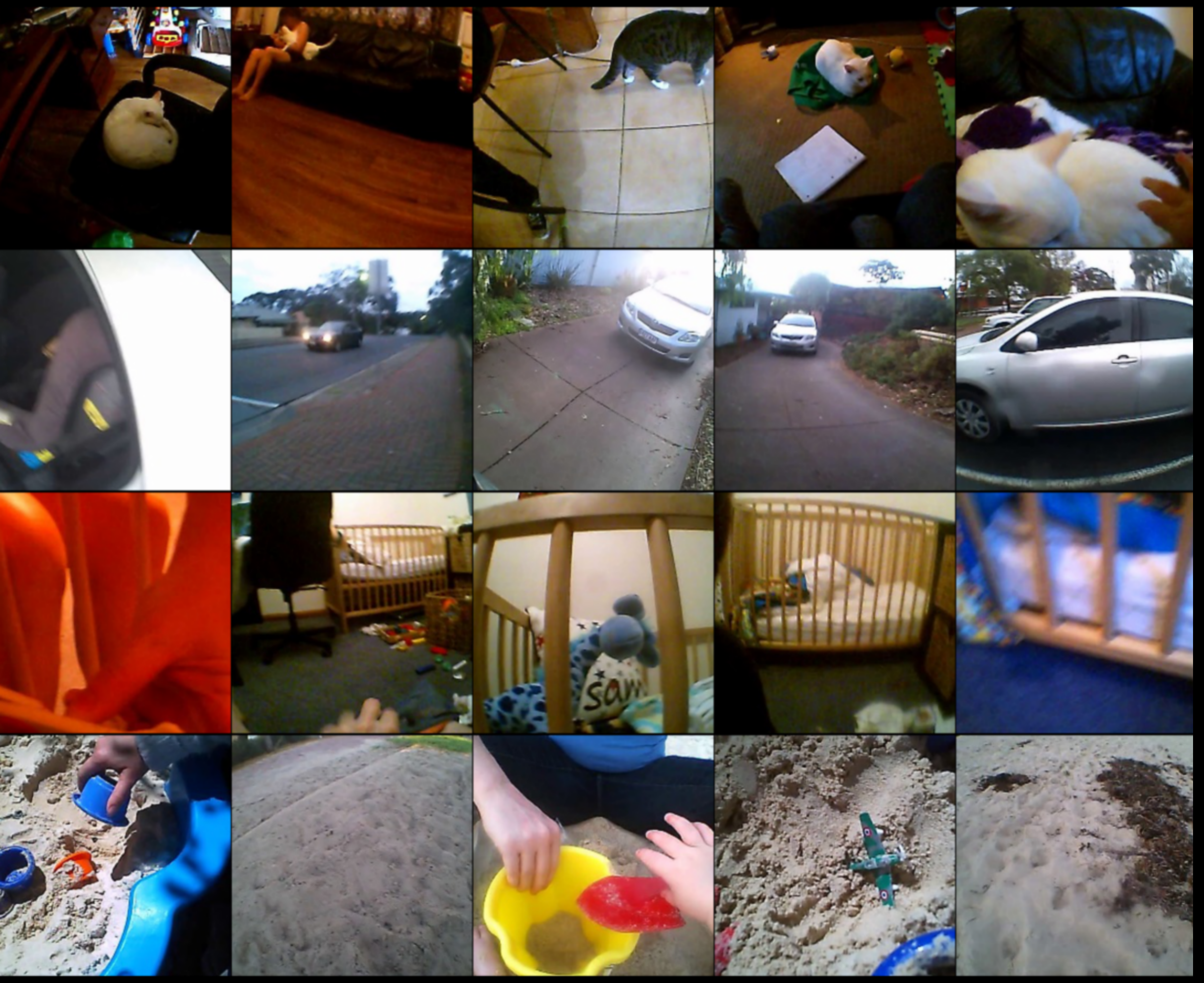
AAAI 2023

Jacob Andreas

lingo.csail.mit.edu

How do we learn?

from observations



[Sullivan et al. 2020]

from exploration

[Legare 2012]

from demonstrations

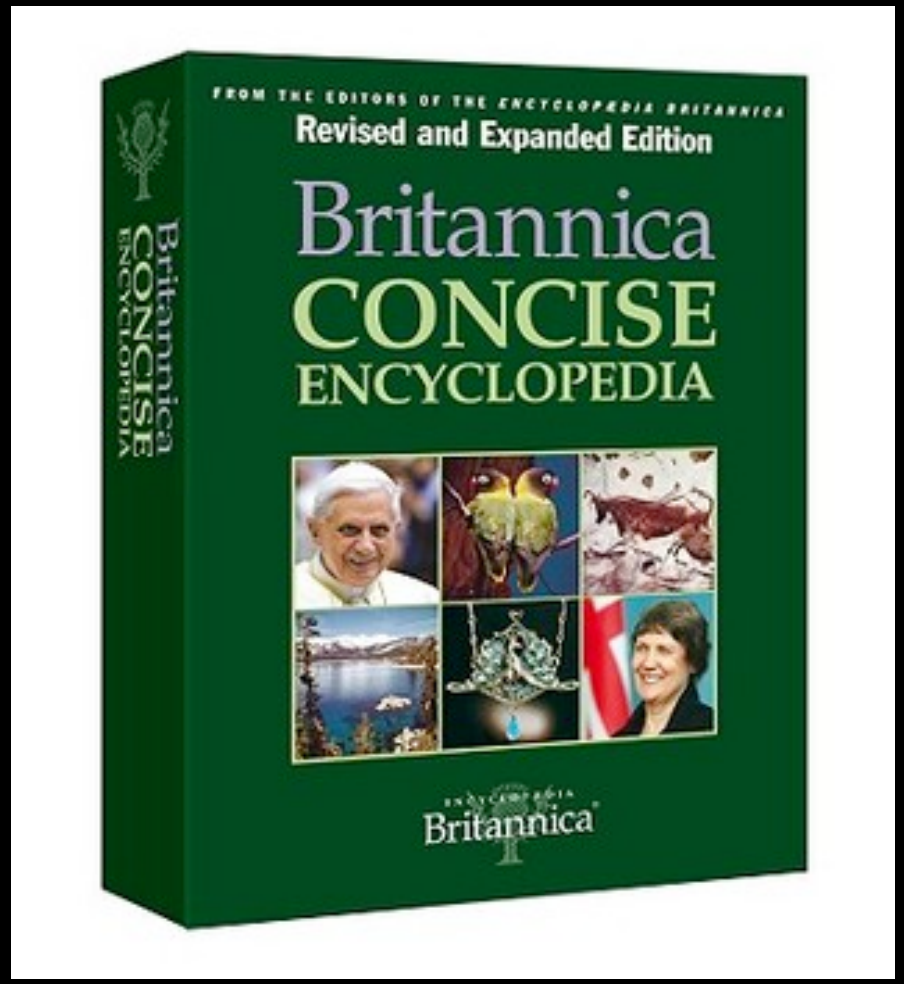
[Buchsbaum et al. 2010]

from language

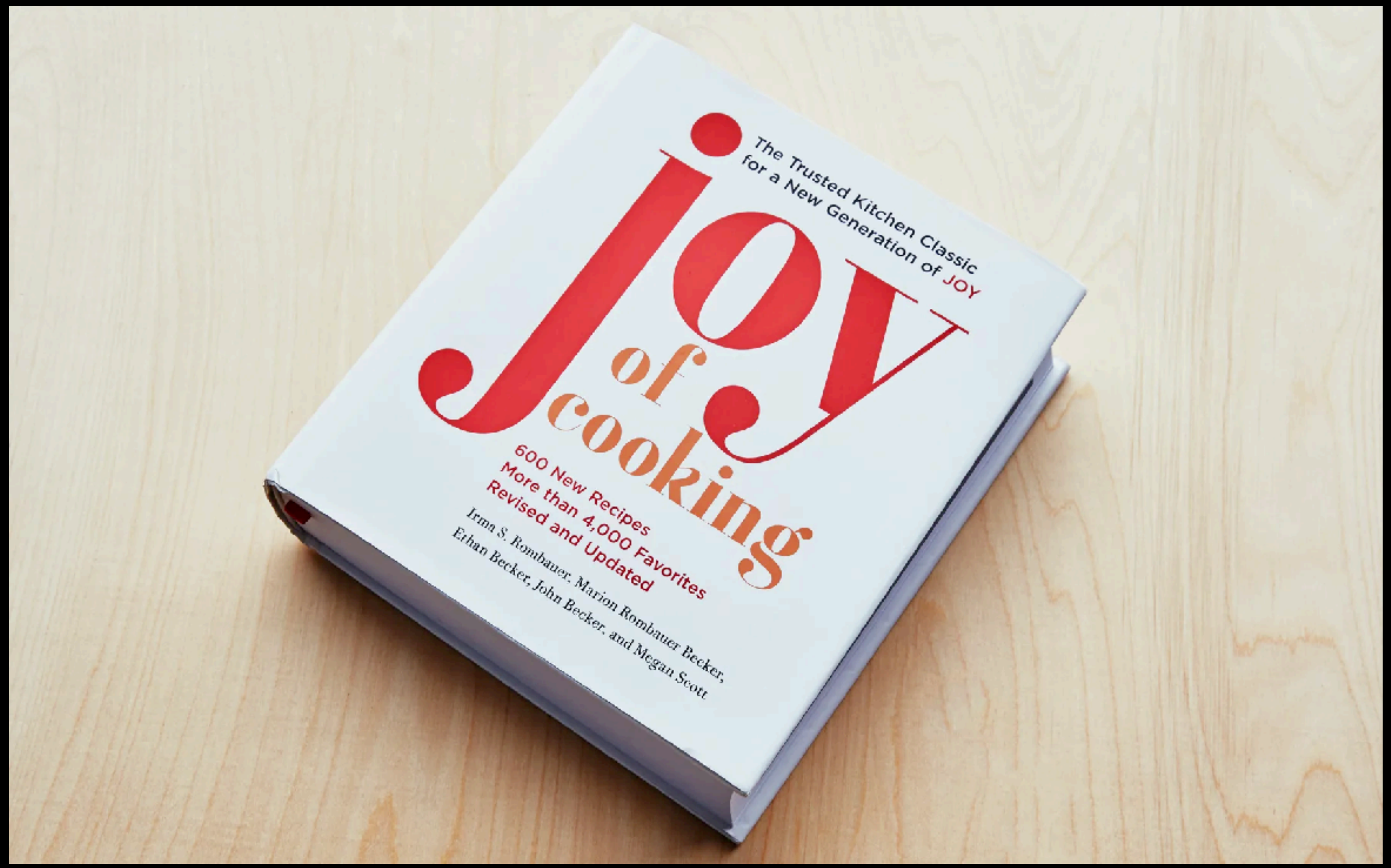
[Morgan et al. 2015]

What do we learn **from language**?

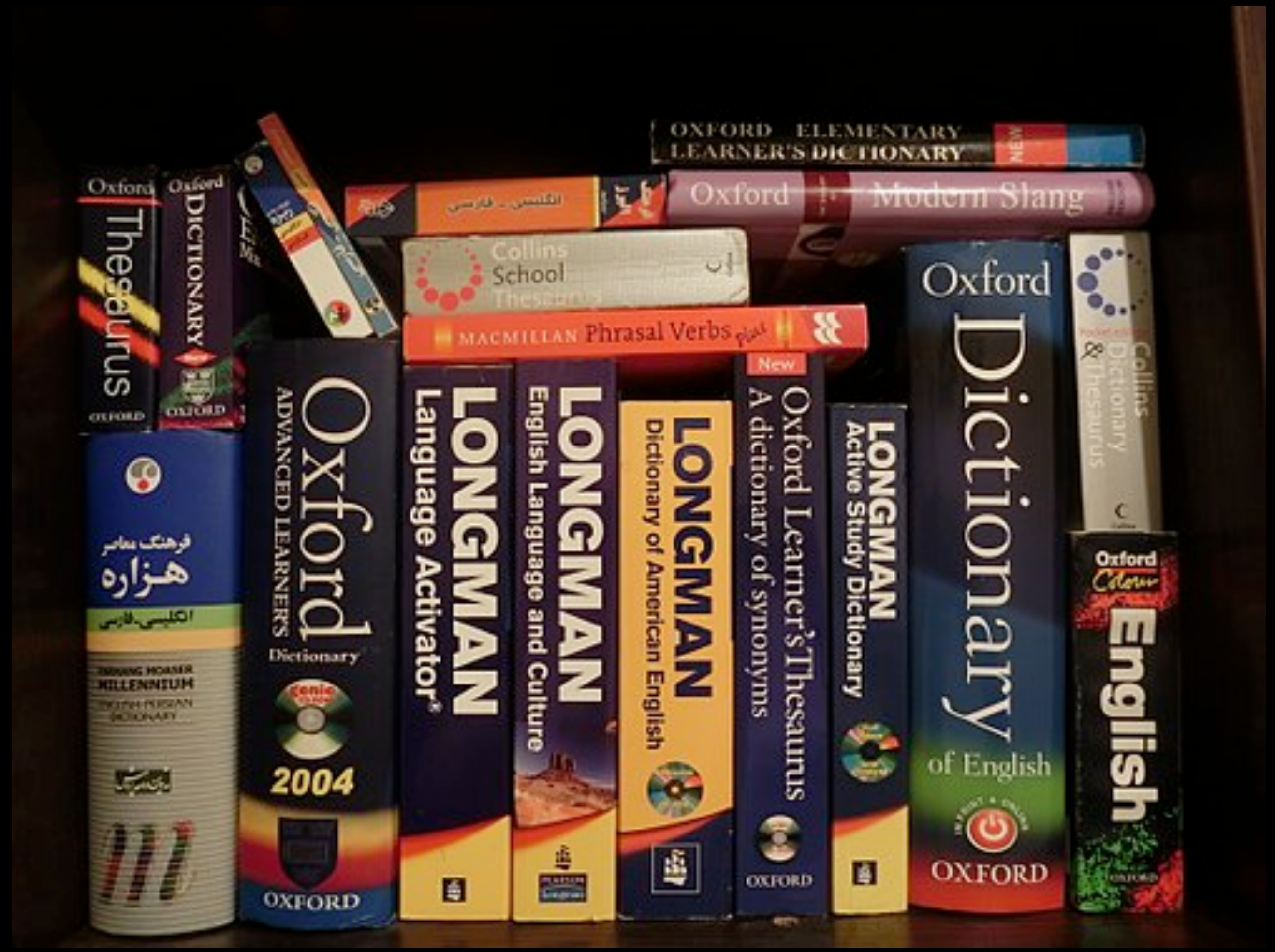
facts



procedures



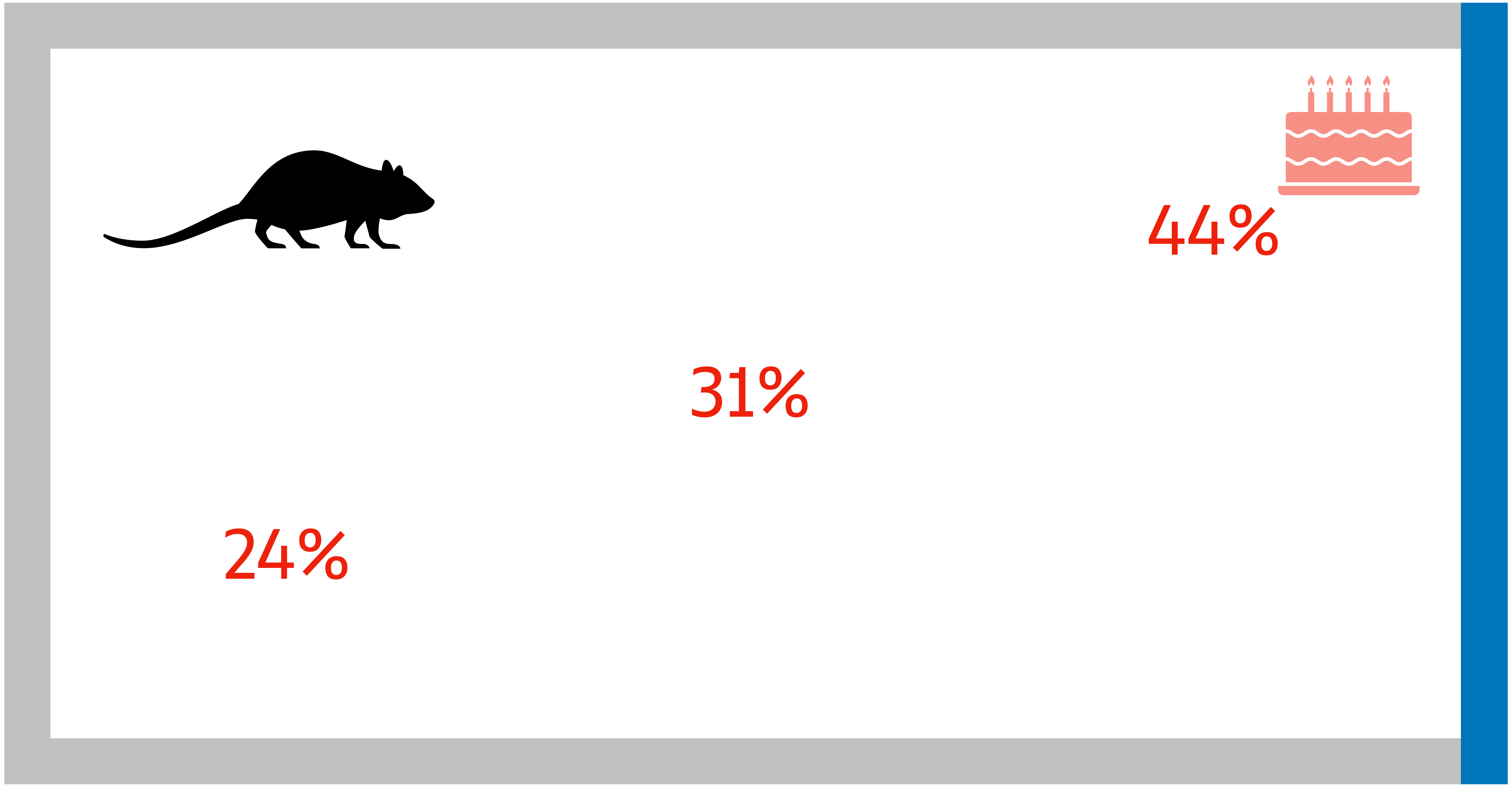
language!



Language in human decision-making

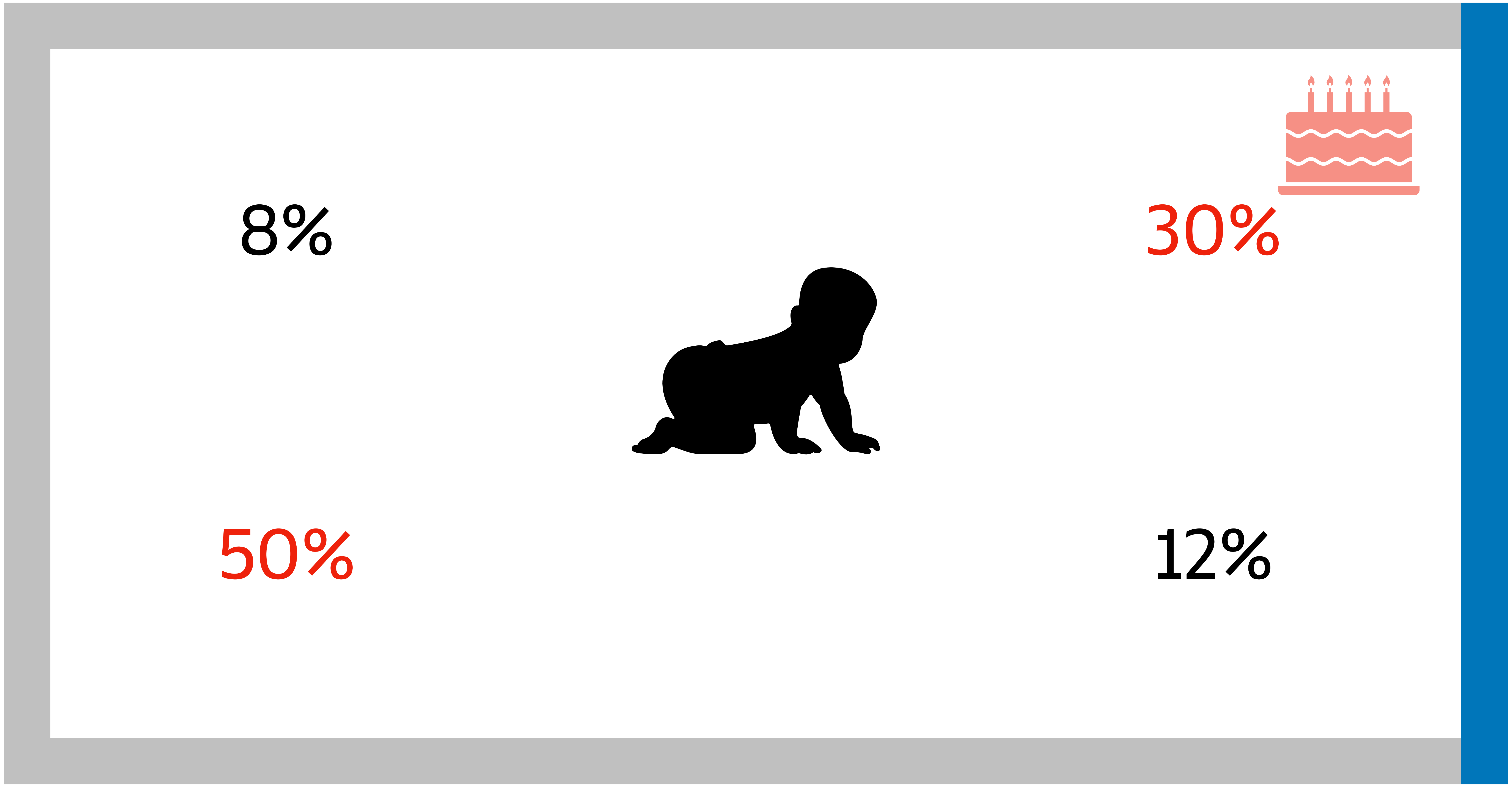
[Hermer-Vazquez, Spelke, Katznelson 1999]

Language in human decision-making



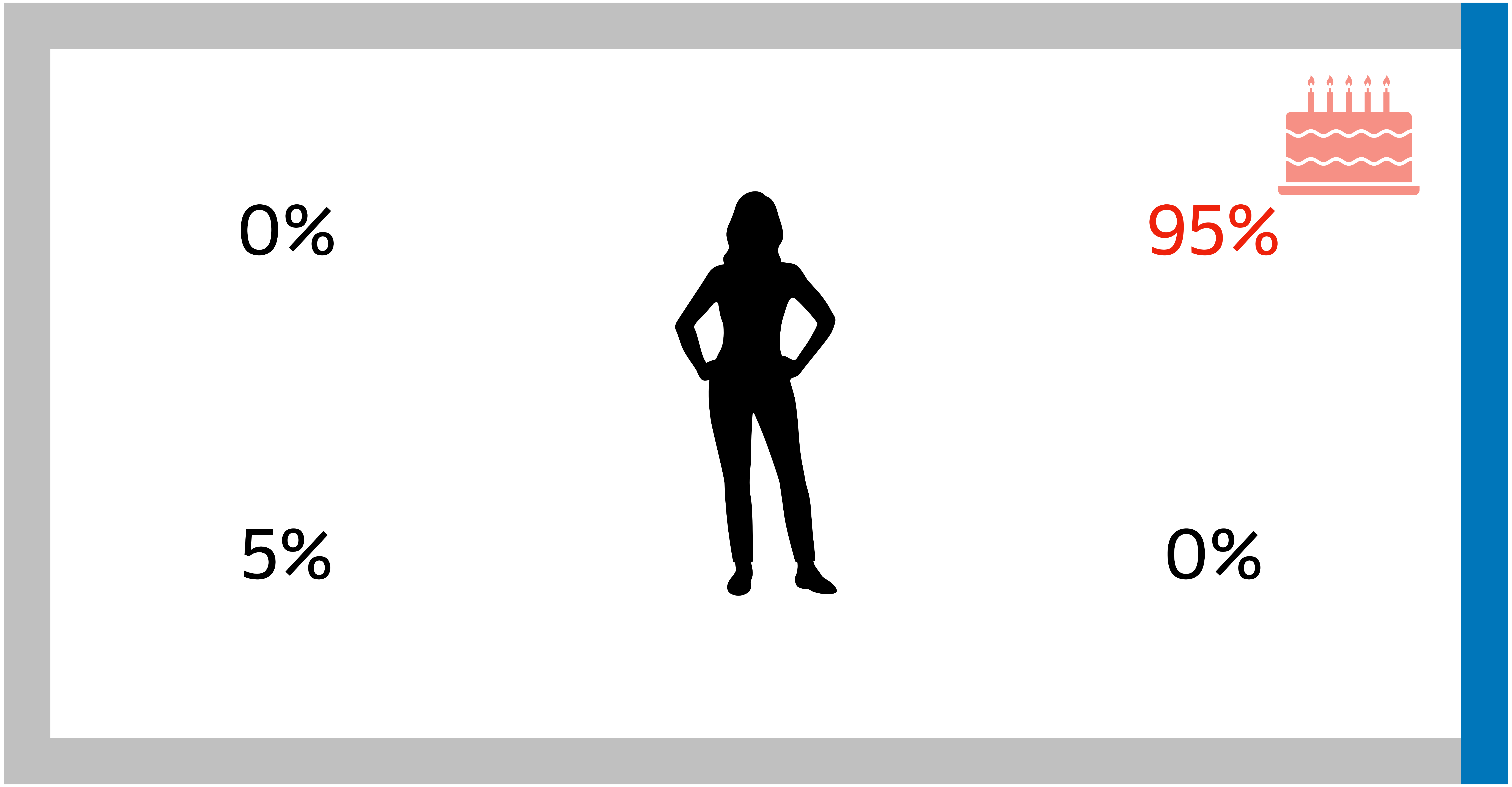
[Hermer-Vazquez, Spelke, Katznelson 1999]

Language in human decision-making



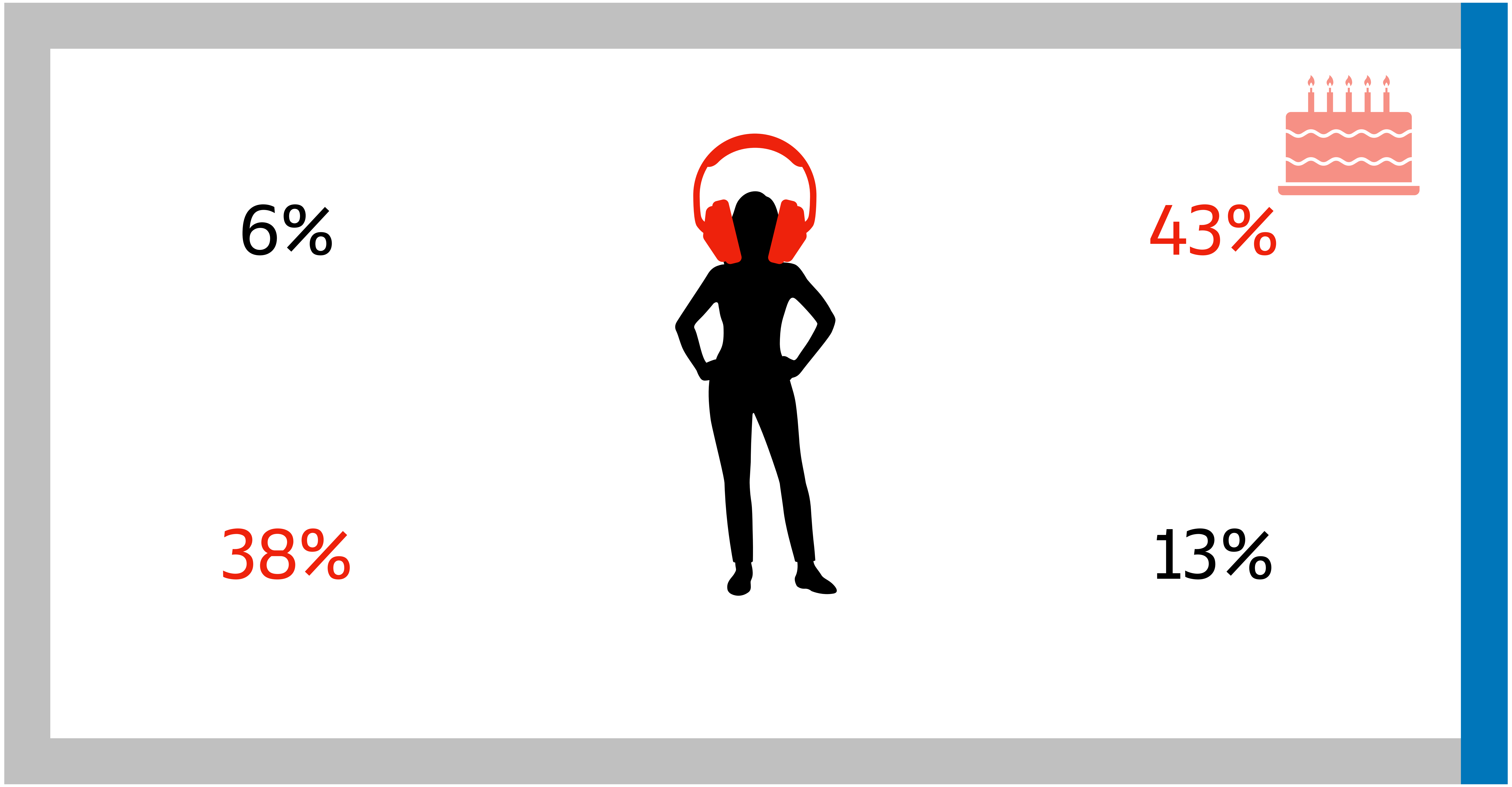
[Hermer-Vazquez, Spelke, Katznelson 1999]

Language in human decision-making



[Hermer-Vazquez, Spelke, Katznelson 1999]

Language in human decision-making



[Hermer-Vazquez, Spelke, Katznelson 1999]

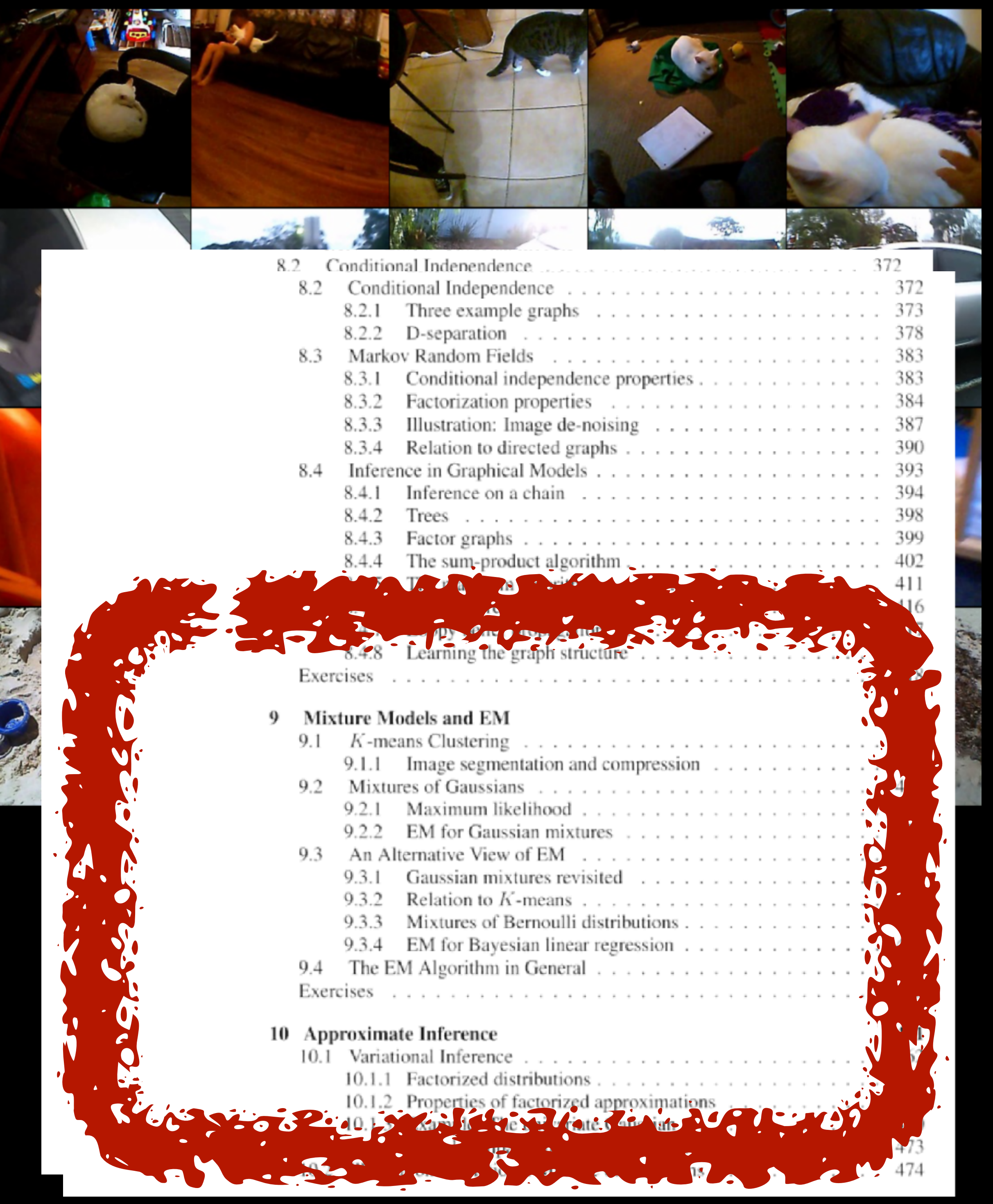
How do machines learn?

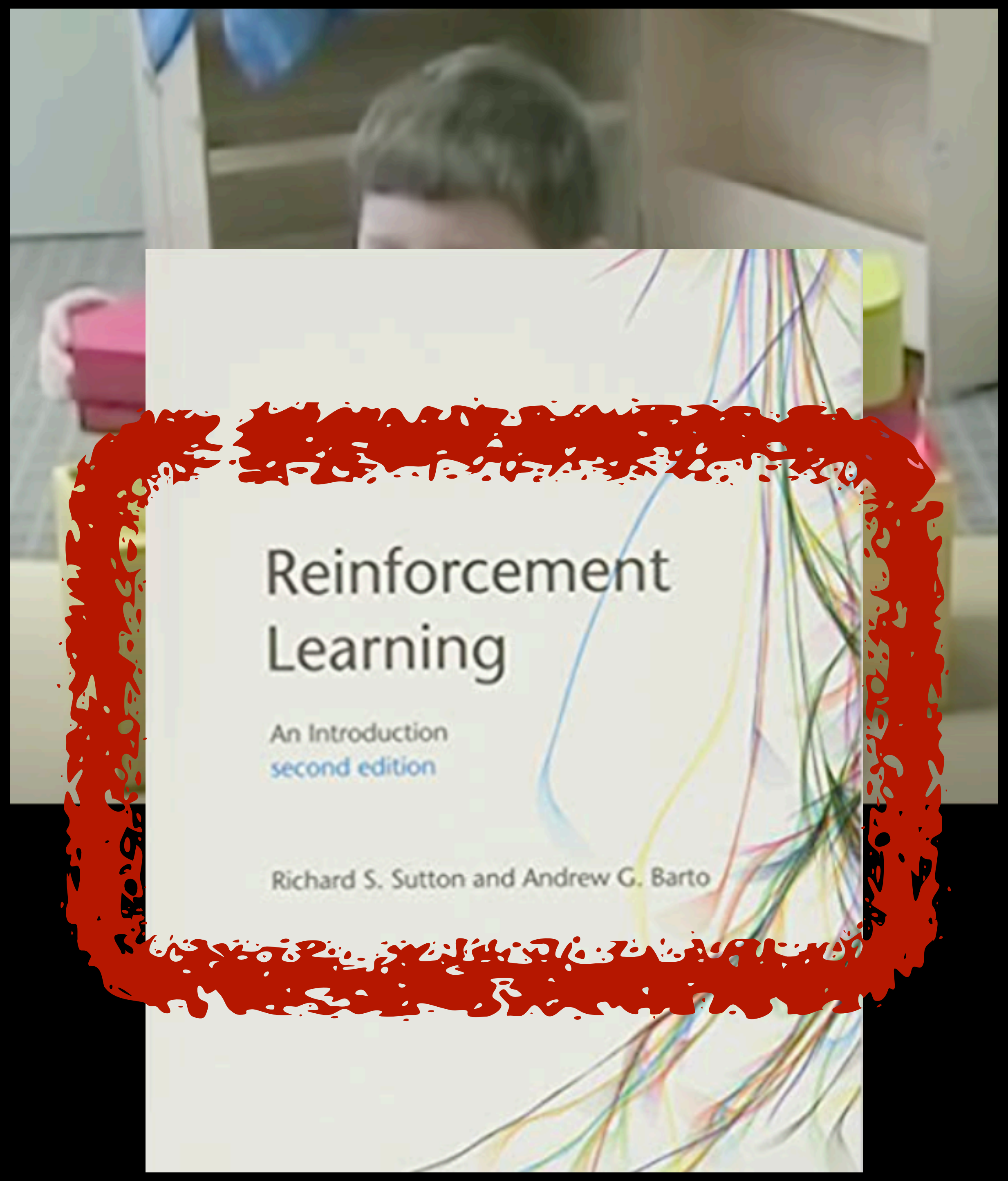
from observations

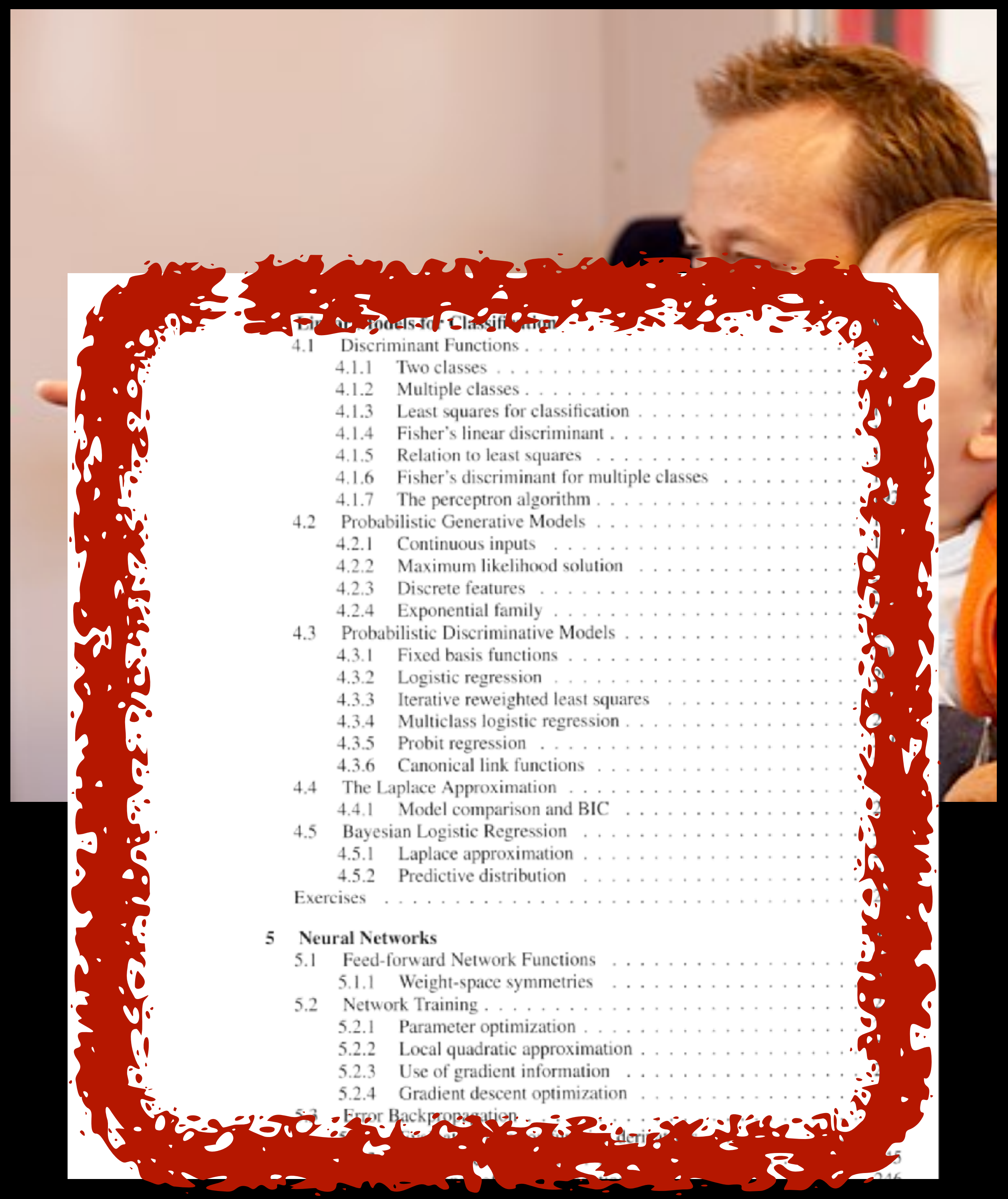
from exploration

from demonstrations

from language







Today's talk

Learning to act

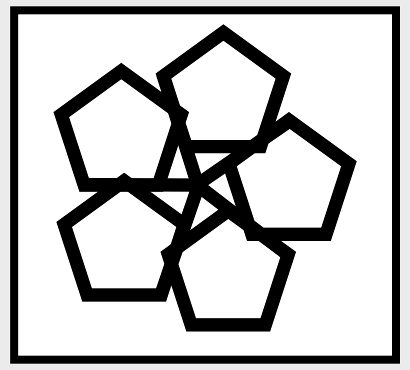
cook an egg →

```
turn(90);
pick(pan);
goto(stove);
place(pan)…
```

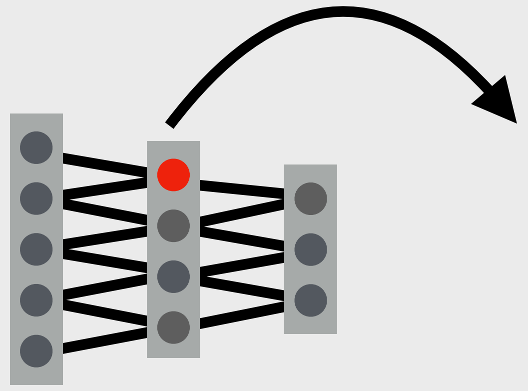
Learning to program

```
(f24 5 (λ
(X) (get/set
(λ (y) (f2 1
(f41 5 y)))
x)) z)
```

←



Learning to explain



dog faces & wheels

Learning skills from demonstrations and instructions

Pratyusha Sharma

+ Antonio Torralba

[Skill Induction & Planning w/ Latent Language. ACL 2022.]

https://www.dailymail.co.uk/sciencetech/article-6591753

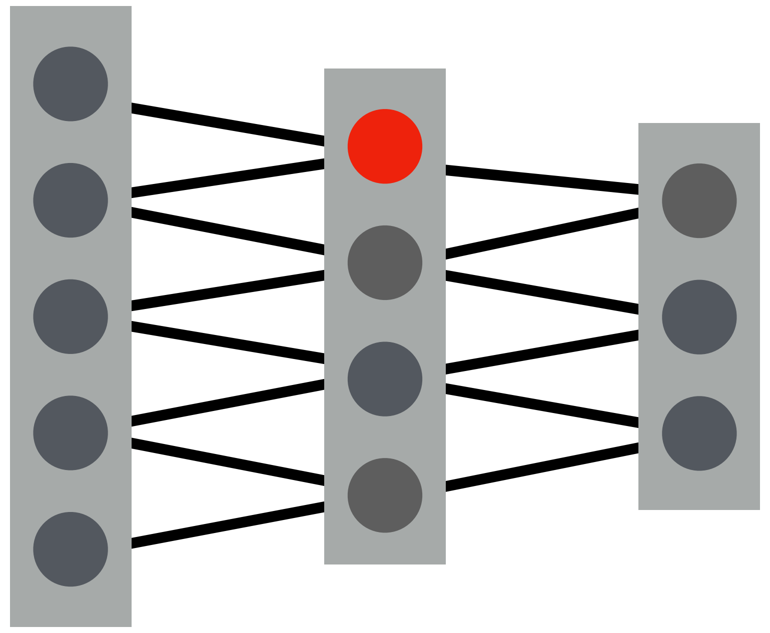
Goal: "Put a clean bowl of water on the kitchen island"

[Shridhar et al. 2020]

Learning to act

```
put a sliced tomato on the
kitchen counter
```

Can we train a fixed model to map goals → actions?

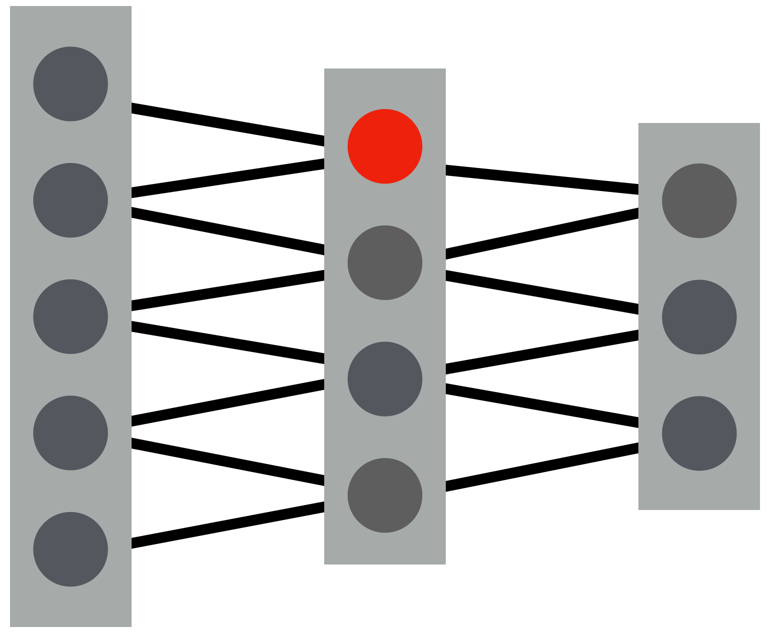


```
turn(left)
```

Learning to act

put a sliced tomato on the kitchen counter

Can we train a fixed model to map goals → actions?



`turn(left)`

0% success rate!

Long horizon tasks

put a sliced tomato on the kitchen counter

↓

look(down) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) rotate(right) forward(10)
forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10)
forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) rotate(right) look(down)
pick(knife) look(up) rotate(right) forward(10) forward(10) rotate(left) forward(10) forward(10) rotate(right
forward(10) look(down) slice(tomato) look(up) rotate(left) rotate(left) forward(10) rotate(right) forward(10)
forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10)
forward(10) rotate(right) look(down) put(knife,sink) look(up) rotate(left) forward(10) forward(10) forward(10)
forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) rotate(left)
forward(10) look(down) pick(sliced(tomato)) look(up) rotate(left) forward(10) rotate(right) forward(10)
forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10)
rotate(right) look(down) open(fridge) put(sliced(tomato),fridge) close(fridge) open(fridge) pick(sliced(tomato))
close(fridge) look(up) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10) forward(10)
forward(10) forward(10) forward(10) rotate(left) put(sliced(tomato),counter)

Hierarchical policies

Explicitly decomposing tasks into subtasks makes this problem easier…

```
put the knife in the drawer
    find(knife)
        turn(left)
        forward(10)
        turn(right)
    grasp(knife)
        open(box)
        grasp(knife)
        ...
    ...
```

[Hand-engineered hierarchies: Parr & Russell, 1998; Andre & Russell, 2002]

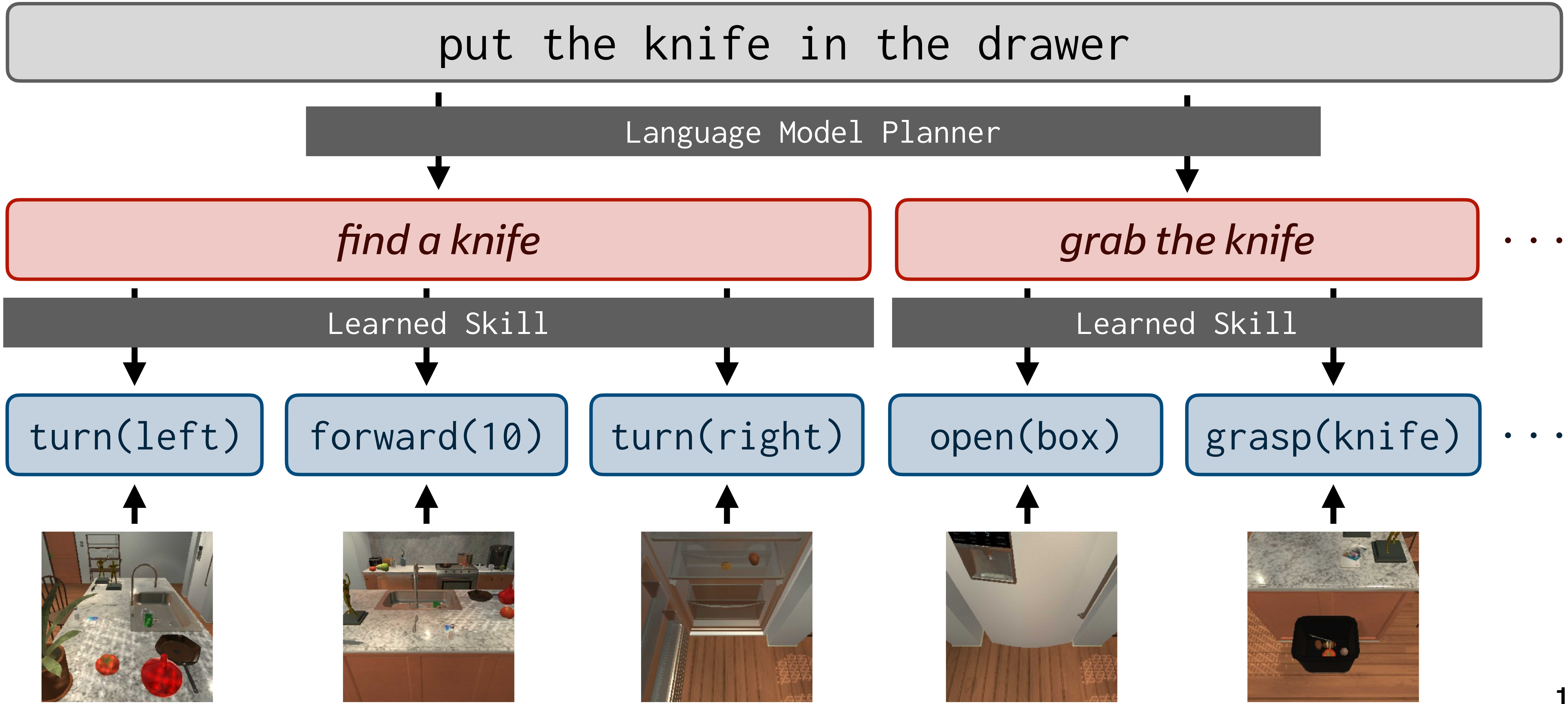
[Supervised training of sub-policies: Kearns & Singh 02, Kulkarni et al. 16]

[Fully unsupervised: Stolle & Precup 02, Fox & Krishnan et al. 16]

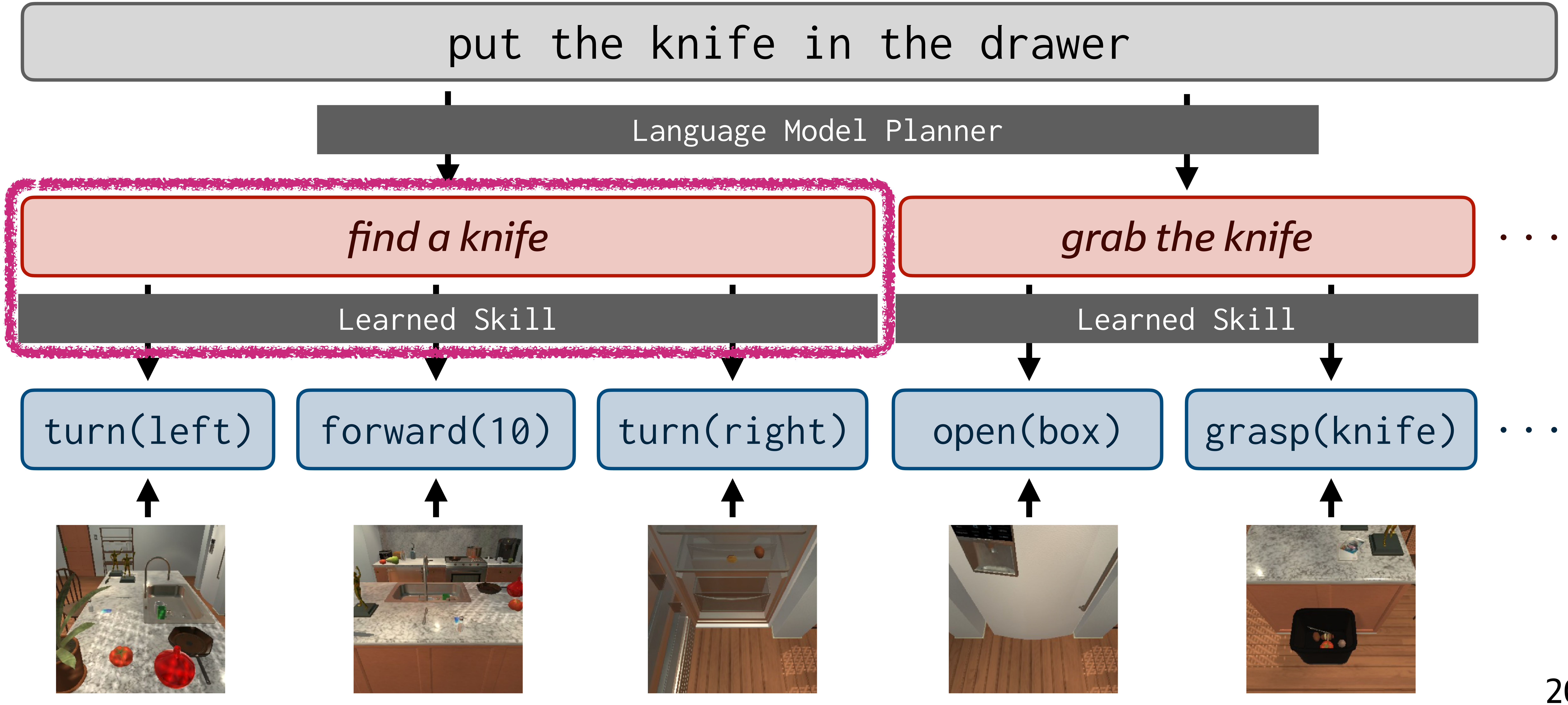
…but existing methods require lots of domain-specific engineering.

How can language help?

A hierarchical policy with **latent language**



Language as a repr. of composable skills



Instructions as a source of easy supervision

put the knife in the drawer

Language Model Planner

find a knife | *grab the knife* . . .

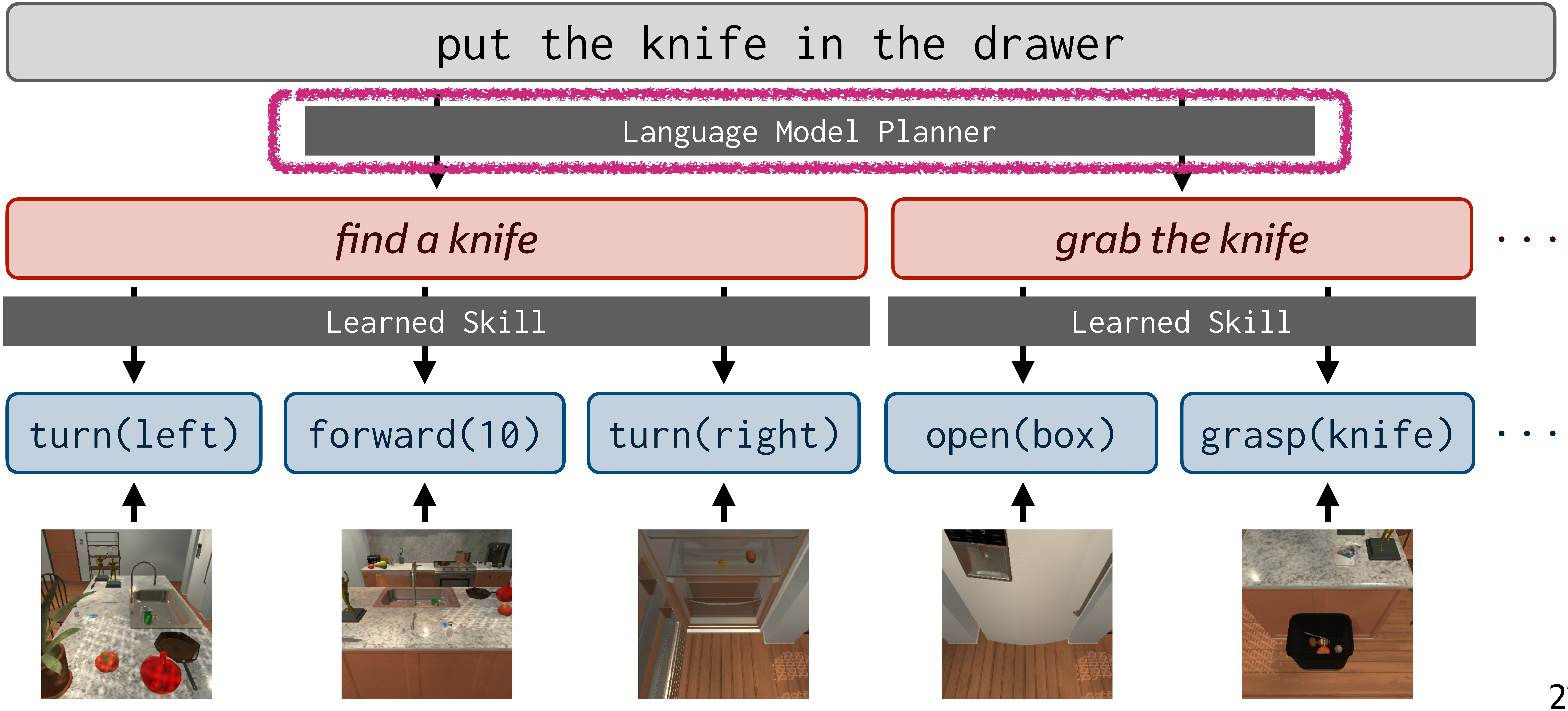
Learned Skill | Learned Skill

turn(left) | forward(10) | turn(right) | open(box) | grasp(knife) . . .





Text corpora as priors on plausible plans



How can language help?

Representation of composable skills

Supervision for a planning model

Generalization to novel goals

Learning to plan and act with language

`put the knife in the drawer`

`turn(left)` `forward(10)` `turn(right)` `open(box)` `grasp(knife)` . . .





Learning to plan and act with language

`put the knife in the drawer`

find a knife | *grab knife* | *go to the drawer* | . . .

?

`turn(left)` | `forward(10)` | `turn(right)` | `open(box)` | `grasp(knife)` | . . .





Learning to plan and act with language

`put the knife in the drawer`

find a knife | *grab knife* . . .

`turn(left)` `forward(10)` . . .

Annotated demonstrations

`cook the egg`

`grasp(egg)` `forward(9)` . . .

Unannotated demonstrations (x9)

Learning to plan and act with language

`put the knife in the drawer`

find a knife | *grab knife* | . . .

`turn(left)` `forward(10)` . . .

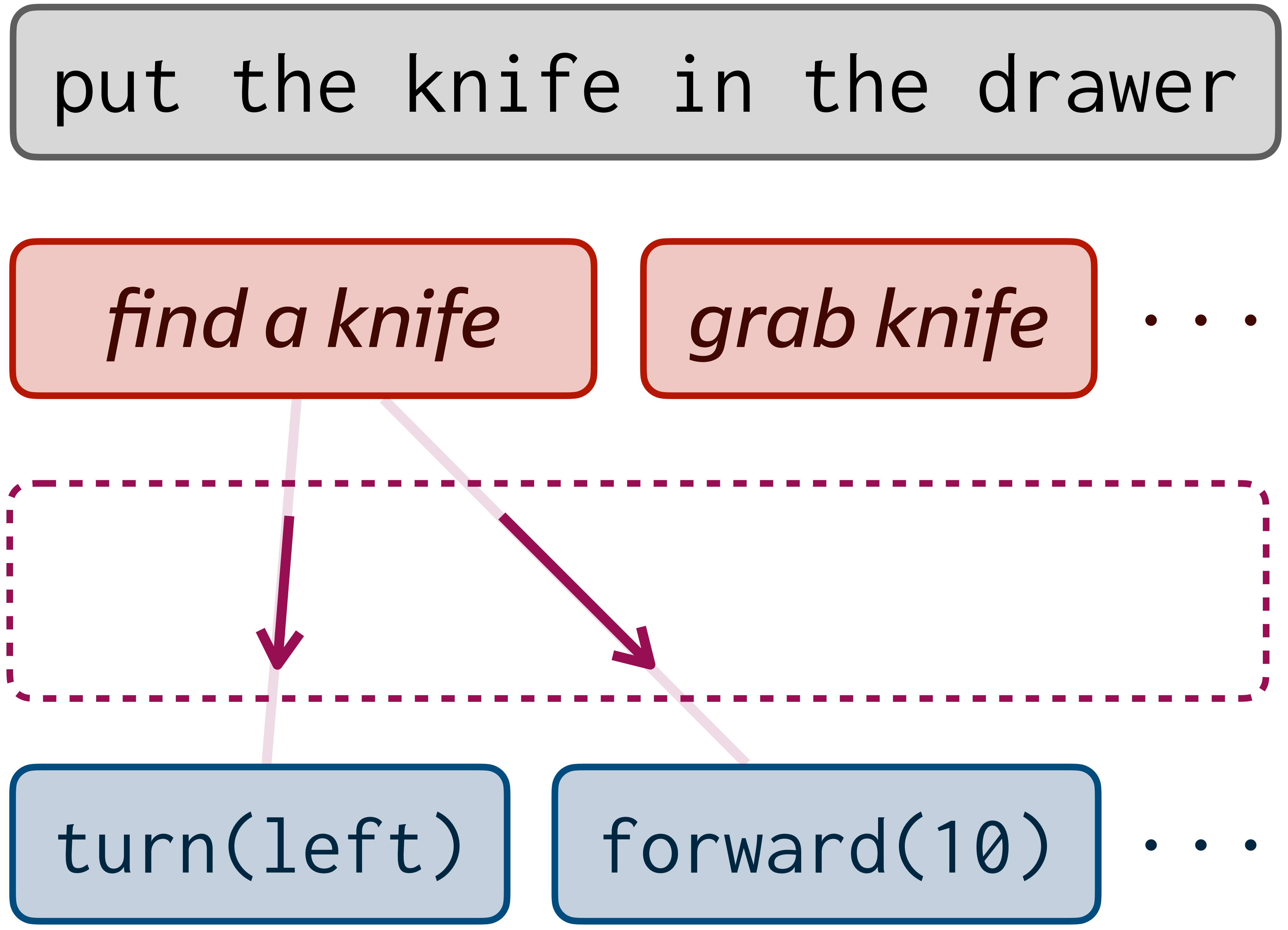
Latent alignments!

`cook the egg`

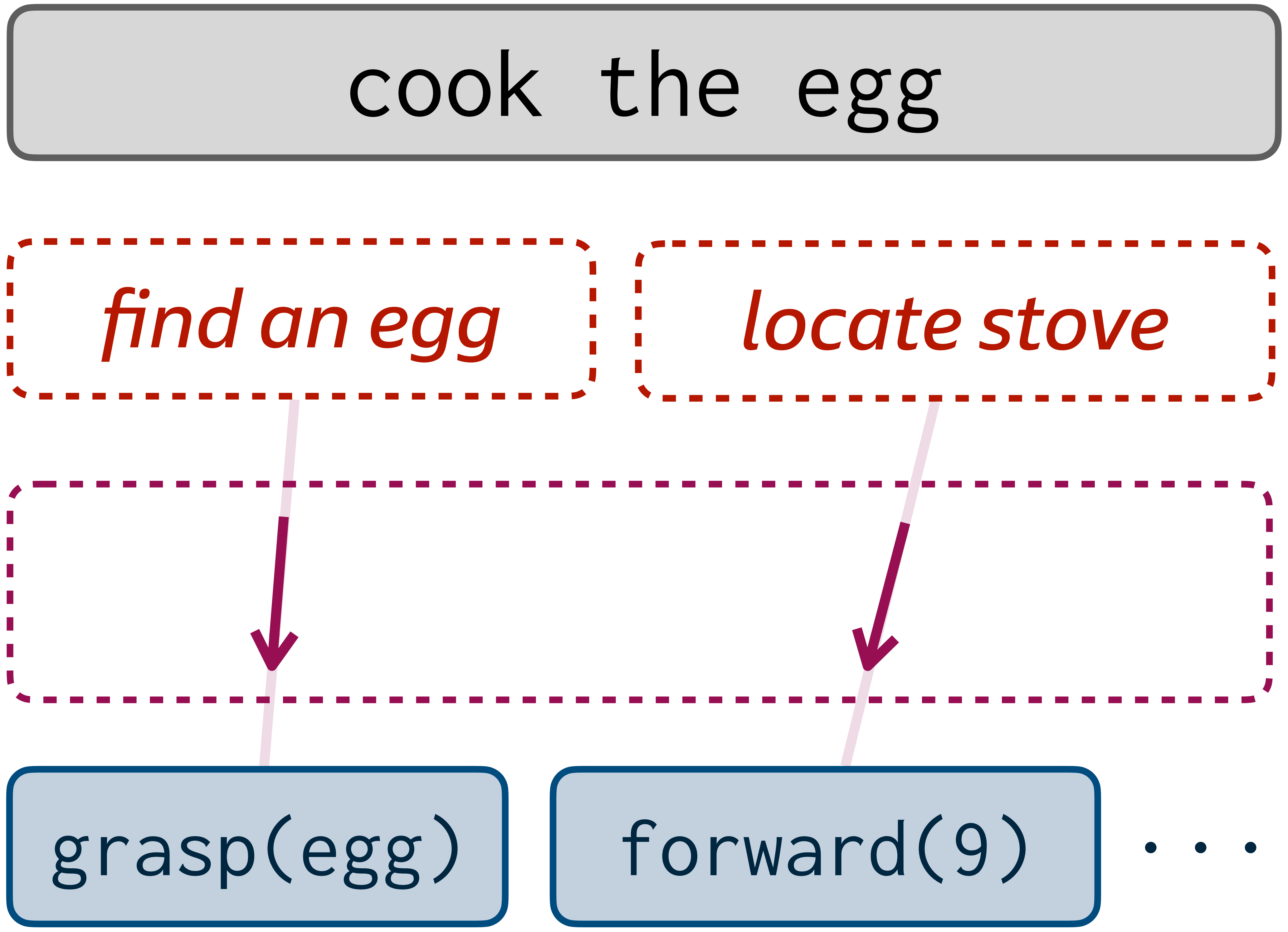
`grasp(egg)` `forward(9)` . . .

Latent plans and alignments!

Learning to plan and act with language

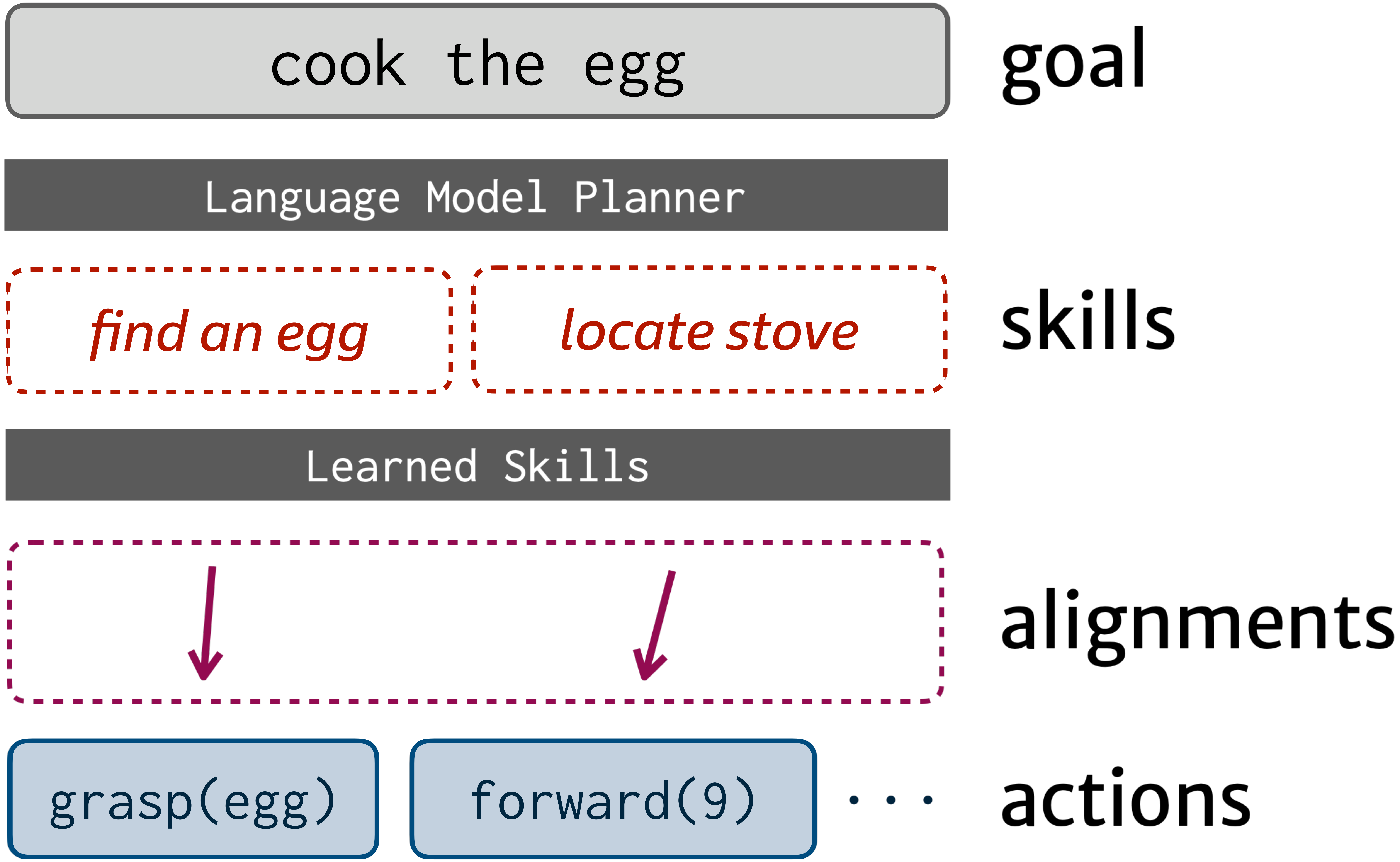


Latent alignments!

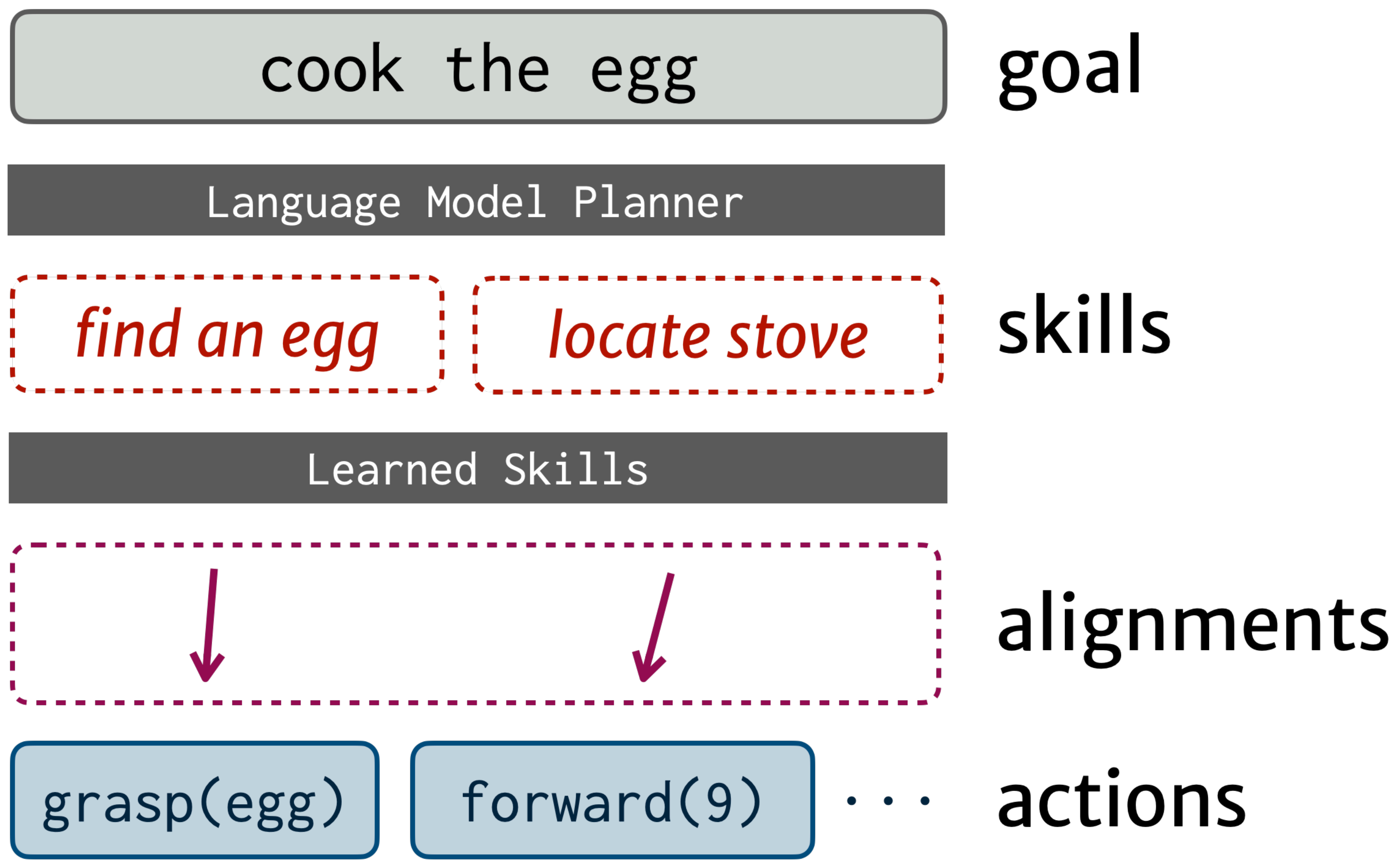


Latent plans and alignments!

(SL)^3: Semi-supervised skill learning w/ latent lang.



(SL)^3: Semi-supervised skill learning w/ latent lang.



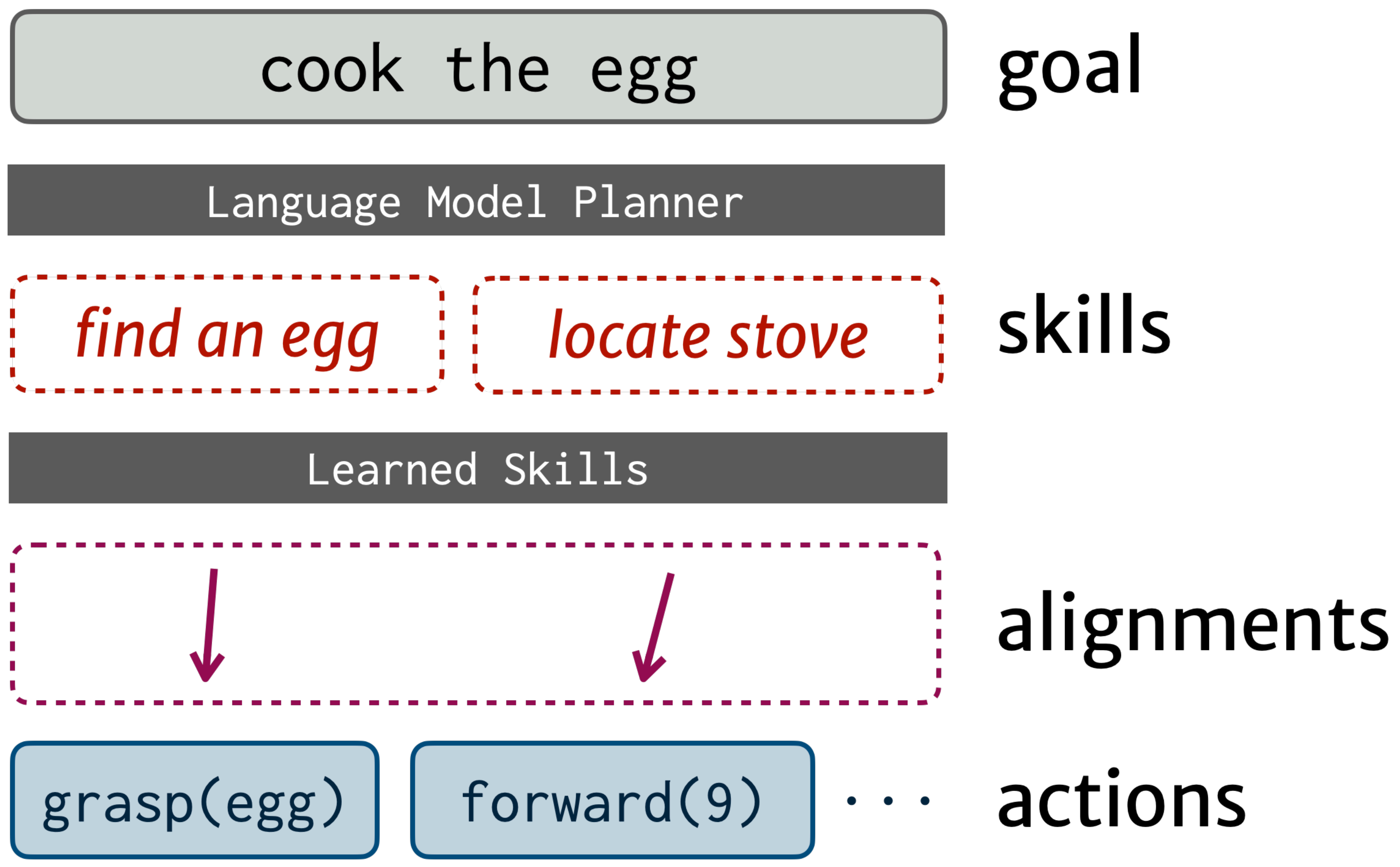
$$\underset{\substack{\theta \\ \text{alignments} \\ \text{subtasks}}}{\operatorname{argmax}} \log p_\theta(\boxed{\text{goal}}, \boxed{\text{subtasks}}, \text{alignments}, \boxed{\text{actions}})$$

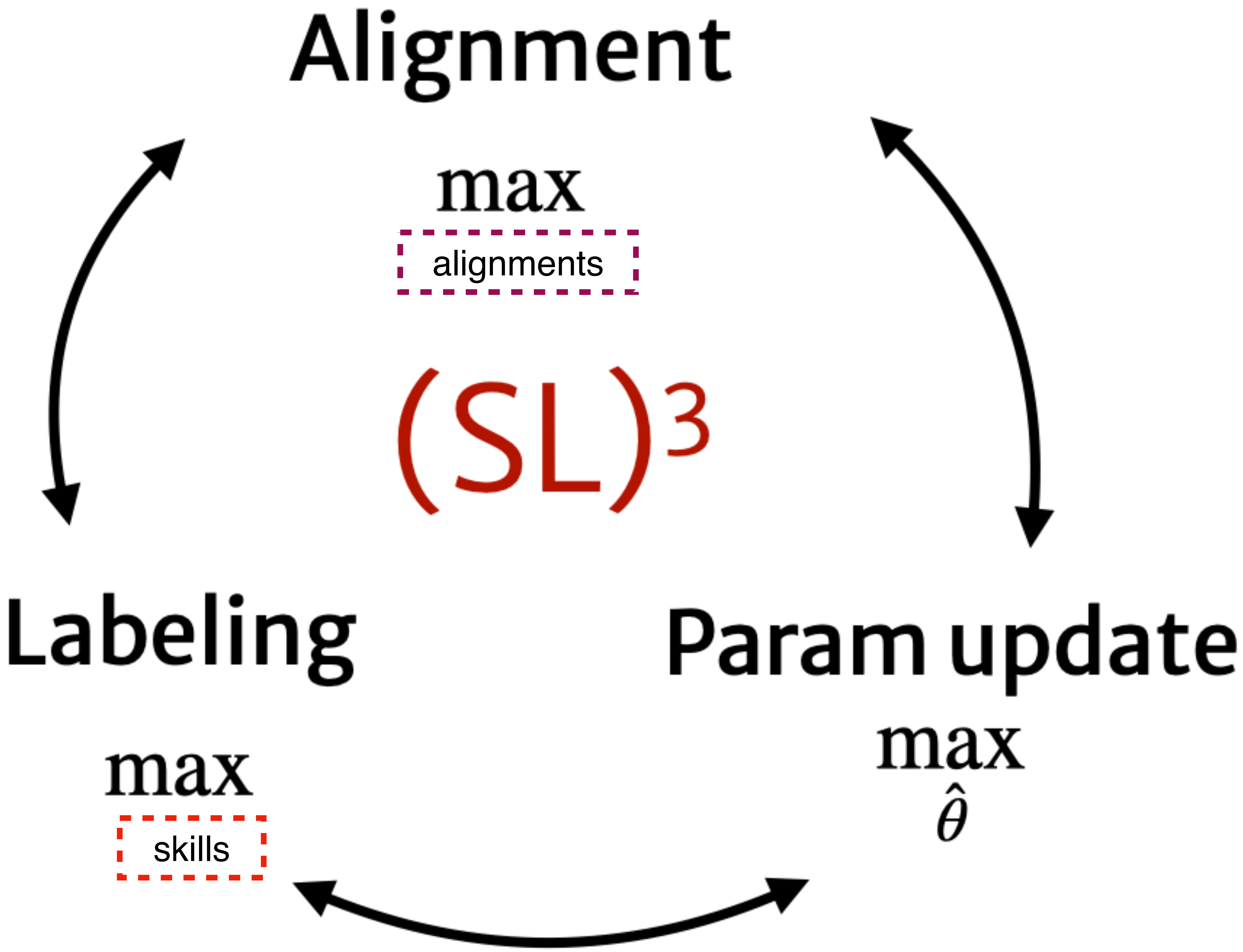
annotated demos

$$+ \log p_\theta(\boxed{\text{goal}}, \text{subtasks}, \text{alignments}, \boxed{\text{actions}})$$

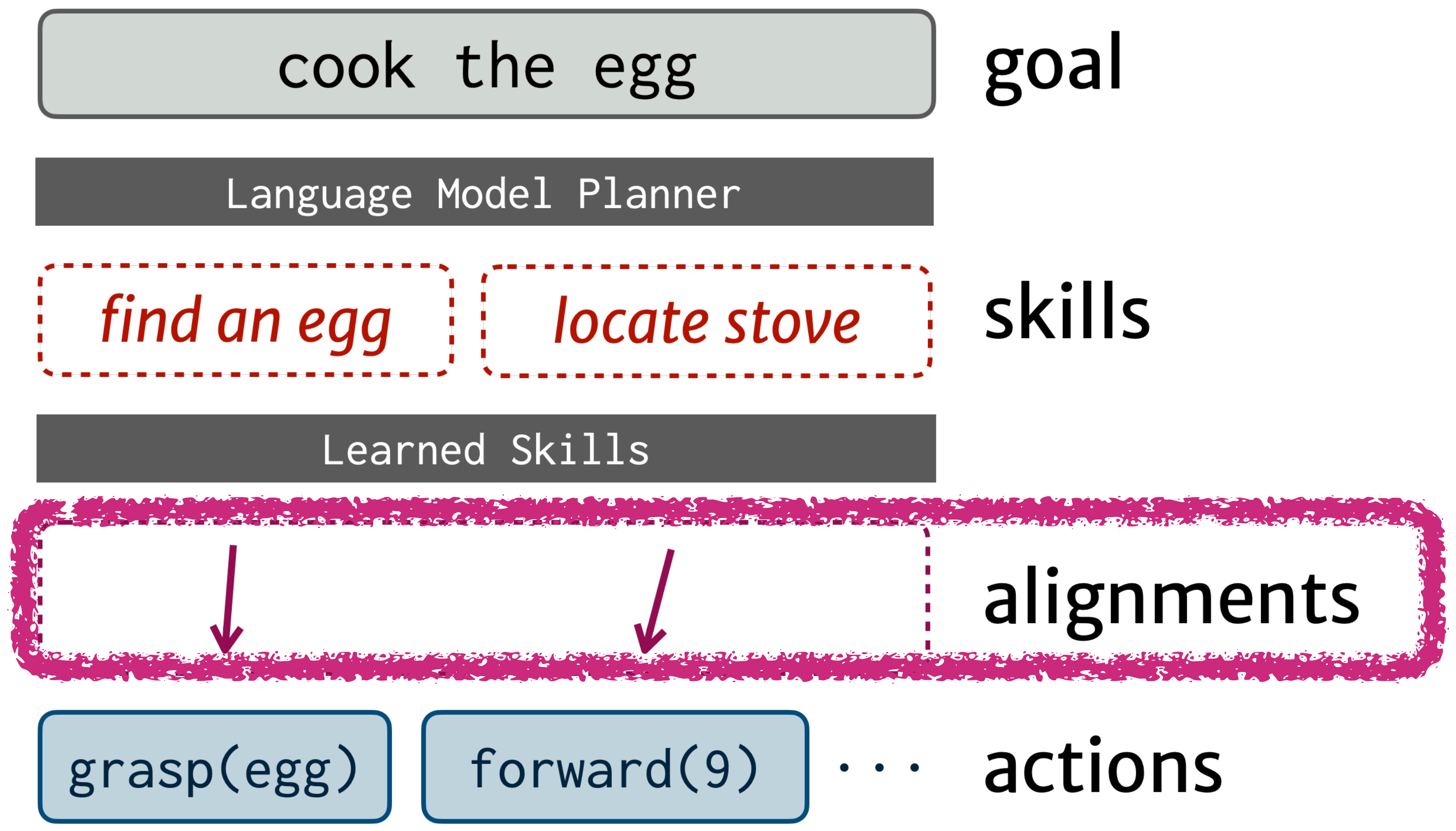
unannotated demos

(SL)³: Semi-supervised skill learning w/ latent lang.

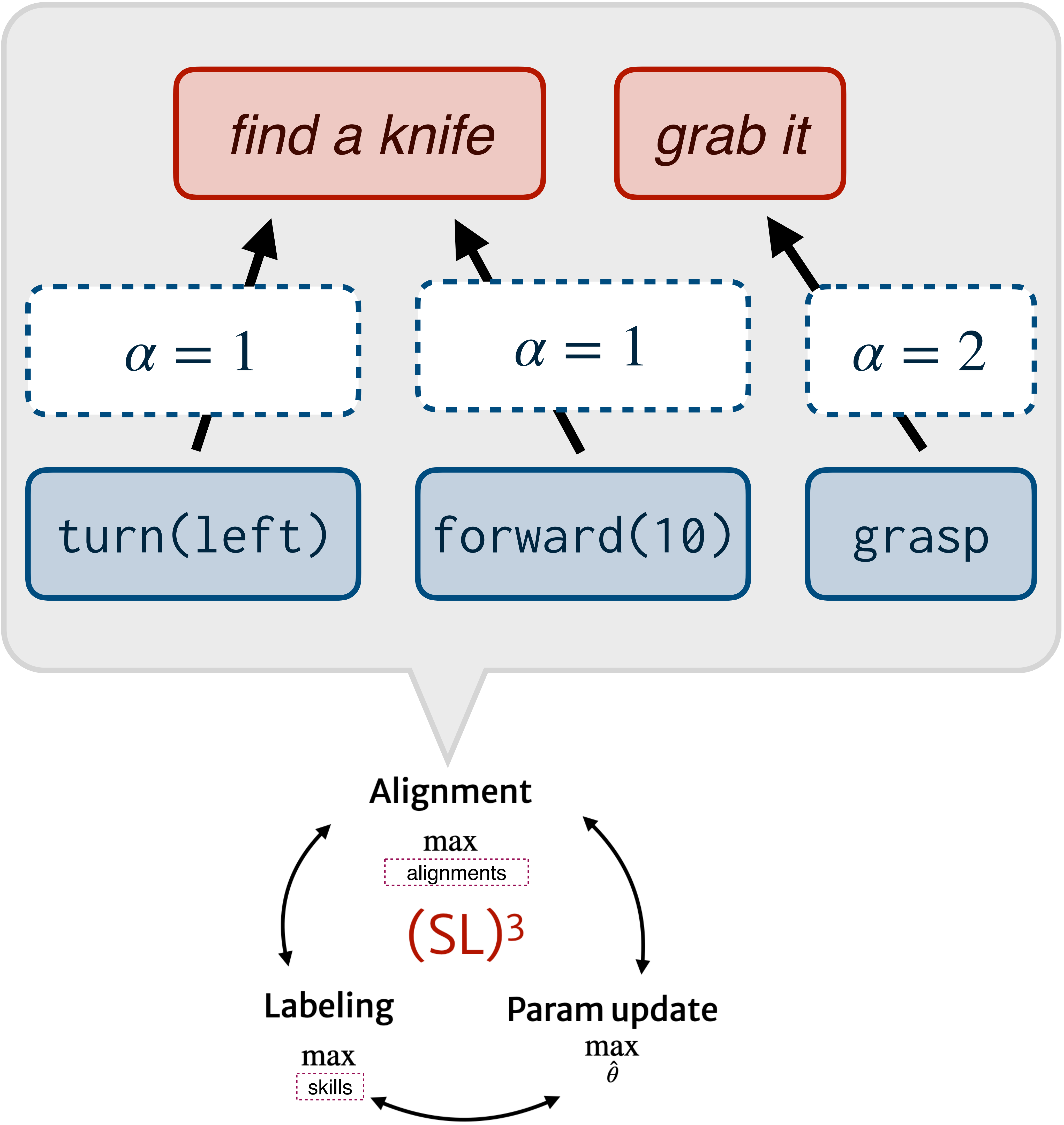




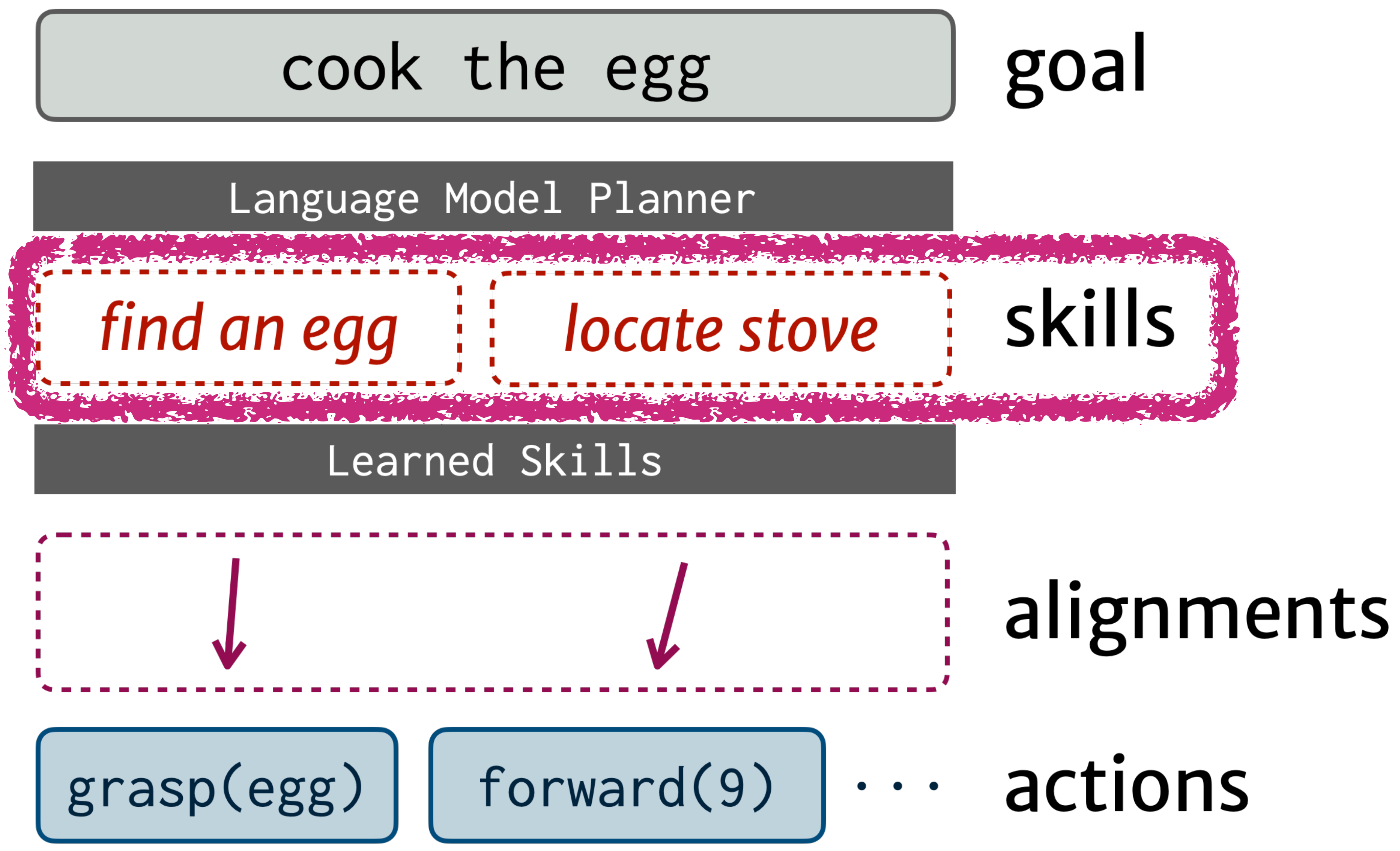
$(SL)^3$: Semi-supervised skill learning w/ latent lang.



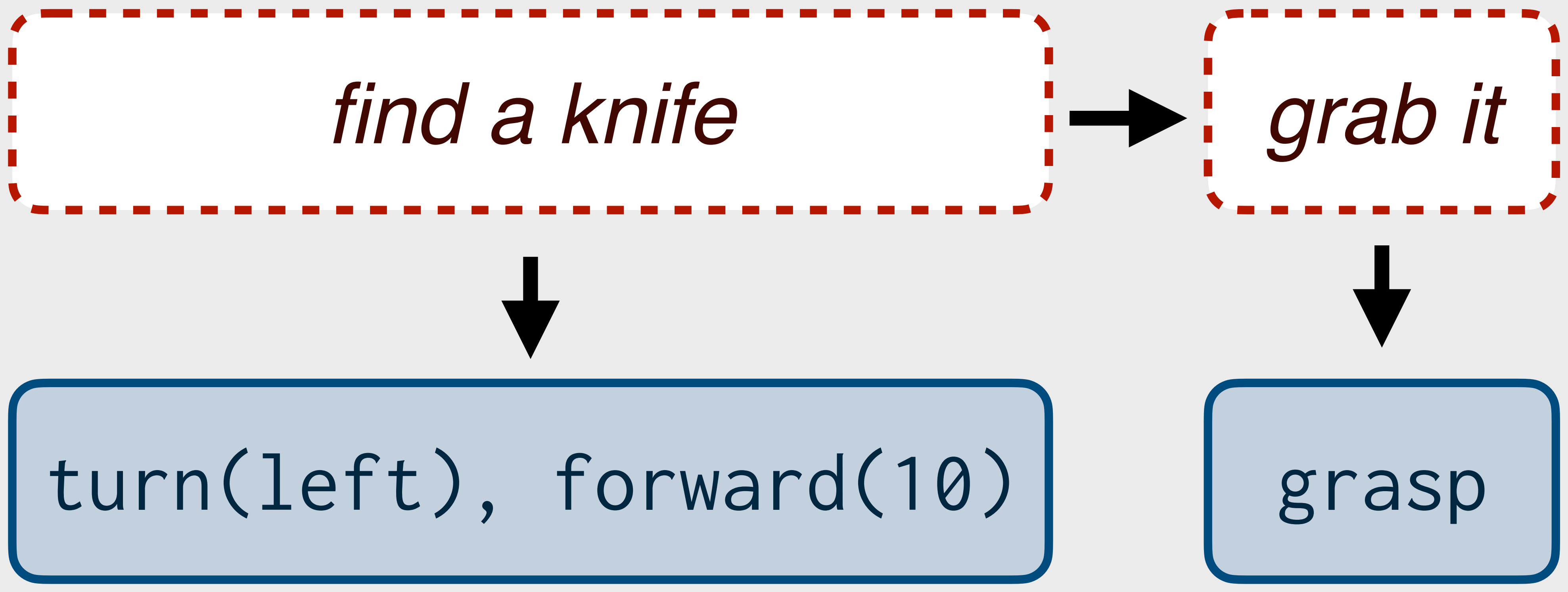
1. Improve alignments



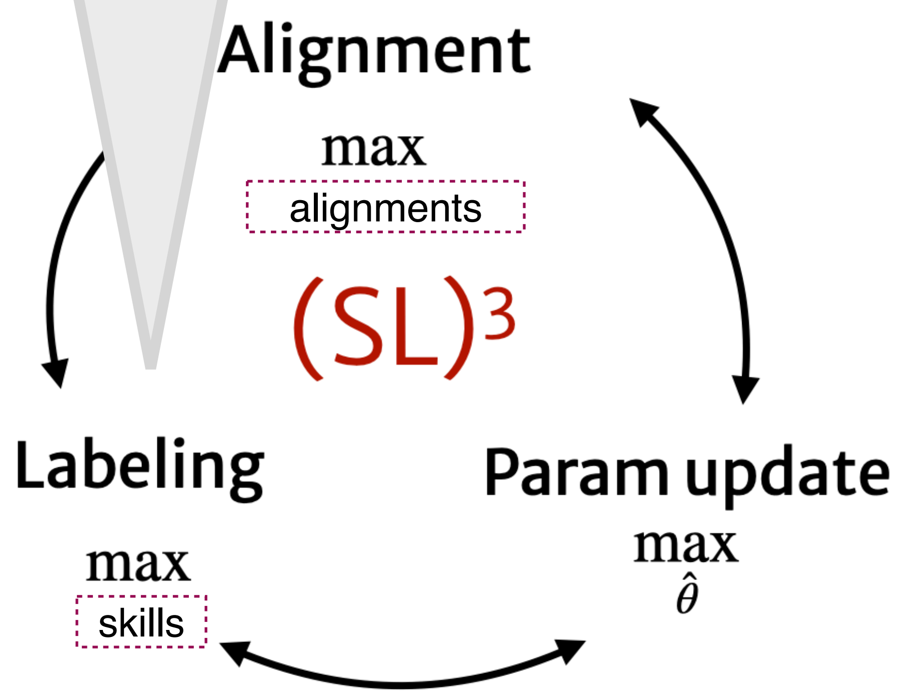
$(SL)^3$: Semi-supervised skill learning w/ latent lang.



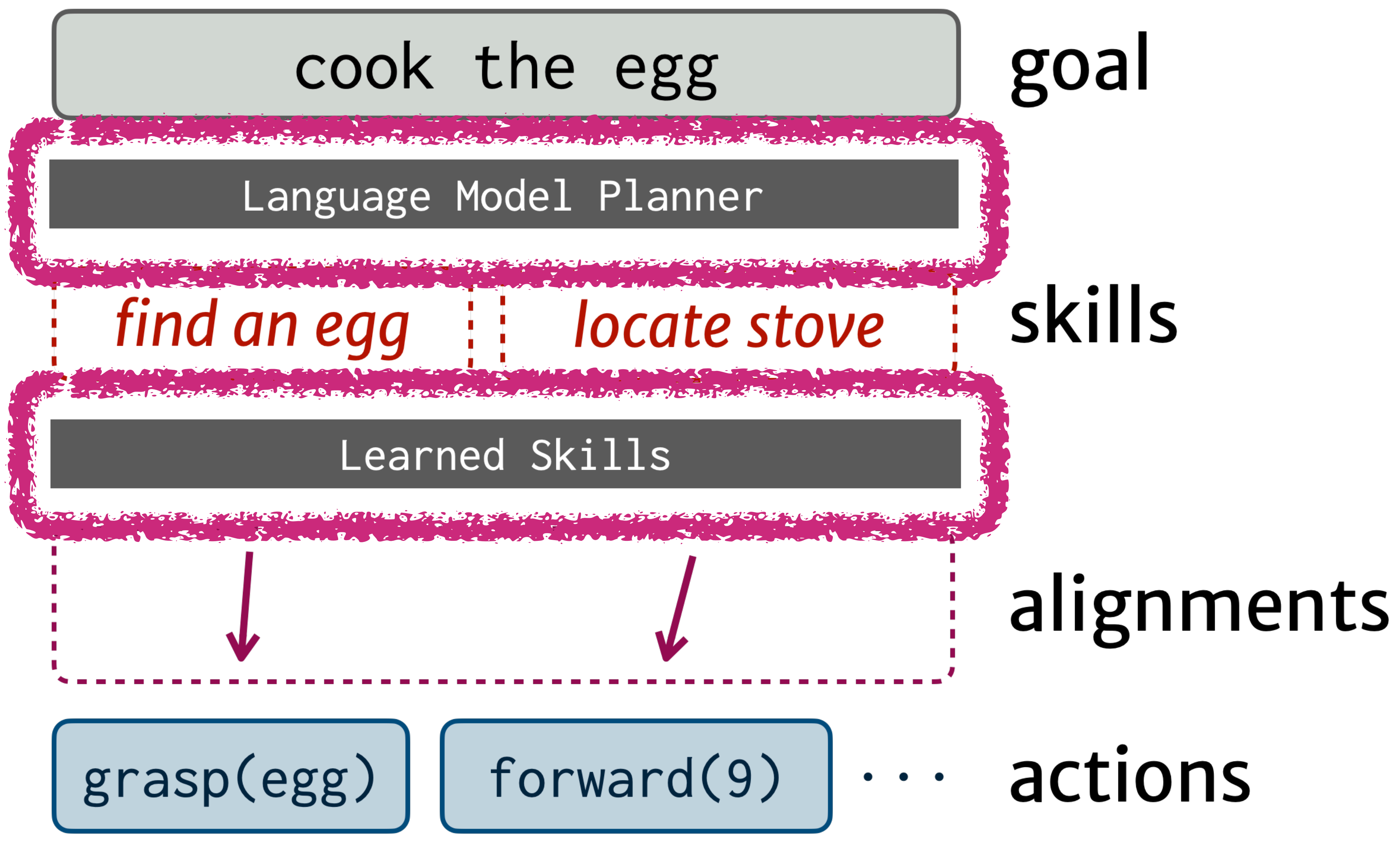
2. Improve labels



Looks like inference in an HMM with every NL string as a possible hidden state!

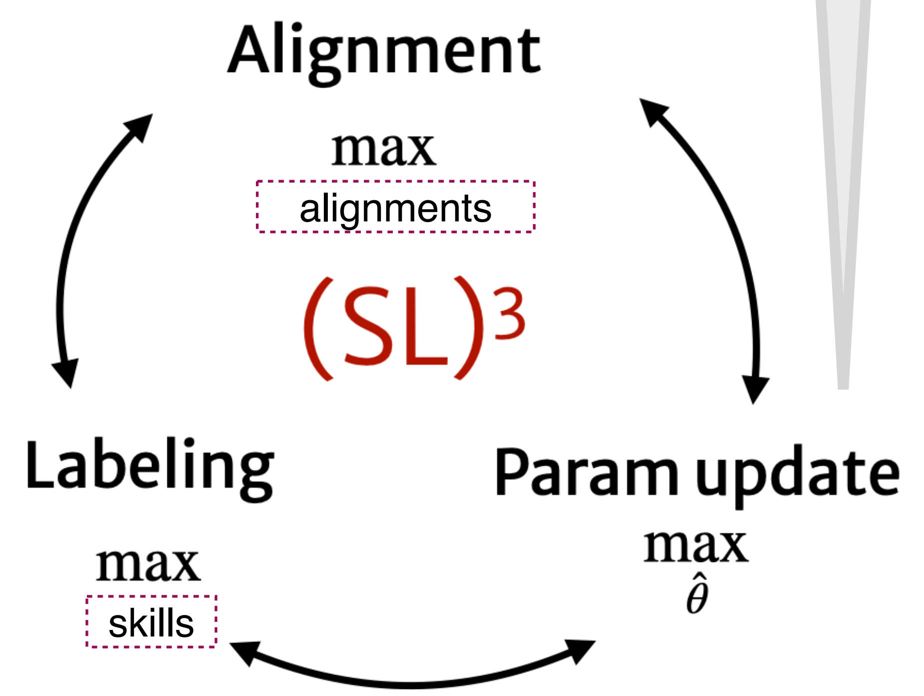


(SL)³: Semi-supervised skill learning w/ latent lang.



Just gradient descent.

3. Improve parameters



Inferred task decompositions

find the butter knife

look(down), forward(6), rotate(90), forward(17), rotate(90),

grab the knife on the counter

forward(3), rotate(90), look(down), pick(obj1), look(up),

find the tomato

rotate(90), forward(2), rotate(270), forward(1), rotate(90),

cut the tomato on the table into slices *go to the drawer*

forward(1), look(down), cut(obj2, obj1), look(up), rotate(270),

rotate(270), forward(1), rotate(90), forward(14), rotate(90),

put the knife in the drawer

look(down), open(obj3), put(obj1, obj3), close(obj3), look(up),

find the tomato *pick up a slice of tomato from*

rotate(90), forward(20), rotate(90), look(down), pick(obj4),

the table *go to the fridge*

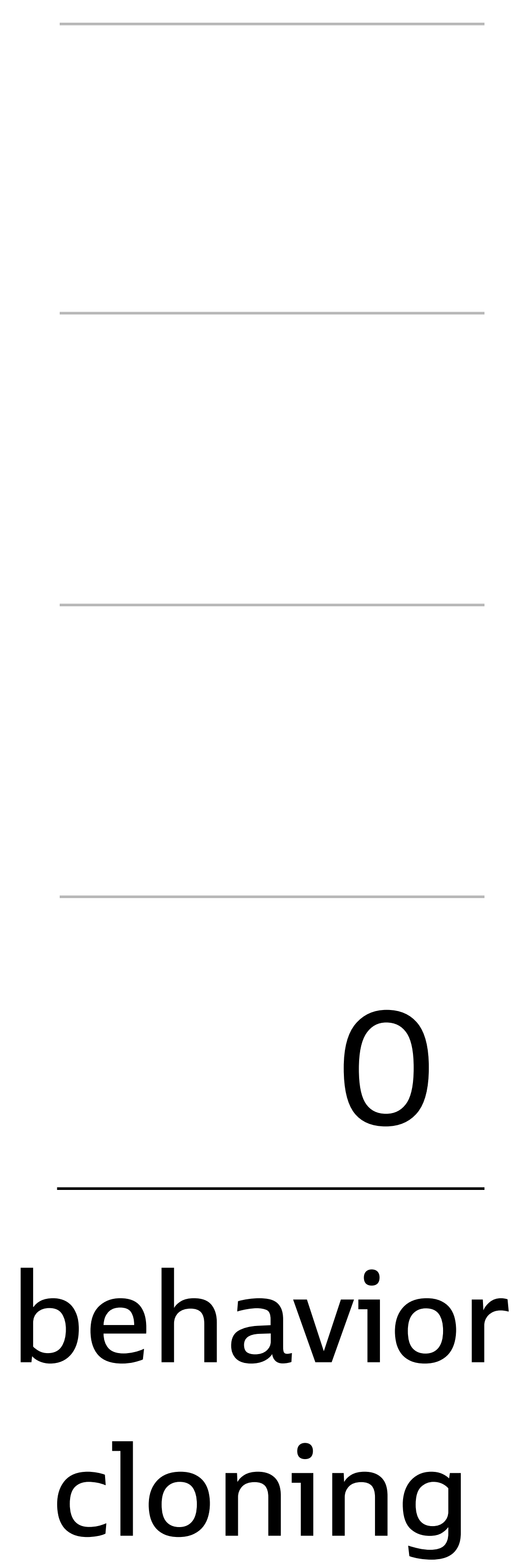
look(up), rotate(270), rotate(270), forward(1), rotate(90),

put the tomato slice on the top shelf of the refrigerator

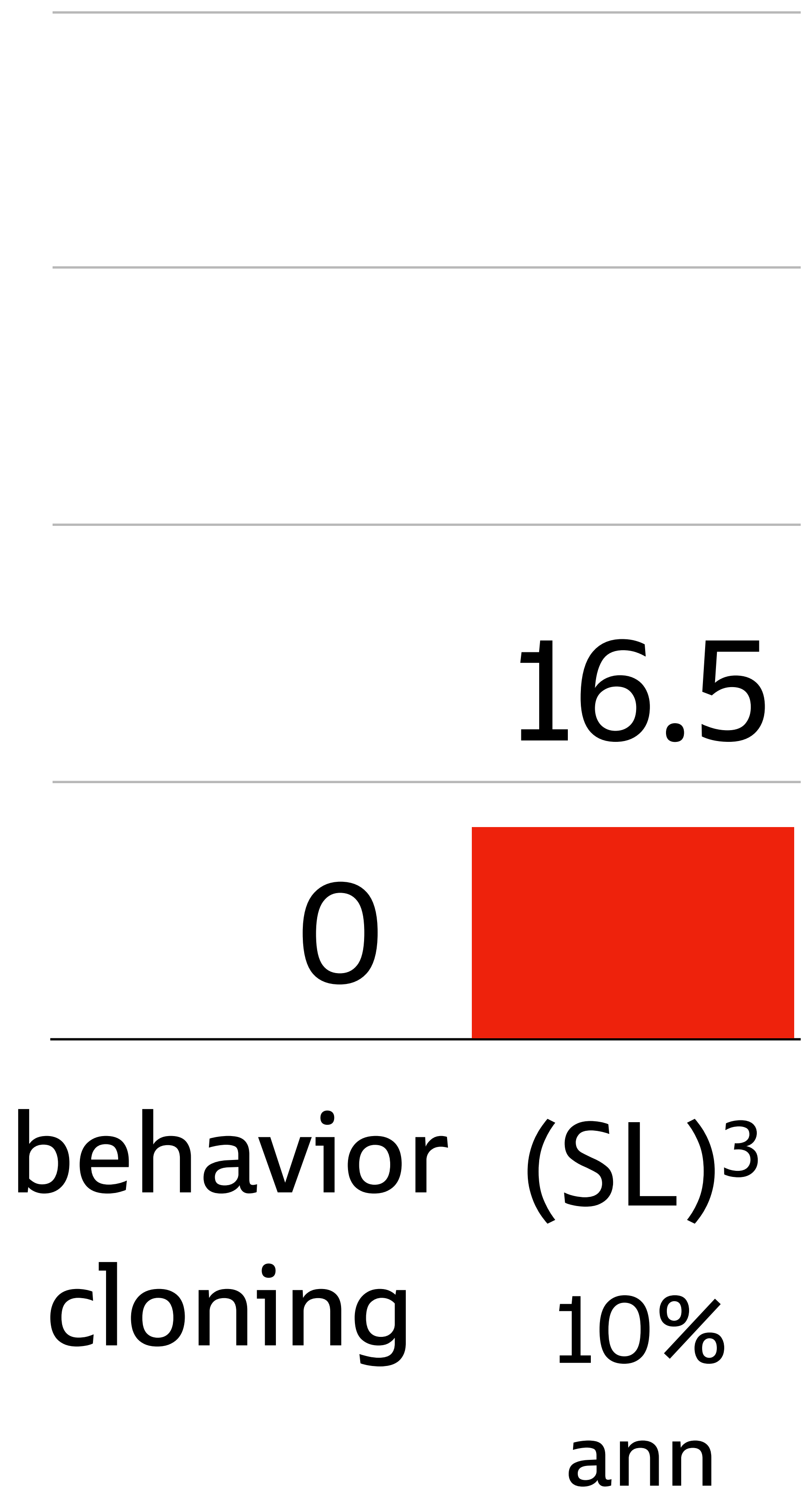
forward(12), rotate(90), look(down), open(obj3), put(obj4, obj3),

place a washed pan on the counter.

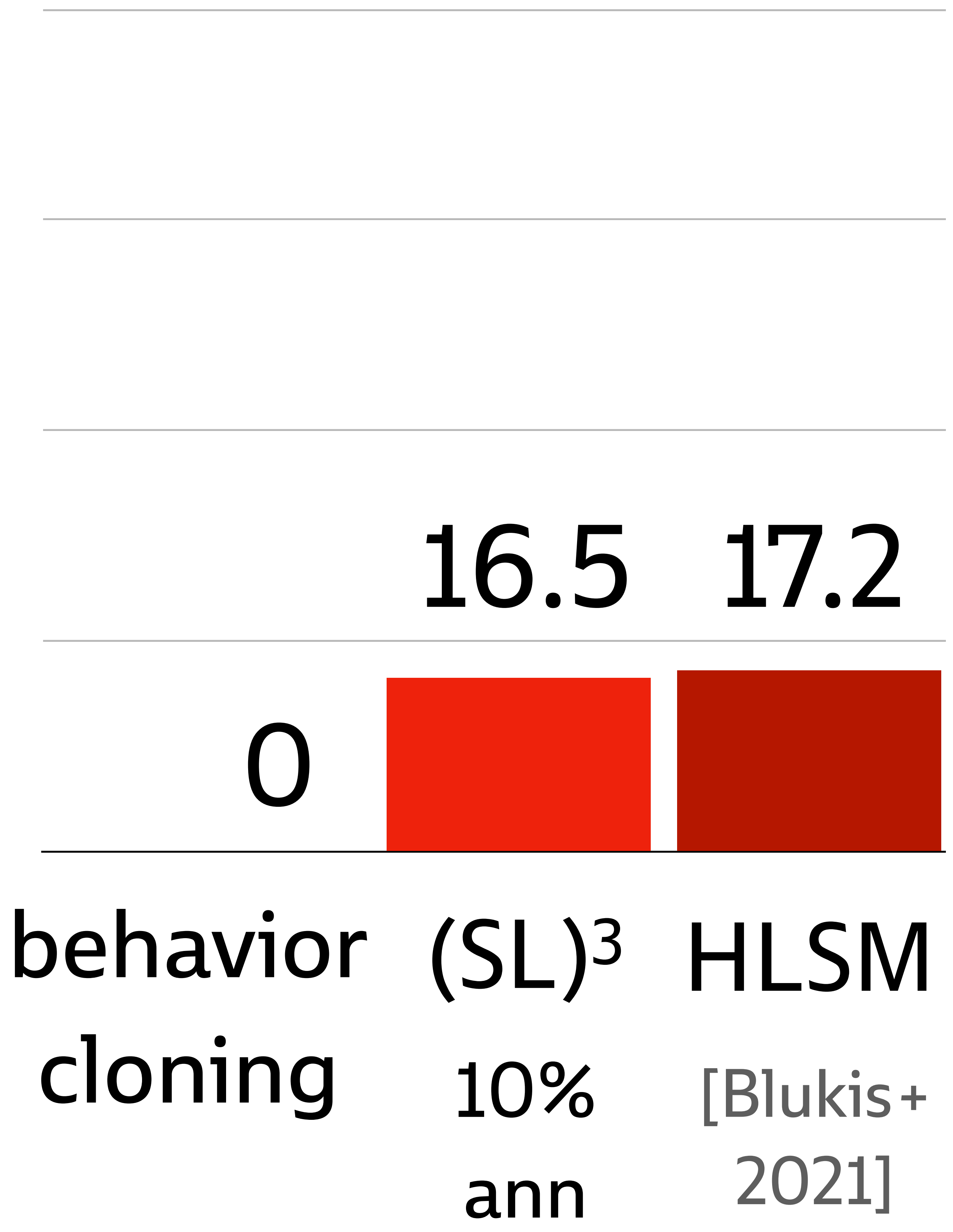
End to end success rates



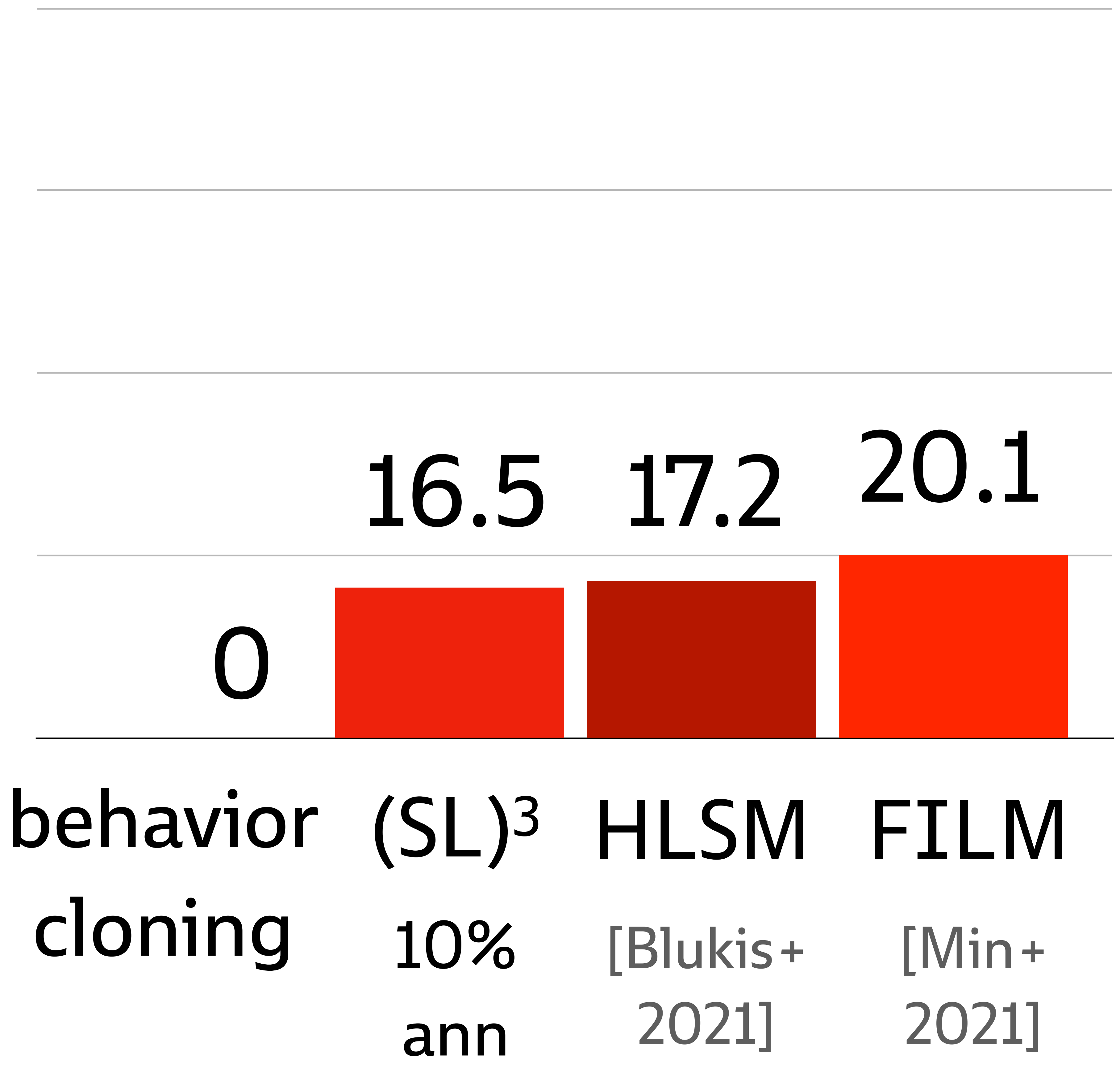
End to end success rates



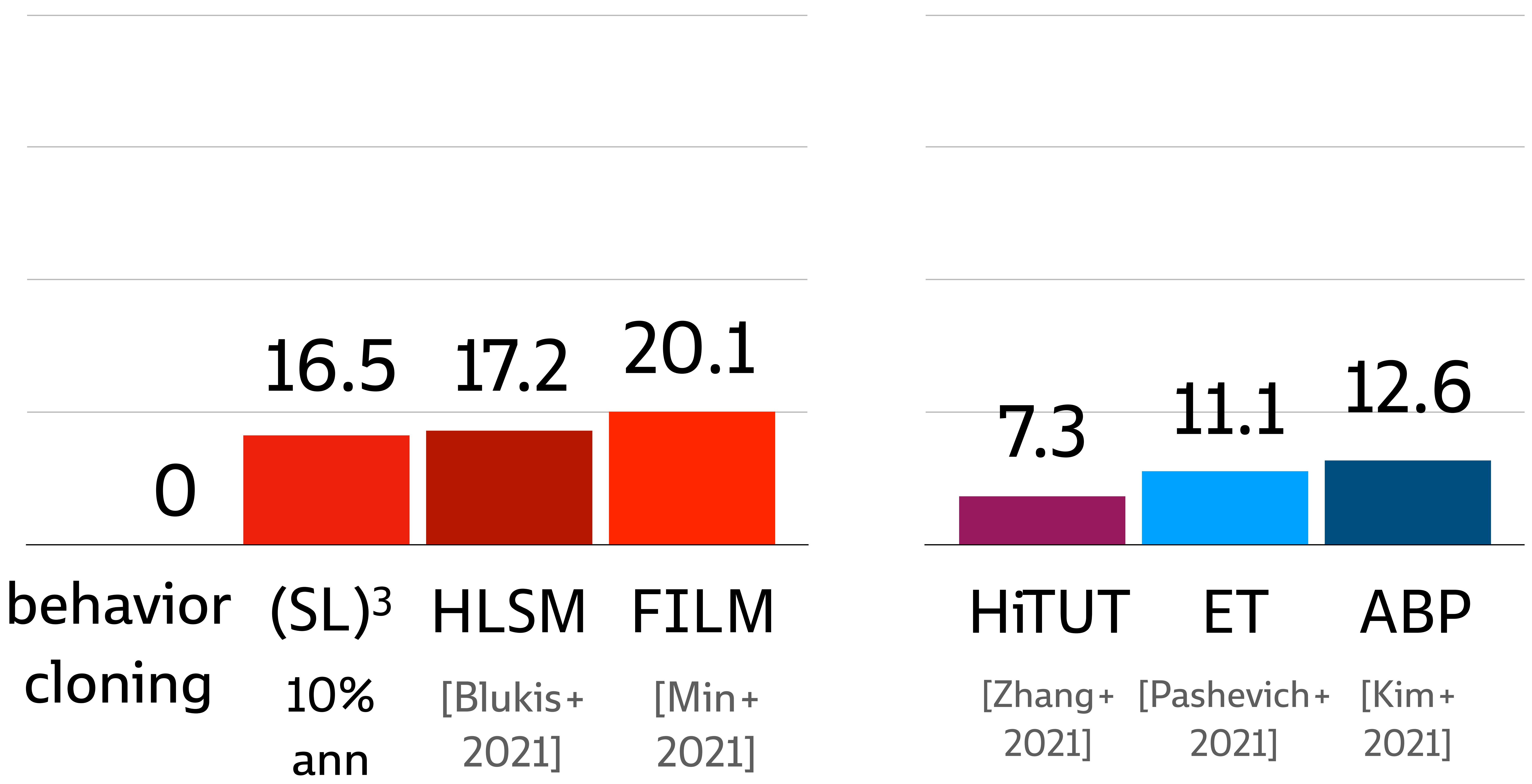
End to end success rates



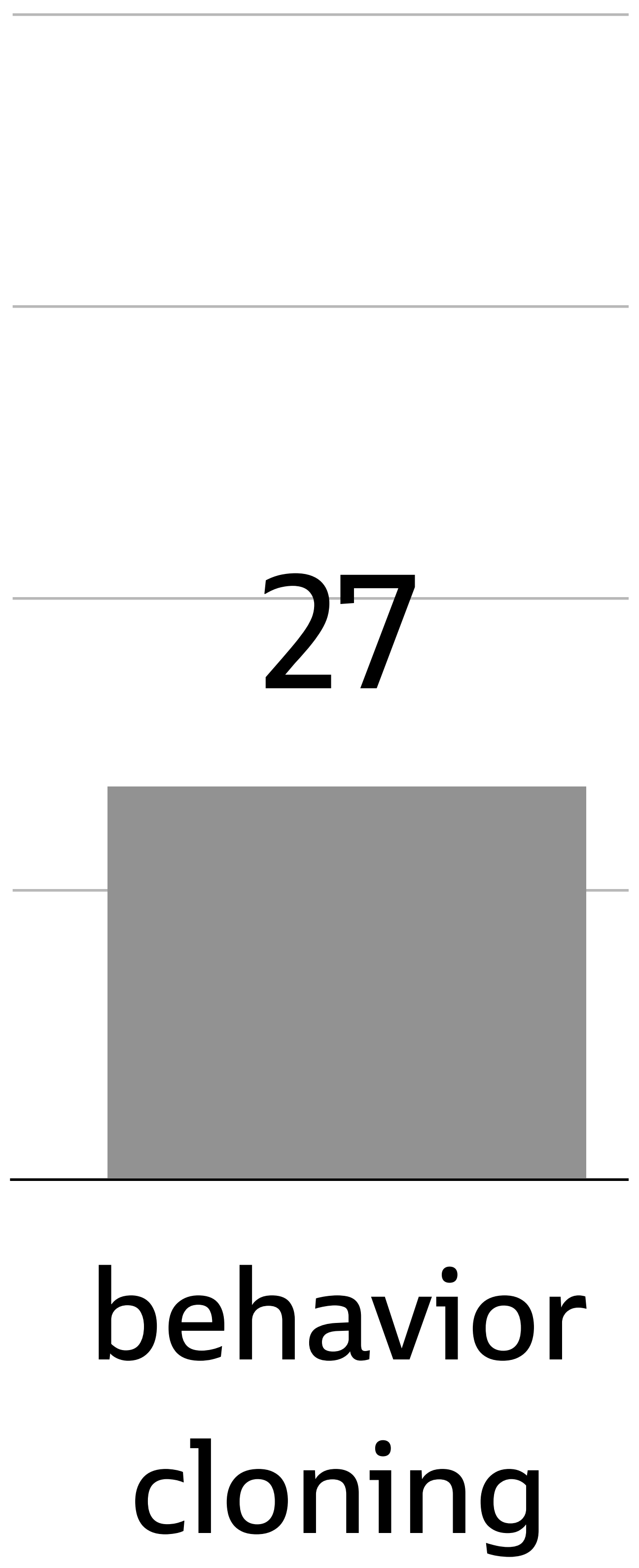
End to end success rates



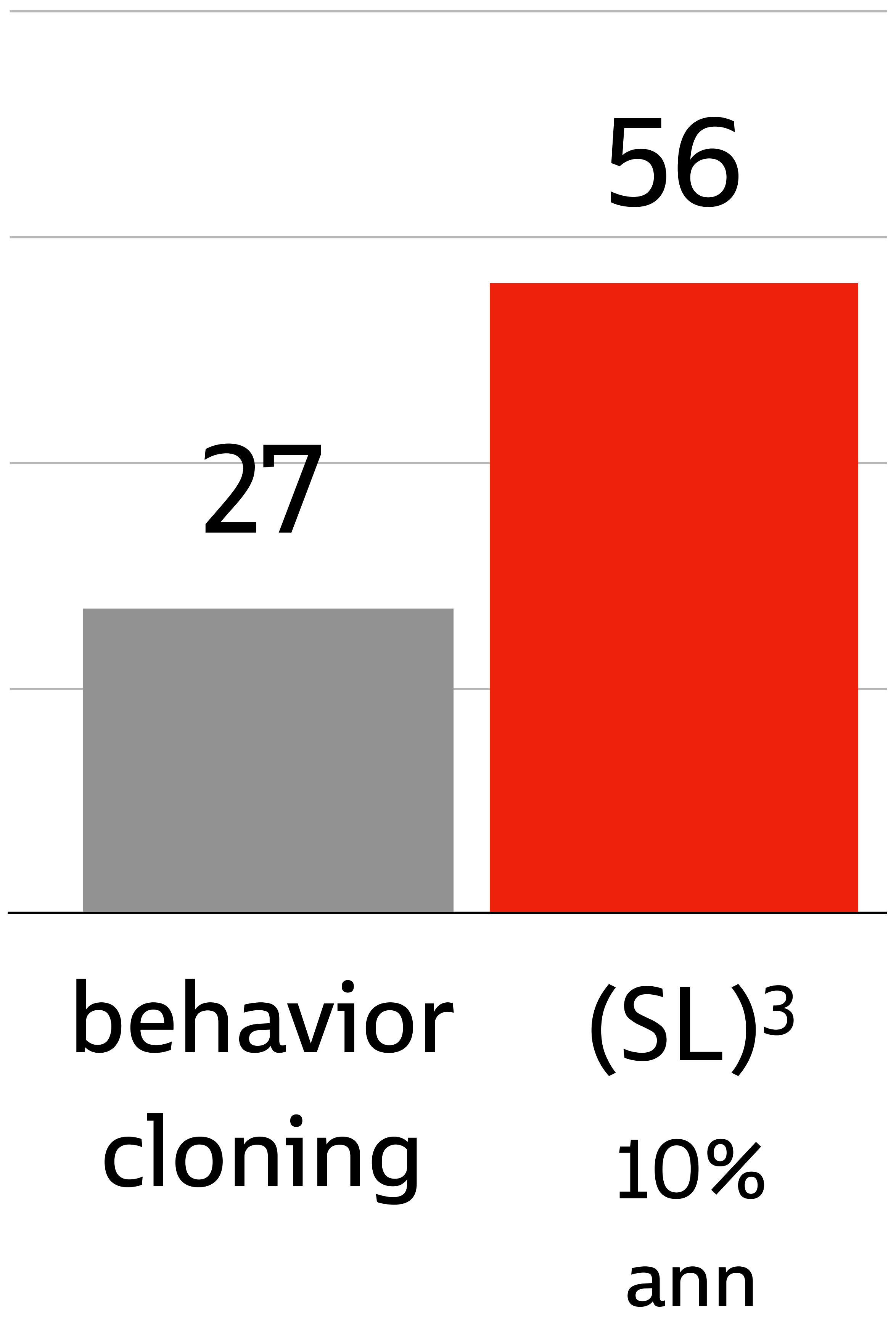
End to end success rates



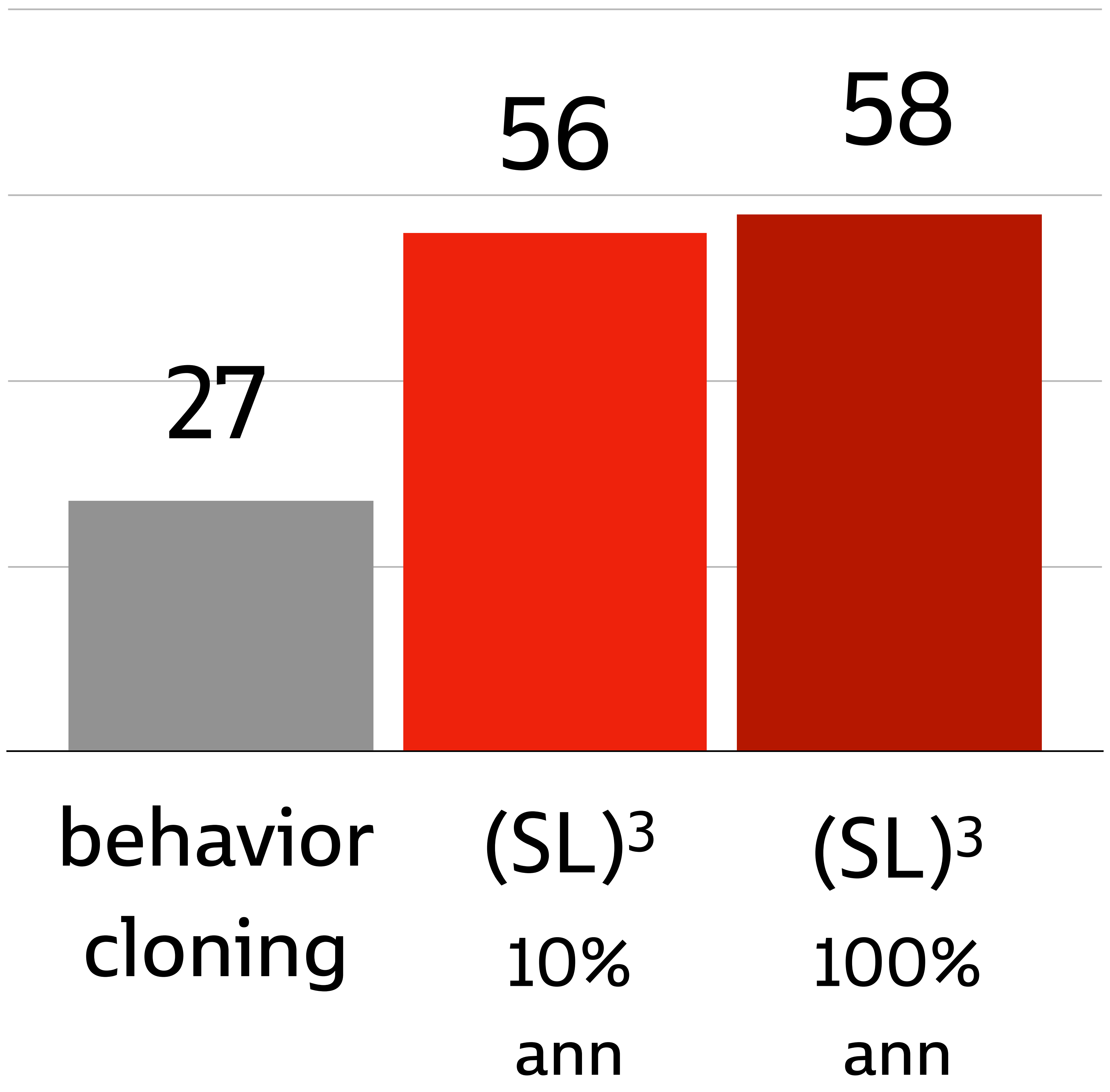
Subtask success rates (excl. navigation)



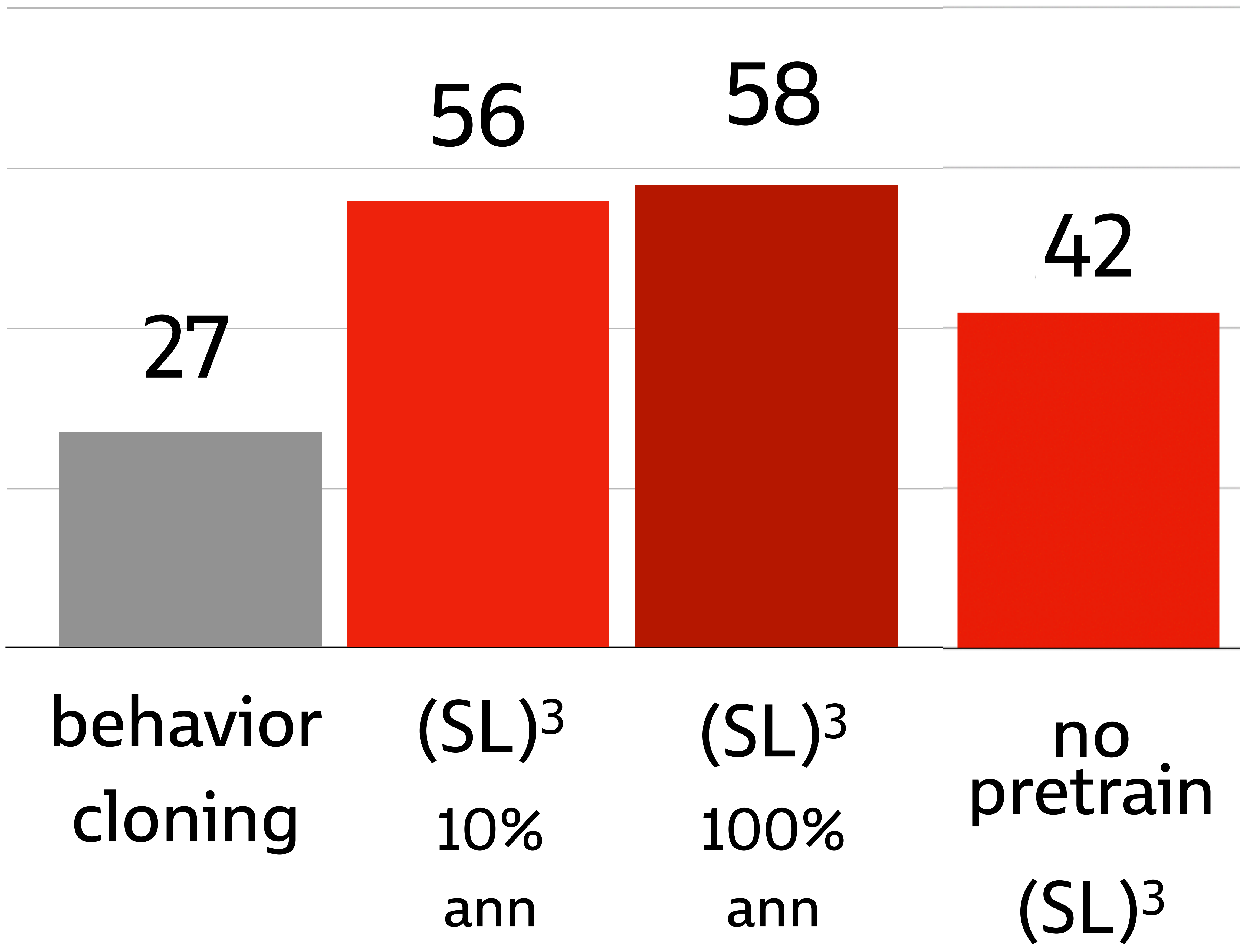
Subtask success rates (excl. navigation)



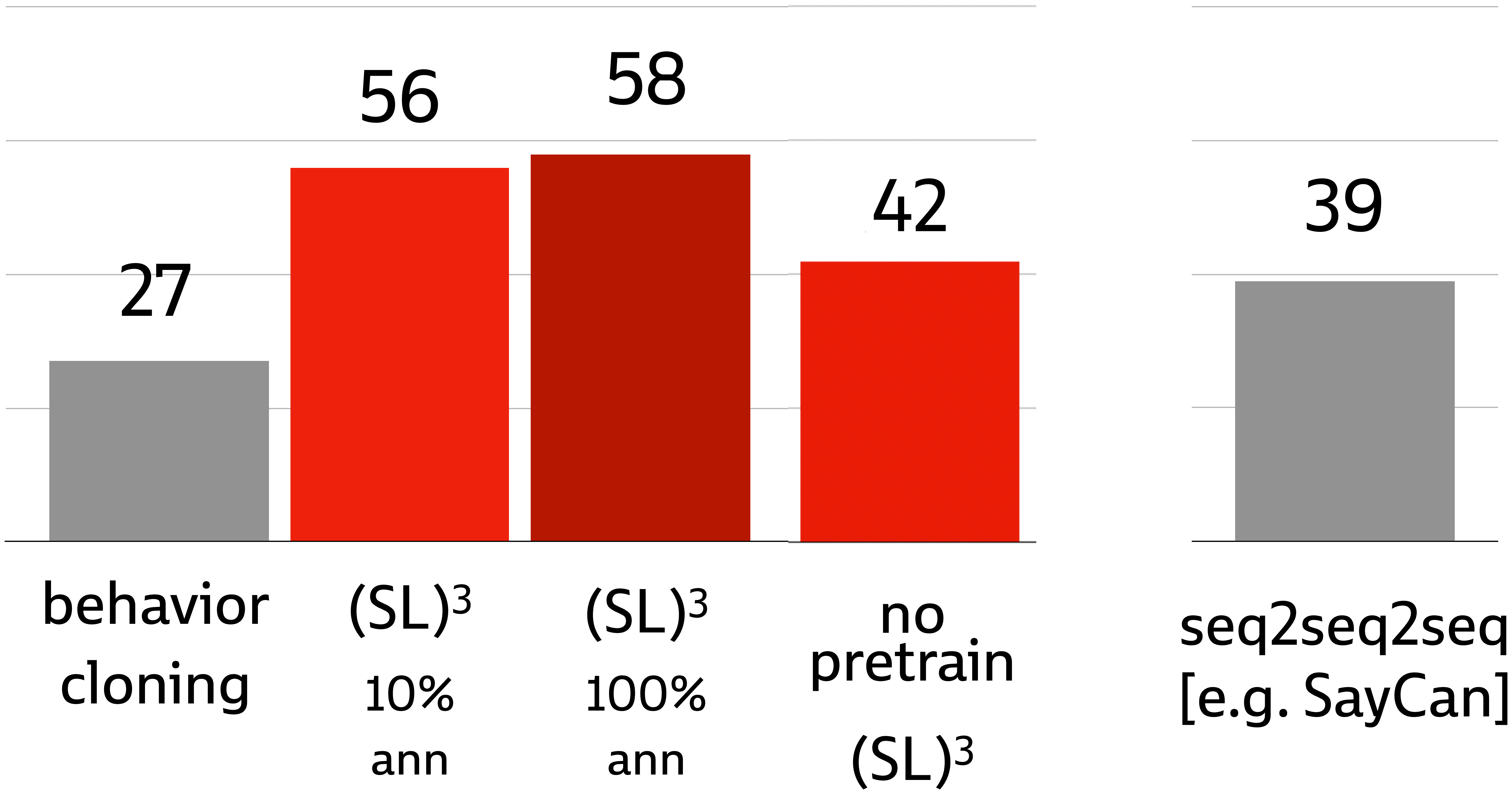
Subtask success rates (excl. navigation)



Subtask success rates (excl. navigation)



Subtask success rates (excl. navigation)



Planning with latent language?

put the knife in the drawer

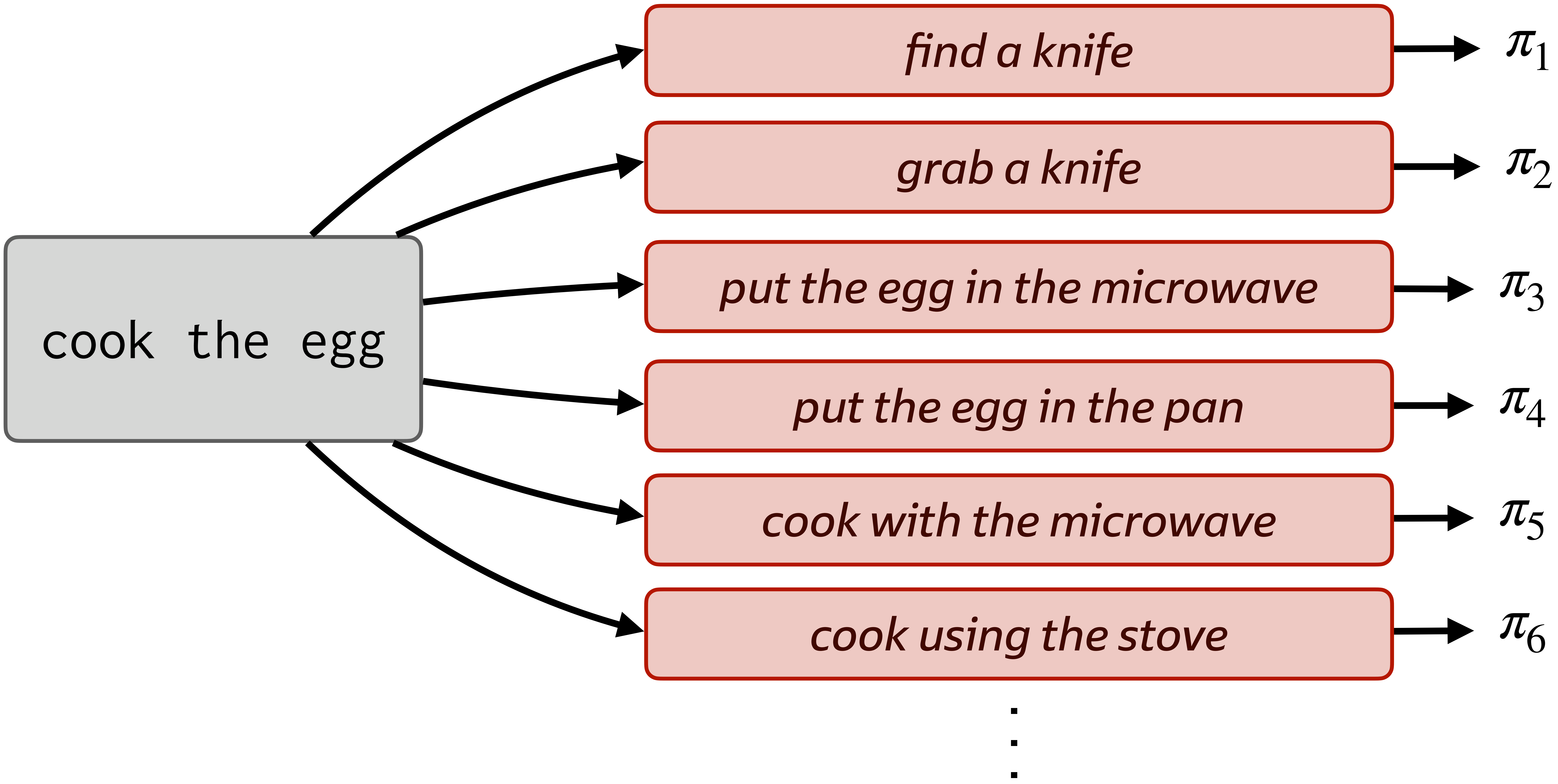
find a knife | *grab it* . . .

turn(left) | forward(10) | turn(right) | open(box) | grasp(knife) . . .





An implicit "library" of reusable skills



Learning from **text corpora**

Query	Prediction
The color of a banana is [?].	green
I can use a [?] to chop a carrot.	knife
I can use a [?] to scrub a carrot.	brush
Plates are found in the [?] room.	dining
If I drop a glass, it will [?].	explode

[Devlin et al. NAACL 2019]

A hierarchical policy with **latent language**

put the knife in the drawer

Language Model Planner

find a knife	*grab it*	. . .

Learned Skill	Learned Skill

turn(left)	forward(10)	turn(right)	open(box)	grasp(knife)	. . .





Approach: learning from text corpora

pick two apples then heat them

Find the apple. Pick up the apple on the table. Go to the microwave. Heat the apple in the microwave. Go to the countertop. put the apple on the counter. Find the apple. pick up another apple on the table. Go to the microwave. open the microwave, put the apple in, close the door, heat it, then remove the apple from the microwave. Go to the diningtable. put the apple next to the other apple.

slice a heated apple

Find the apple. pick up the apple that is on the counter. Go to the microwave. open the microwave and place the apple inside then close the door and turn on the microwave for five seconds. Find the knife. pick up the yellow knife that is on the counter. Find the apple. slice the apple that is in the microwave.

clean and cool an apple

Find the apple. pick up the apple from the counter. Go to the sinkbasin. place the apple in the sink, clean it with water, take apple out. Go to the fridge. open the fridge, place apple on shelf to the left of the apple, close the fridge.

(a)

clean and cool a carrot

Find the lettuce. pick up the carrot from the island. Go to the sinkbasin. place the carrot in the sink and turn on the water. turn off the water and pick up the red carrot. Go to the fridge. open the fridge and place the carrot inside.

place two iPhones on the table

Find the cellphone. pick up the iphone from the table. Go to the sidetable. put the iphone on the table. Find the cellphone. pick up the other iphone from the table. Go to the sidetable. put the iphone on the table.

rinse some tomatoes

'Find the tomato. pick up the tomato sitting on the table. Go to the sinkbasin. put the tomato in the sink and rinse it. Go to the sidetable. put the tomato on the table.

(b)

Learning skills: summary

What: Hierarchical policy learning from demonstrations with (sparse) natural language supervision.

How: Automatic "parsing" of annotated & unannotated demos with dynamic programs for alignment and inference of string-valued latent variables.

Why: Instructions are easy to collect; training with <1k of them gives performance comparable to state-of-the-art models evaluated with *ground truth* plans.

Learning functions from denotations
and descriptions

Lio
Wong

+ Josh Tenenbaum

[Leveraging Language for Program Search and Abstraction Learning. ICML 2021.]

Inferring programs from specifications

previously:

Predict a program to execute given a high-level goal.

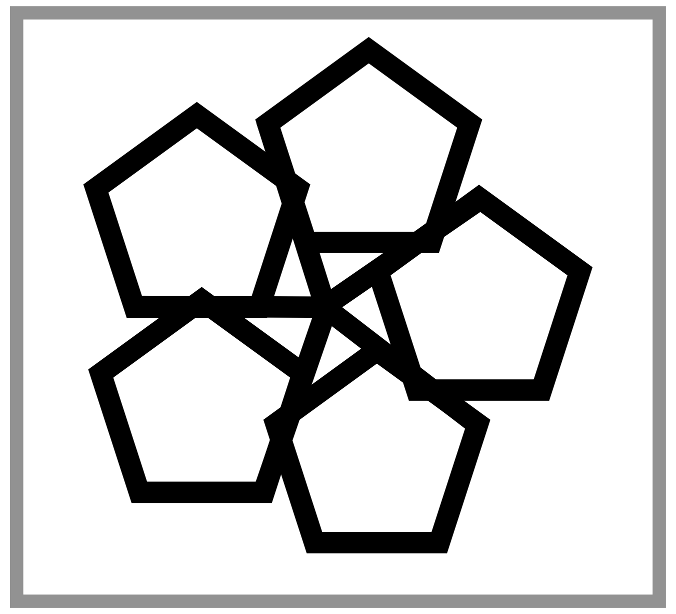
cooked(egg)

grasp(egg) forward(9)

now:

Infer a program given the results of execution.

```
(f24 5 (λ (X) (get/set (λ (y)
(f2 1 (f41 5 y))) x)) z)
```



Inferring programs from specifications

Many learning problems are naturally formulated as program synthesis.

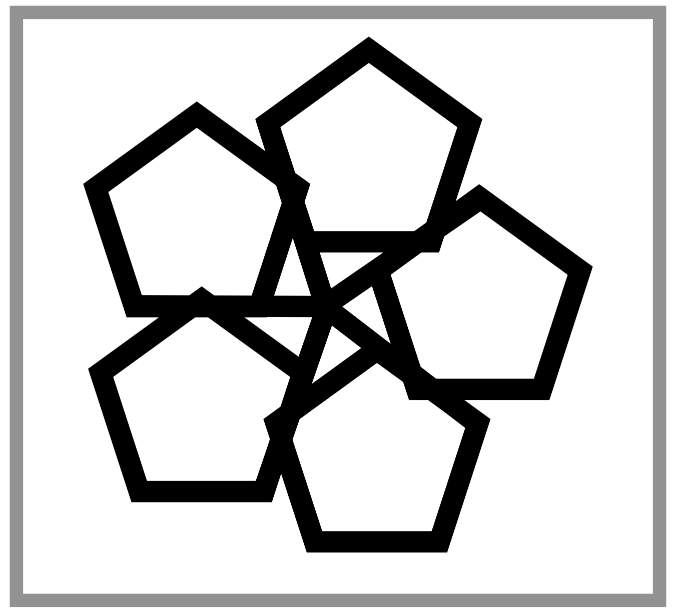
```
s/Figure/Fig./g
```

Figure 1 → Fig. 1

as in Figure 6a → as in Fig. 6a

a striking figure → a striking figure

```
(f24 5 (λ (X) (get/set (λ (y)
(f2 1 (f41 5 y))) x)) z)
```



Language & program abstractions

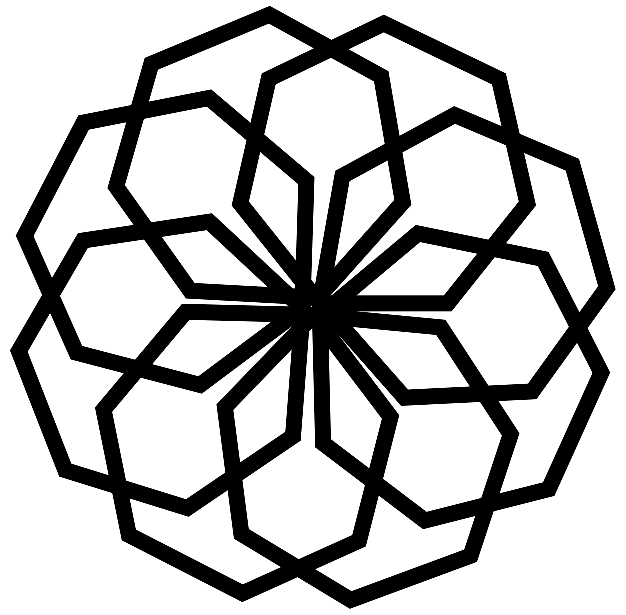
Programs are compositional.

```
look(down), forward(6), rotate(90), forward(17),
forward(3), rotate(90), look(down), pick(obj1)
rotate(90), forward(2), rotate(270), forward(1),
forward(1), look(down), cut(obj2, obj1), look(up),
```

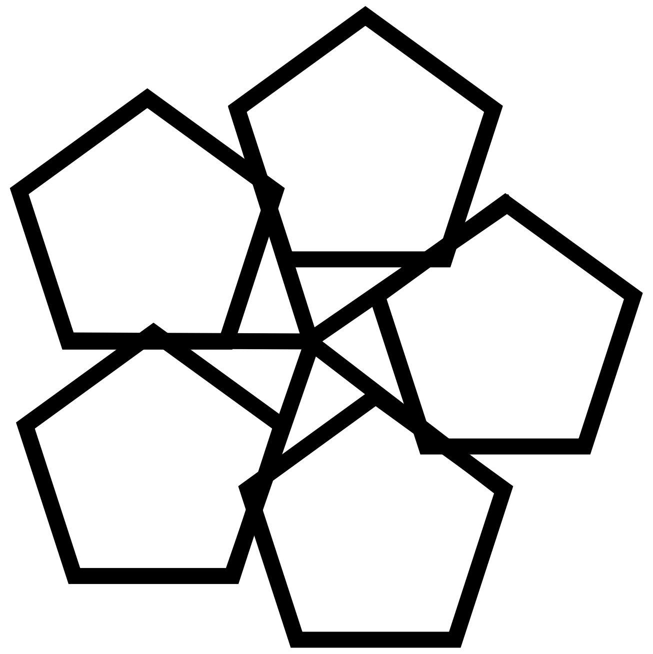
```
for i in range(8):
  for j in range(7):
    pendown()
    forward(1cm)
    penup()
    rotate(129)
  rotate(45)
```

```
for i in range(5):
  for j in range(5):
    pendown()
    forward(2cm)
    penup()
    rotate(108)
  rotate(72)
```

```
f1(8,
  f2(7, 1cm))
```



```
f1(5,
  f2(5, 2cm))
```



Language & program abstractions

Programs are compositional.

This compositional structure is reflected in language!

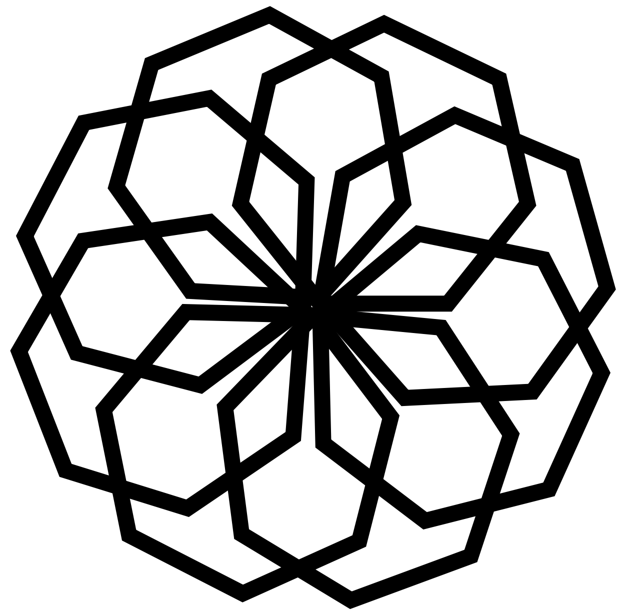
pick up a knife, find a tomato, then go to the counter and slice it

a pinwheel made of 8 heptagons

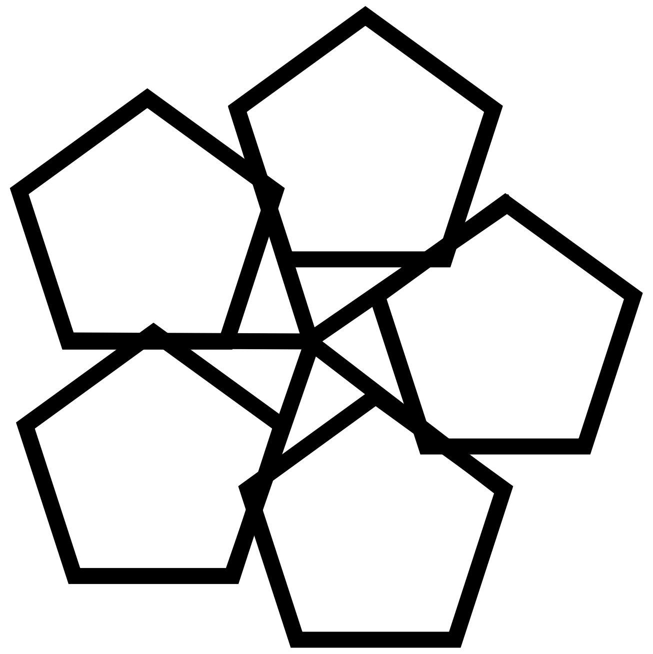
a pinwheel made of 5 pentagons

find_knife
```
look(down), forward(6), rotate(90), forward(17),
```
find_tomato
```
forward(3), rotate(90), look(down), pick(obj1)
```
goto_counter
```
rotate(90), forward(2), rotate(270), forward(1),
```
slice
```
forward(1), look(down), cut(obj2, obj1), look(up),
```

```
pinwheel(8,
  gon(7, 1cm))
```



```
pinwheel(5,
  gon(5, 2cm))
```



Language & program abstractions

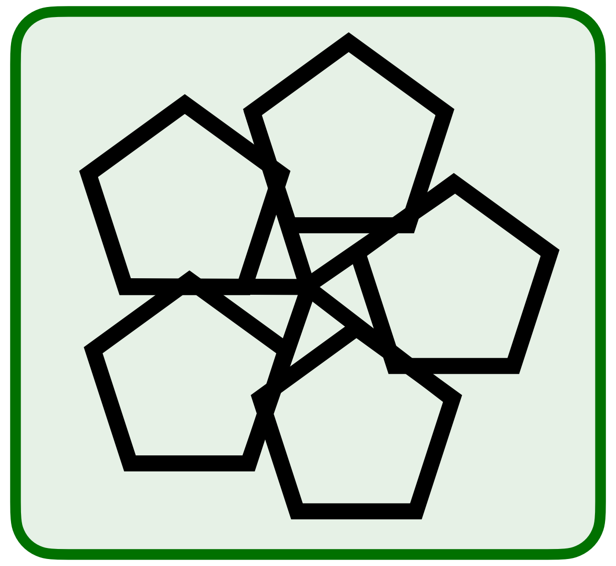
find a knife → `turn(left)` `forward(10)`

grab it → `grasp`

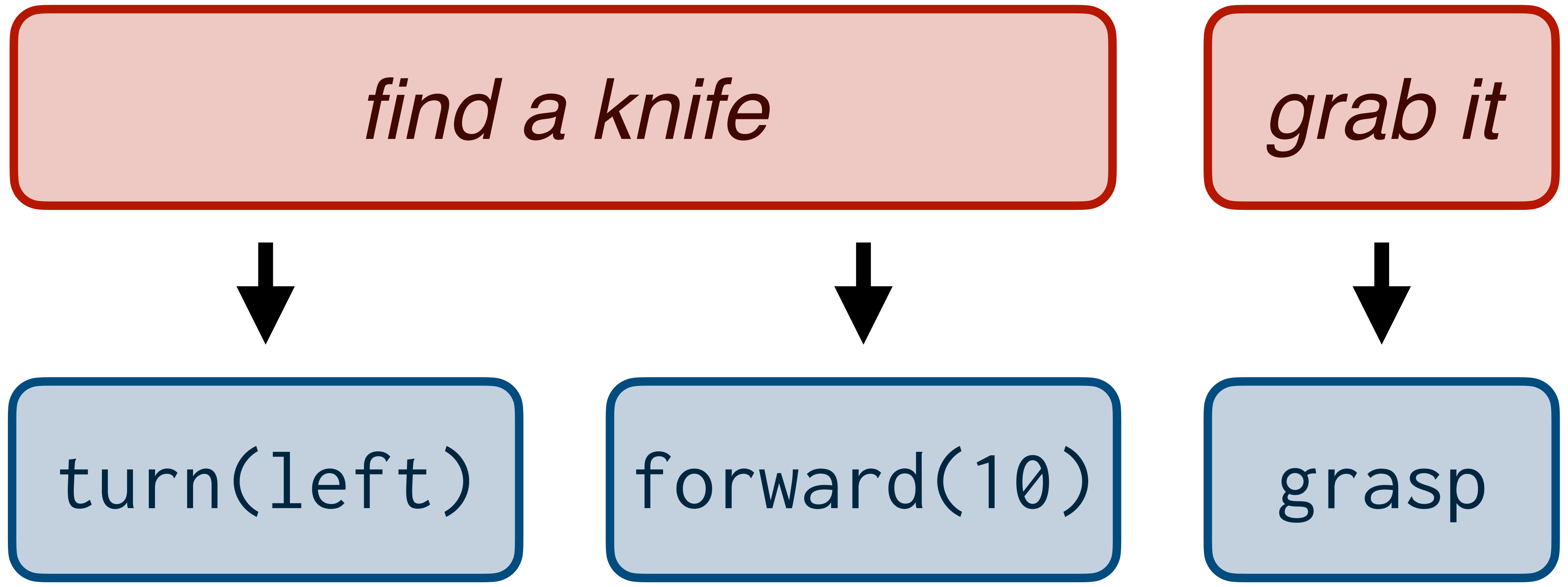
Can we use language to learn from <u>denotations</u> the same way we used it to learn with full supervision?

a pinwheel made of *5* *pentagons*

```
for i in range(5):
  for j in range(5):
    pendown()
    forward(2cm)
    penup()
    rotate(72)
  rotate(72)
```

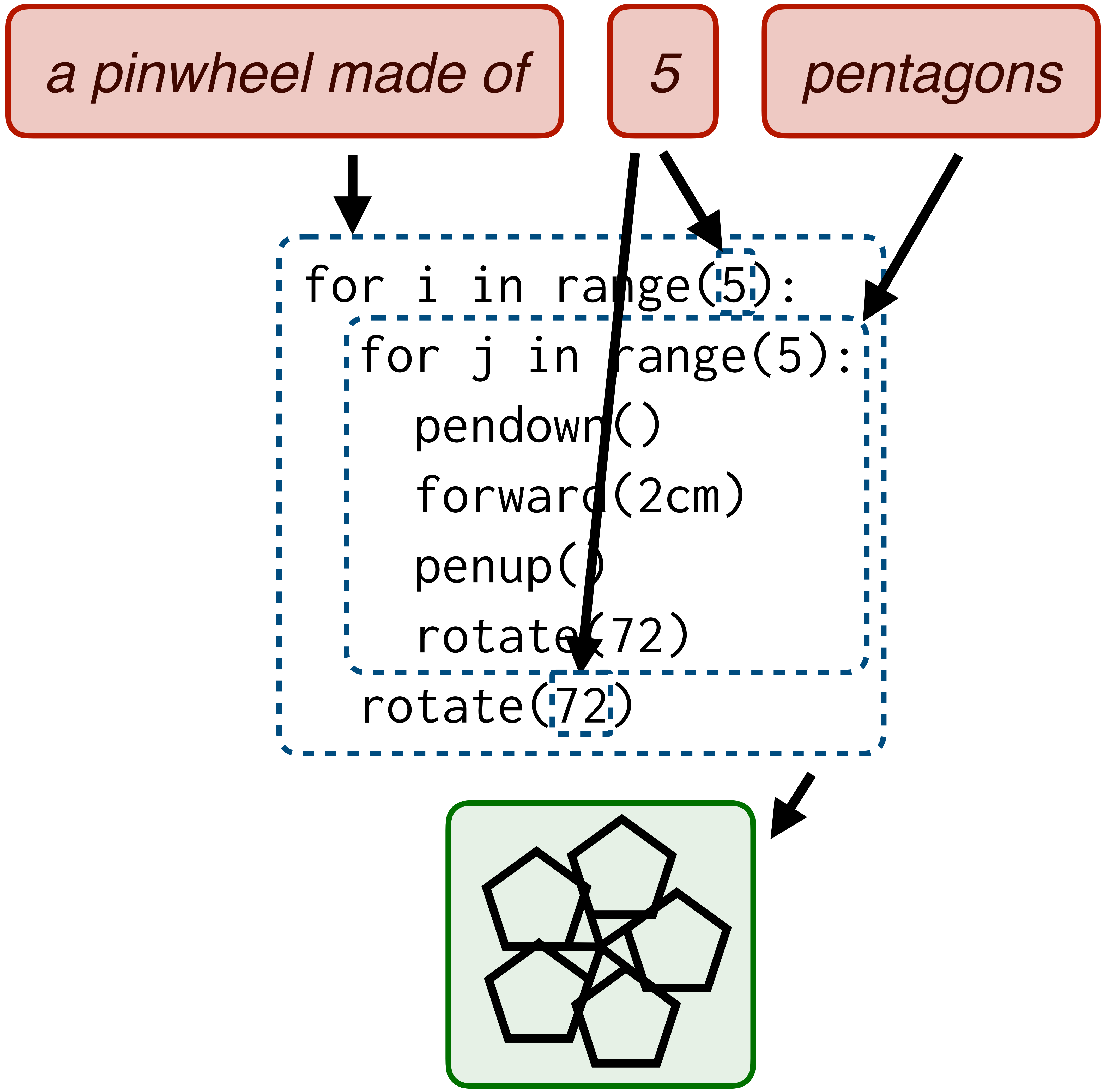


Language & program abstractions



Can we use language to learn from <u>denotations</u> the same way we used it to learn with full supervision?

Key challenge: we need to infer programs along with all the other latent vars!



LAPS: lang. for abstraction & program search

a pinwheel made of

annotation

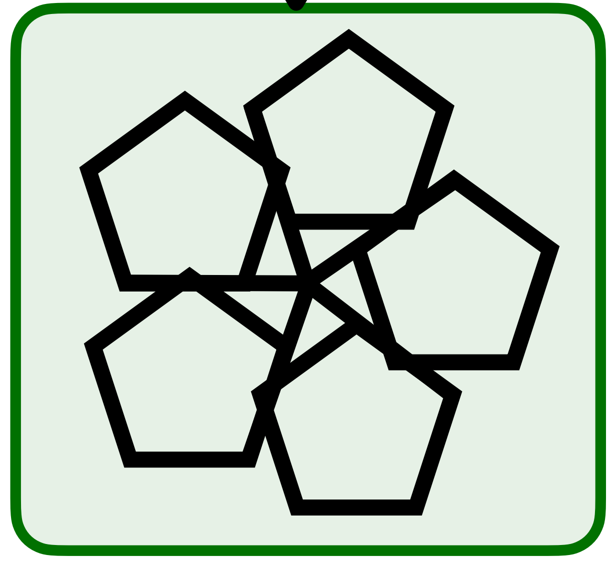
Program generator

alignment

```
lambda x, f:
  for i in range(x):
    f()
  rotate(360/x)
```

program

Execution model



output

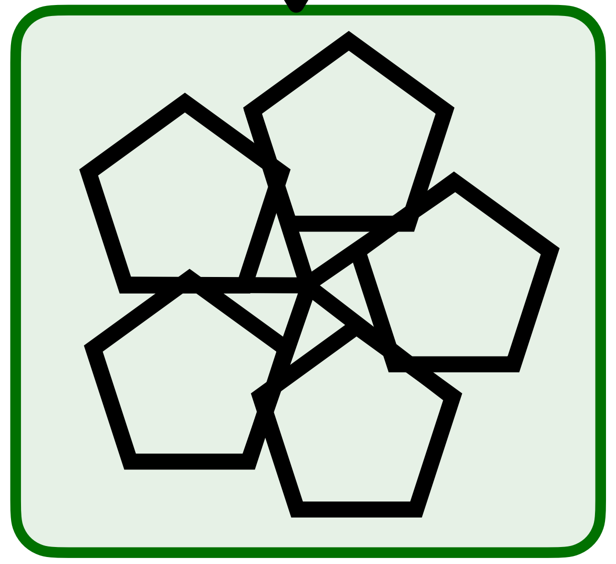
An explicit library of reusable functions

a pinwheel made of

Program generator

```
lambda x, f:
  for i in range(x):
    f()
  rotate(360/x)
```

Execution model



Generative model of programs

f0=(λ (x y z) (for x (λ (u v) (move z y v))))

move pen in parameterized loop

f9=(**f0** ∞ ε)

0.07 | *semicircle*

smooth curve

f4=(λ (x y z) (**f0** x (/ 2π y) 1 z))

0.09 | *small*

rotates and draws a unit line

f5=(λ (x y) (**f4** x x y))

0.27 | *gon*
0.22 | *small*

rotational symmetry by number of sides

...

f17=(λ (x) (pen-up (λ (y) (**f16** x y))))

0.67 | *separated*
0.06 | *space*

lift pen between consecutive shapes

...

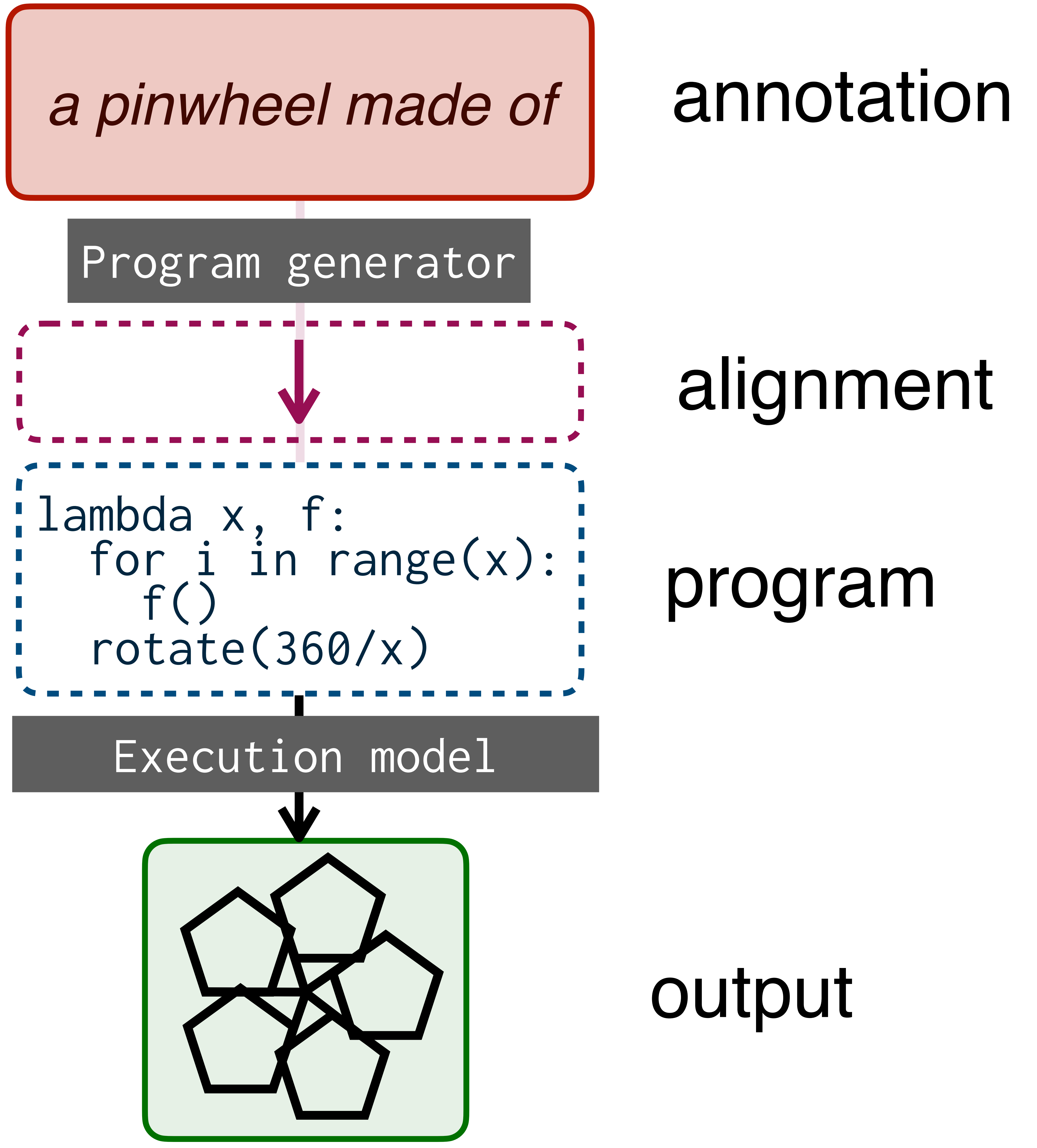
f24=(λ (x y) (**f23** (λ (z u) (**f21** y 0 x u))))

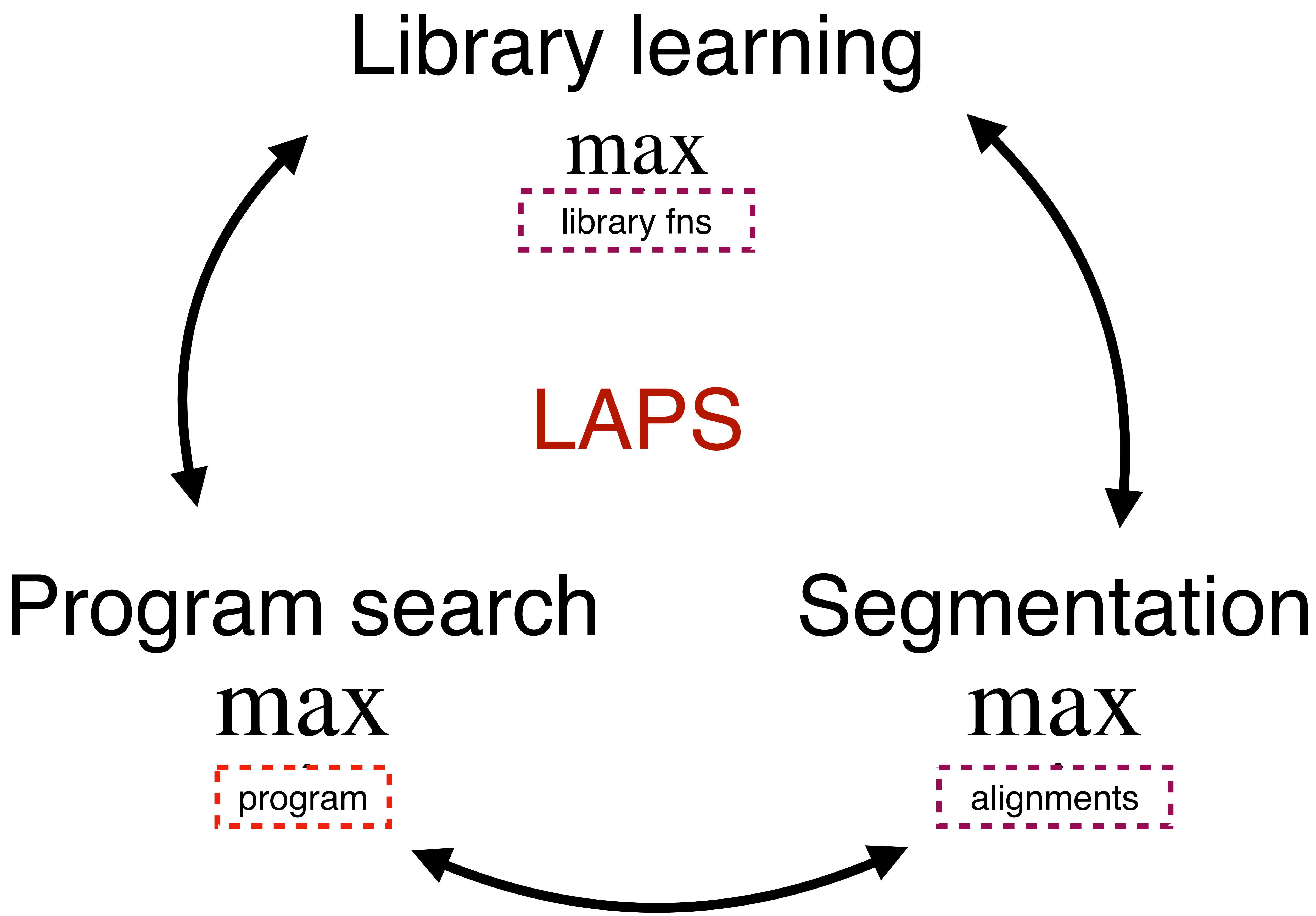
0.09 | *snowflake*
0.09 | *arms*

rotate shapes around axis

composes functions from a library of program fragments given words

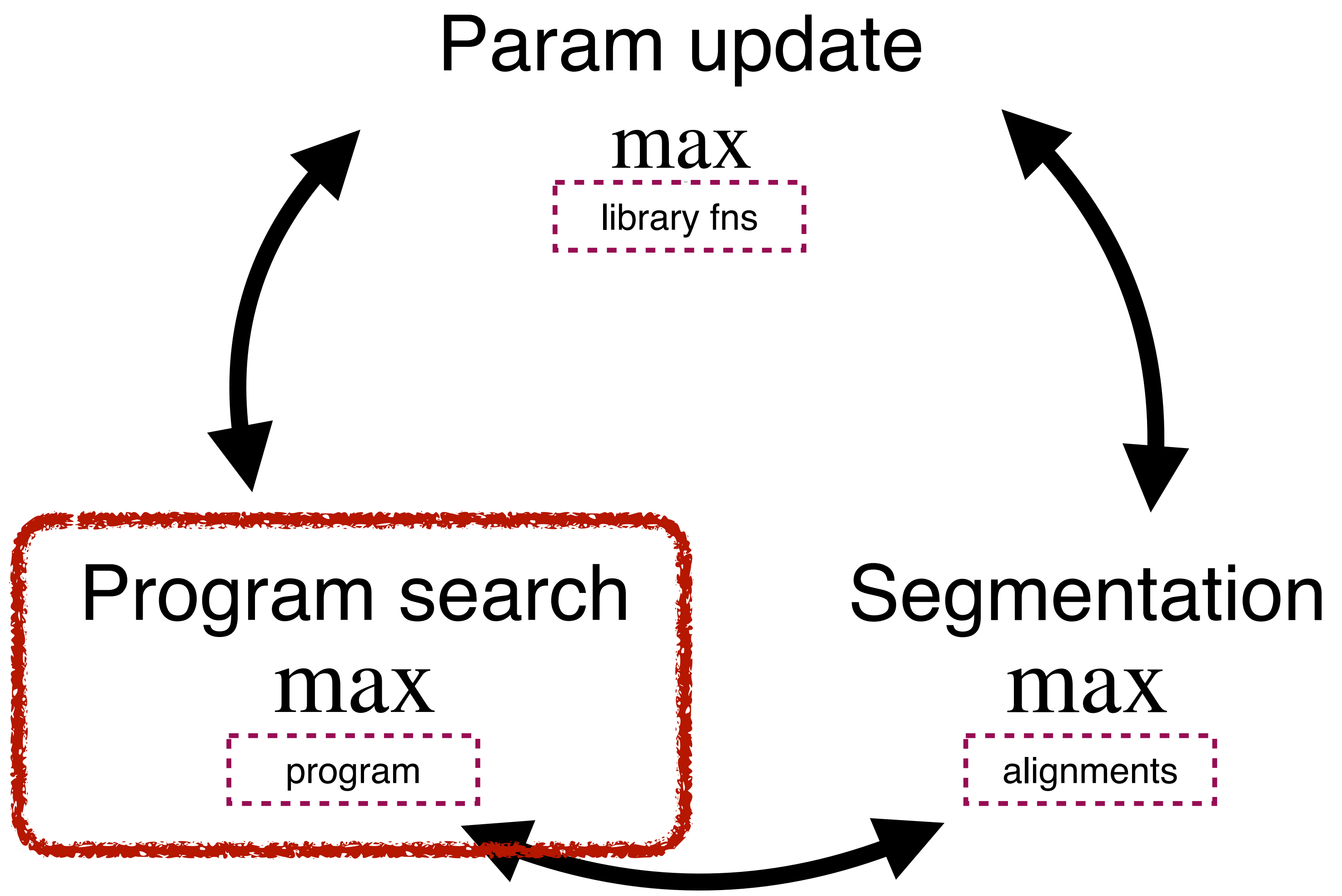
LAPS: lang. for abstraction & program search

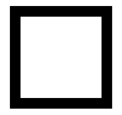


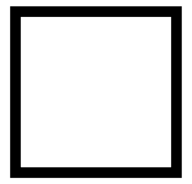


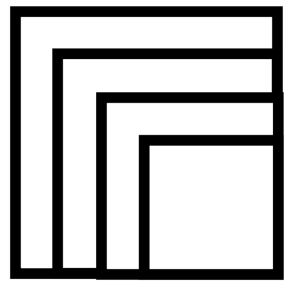
Growing the library and the set of solved programs

[c.f. Ellis et al. 21, *DreamCoder.*]



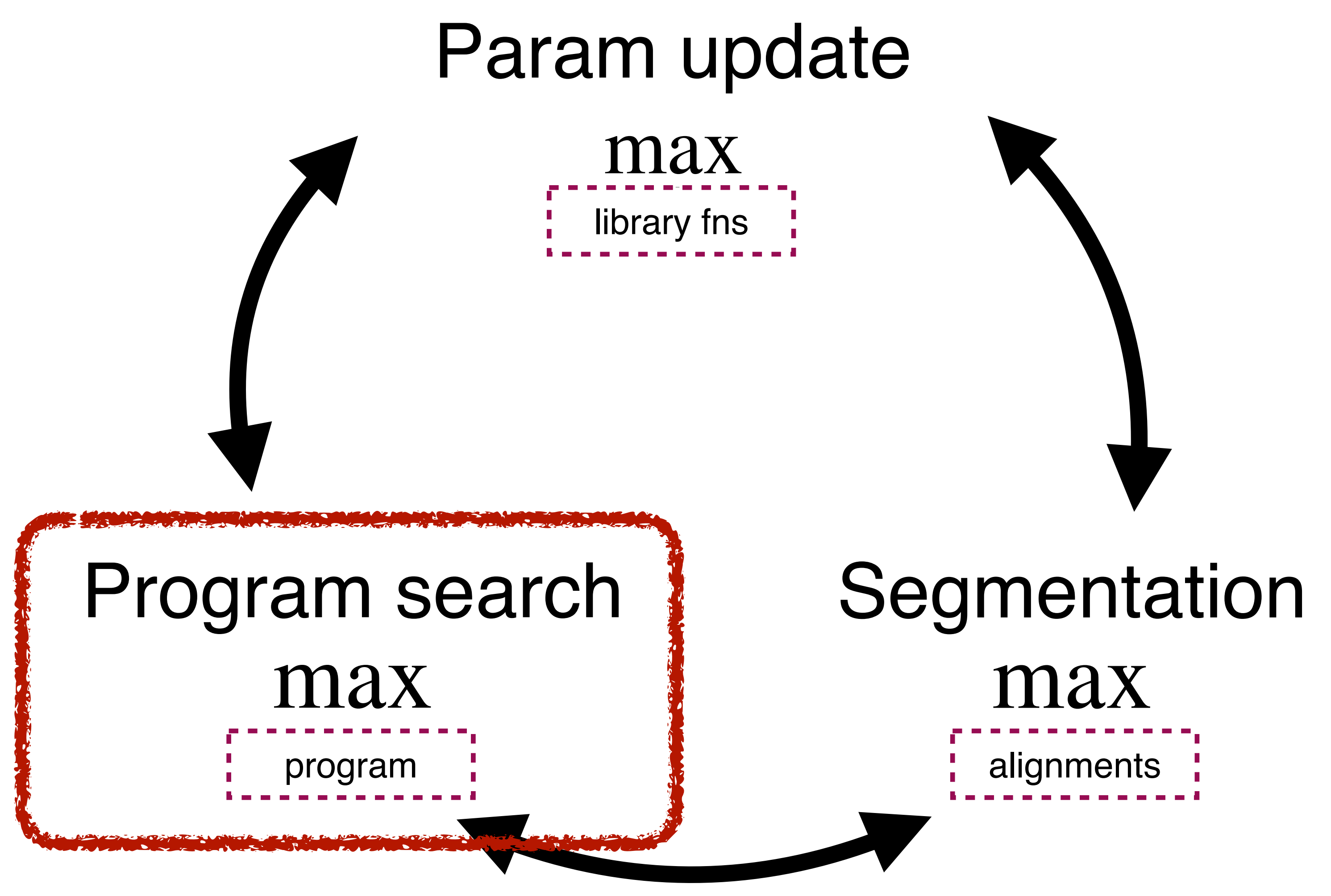
 *a small square*

 *a medium square*

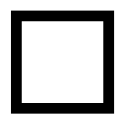
 *4 nested squares*

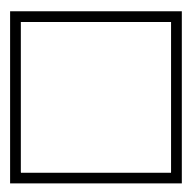
Growing the library and the set of solved programs

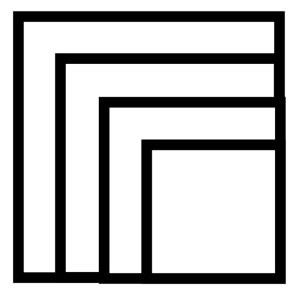
[c.f. Ellis et al. 21, *DreamCoder.*]




```
for i in range(4):
  pendown()
  forward(1)
  penup()
  rotate(90)
```

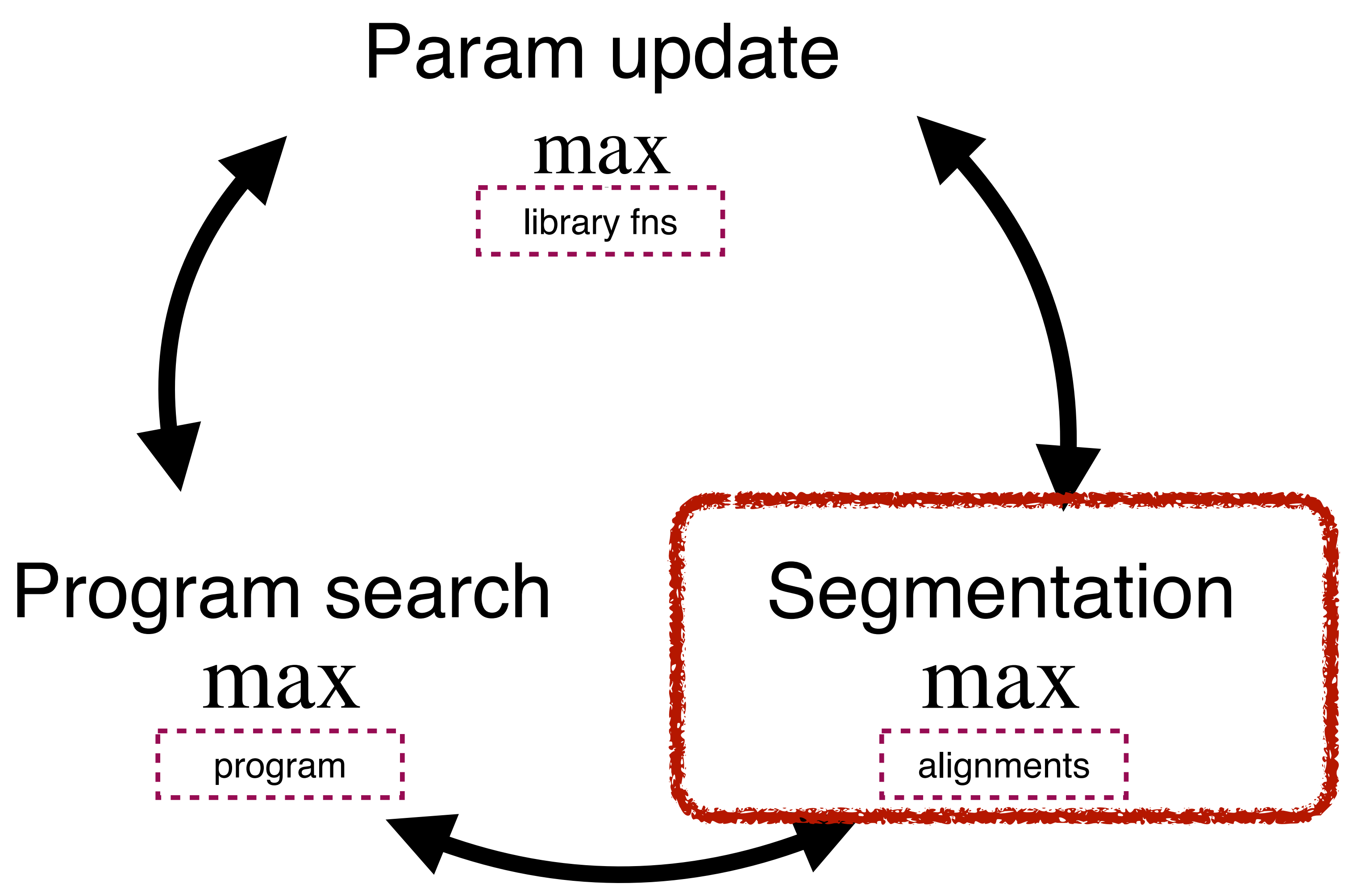
 *a small square*

 *a medium square*

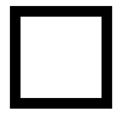
 *4 nested squares*

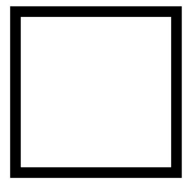
Growing the library and the set of solved programs

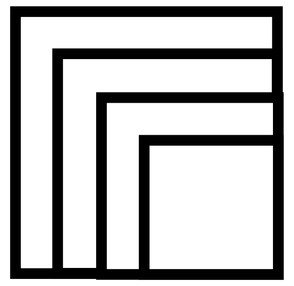
[c.f. Ellis et al. 21, *DreamCoder.*]




```
for i in range(4):
  pendown()
  forward(1)
  penup()
  rotate(90)
```

 *a small square*

 *a medium square*

 *4 nested squares*

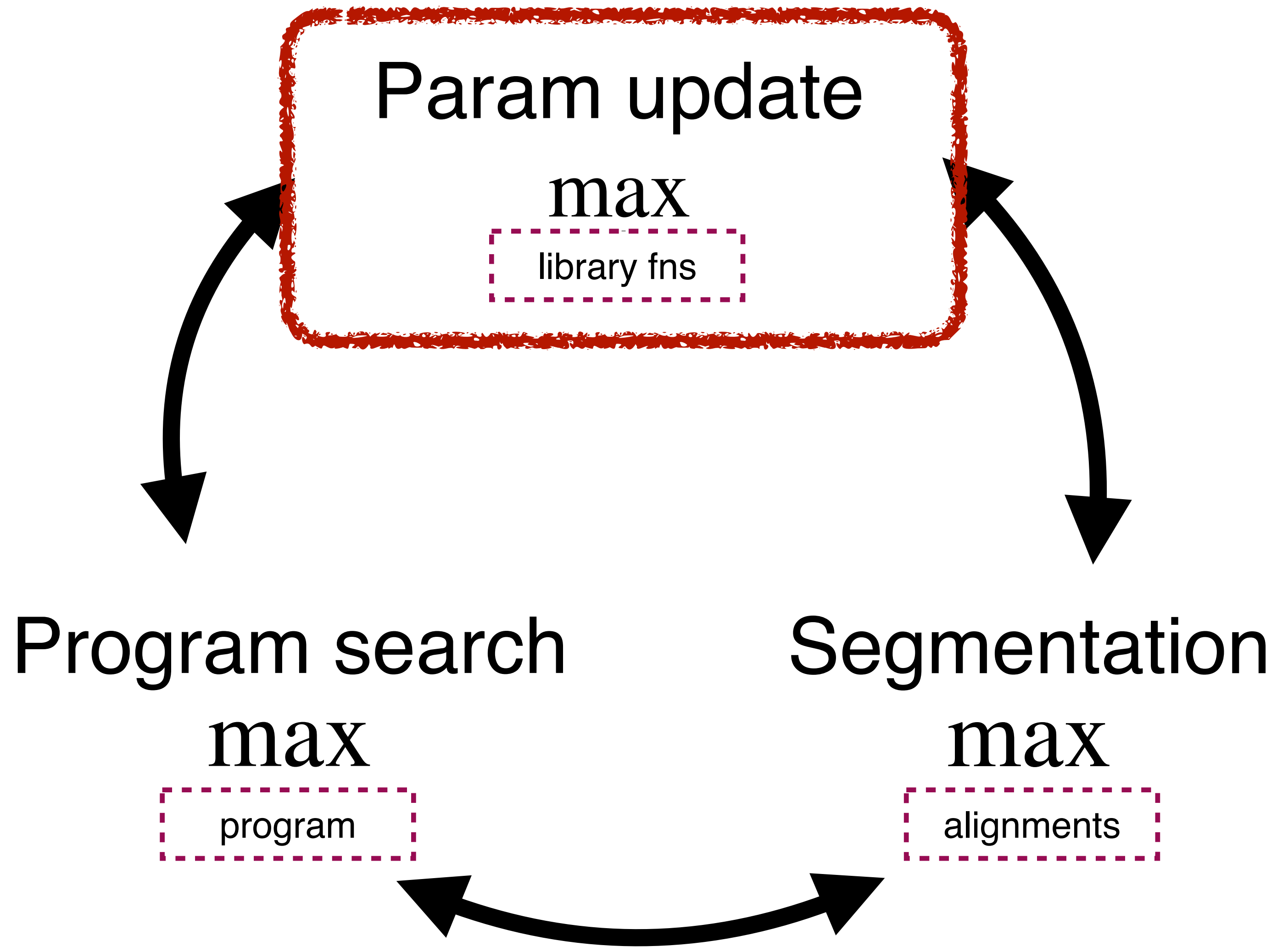
Growing the library and the set of solved programs

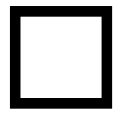
[c.f. Ellis et al. 21, *DreamCoder.*]

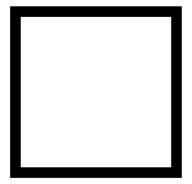
square

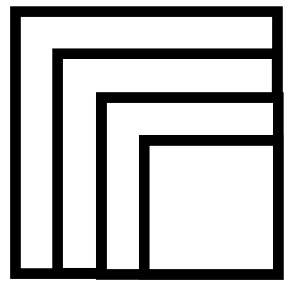
```
def fn0(fn):
  for i in range(4):
    fn()
    rotate(90)
```

```
fn0(
  lambda: (
    pendown();
    forward(1);
    penup()
))
```



 *a small square*

 *a medium square*

 *4 nested squares*

Growing the library and the set of solved programs

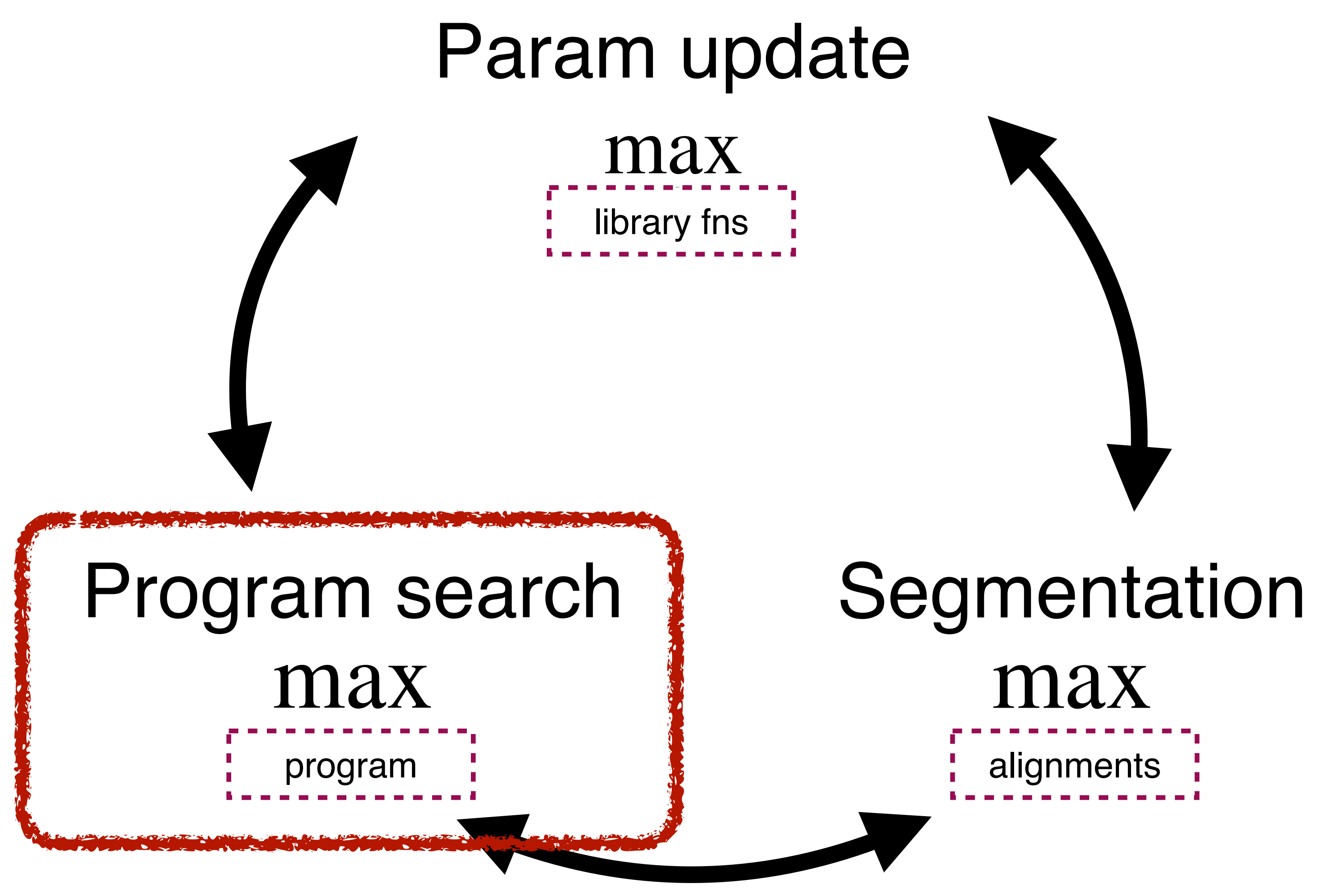
[c.f. Ellis et al. 21, *DreamCoder.*]

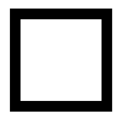
square

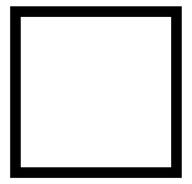
```
def fn0(fn):
  for i in range(4):
    fn()
    rotate(90)
```

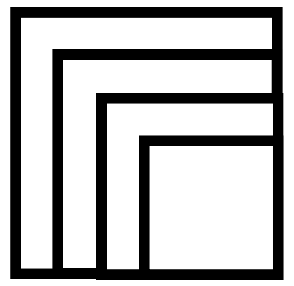
```
fn0(
  lambda: (
    pendown();
    forward(1);
    penup()
))
```

```
fn0(
  lambda: (
    pendown();
    forward(2);
    penup()
))
```



 *a small square*

 *a medium square*

 *4 nested squares*

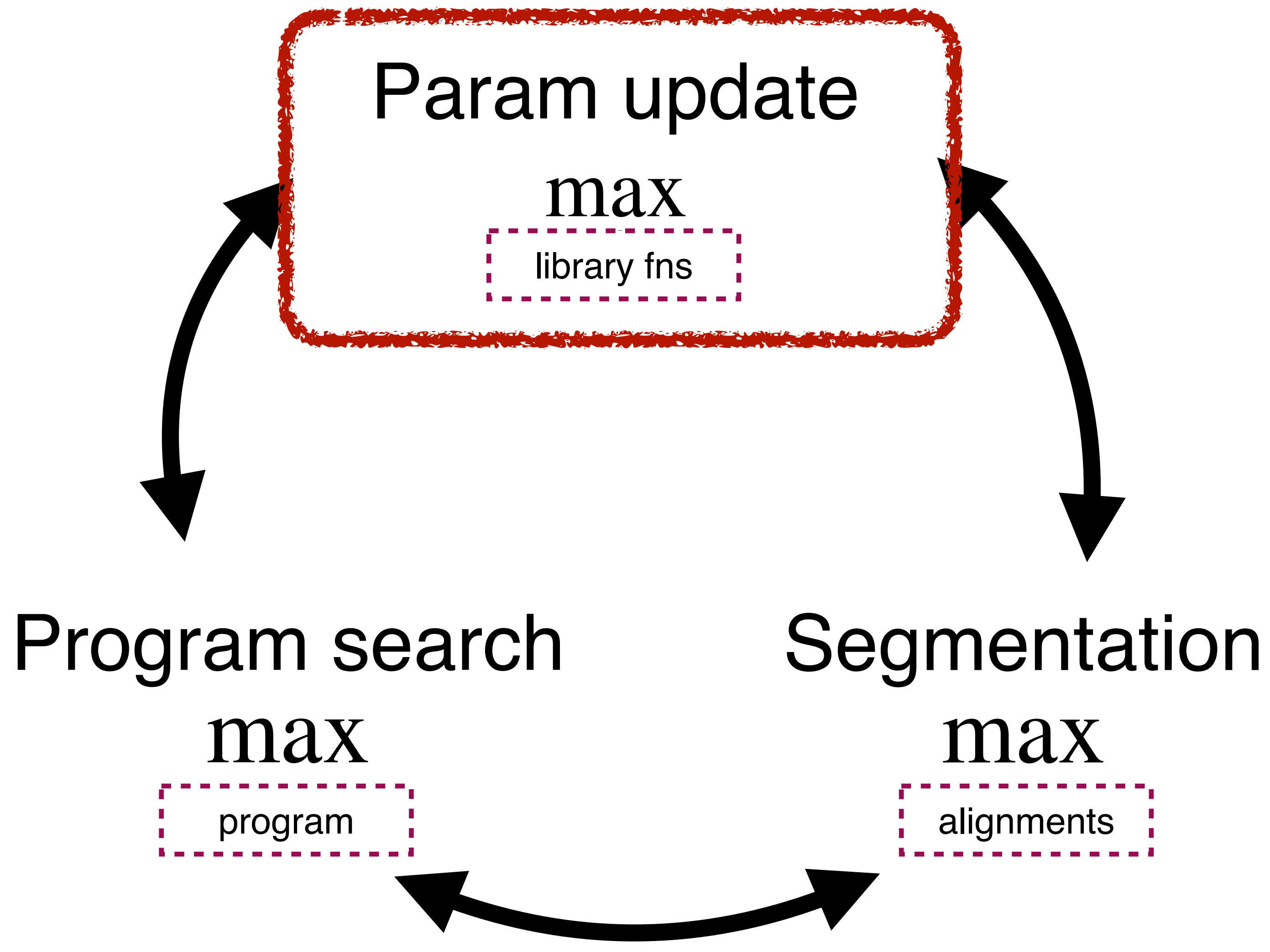
Growing the library and the set of solved programs

[c.f. Ellis et al. 21, *DreamCoder.*]

square

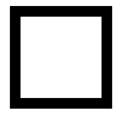
```
def fn0(fn):
  for i in range(4):
    fn()
    rotate(90)
```

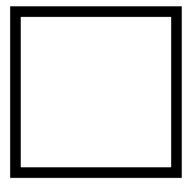
```
def fn1(x):
  pendown()
  forward(x)
  penup()
```

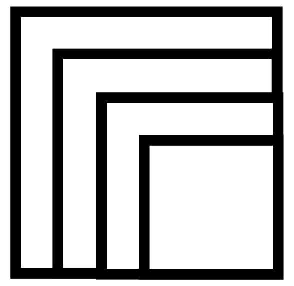



```
square(fn1(1))
```

```
square(fn1(2))
```

 *a small square*

 *a medium square*

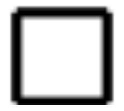
 *4 nested squares*

Data: inverse graphics

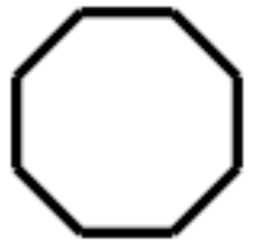
200 training images:

Simple shapes

a small triangle
small triangle



a medium square
one medium square



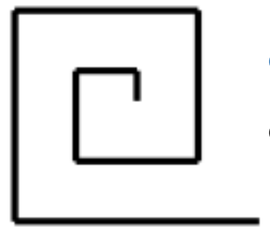
a medium eight gon
octogon

a big circle
just a circle

Complex objects

a seven pointed star
a seven sided snowflake with long triangles as arms

a four stepped zigzag
four step ladder going from top to bottom



a greek spiral with eight turns
a long line that curls in on itself at right angles

Compositional objects and relations

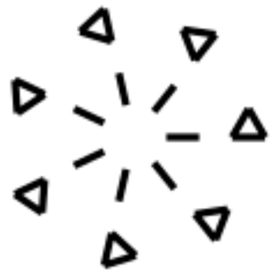
a small five gon next to a small seven gon
a five sided gon beside a seven sided gon

a small nine gon separated by a big space from a small circle
nine gon on left with small circle on right not connected

a small triangle connected by a big line to a medium triangle
a small triangle with a long line and a medium triangle

four nested squares
four stacked squares

six small five gons in a row
six overlapped pentagons going left to right



seven sided snowflake with a short space and a short line and a short space and a small triangle as arms
a seven sided snowflake with seven triangles and line

Results: inverse graphics

a small semicircle
`(f19 (f9 0 x))`


a medium semicircle
`(f3 (f9 0 x))`


a big semicircle
`(f9 (* (/ ε 1) 5) x)`


a small five gon
`(f5 5 x)`


a small nine gon
`(f5 9 x)`

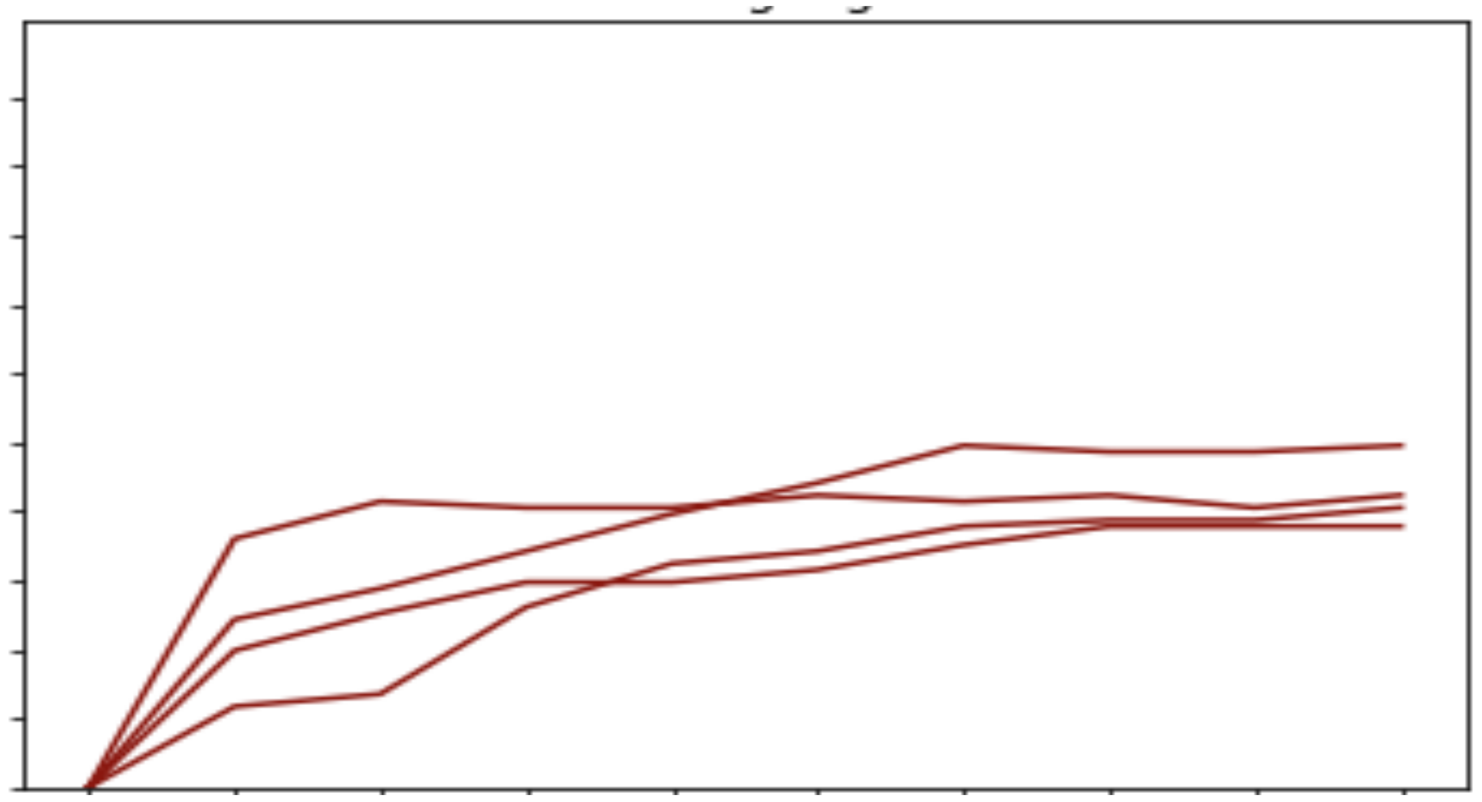

a medium seven gon
`(f5 2 (f20 7 x))`


eight sided snowflake with a small seven gon as arms
`(f24 7 8 x)`

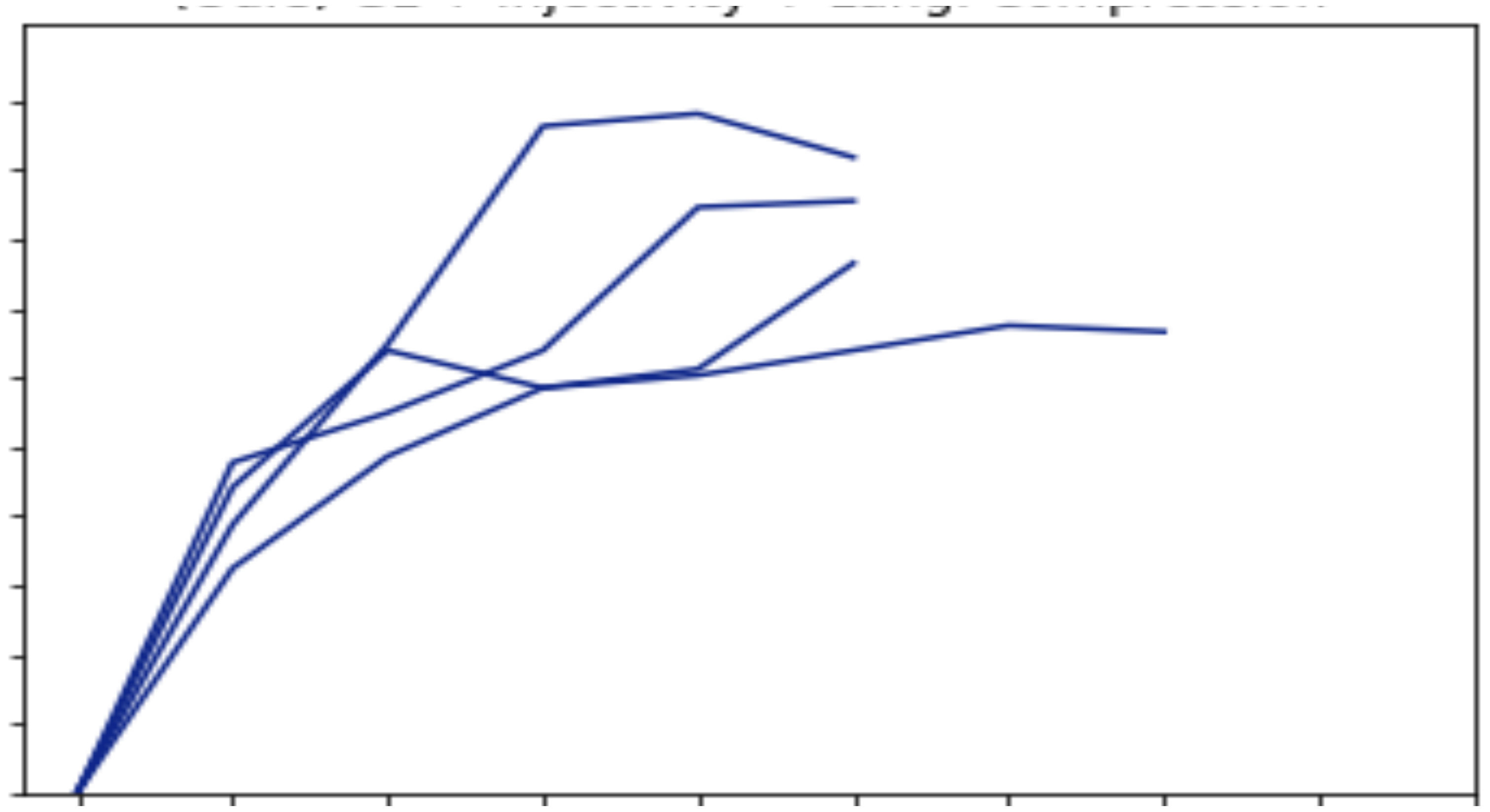

five sided snowflake with a short line and a medium five gon as arms
`(f24 5 (λ (x) (get/set (λ (y) (f2 1 (f41 5 y)))x)) z)`

% Solved (0 – 100%)

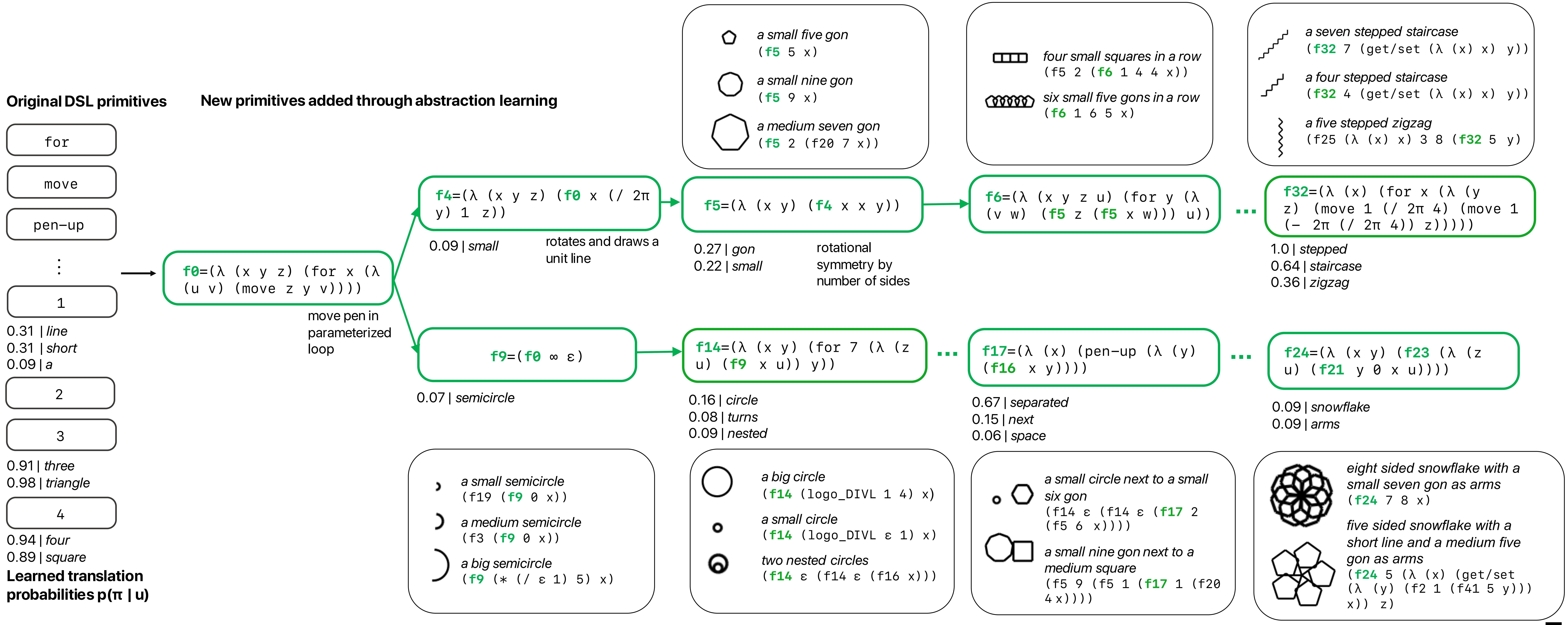
DreamCoder (no language)



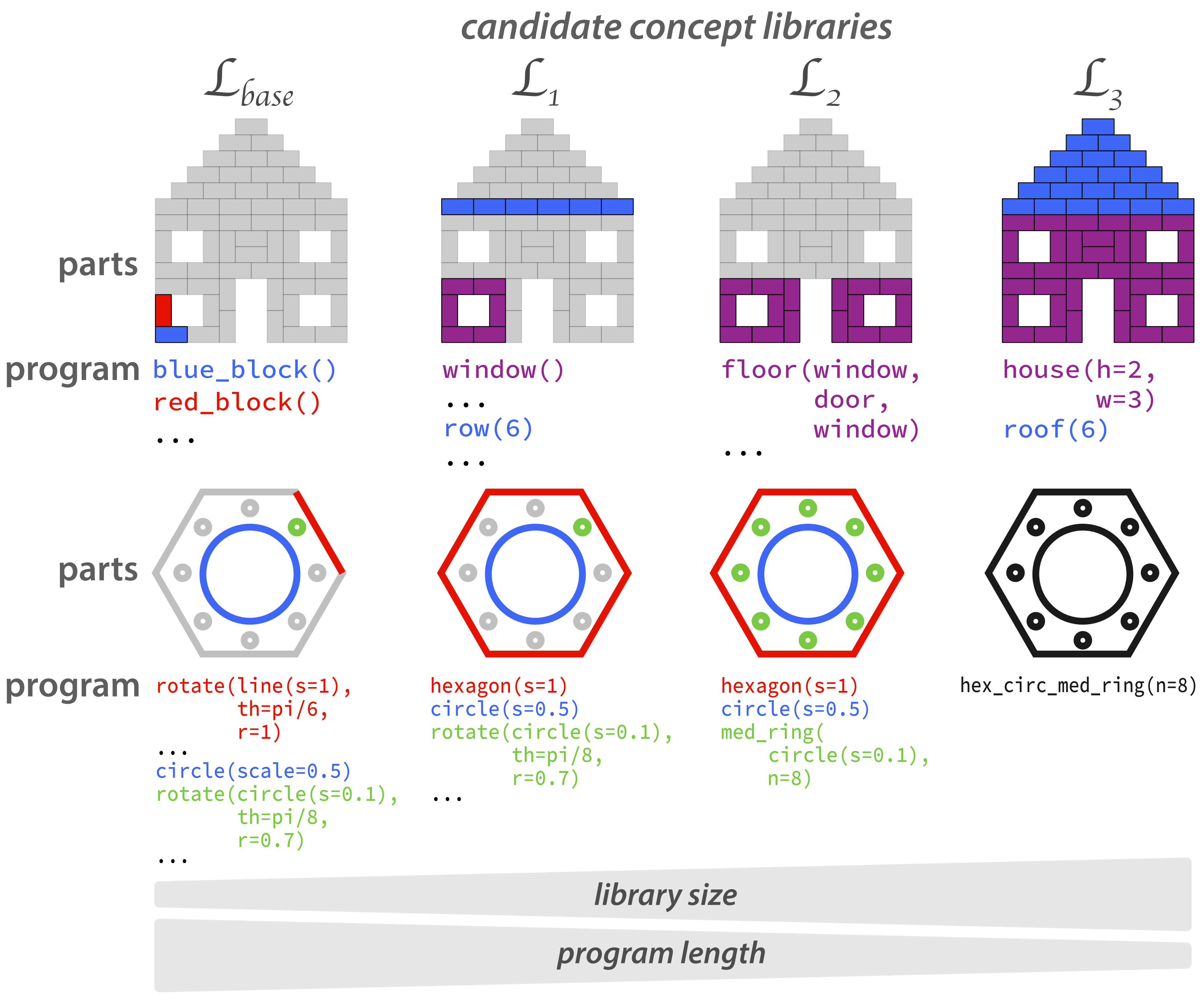
LAPS + ME + lang. compression

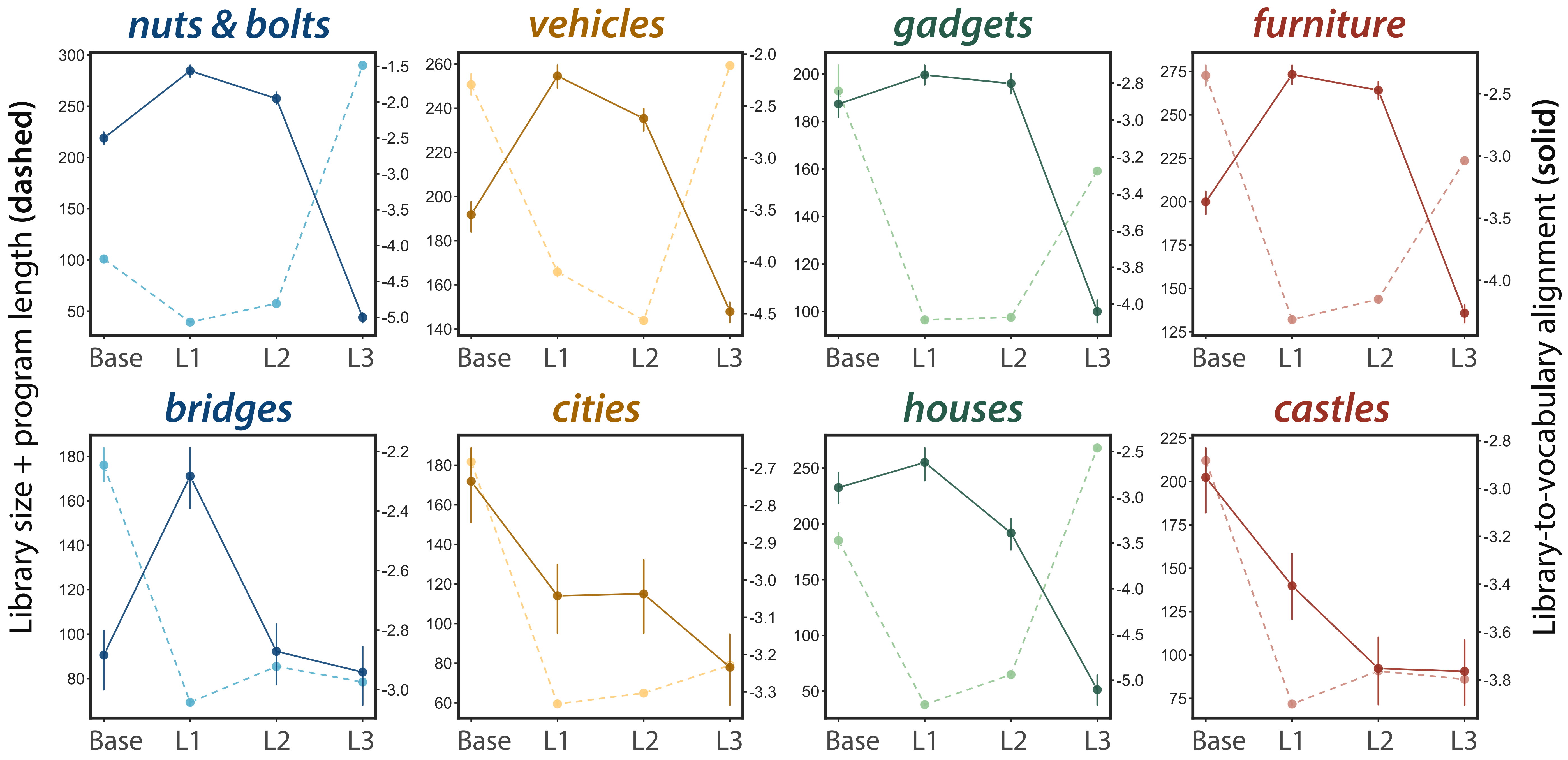


Language guides discovery of program abstractions



Library learning as a scientific tool





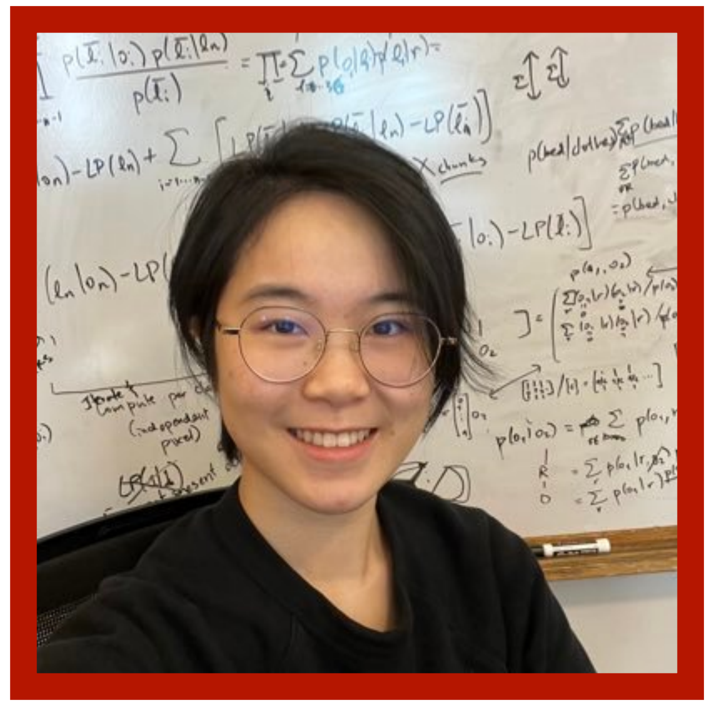
Learning libraries: summary

What: Inductive program synthesis with natural language guidance.

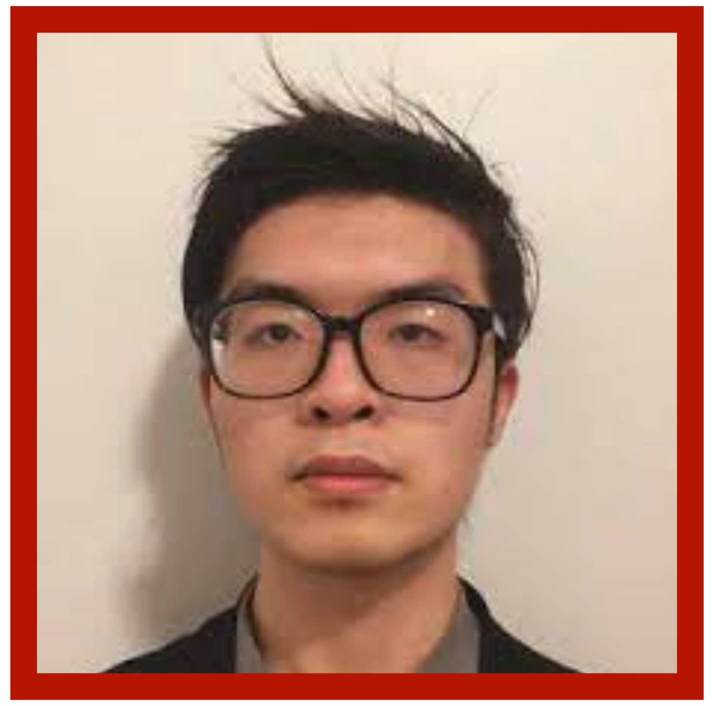
How: Discovery of reusable program fragments using language to guide a library learning procedure.

Why: With only 100s of annotations, solve 72% more program synthesis tasks than a leading synthesizer.

Learning from
unannotated text alone



Belinda Li



Will Chen

Pratyusha Sharma

[LaMPP: Language Models as Probabilistic Priors for Perception and Action. arXiv 2023]

Language as a latent variable, LMs as priors

cook the egg

Language Model Planner

find an egg

locate stove

Learned Skills

grasp(egg)

forward(9)

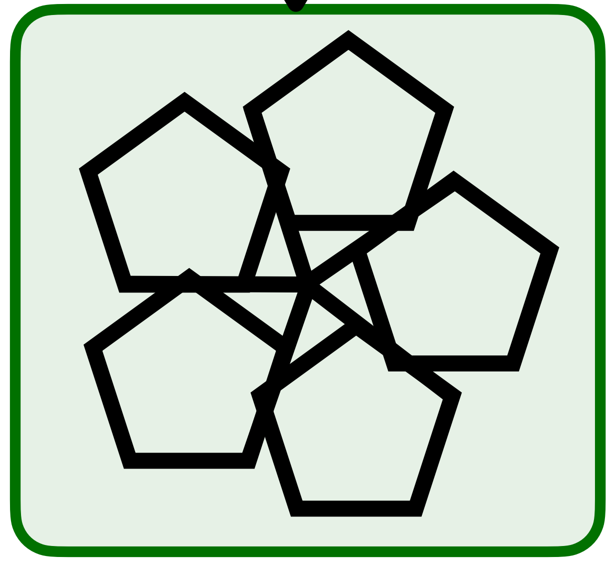
...

a pinwheel made of

Program generator

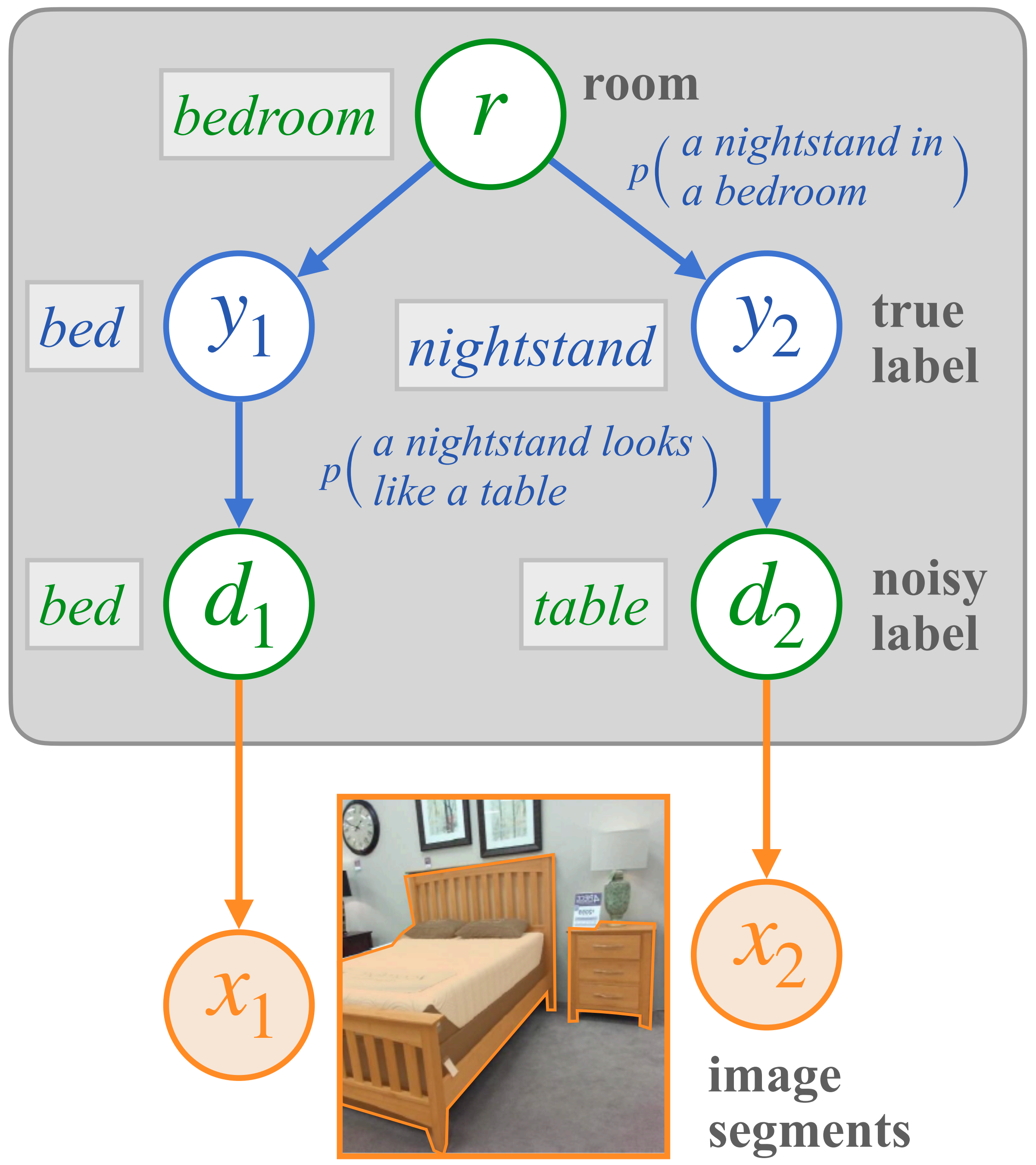
```
lambda x, f:
  for i in range(x):
    f()
  rotate(360/x)
```

Execution model

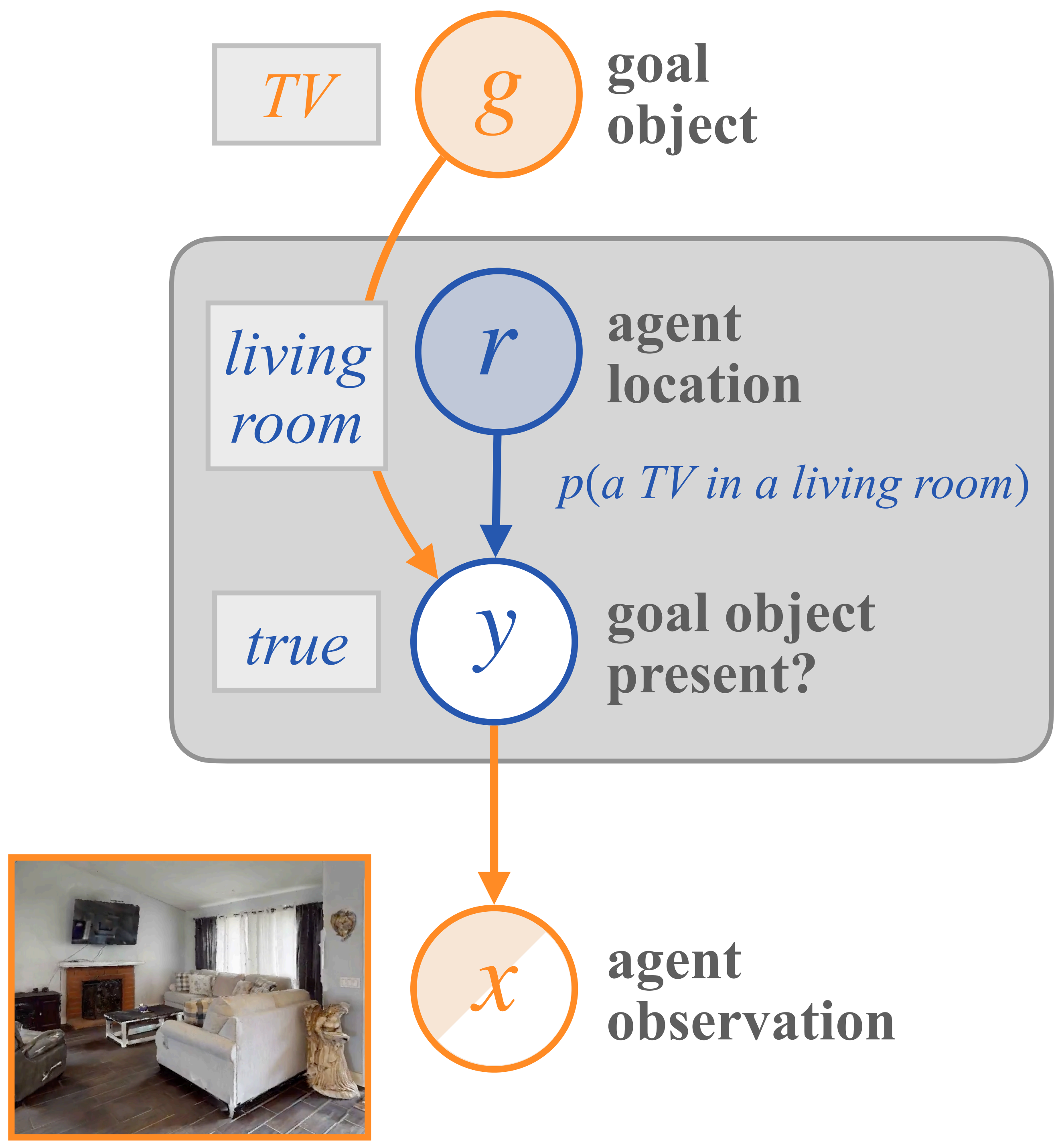


LM priors for vision & beyond!

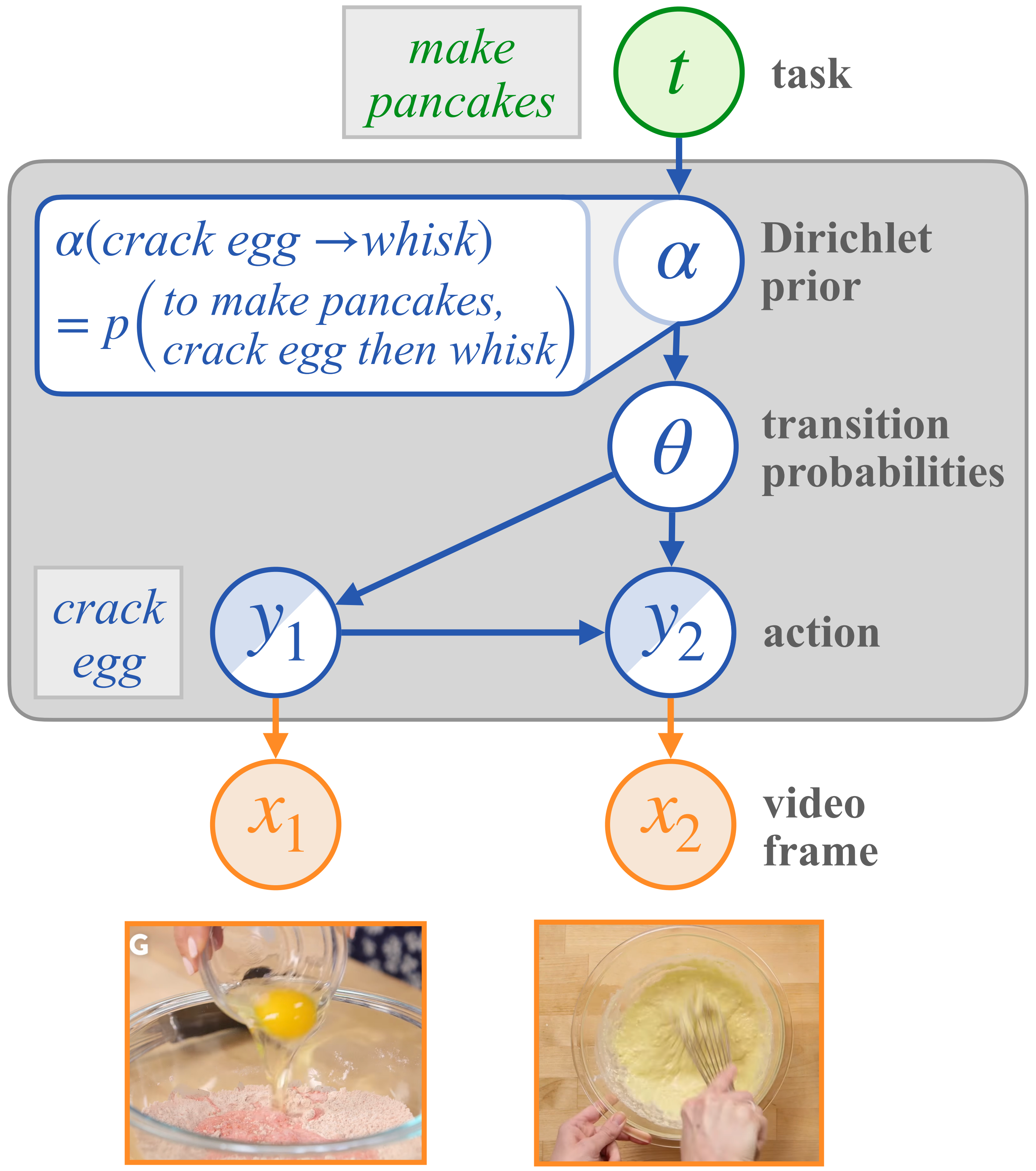
Semantic segmentation!



Household navigation!



Activity recognition!



LM priors improve generalization

	Model	mIoU	Best/Worst Object (ΔIoU)
ID	Base model	47.8	-
	Model chaining	37.5	shower curtain (+16.9) toilet (−37.2)
	LAMPP	48.3	shower curtain (+18.9) desk (−2.16)
OOD	Base model	33.8	-
	LAMPP	34.0	nightstand (+8.92) sofa (−2.50)

Model	Success rate Class	Freq.	Best/Worst Object (ΔSR)
Base model	52.7	53.8	-
Uniform prior	52.1	51.7	-
Model chaining	61.2	65.3	Toilet (+20.9) TV Monitor (−4.2)
LAMPP	66.5	65.9	TV Monitor (+33.0) Plant (−0.0)

LM priors: summary

What: Language as a source of background knowledge in general probabilistic models.

How: Query LMs to *parameterize* domain-specific graphical models.

Why: Big increases on accuracy on rare labels, input configurations.

Learning interactively

Jesse Mu

[Compositional Explanations of Neurons. NeurIPS 2020.]

Evan Hernandez

Teona Bagashvili

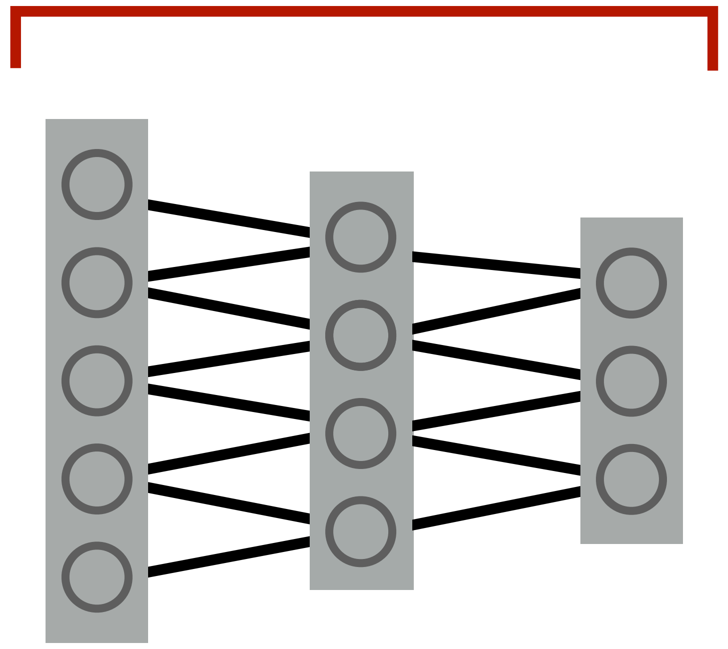
Sarah Schwettmann

+ David Bau and Antonio Torralba

[Natural Language Descriptions of Deep Visual Features. ICLR 2022.]

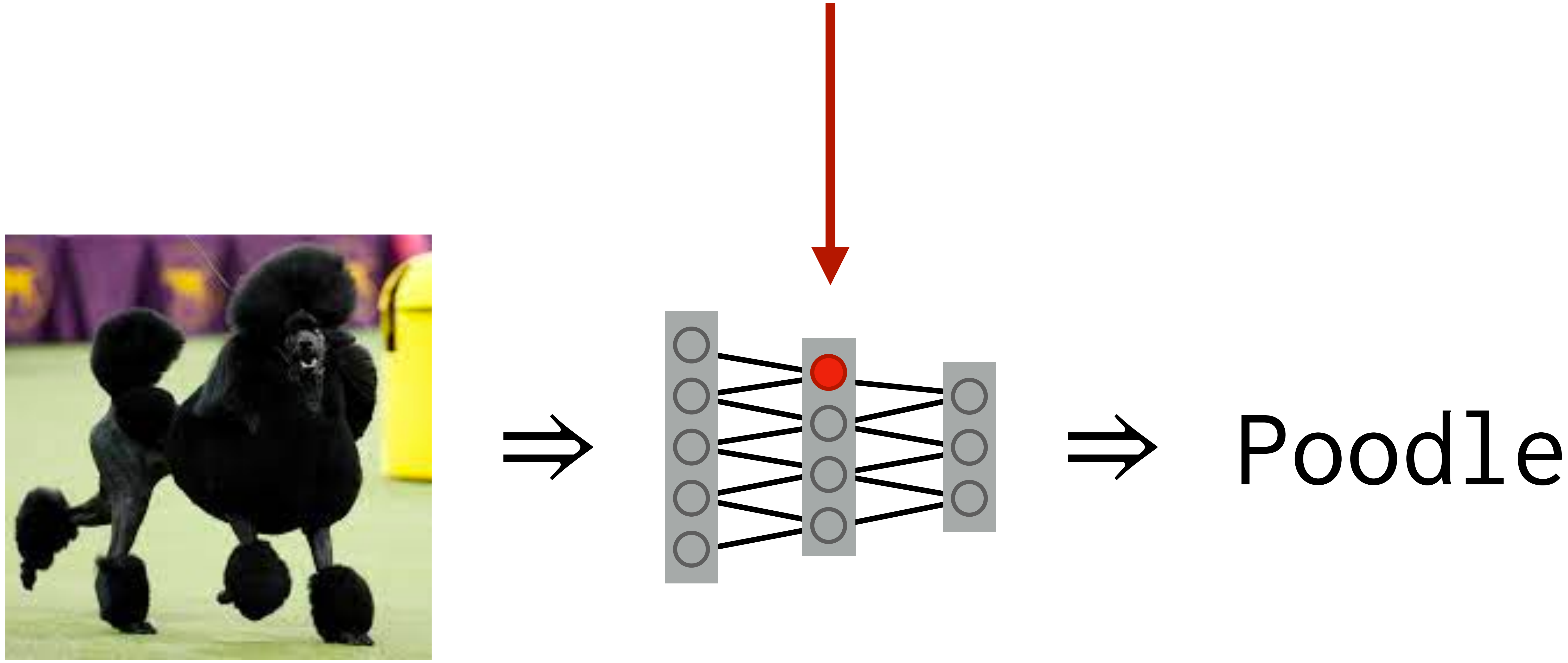
Understanding deep networks

What has this network learned?

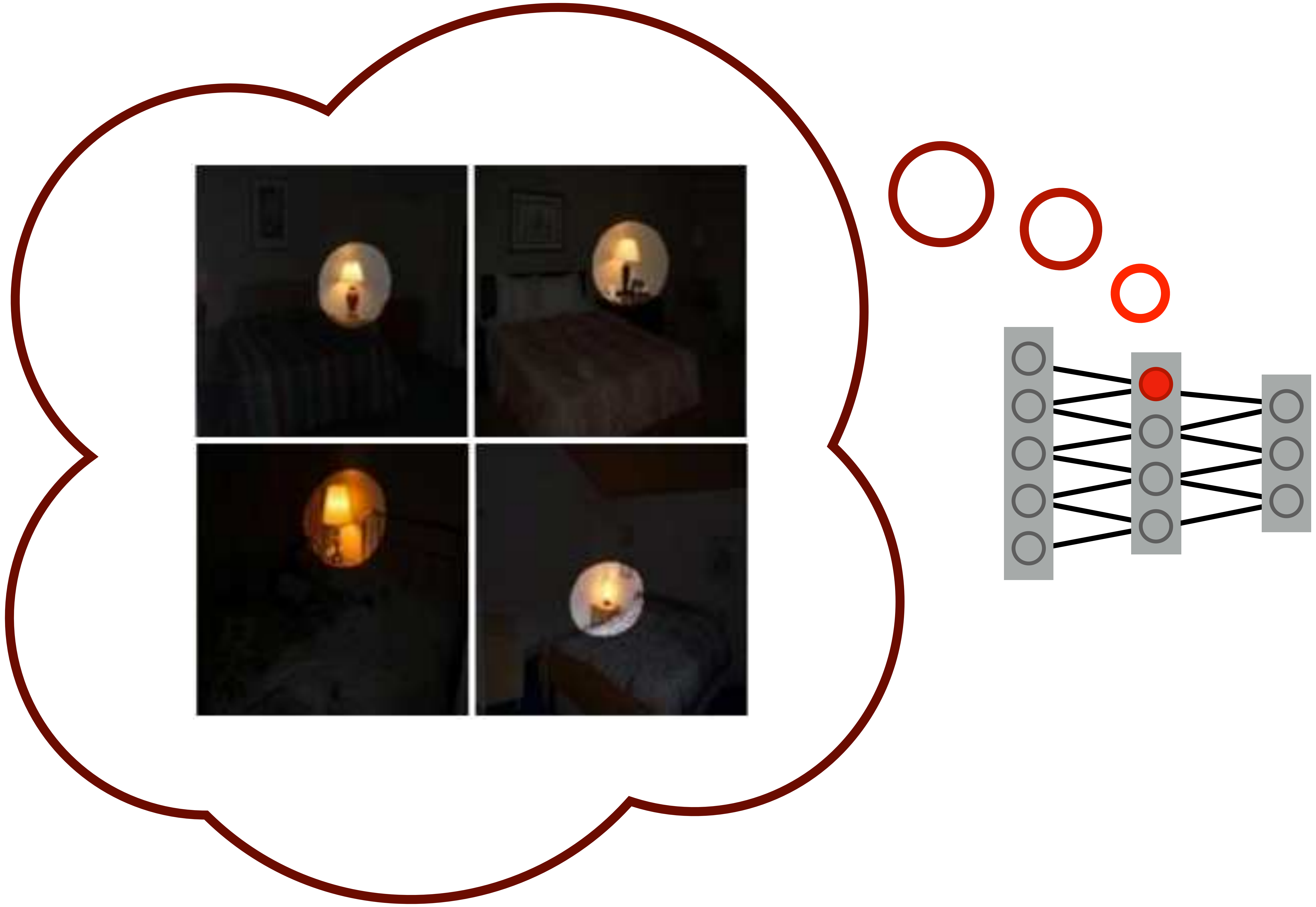
 $\Rightarrow$  $\Rightarrow$ Poodle

Understanding features in deep networks

What is the function of this neuron?



Labeling neurons **with visualizations**



Idea: determine a neuron's function by identifying input (regions) that activate it.

Extremely labor-intensive!

Labeling neurons **with language**

neuron masks $M_{483}(\mathbf{x})$

$\rightarrow$

$$\lambda$$

$$\mathrm{pmi}(\,\cdot\,;\,\cdot\,)$$

$\rightarrow$

water that is not blue

$$\max \log p(\text{description} \mid \text{mask}) - \log p(\text{description})$$

Machine-generated neuron descriptions

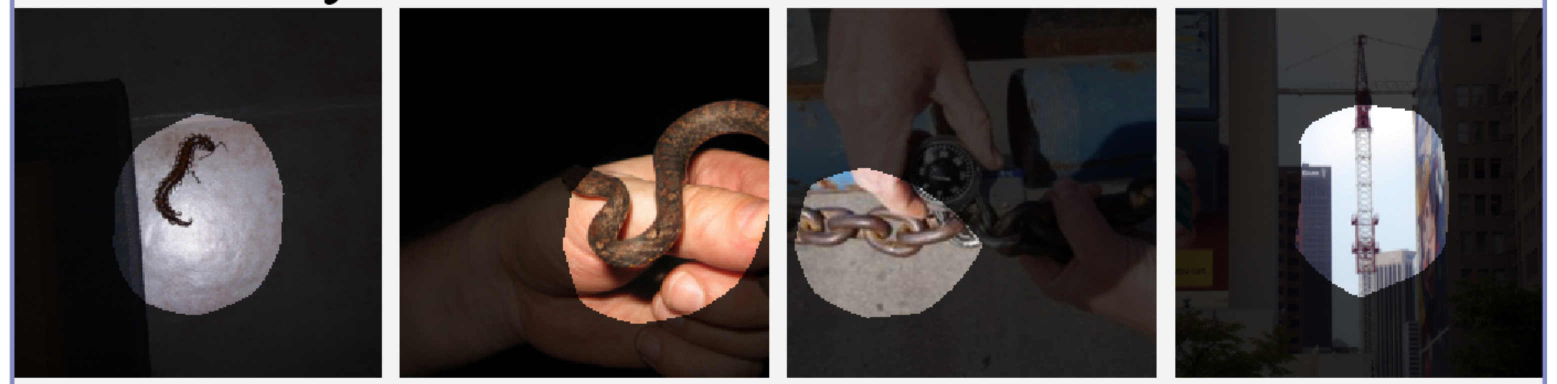
Generalization across architecture

AlexNet → ResNet

ResNet layer2-45

Human: the area on top off the line
MILAN: The top boundary of horizontal objects

ResNet layer4-1335

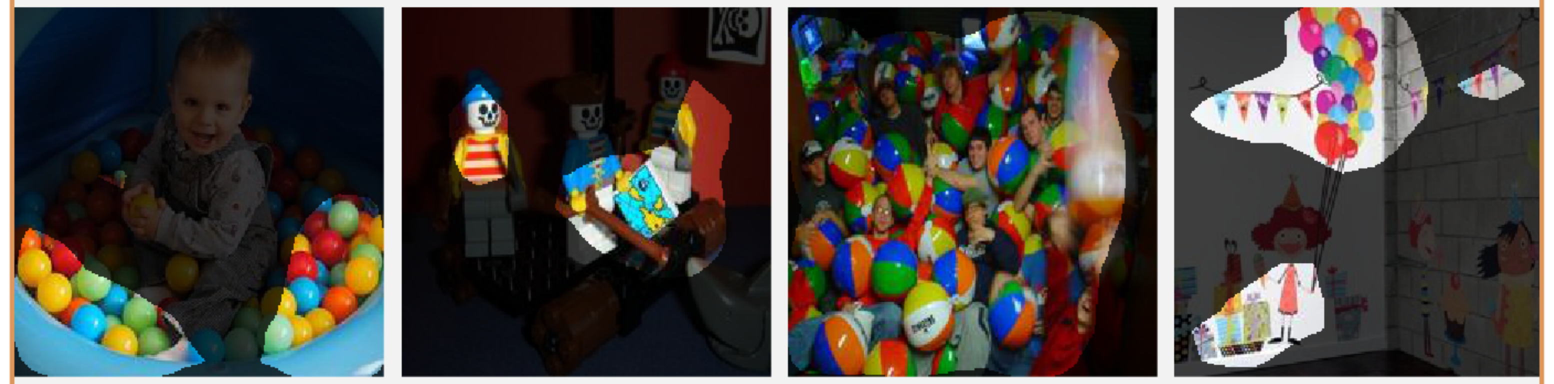


Human: long, thin objects
MILAN: Long slender objects

Generalization across dataset

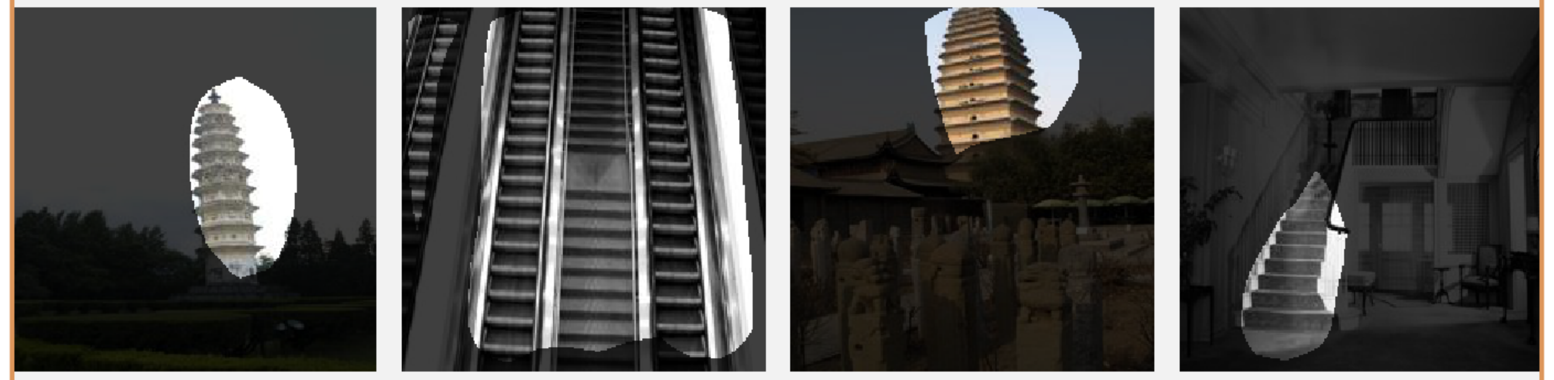
ImageNet → Places

AlexNet conv4-25



Human: colorful balls and parts from pictures
MILAN: colorful toys

AlexNet conv4-163



Human: buildings and stairs
MILAN: Objects with ridges

Generalization across task

CNN → GAN

BigGAN layer4-26

Human: houses built in the mountain cliff
MILAN: Rocks and stone walls

BigGAN layer1-528

Human: keyboards
MILAN: keyboards

Editing models

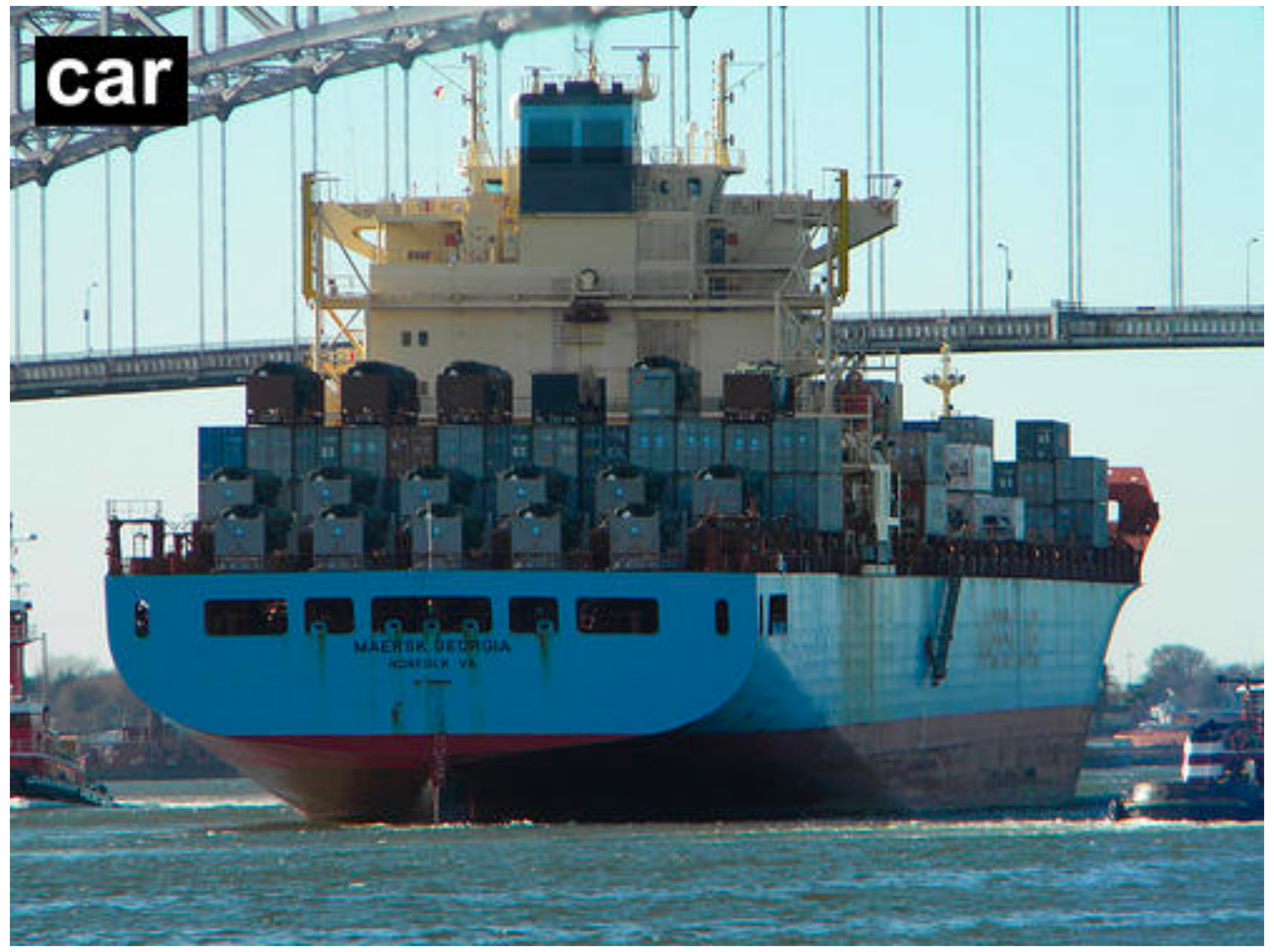
(a) training dataset

(b) adversarial test dataset

layer3-134, “words and letters”

(c) text neuron

Editing models



⇒ car

⬇

ship

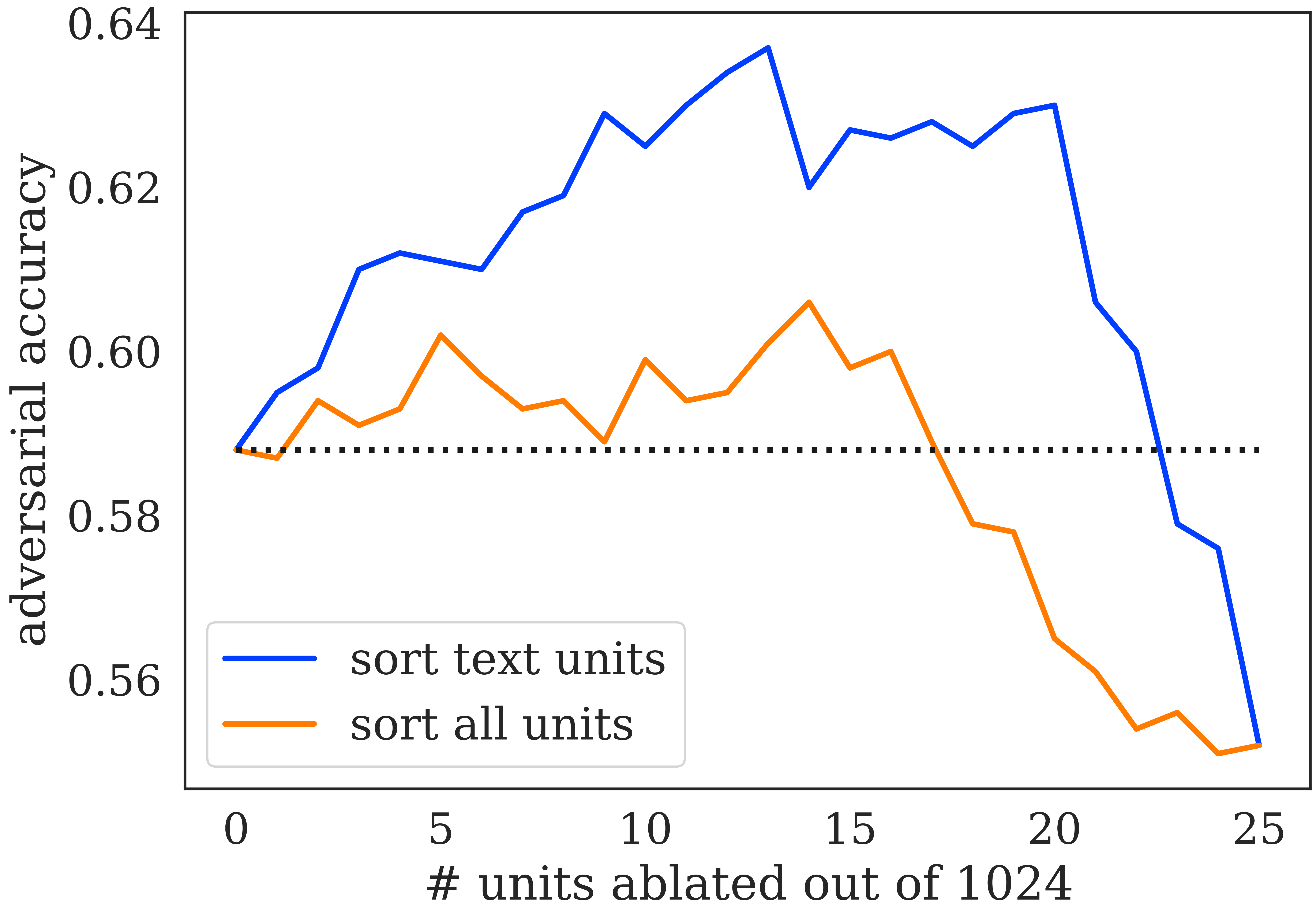
⇒ chihuahua

⬇

frog

Delete neurons labeled as text recognizers
→ 12% decrease in error rate!

Editing models

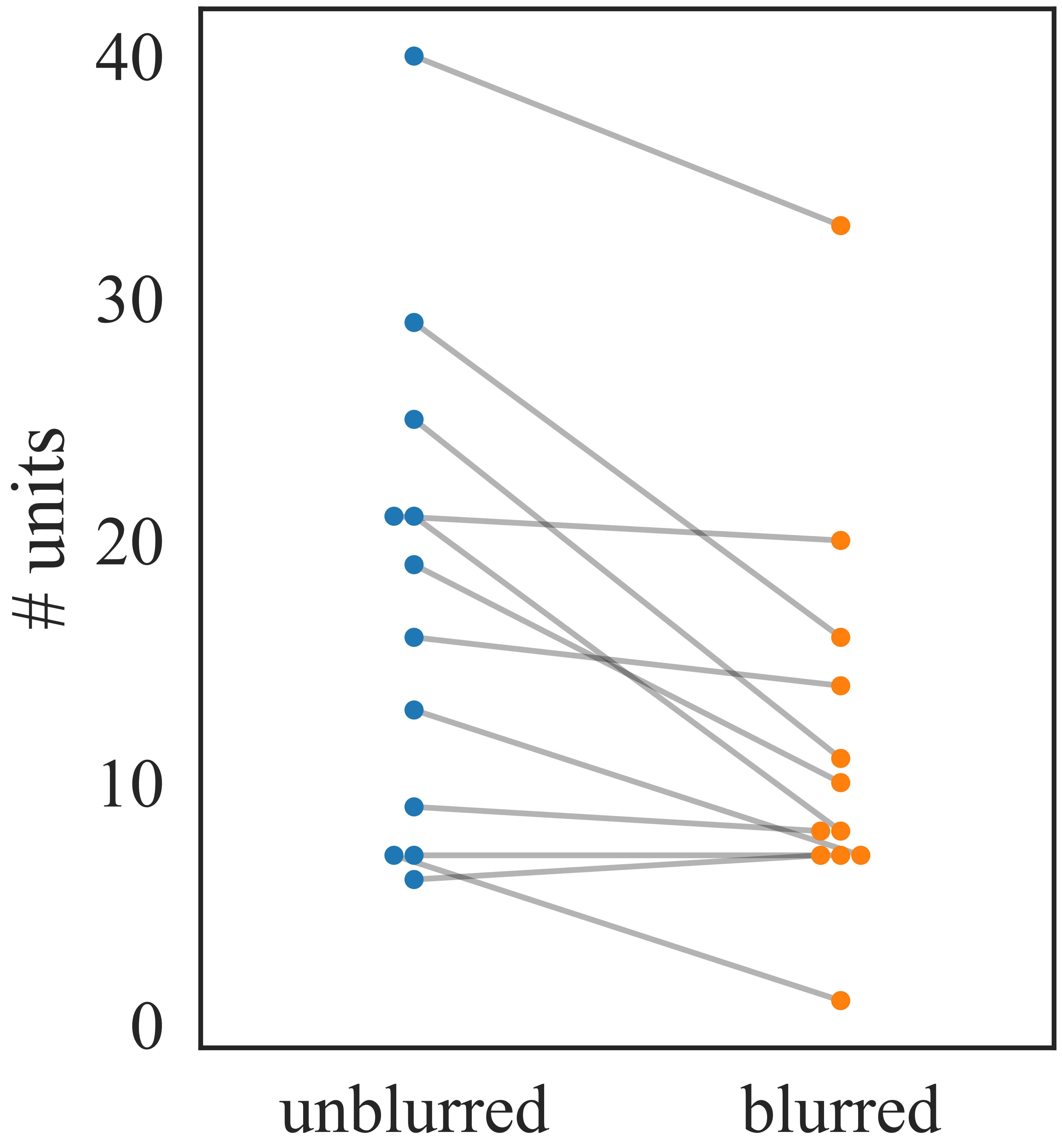


Auditing models

Blurred ImageNet [Yang et al. 2021]

Auditing models

Blurring reduces the number of face-sensitive neurons across 12 models...

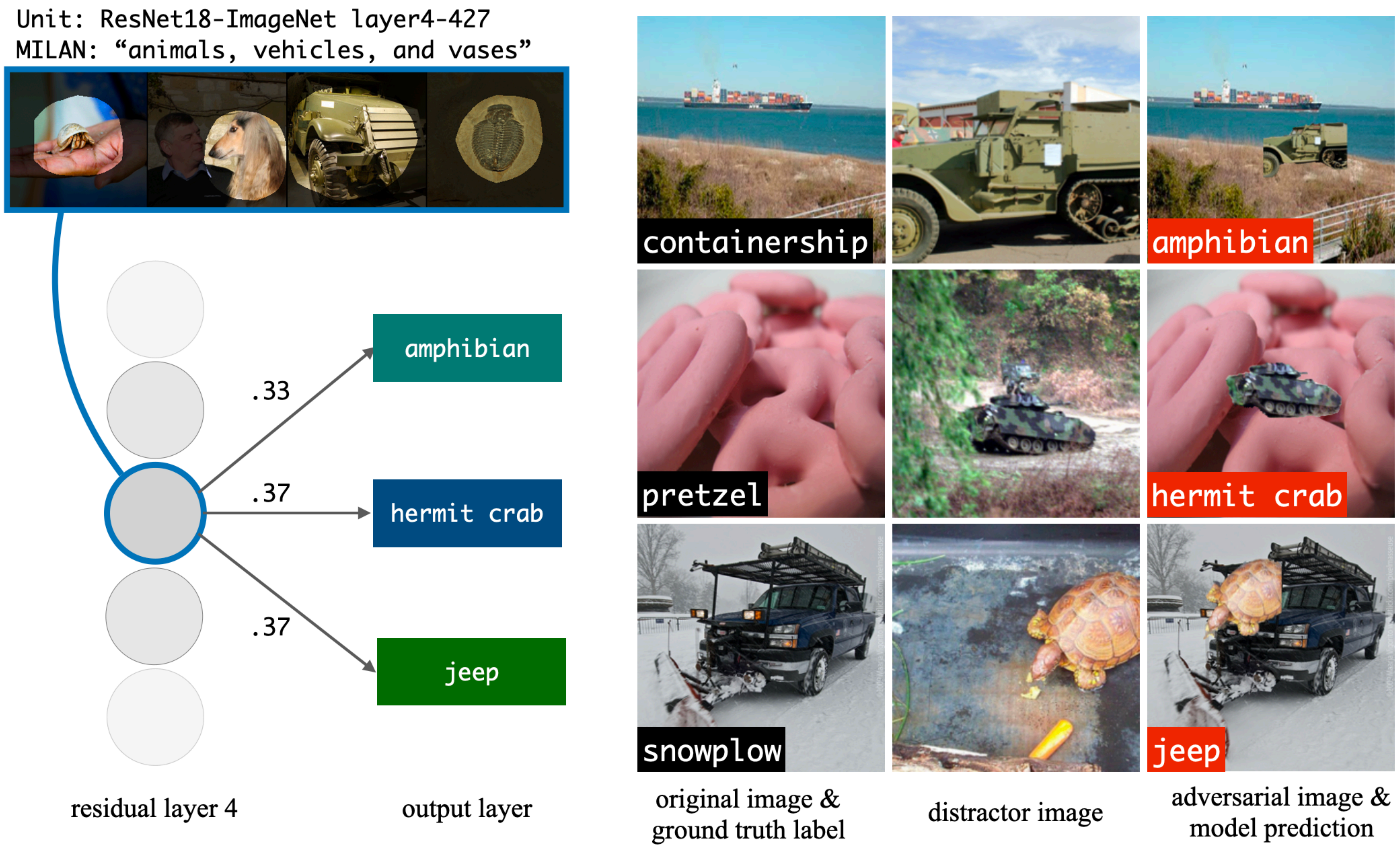


Auditing models

...but not entirely!

142 face-selective neurons across 12 models trained on blurred faces.

Adversarial examples

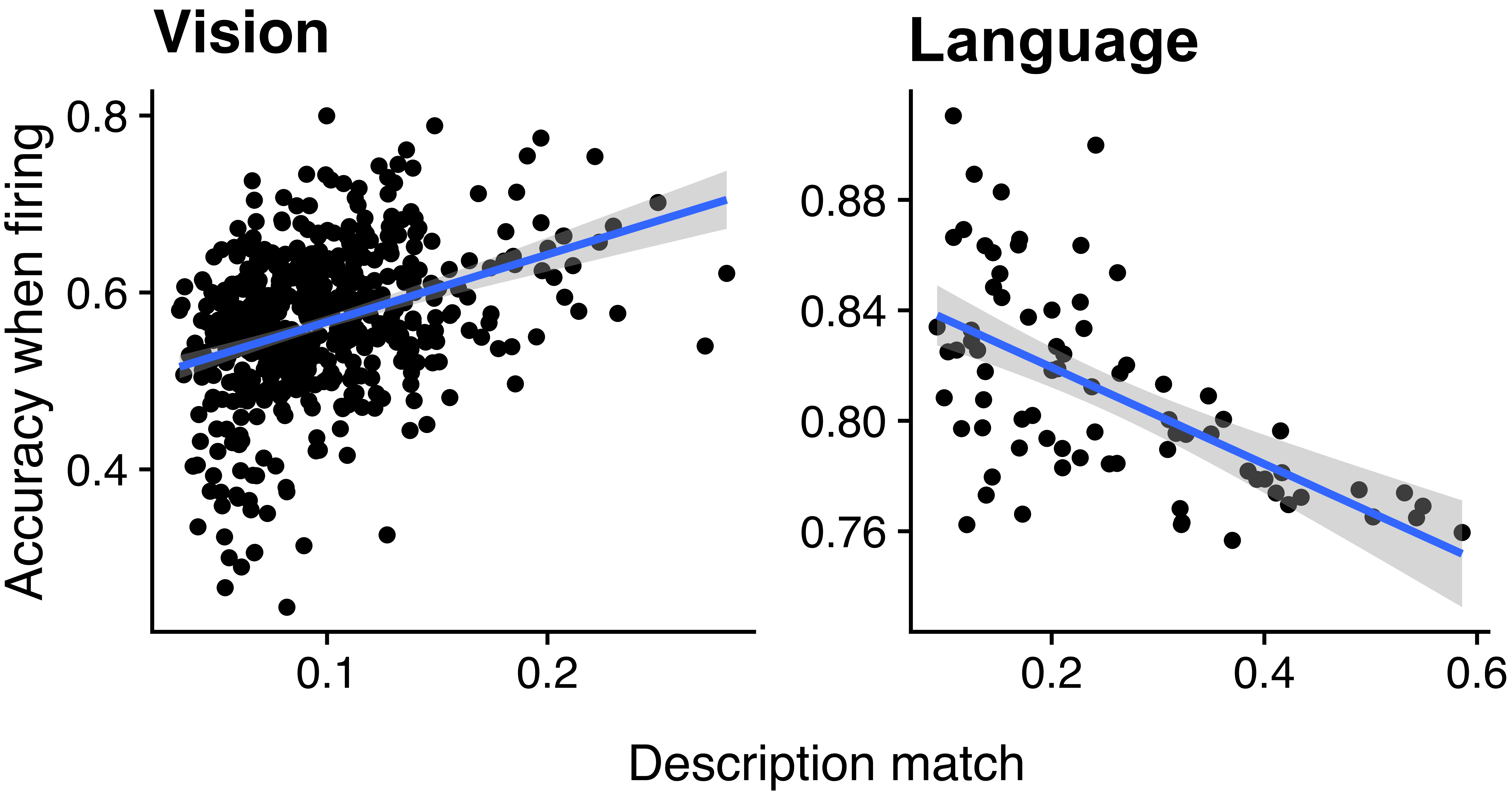


Adversarial examples in NLP

Unit 133 (couch words in hypothesis)

hypothesis contains: synonyms of couch or one of inside, indoors, home, eating.

Explainability and model accuracy



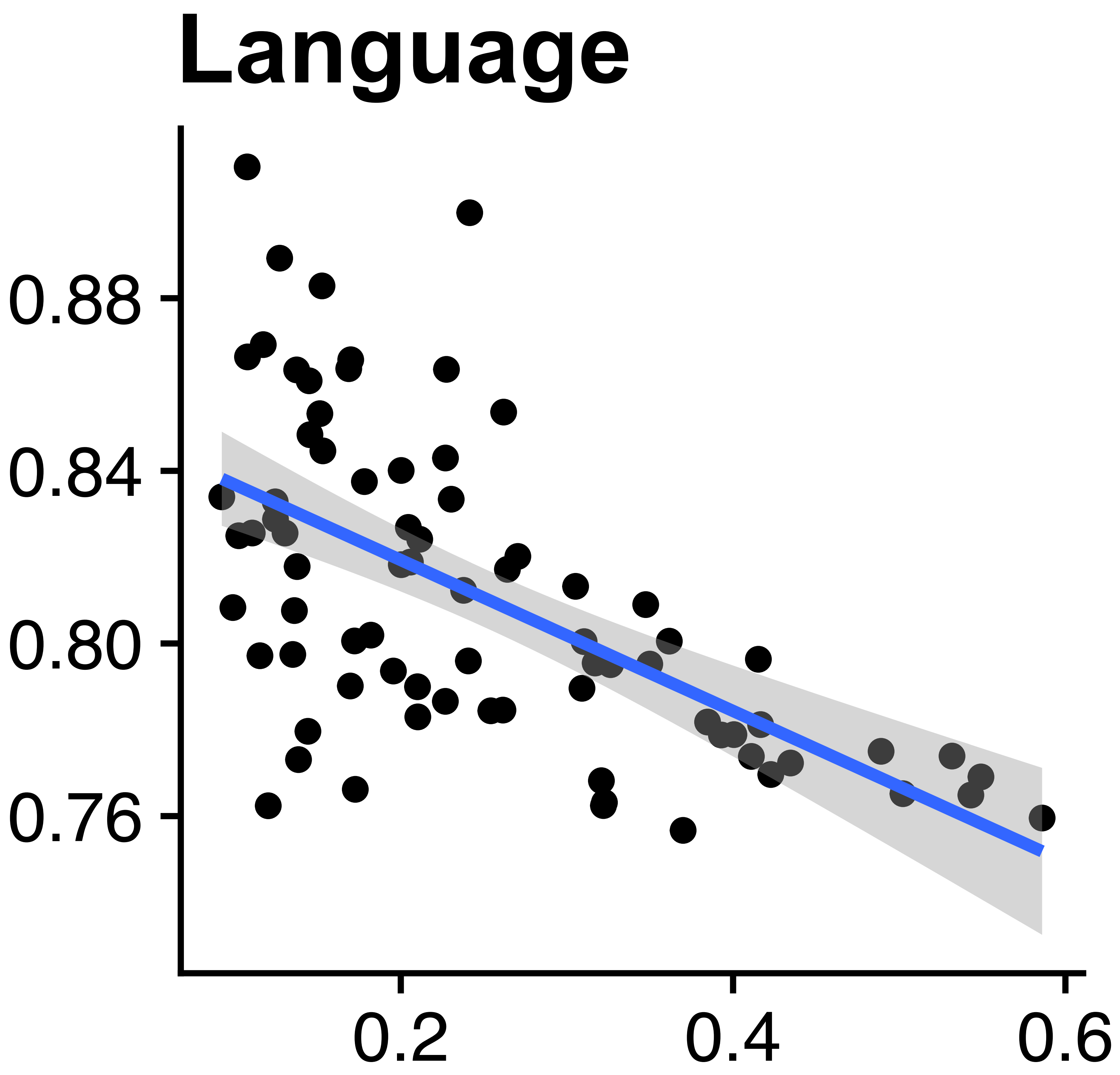
Explainability and model accuracy

Accuracy when firing

better explanations

↓

worse predictions!



The next challenge: *relevant* explanation

Green and yellow animals, a yellow smiley face, and a firefighter's head and jacket.
The heads of animals
about animals
foods in the packet
Dog and fox eyes.
This is a animal head.
Body part of the birds
The bottom portions of faces of animals.
Dog heads with black and white.
Anything that has the color blue in it.
These are animal heads.
These are animal heads.
about animals
This is showing both parts of animals and parts of wheels.
The color white
These are diagonal lines.
Shifting contrast colors, either light-dark-light or dark-light-dark.
Doors, windows, and see-through objects.
Turtle shells and regular overlapping patterns are

Red clothing, vehicles, plants, and objects.
Human skin
The black areas are highlighted in the images.
The images show body coverings of animals including fur, feathers, hair, and claws.
It shows an image that has a bit of white in it.
These are flowers and animal fur.
This regions is that is being highlighted are spots.
eyes and mouth
face of all animals
This is fruit and other circles.
Face of dogs
Dog faces and bodies.
The face area is highlighted in the images.
eyes and beak
People's faces and other body parts.
Green grass, plants, and objects.
This is black and white grids.
yellow spots surrounded by uniform colors
body of the dogs
arches
all the above are in green color

This is text.
The regions depict lines,
center.
They are the west or 9 o
objects that contain conc
This is the area above d
They are brownish fur.
Texts and blue or yellow
This is very natural area
side angle part of all obje
letters of the all images
Objects with curved edg
Core hours are set the ti
outside in the office.The
flexible.
All images are made up
They are circular objects
Blue colored objects.
Animal skins are in the i
The shiny white part of v
They are the midsection
Eyes of various animals,

Generating explanations: summary

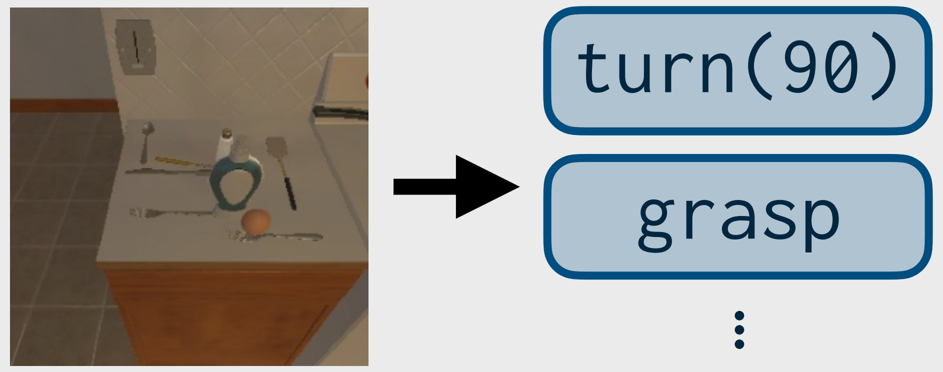
What: Automatic natural language labels for every neuron in a deep network.

How: Textual summaries of neuron visualizations using program synthesis & image captioning techniques.

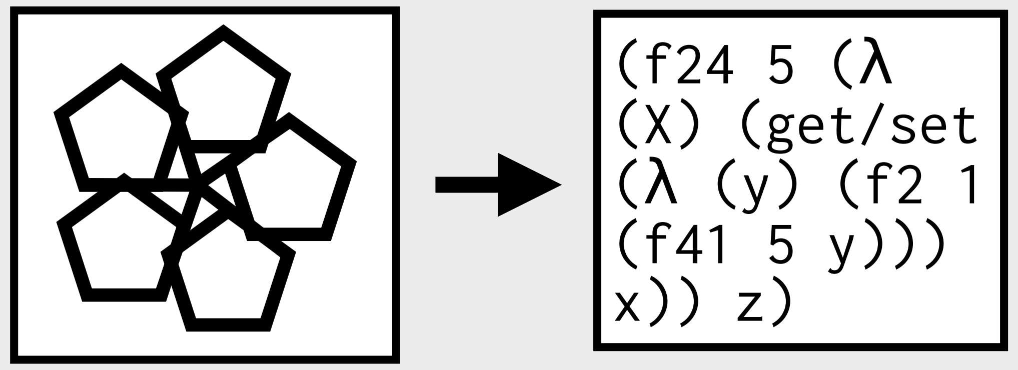
Why: Neuron explanations surface adversarial vulnerabilities, expose sensitive features, improve model robustness.

Toward natural language supervision

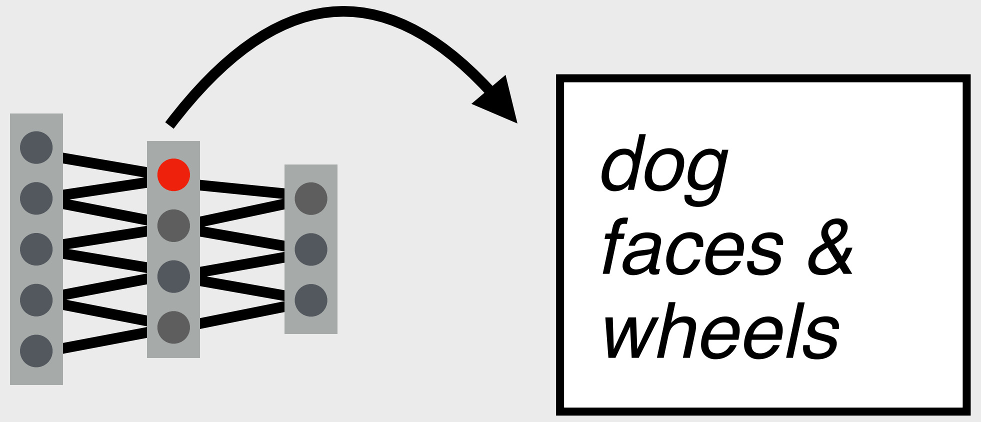
Learning from demonstrations



Learning from observations



Learning to explain



Effective & efficient natural language supervision is possible for lots of interesting learning problems!

Lio
Wong

Pratyusha
Sharma

Evan
Hernandez

Jesse
Mu

Teona
Bagashvili

Sarah
Schwettmann

Thank you!

NSF

SONY

intel

MIT CSAIL

Text-based image editing

original (a) “purple flowers” (b) “horizontal lines”

Force neurons with the desired label to activate:
controlled manipulation of image content!

Text-based image editing by activating descriptions