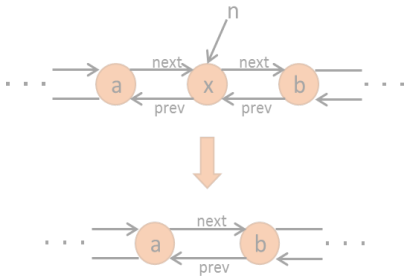
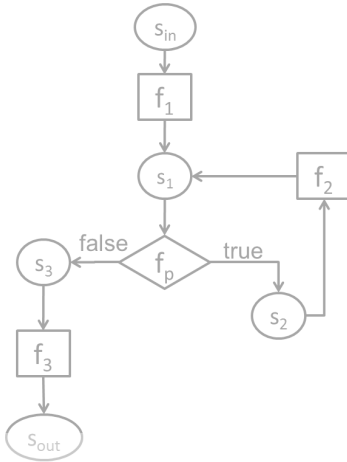


$$\exists c \forall in \ Q(c, in)$$

```

/* Average of x and y without using x+y (avoid overflow)*/
int avg(int x, int y){
    int t = expr({x/2, y/2, x%2, y%2, 2 }, {PLUS, DIV});
    assert t == (x+y)/2;
    return t;
}

```

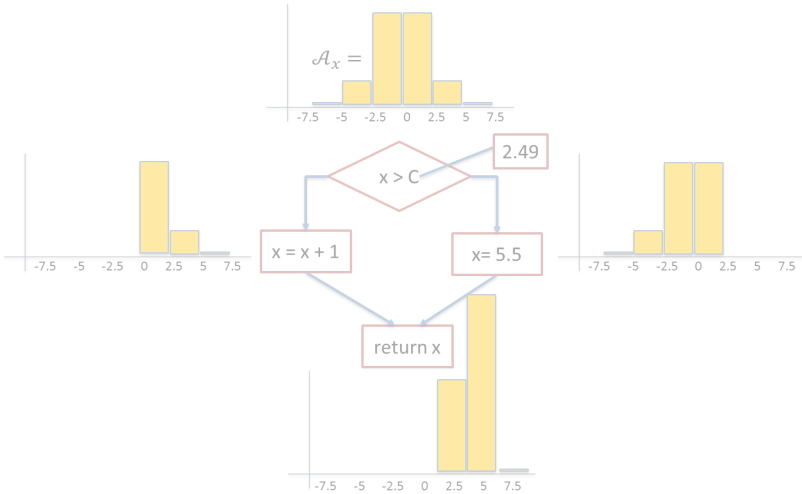
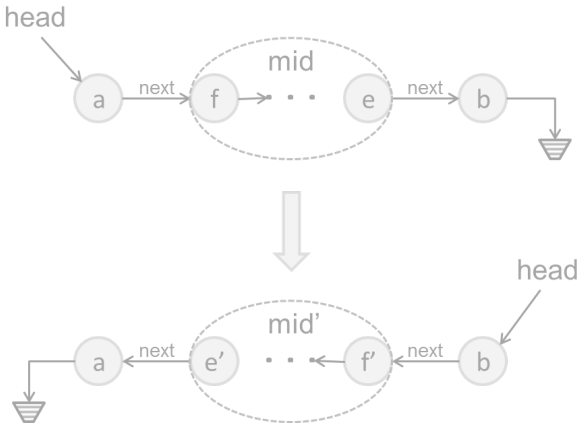


```

{
    s = n.succ;
    p = n.pred;
    p.succ = s;
    s.pred = p;
}

```

Module III: Advanced Topics



$$\varphi(p)$$

$$Sk[c](in)$$

Lecture 14

Human Interaction in Synthesis

Nadia Polikarpova

(with slides from Hila Peleg)

Project Logistics

Project presentations

- Tuesday Mar 21, 3-5:20pm
- 20 min per team (15 min presentation + questions)
- Structure: motivation, **demo**, technique, evaluation
- If your slot doesn't work for you, let me know

Project reports

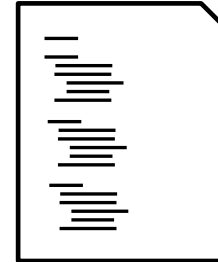
- Due on Mar 24 (start early!)
- Format: see course organization page (3-5 pages, ACM format)

What we've seen so far

specification



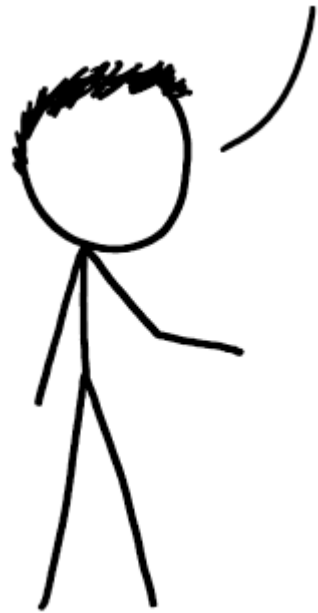
code



focus on the search

The big picture

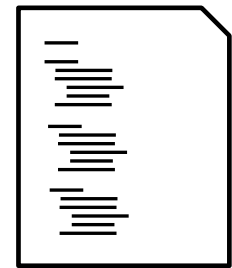
This is what I want



specification



code



Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	
4	Michelle	Casbon	15:55	
5	Mikael	Vidstedt	10:30	
6	Sam	Aaron	16:40	
7	Anita	Sengupta	17:40	
8	Jessica	Kerr	09:00	
9	Dave	Thomas	11:30	
10	Chris	Richardson	14:35	

Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

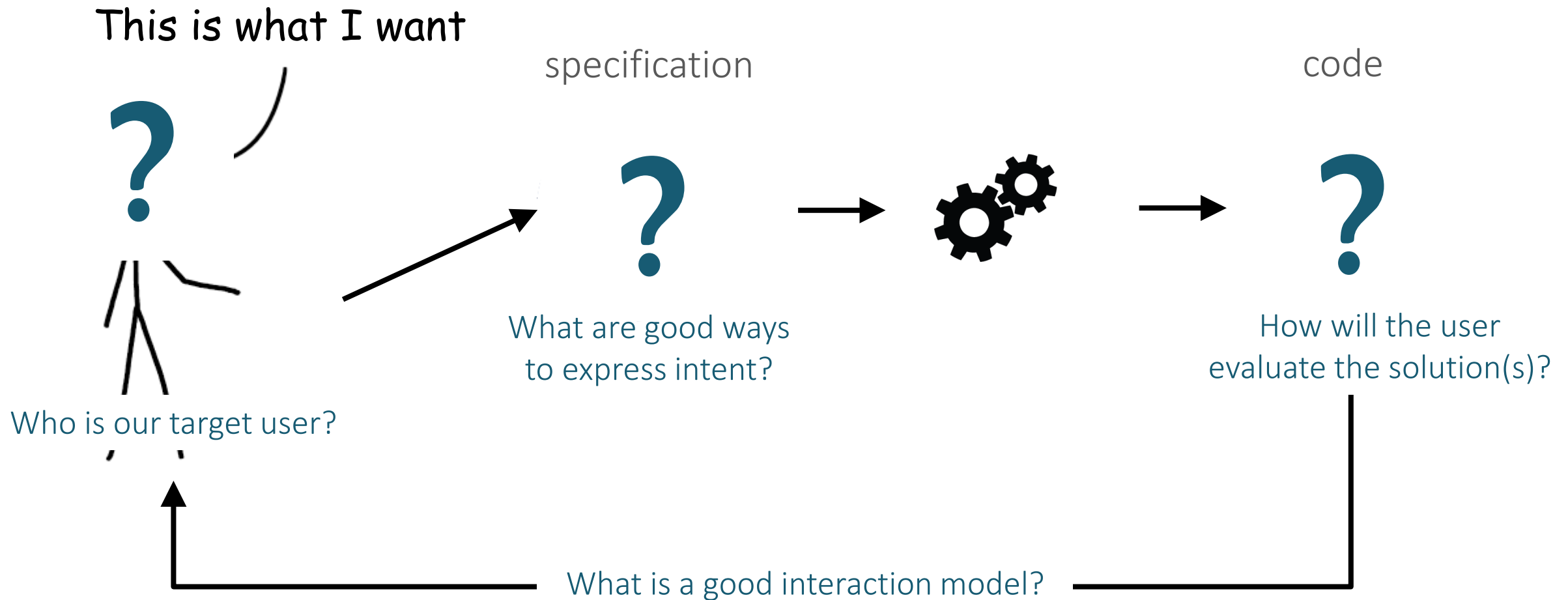
Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raika Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raika Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

The big picture



This week

Today: synthesis for programmers

- Snippy [Ferdowsifard et al, UIST'20; OOPLSA'21]
- Hoogle+ [James et al, OOPSLA'20]
- RESL [Peleg et al, OOPLSA'20; Peleg et al, ICSE'18]

Thursday: synthesis for non-programmers

- Rousillon [Chasins, Meuller, Bodik, UIST'18]
- Wrex [Drosos et al, CHI'20]
- Regae [Zhang et al, UIST'20]

Snippy

[Ferdowsifard et al, UIST'20]

Live Programming + Programming by Example

Live Programming

Visualize program state

name words
'Augusta Ada King' ['Augusta', 'Ada', 'King']
'Alonzo Church' ['Alonzo', 'Church']

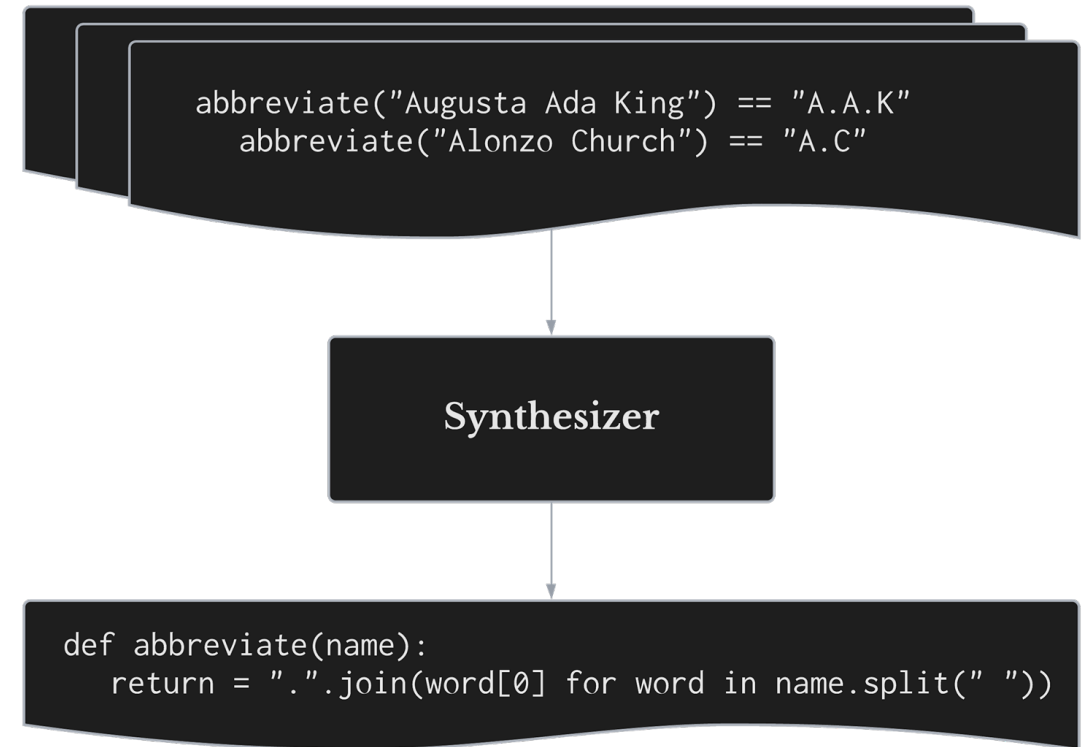
name words abbr
'Augusta Ada King' ['Augusta', 'Ada', 'King'] ''
'Alonzo Church' ['Alonzo', 'Church'] ''

name words abbr word c
0 'Augusta Ada King' ['Augusta', 'Ada', 'King'] '' 'Augusta' 'A'
1 'Augusta Ada King' ['Augusta', 'Ada', 'King'] '' 'Ada' 'A'
2 'Augusta Ada King' ['Augusta', 'Ada', 'King'] '' 'King' 'A'
0 'Alonzo Church' ['Alonzo', 'Church'] '' 'Alonzo' 'A'
1 'Alonzo Church' ['Alonzo', 'Church'] '' 'Church' 'A'

name words abbr word c rv
'Augusta Ada King' ['Augusta', 'Ada', 'King'] '' 'King' 'A' ''
'Alonzo Church' ['Alonzo', 'Church'] '' 'Church' 'A' ''

Programming By Example

Generate programs from examples



Live Programming

Visualize program state

#	name	words
	'Augusta Ada King'	['Augusta', 'Ada', 'King']
	'Alonzo Church'	['Alonzo', 'Church']

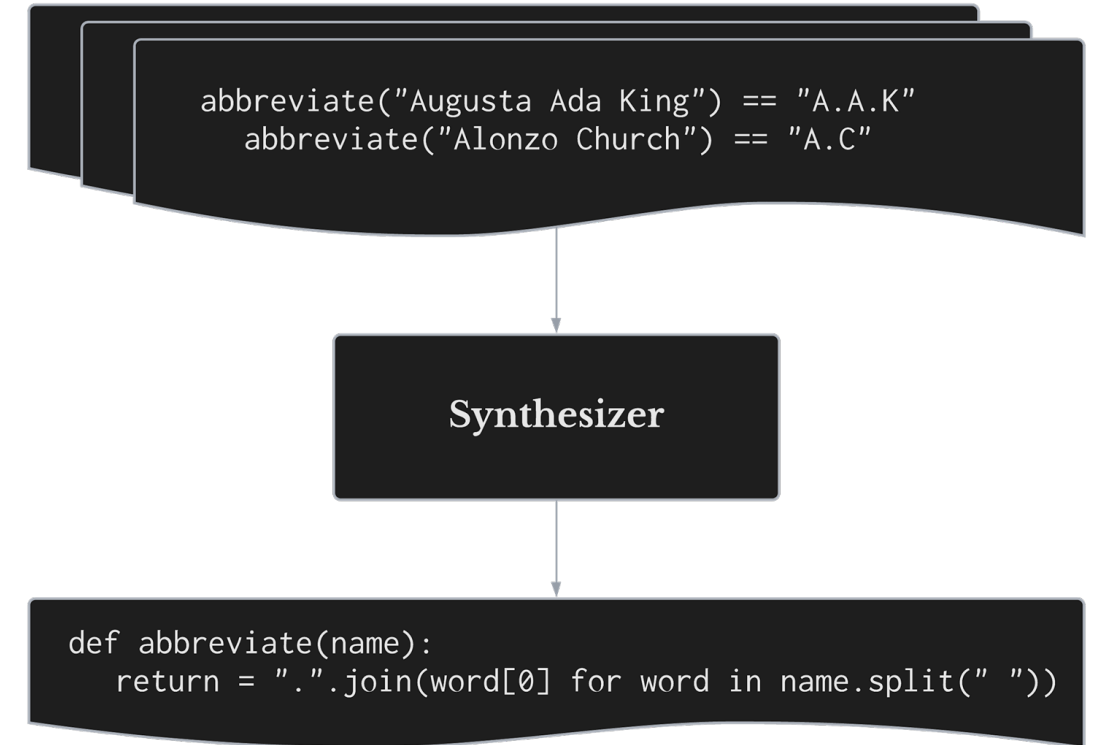
#	name	words	abbr
	'Augusta Ada King'	['Augusta', 'Ada', 'King']	''
	'Alonzo Church'	['Alonzo', 'Church']	''

#	name	words	abbr	word	c
0	'Augusta Ada King'	['Augusta', 'Ada', 'King']	''	'Augusta'	'A'
1	'Augusta Ada King'	['Augusta', 'Ada', 'King']	''	'Ada'	'A'
2	'Augusta Ada King'	['Augusta', 'Ada', 'King']	''	'King'	'A'
0	'Alonzo Church'	['Alonzo', 'Church']	''	'Alonzo'	'A'
1	'Alonzo Church'	['Alonzo', 'Church']	''	'Church'	'A'

#	name	words	abbr	word	c	rv
	'Augusta Ada King'	['Augusta', 'Ada', 'King']	''	'King'	'A'	''
	'Alonzo Church'	['Alonzo', 'Church']	''	'Church'	'A'	''

Programming By Example

Generate programs from examples

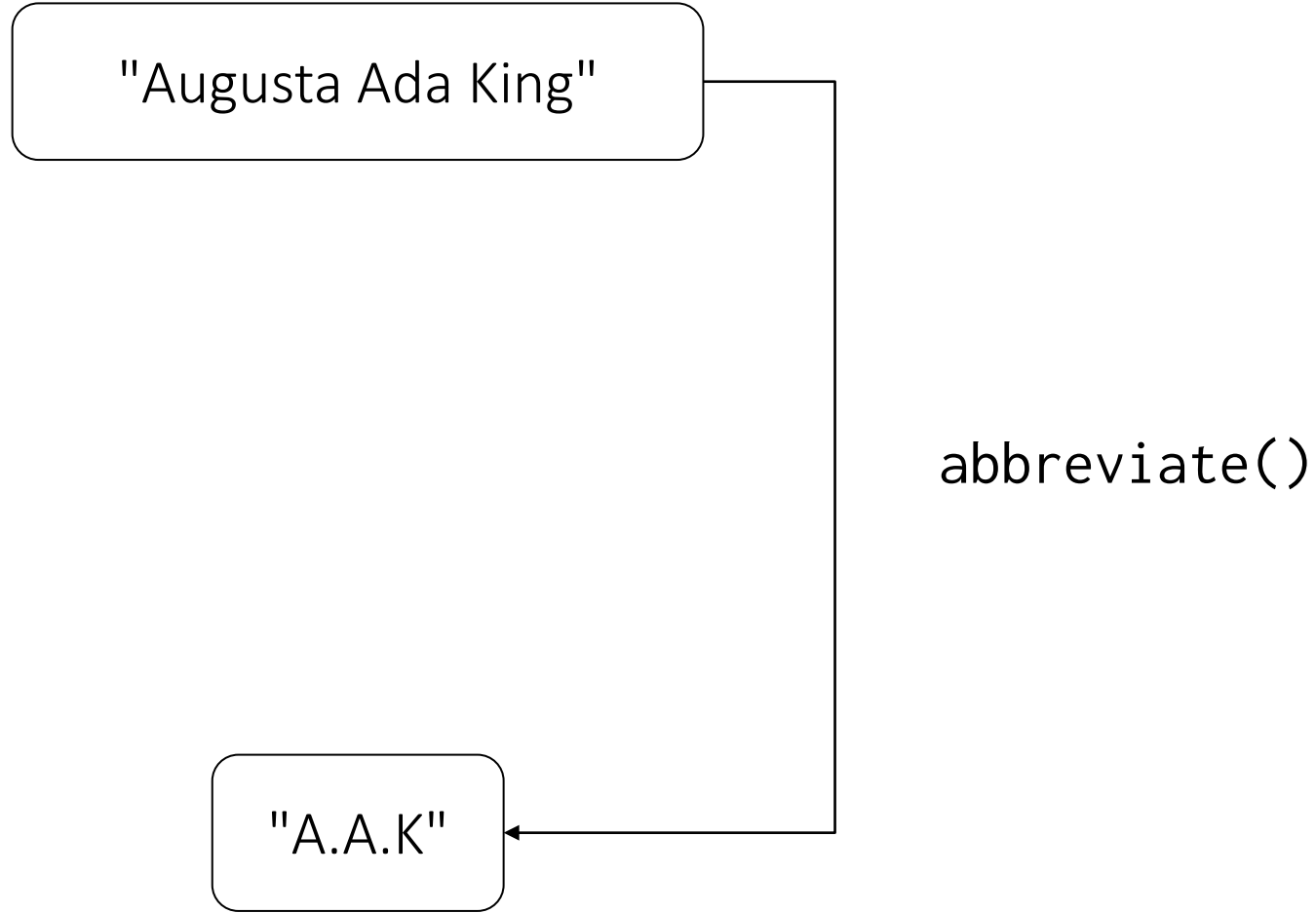


Snippy

Idea: Live Programming is a great environment for synthesis

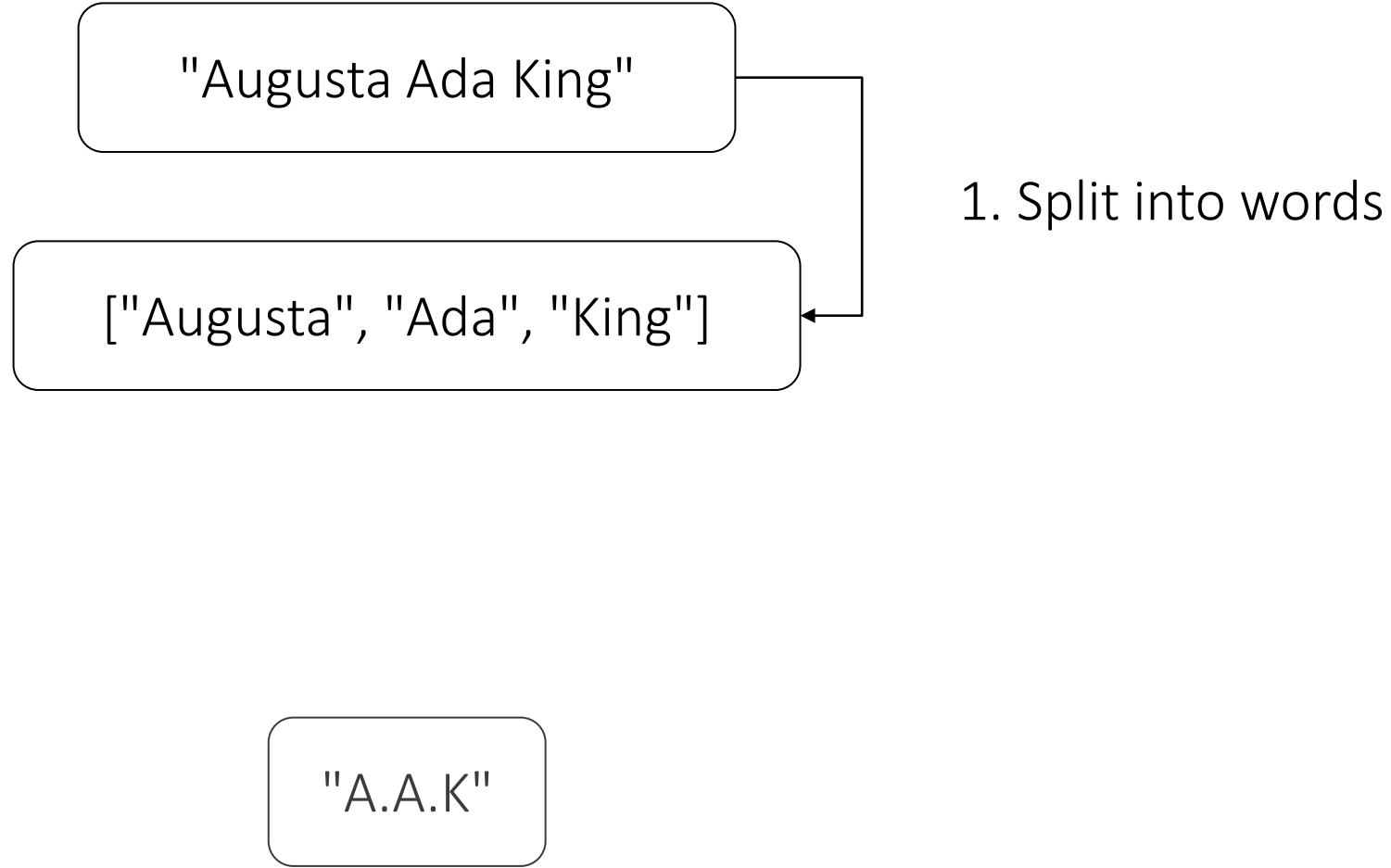
- inputs are already there (only need to add output)
- encourages “small-step” PBE (easier for the synthesizer)

Problem: Abbreviate*



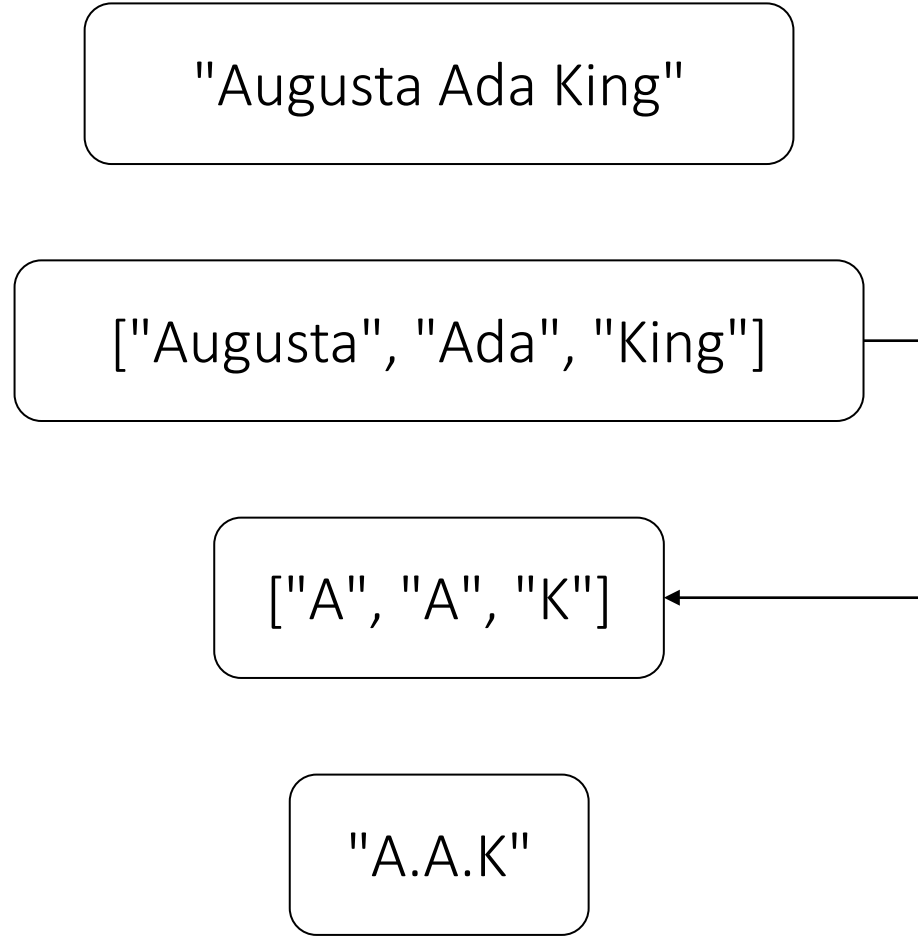
* <https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3>

Problem: Abbreviate*



* <https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3>

Problem: Abbreviate*



1. Split into words

2. Get the first letter of each

* <https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3>

Problem: Abbreviate*

"Augusta Ada King"

["Augusta", "Ada", "King"]

["A", "A", "K"]

"A.A.K"

1. Split into words

2. Get the first letter of each

3. Put dots in between

* <https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3>

Problem: Abbreviate*

"Augusta Ada King"

["Augusta", "Ada", "King"]

["A", "A", "K"]

"A.A.K"

1. Split into words
2. Get the first letter of each
3. Put dots in between

* <https://www.codewars.com/kata/57eadb7ecd143f4c9c0000a3>

User Study

- Within-subject study of 13 participants
- 4 programming tasks in 2 pairs
- Two groups:
 - SnipPy
 - Projection Boxes + Web Browser

SnipPy vs. The Internet

- More limited
- + Faster
- + Lower cognitive burden
- + More compact solutions

Loopy

[Ferdowsifard et al, OOPSLA'21]

Snippy for loop bodies

```
1 def compress(s):
2     rs = ''
3     count = 1
4     last = s[0]
5     for c in s[1:]:
6         last, count, rs = ??
7     return rs
8
9 compress('aabccca')
```

#	c	s	rs _{in}	count _{in}	last _{in}	last _{out}	count _{out}	rs _{out}
0	'a'	'aabccca'	''	1	'a'	'a'	2	''
1	'b'	'aabccca'	''	2	'a'	'b'	1	'2a'
2	'c'	'aabccca'	'2a'	1	'b'	'c'	1	'2a1b'
3	'c'	'aabccca'	'2a1b'	1	'c'	'c'	2	'2a1b'
4	'c'	'aabccca'	'2a1b'	2	'c'	'c'	3	'2a1b'
5	'a'	'aabccca'	'2a1b'	3	'c'	'c'	3	'2a1b'

Loopy

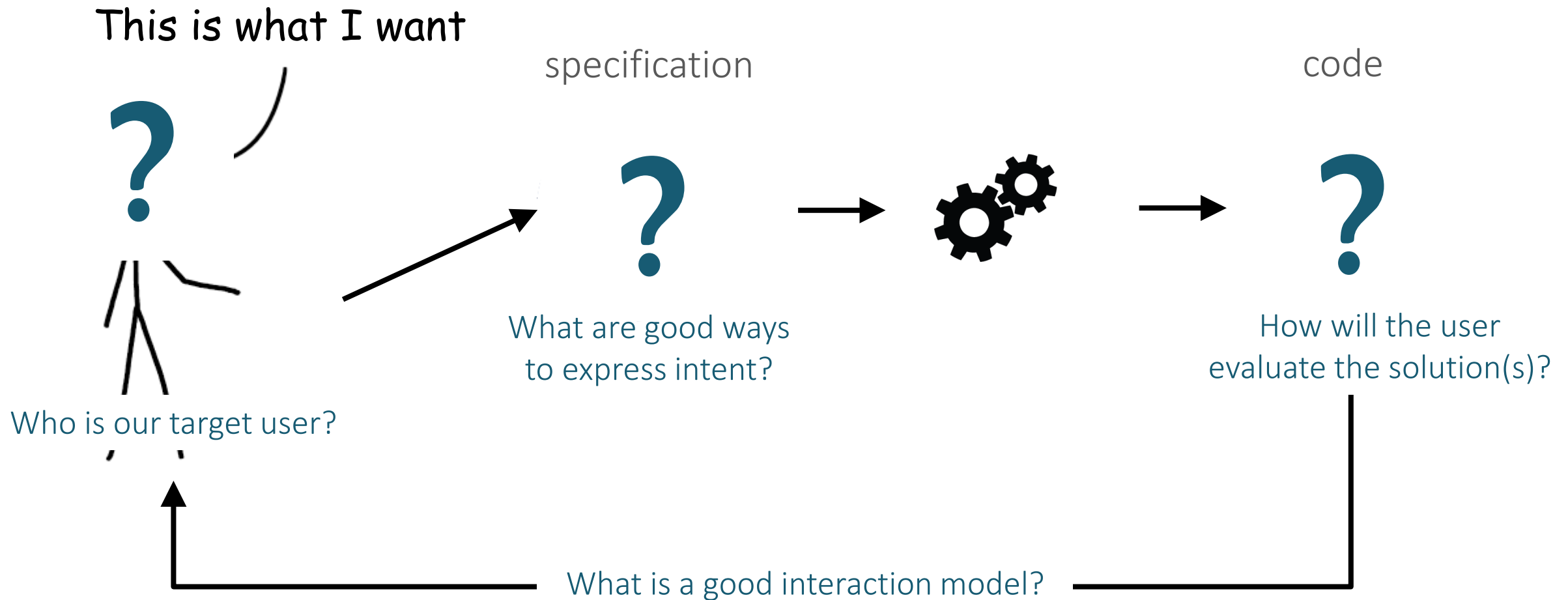
[Ferdowsifard et al, OOPSLA'21]

```
1 def compress(s):
2     rs = ''
3     count = 1
4     last = s[0]
5     for c in s[1:]:
6         last, count, rs = ??
7     return rs
8
9 compress('aabccca')
```



```
1 def compress(s):
2     rs = ''
3     count = 1
4     last = s[0]
5     for c in s[1:]:
6         if c == last:
7             count += 1
8         else:
9             rs = rs + str(count) + last
10            count = 1
11            last = c
12    return rs
```

The big picture



API discovery for Haskell

Inspiration: Hoogle

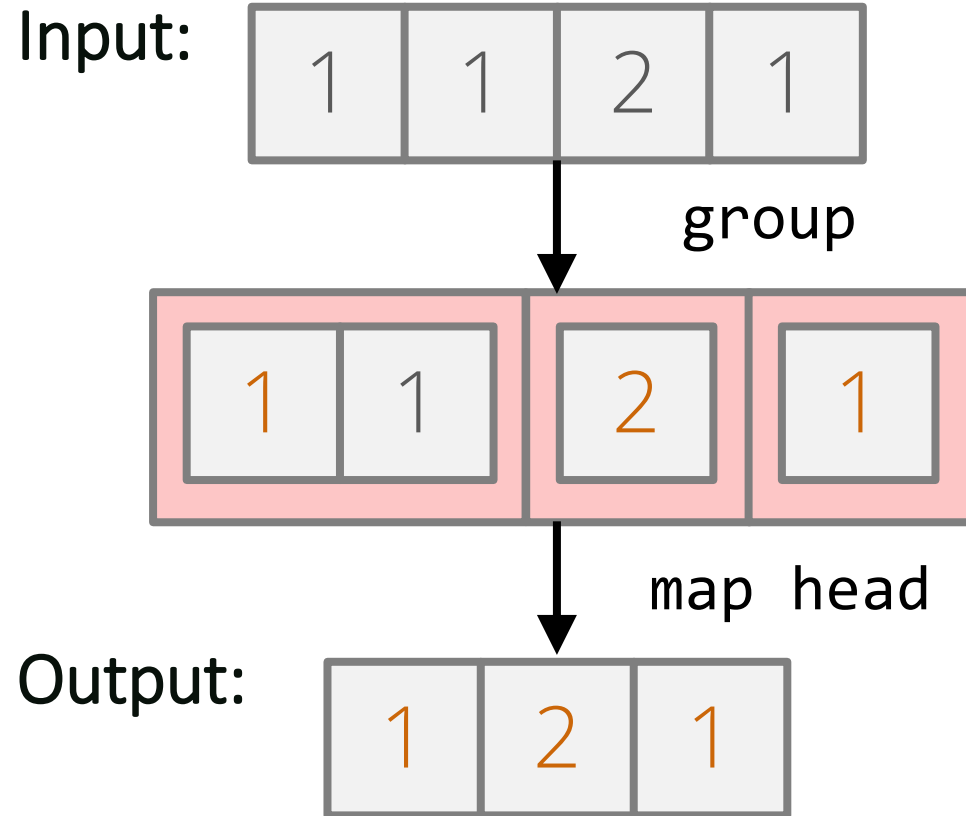
Hoogle

```
split :: Char -> String -> [String]
```

```
ghc Util
```

cannot compose functions!

Example: compress a list



Compress: specification

compress :: [a] → [a]

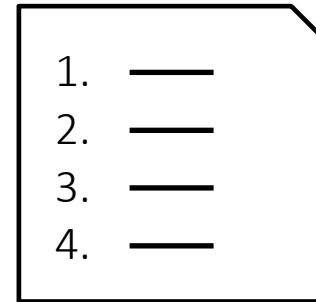
Hoogle+

specification

$\text{Eq } a \Rightarrow [a] \rightarrow [a]$



programs



Haskell
libraries

Challenges

- Types are ambiguous
 - Which solution is the right one?
 - Helping beginners with types

Too many irrelevant results

Hoogλe+

`\xs -> []`



ignores the argument!

`\xs -> xs`



ignores the type class!

`\xs -> head []`



always crashes!

`\xs -> tail xs`



ignores the type class!

Too many irrelevant results

Hoogλe+

```
\xs -> head (group xs)
```

```
\xs -> init (head (group xs))
```

```
\xs -> tail (head (group xs))
```



duplicate!

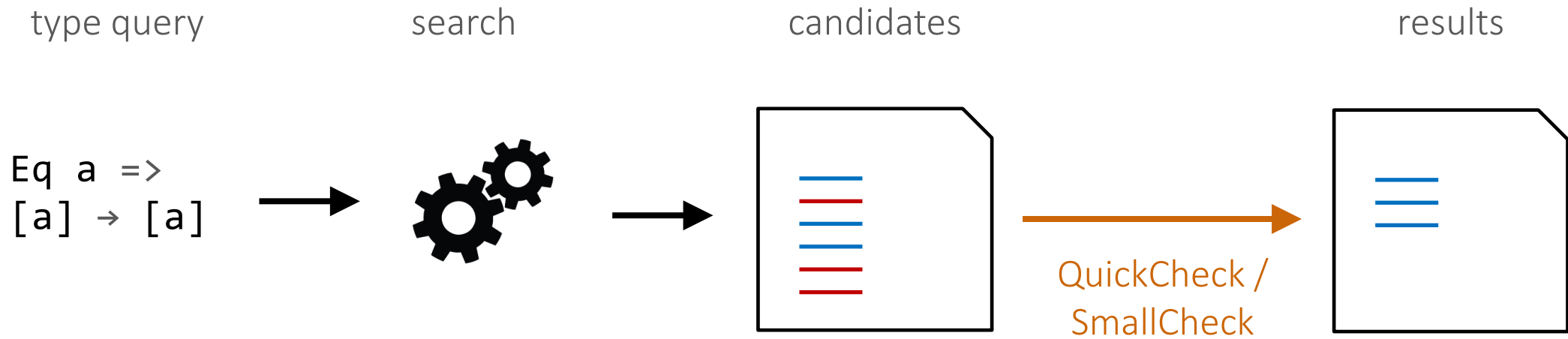
Challenges

- Types are ambiguous
 - sometimes can fix by asking the user for IO examples
 - but not for all queries, e.g. **Int** -> **ByteString**
 - should be able to filter out irrelevant results without user's help

Which solution is the right one?

Helping beginners with types

Test-based filtering



1. does it crash on all inputs?
2. is the output always the same as another candidate?
3. does the output stay the same when changing an input?

Challenges

Types are ambiguous

- Which solution is the right one?
- Helping beginners with types

Comprehension

Hoogλe+

```
\xs -> concat (group xs)
```

```
\xs -> head (group xs)
```

```
\xs -> last (group xs)
```

```
\xs -> map head (group xs)
```

how do I know what
these programs do?

Test-based comprehension

Hoogλe+

`Eq a => [a] → [a]`

Search

`\xs -> concat (group xs)`

`[0,1] -> [0,1]`

`[0,0] -> [0,0]`

`\xs -> head (group xs)`

`[0,1] -> [0]`

`[0,0] -> [0,0]`

`\xs -> last (group xs)`

`[0,1] -> [1]`

`[0,0] -> [0,0]`

`\xs -> map head (group xs)`



`[0,1] -> [0,1]`

`[0,0] -> [0]`

Challenges

Types are ambiguous

Which solution is the right one?

→ Helping beginners with types

Hoogle+

specification

Eq
**beginner
unfriendly!**

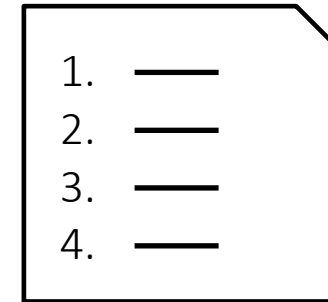
[a]



H+



programs



Can we infer type from tests?

specification

Eq a => [a] → [a] →

H+



programs

1. _____

2. _____

3. _____

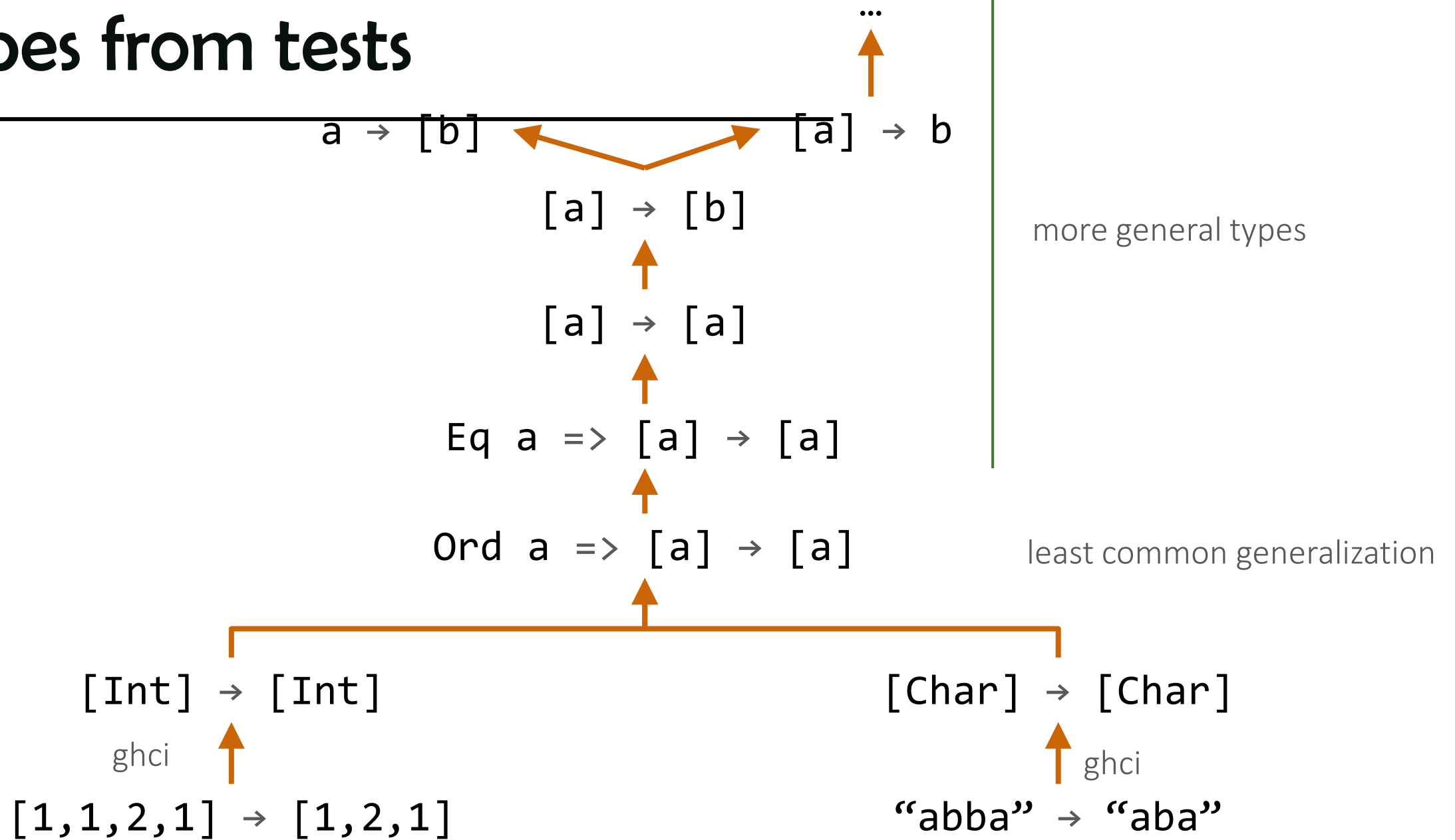
4. _____

[1,1,2,1] → [1,2,1]

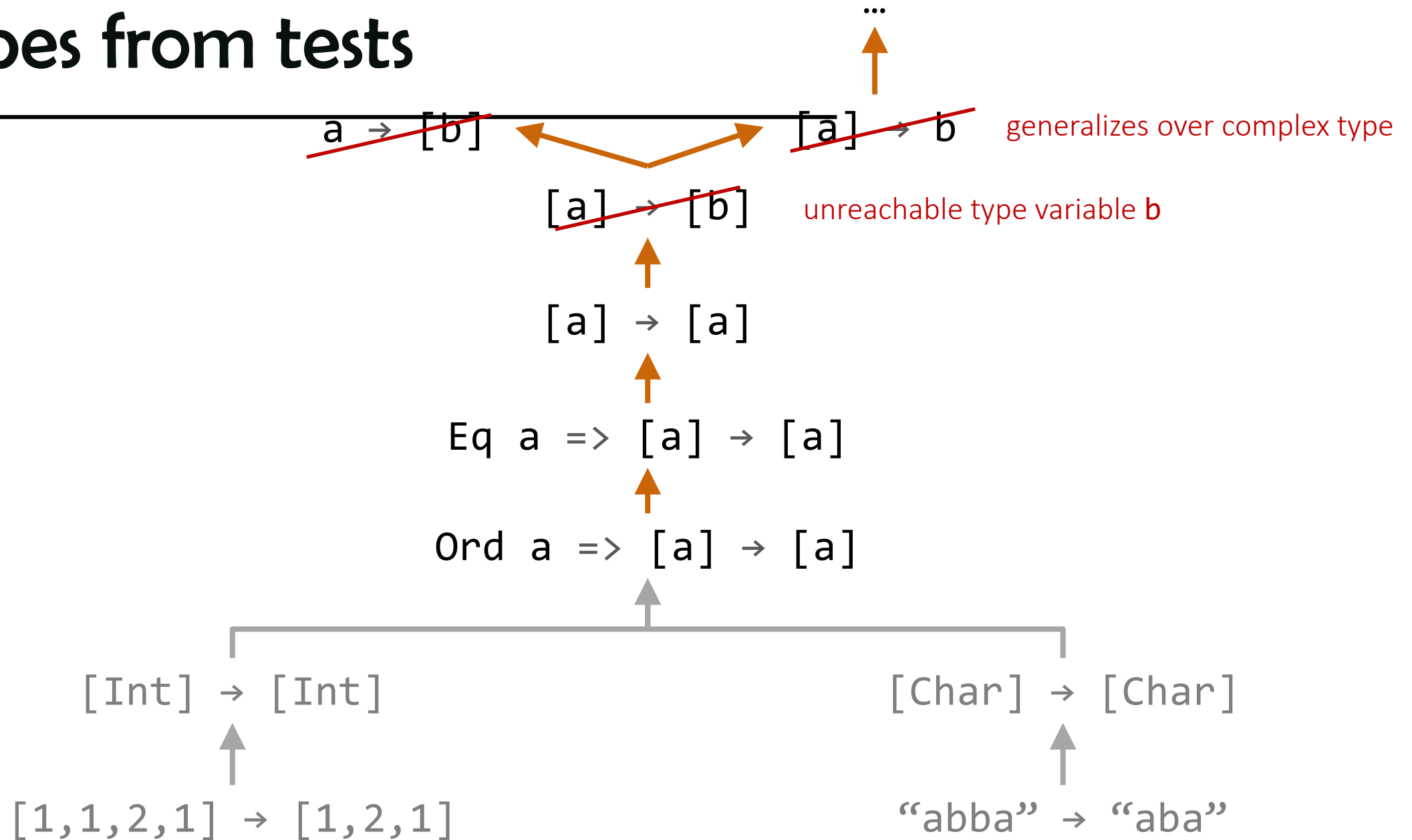
“abba” → “aba”



Types from tests



Types from tests



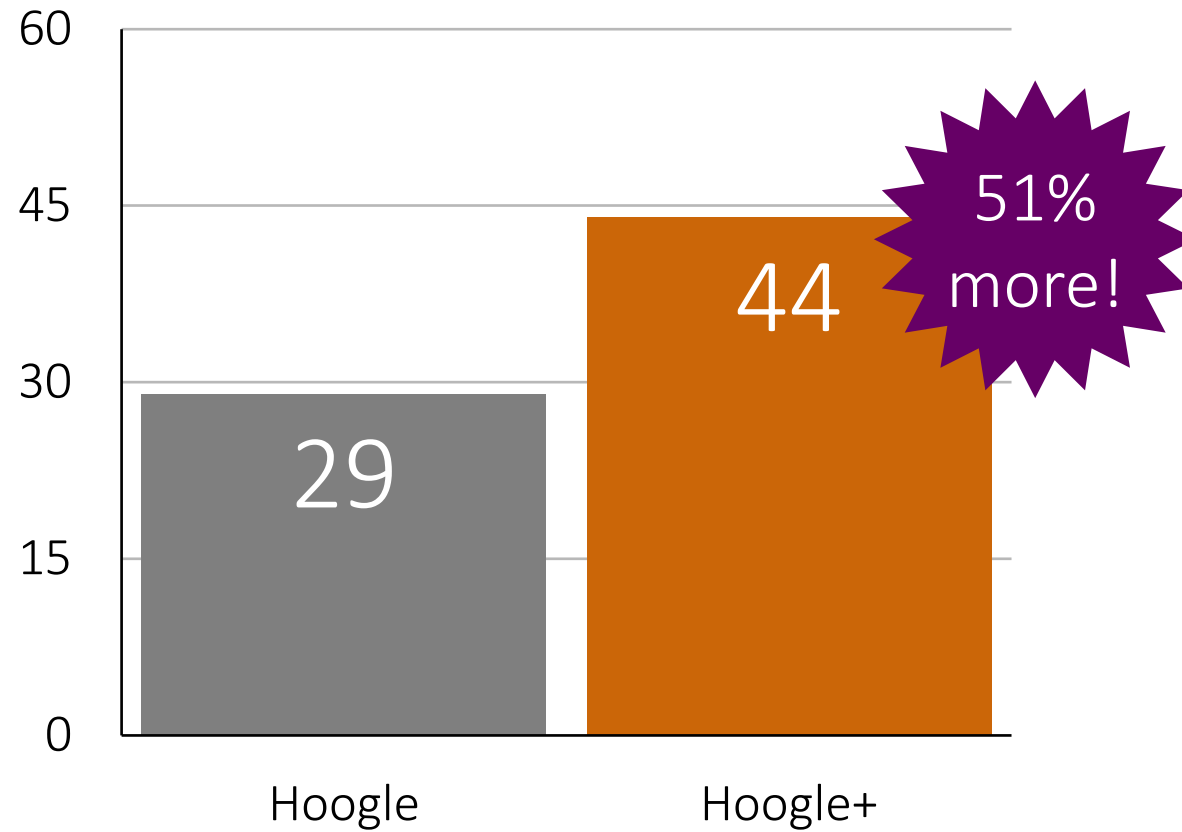
User study

30 participants

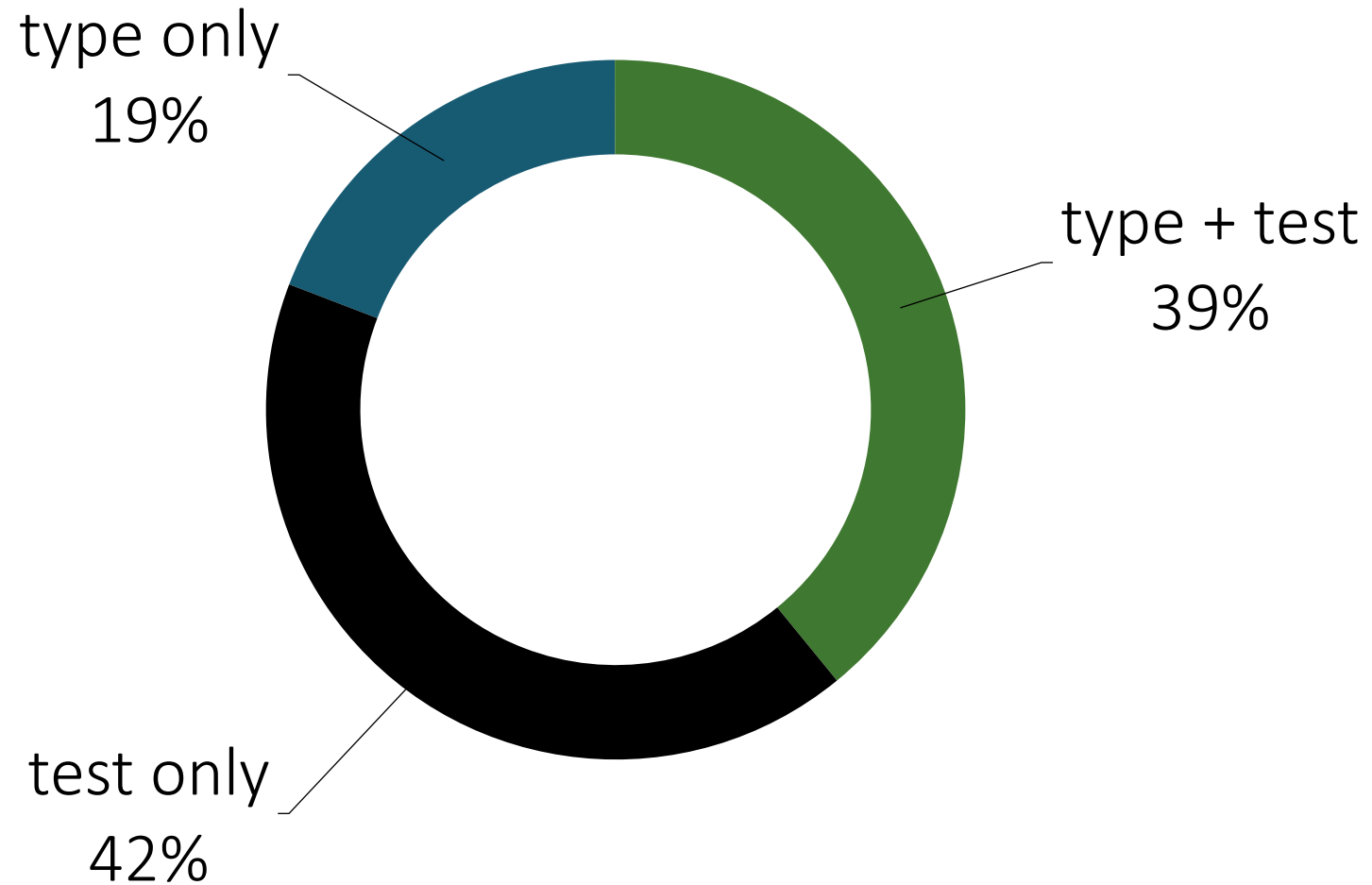
4 tasks

2 with Hoogle, then 2 with Hoolge+

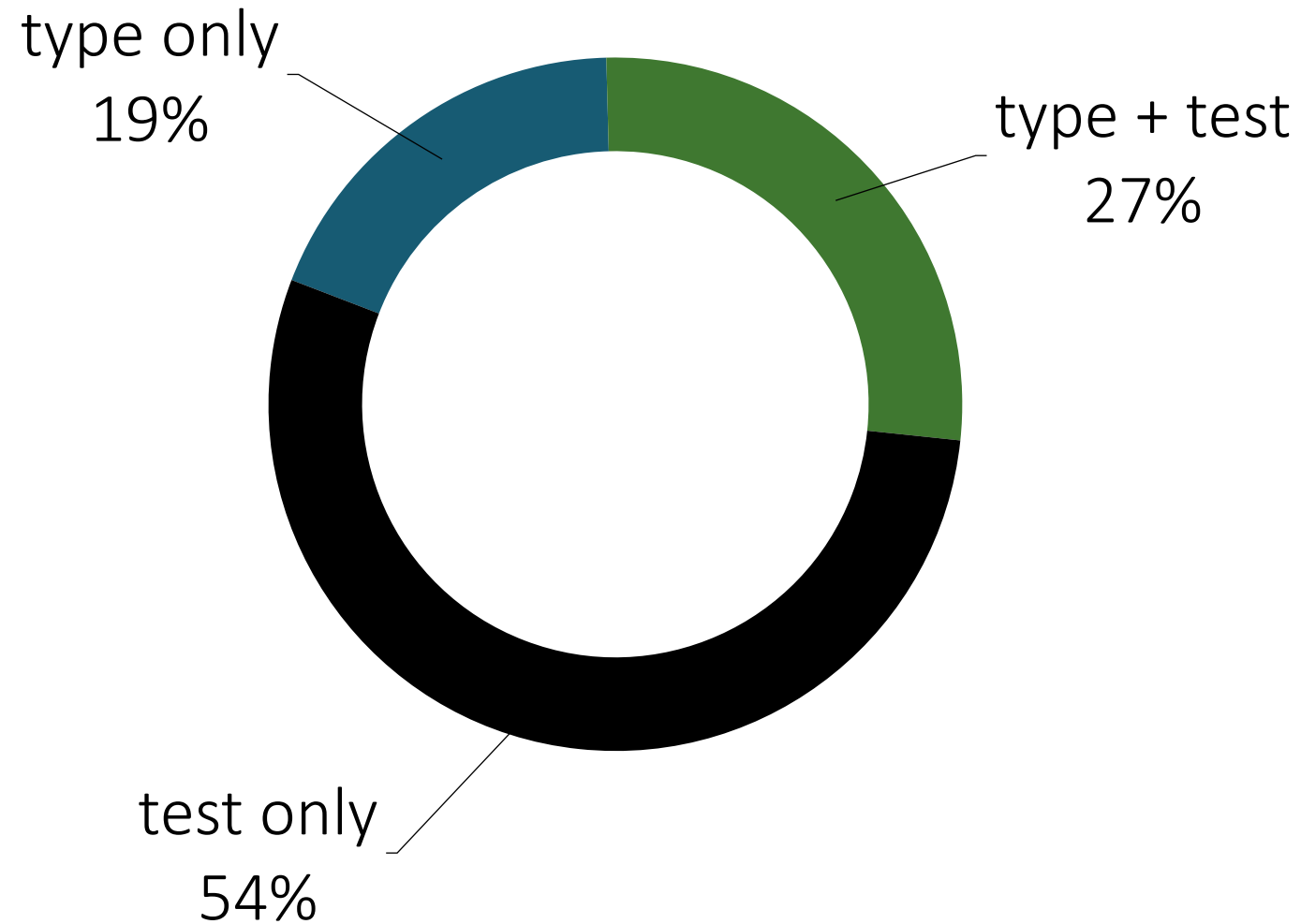
Results: completed tasks



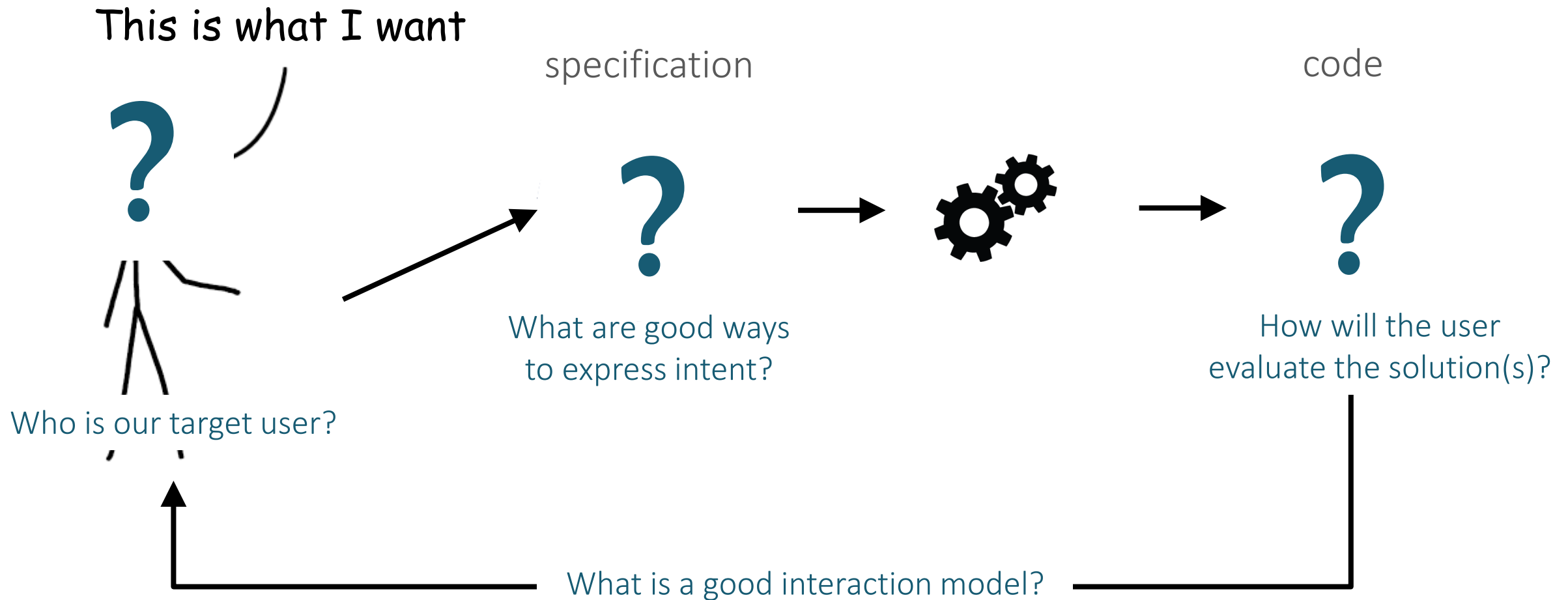
Modes of specification



Modes of specification: **beginners**

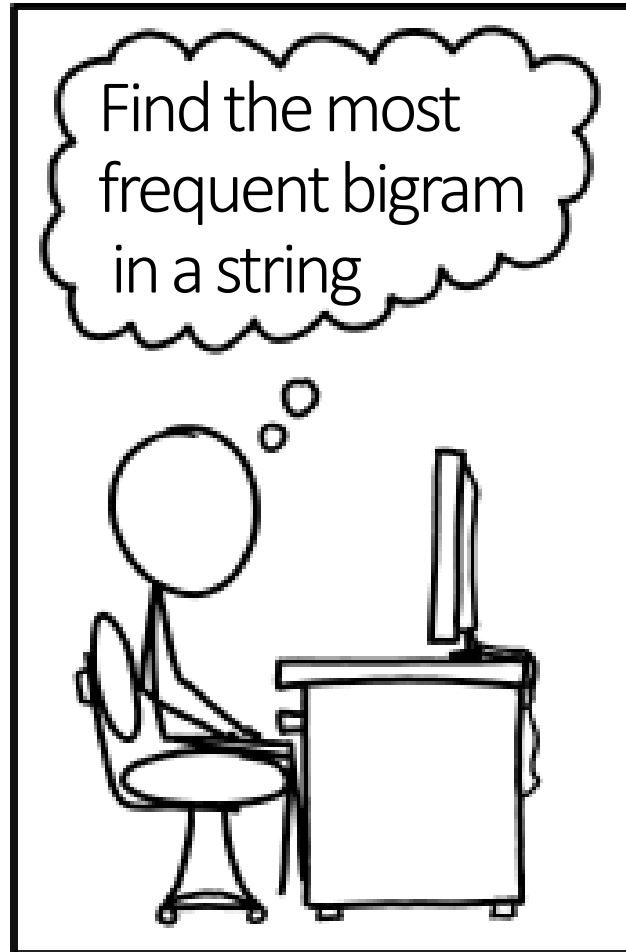


The big picture



→ Synthesis embedded into a REPL
= syntactic specs (aka granular interaction model)
+ sketching
+ debugger

Programming by Example



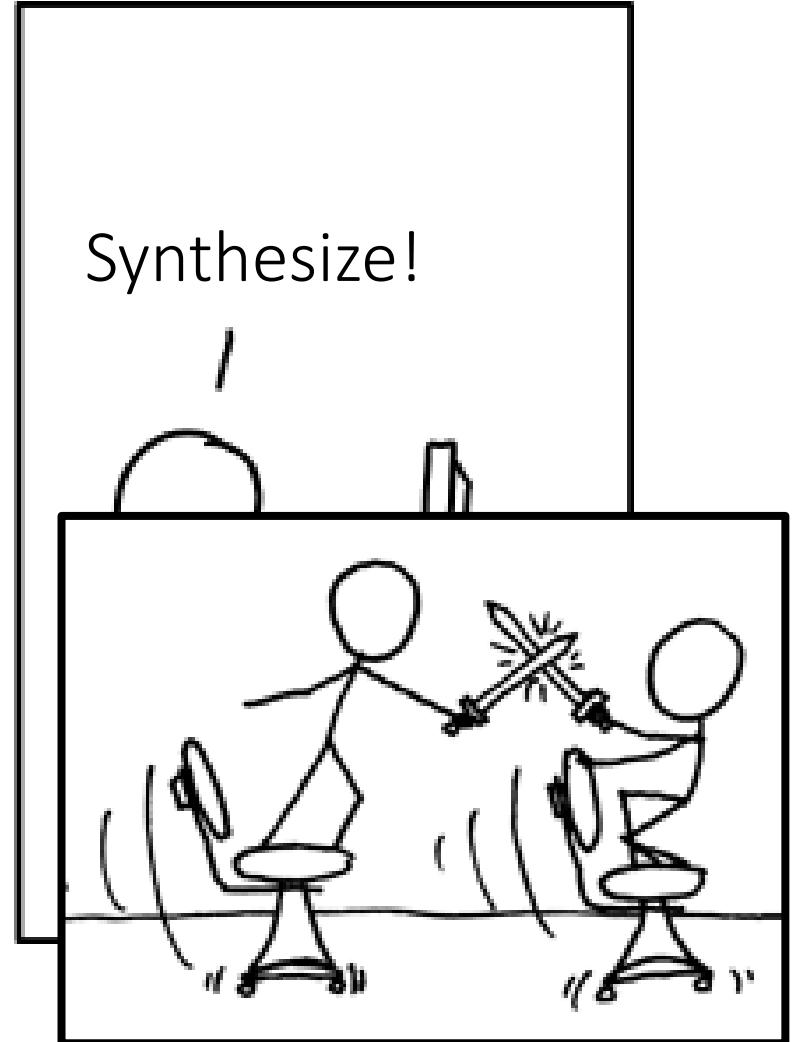
An example of the desired behavior:

Input:

"a**bd**fibfcfde**bd**fd
e**bd**ihgfkjfd**bd**"

Output: "bd"

Synthesize!



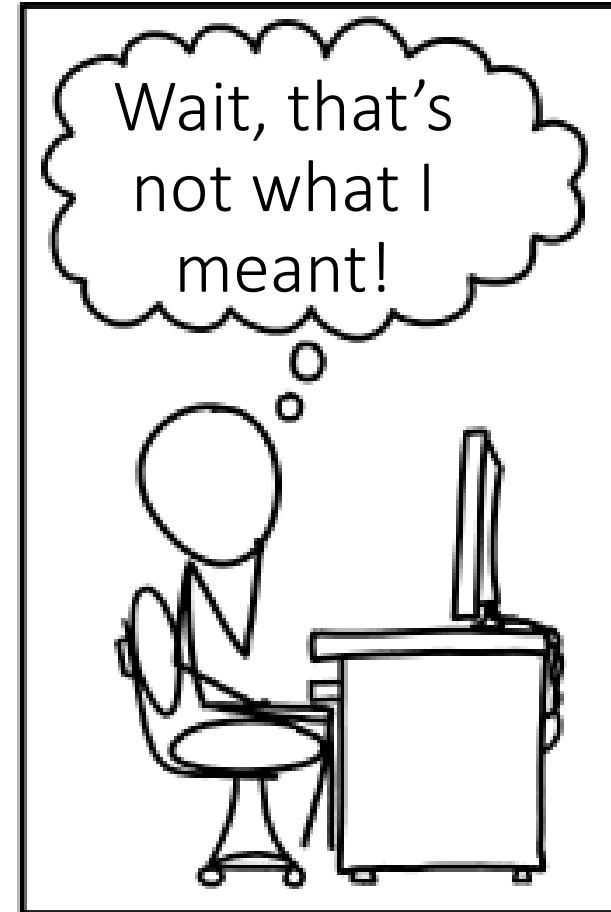
Examples are ambiguous

Input:

"a**bd**fibfcfde**bd**fd
e**bd**ihgfkjfd**bd**"

Output: "bd"

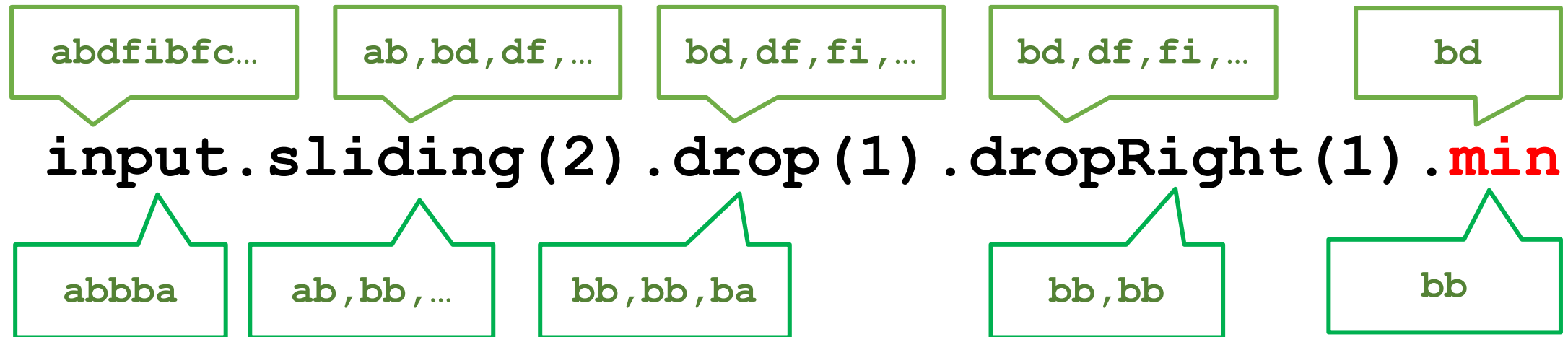
```
input  
.takeRight(2)
```



Problem with examples



Input: "abbba", Output: "bb"



Granular Interaction Model (GIM)

[Peleg et al, ICSE'18]

Idea: Programmers understand code

- they can give syntactic feedback about the candidate solution

Granular Interaction Model (GIM)

```
input
```

```
//ab, bd, df, ...
```

```
.sliding(2)
```

```
//bd, df, fi, ...
```

```
.drop(1)
```

```
//bd
```

```
.min
```

That looks right

Those are wrong

Granular feedback

Exclude

`exclude(f.g.h):`
never show programs
of the form
`input...f.g.h...`

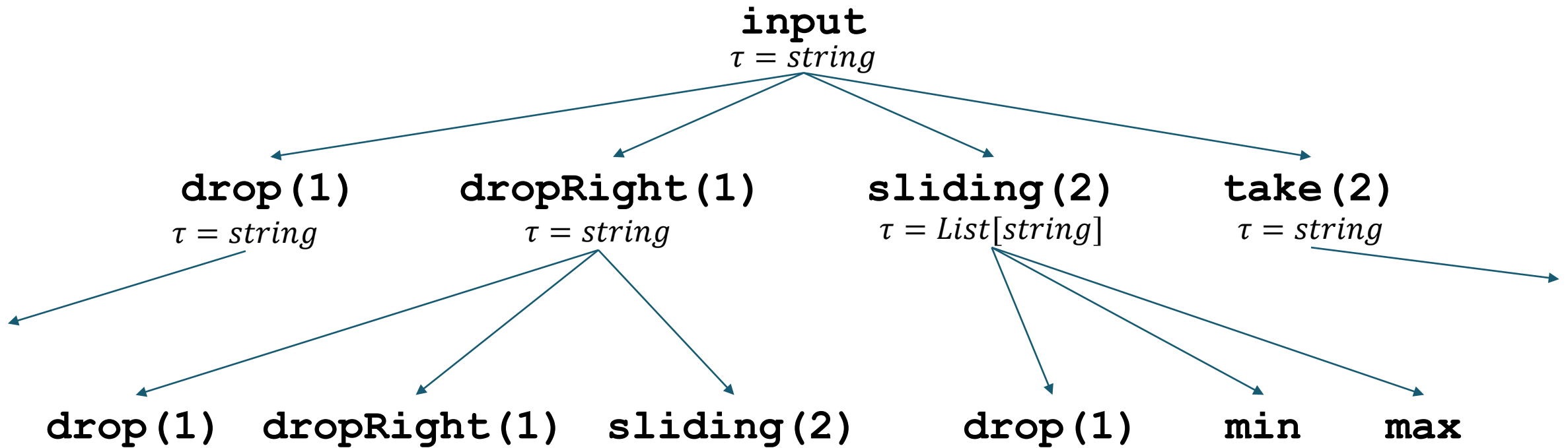
Retain

`retain(f.g.h):`
only show programs of
the form
`input...f.g.h...`

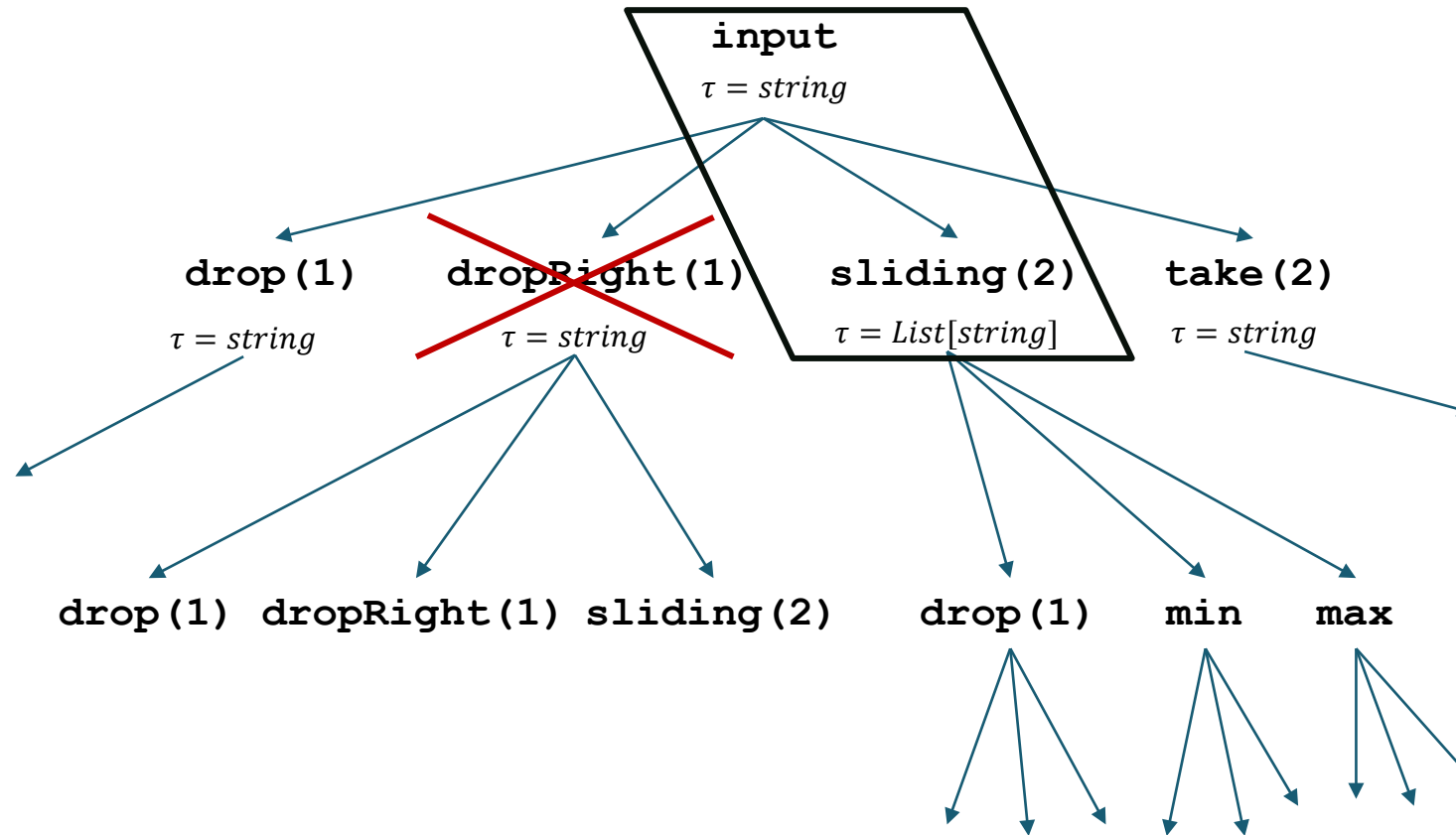
Affix

`affix(f.g.h):`
only show programs of
the form
`input.f.g.h...`

Program space



Program space pruning



Synthesis embedded into a REPL
= syntactic specs (aka granular interaction model)
+ sketching
+ debugger

Running example

All prefixes of printed number

420



['4', '42', '420']

RESL

demo

User study

19 participants

- industry programmers with no JS experience

4 tasks

Control: REPL + docs

Treatment: RESL

User study

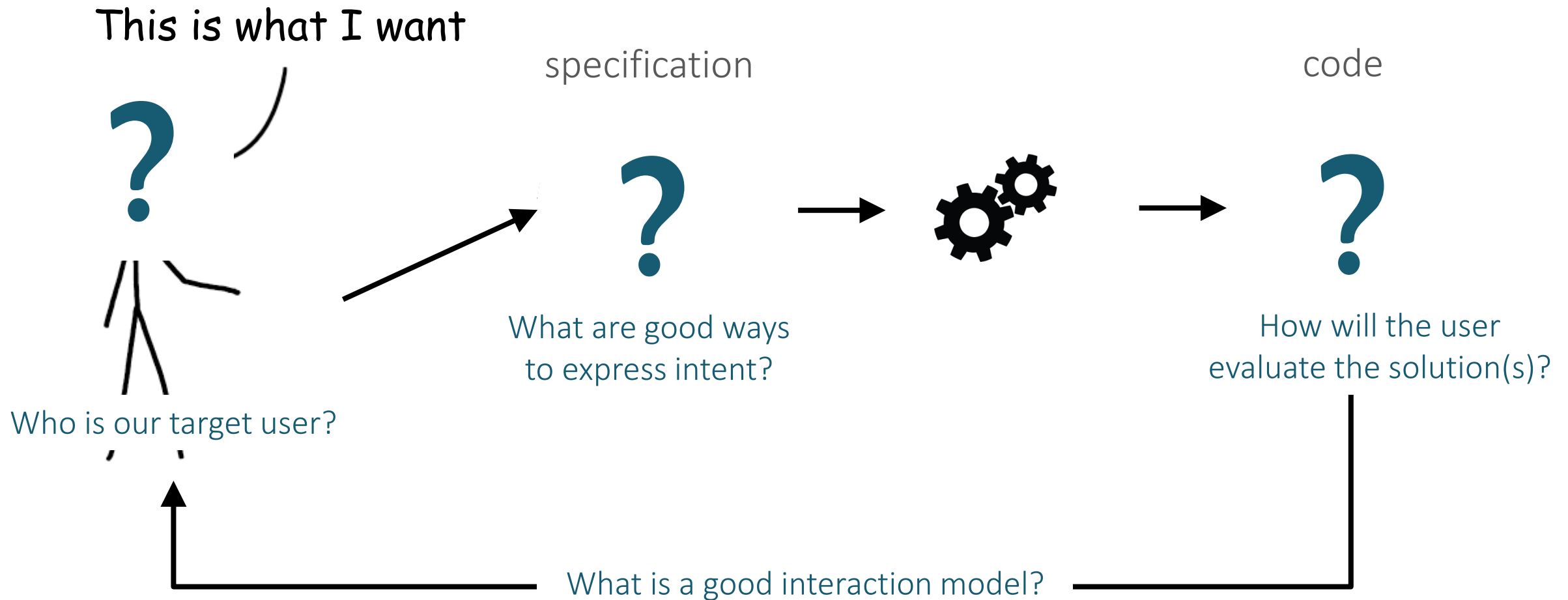
RQ1: Does RESL reduce editing load?

- In 3 out of 4 tasks big reduction in edit iterations
- In 2 out of 4 tasks, over 50% of the code was synthesized

RQ2: Does RESL reduce programmer frustration?

- No RESL user abandoned the task while some REPL users did

The big picture



This week

Tuesday: synthesis for programmers

- Snippy [Ferdowsifard et al, UIST'20; OOPLSA'21]
- Hoogle+ [James et al, OOPSLA'20]
- RESL [Peleg et al, OOPLSA'20; Peleg et al, ICSE'18]

Thursday: synthesis for non-programmers

- Rousillon [Chasins, Meuller, Bodik, UIST'18]
- Wrex [Drosos et al, CHI'20]
- Regae [Zhang et al, UIST'20]

Rousillon / Helena

[Chasins, Meuller, Bodik, UIST'18]

Web scraping for social scientists

The web: a rich source of data!

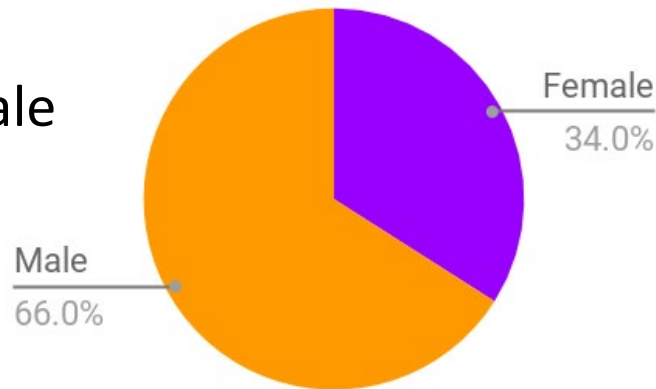
2008: Google indexed **1 trillion** pages

Now: indexes > **60 trillion** pages

→ lots of content out there

Have you written a scraper?

Percentages of Female and Male
Speaking Characters - Top 100
Films of 2017



Woman director or writer: **42%** female speaking roles
Only male directors, writers: **32%** female speaking roles

Martha M. Lauzen. 2018. It's a Man's (Celluloid) World: Portrayals of Female Characters in the 100 Top Films of 2017

Top-US-Grossing Feature Films Released 2017-01-01 to 2017-12-31			
1 to 50 of 11,605 titles Next » View Mode: Compact Detailed			
Sort by: Popularity Alphabetical IMDb Rating Number of Votes			
US Box Office ▼ Runtime Year Release Date			
	1. Star Wars: The Last Jedi (2017)	7.2	☆ Rate +
	2. Beauty and the Beast (2017)	7.2	☆ Rate +
	3. Wonder Woman (2017)	7.5	☆ Rate +
	4. Jumanji: Welcome to the Jungle (2017)	7	☆ Rate +
	5. Guardians of the Galaxy Vol. 2 (2017)	7.7	☆ Rate +
	6. Spider-Man: Homecoming (2017)	7.5	☆ Rate +
	7. It (I) (2017)	7.4	☆ Rate +
	8. Thor: Ragnarok (2017)	7.9	☆ Rate +
	9. Despicable Me 3 (2017)	6.3	☆ Rate +
	10. Justice League (2017)	6.6	☆ Rate +
	11. Logan (2017)	8.1	☆ Rate +
	12. The Fate of the Furious (2017)	6.7	☆ Rate +
	13. Coco (I) (2017)	8.4	☆ Rate +
	14. Dunkirk (2017)	8	☆ Rate +
	15. Solo: A Star Wars Story (2017)	6.7	☆ Rate +

Let's automate!

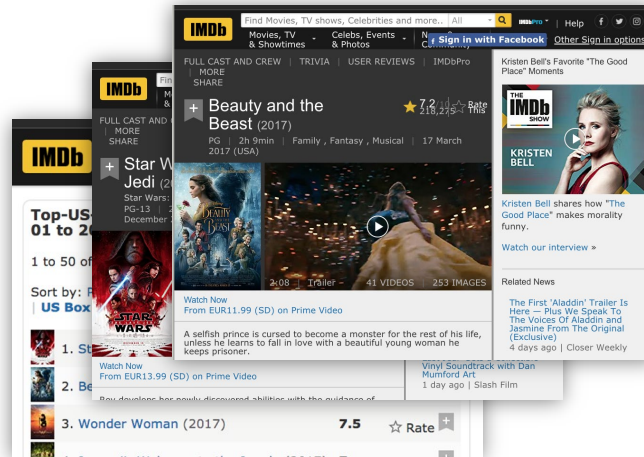


We've got some libraries...

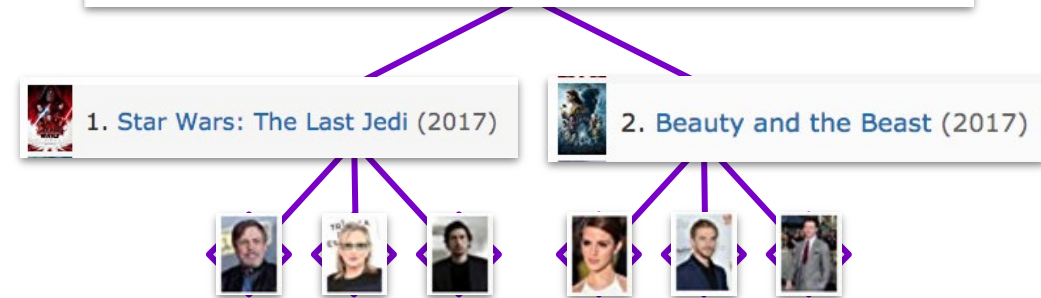
common thread: users
must reverse engineer
target webpages



Formative Study: What kinds of web data?



Top-US-Grossing Feature Films Released 2017-01-01 to 2017-12-31



distributed

must navigate between pages -
e.g., click, use forms + widgets

hierarchical

must traverse and collect tree-
structured data

Formative Study: Can social scientists use...

Traditional
programming?

Skills:

Basic programming

Web DSL

DOM

~~JavaScript~~

~~Server interaction~~



Manual collection?

Skills:

Browser use

But

Slow

Tedious

Small-scale data



Programming by
demonstration?

Skills:

Browser use

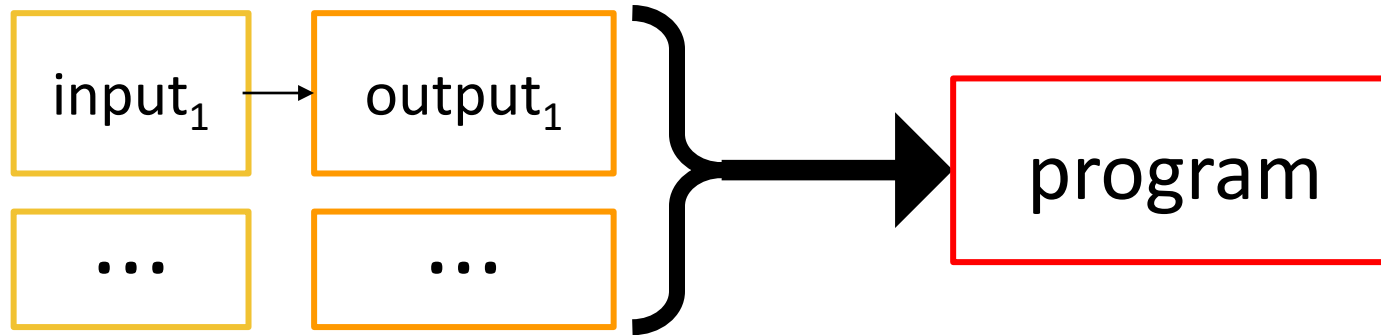
But

Can't collect
distributed,
hierarchical datasets

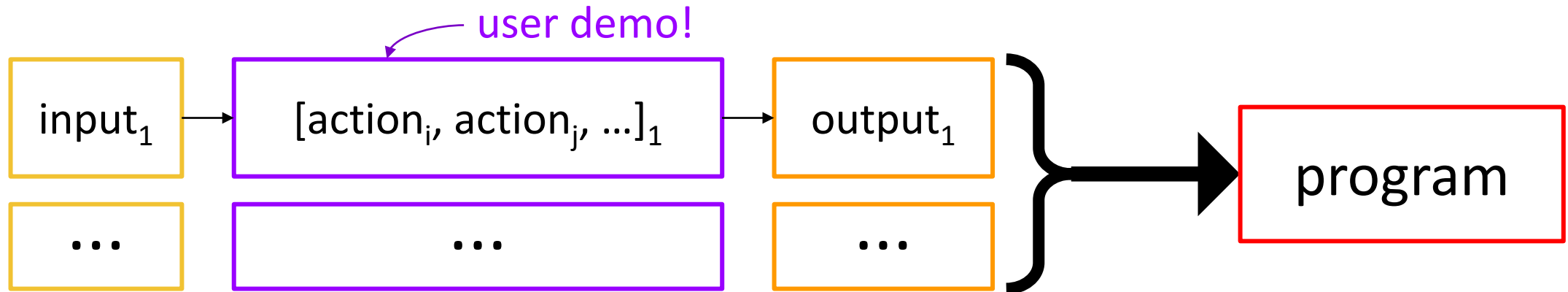


What's Programming by Demonstration (PBD)?

Closely related to Programming by Example (PBE) (e.g., FlashFill)



But PBD (e.g., SMARTedit) gets to see the input being transformed into the output:



Design goals

D1: Expertise – do not require knowledge of HTML, DOM trees, etc.

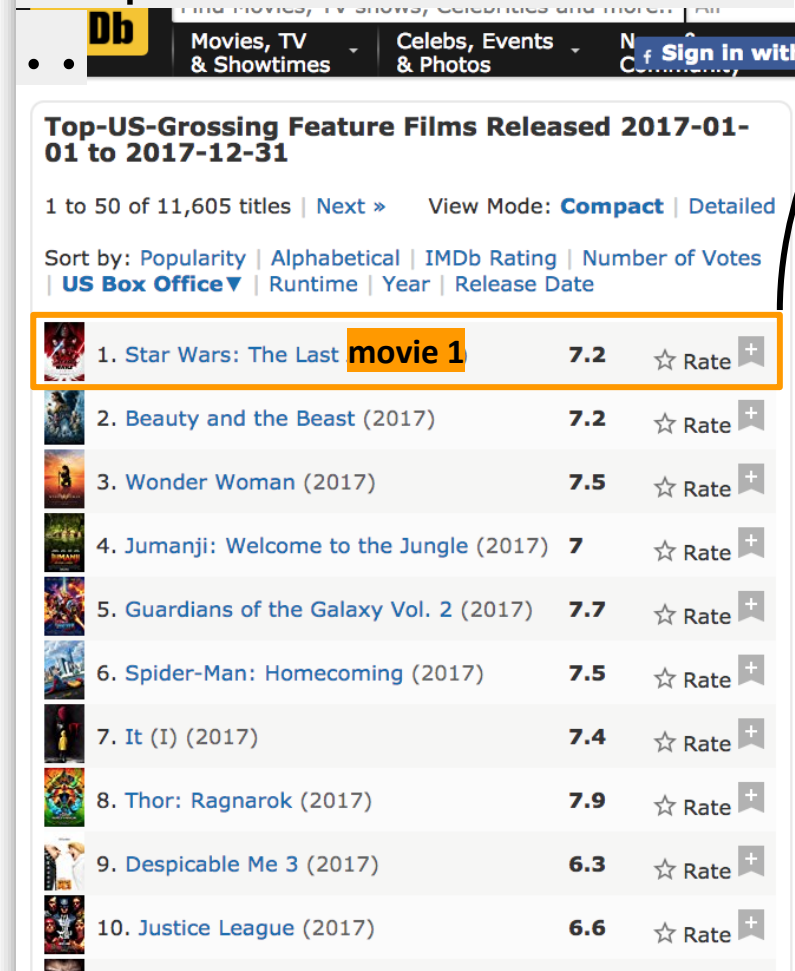
D2: Distributed hierarchical data – handle realistic datasets

D3: Learnability – prioritize learnability by novices over usability by tool experts

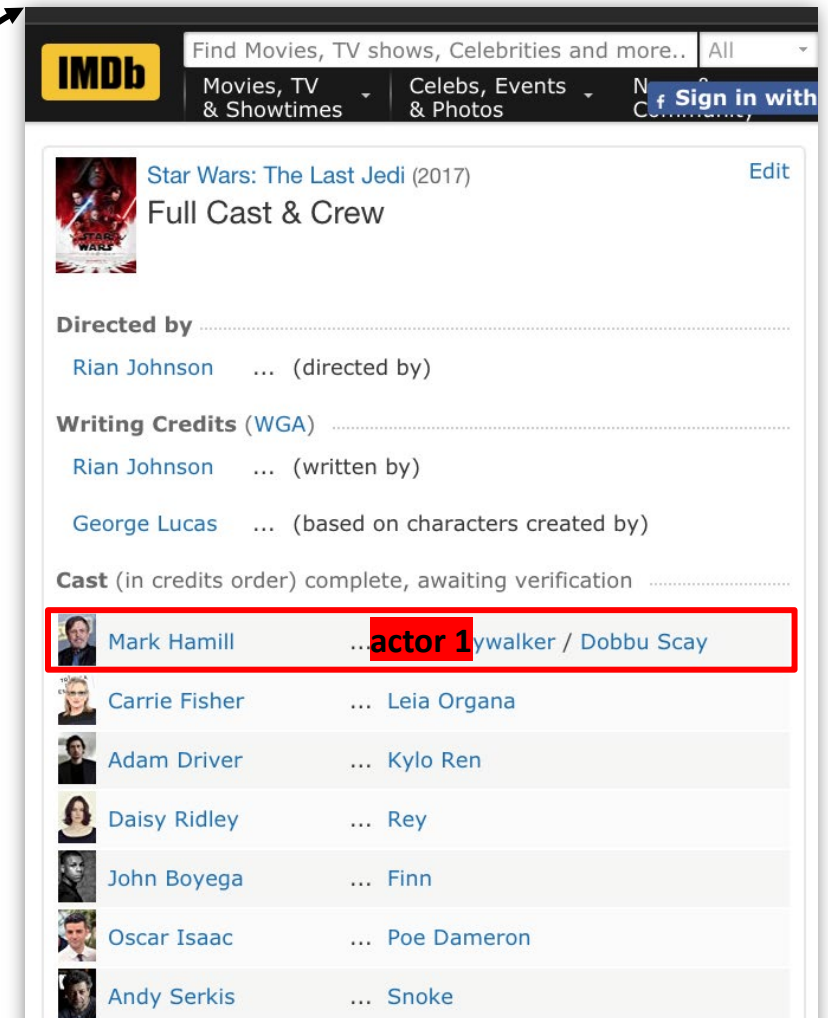
The Interaction Model

load

https://www.imdb.com/..



click



user

demonstrates
how to collect
one joined row

start recording

load www.imdb.com...

collect

movie 1

click

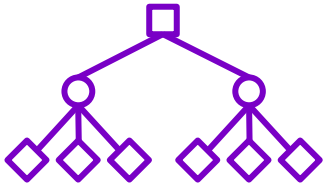
movie 1

collect

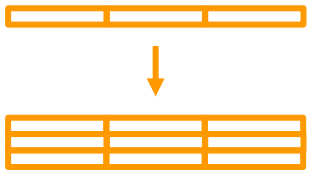
actor 1

end recording

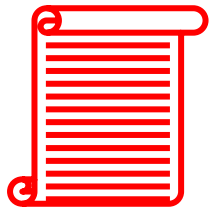
Technical Challenges



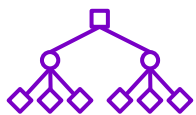
Hierarchical Data: Synthesis of nested loops - needed for hierarchical data - is a long-standing open problem.



Relation Ambiguity: Single row is an ambiguous demo.
Which relation did the user intend to select?

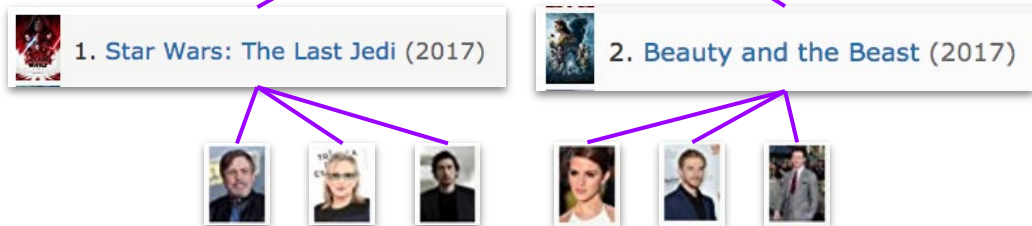


Readability: For robust automation, must run 100s of low-level, unreadable DOM events.



Problem 1: Hierarchical Data

Top-US-Grossing Feature Films Released 2017-01-01 to 2017-12-31



hierarchical data \rightarrow nested loops

The issue:

Nested loop synthesis is an open problem.

```
for movie in movie_list:
    // scrape movie data
    for actor in actor_list:
        // scrape actor data
```

Past solutions:

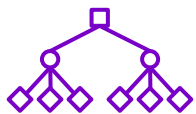
In web automation, none. In other domains, manually marking loop boundaries.

progs
w/ no
loops

progs w/
single-level
loops

progs w/
nested
loops

The space of possible programs is just too big. To pick among all these, our spec is ambiguous.



Problem 1: Hierarchical Data

Solution:

Design user interaction to make search tractable

Contract w/ user: perform one iteration of each loop, ordered from outer to inner

load <https://www.imdb.com/se...> into p1

scrape [Star Wars: The Last Jedi](#) in p1 and call it movie_title movie cell

scrape [\(2017\)](#) in p1 and call it movie_year movie cell

click [Star Wars: The Last Jedi](#) in p1 movie cell

scrape [Mark Hamill](#) in p2 and call it actor_name actor cell

scrape [Luke Skywalker](#) in p2 and call it actor_role actor cell

Label uses of relation cells

movie relation

	1. Star Wars: The Last Jedi (2017)	7.2	☆ Rate +
	2. Beauty and the Beast (2017)	7.2	☆ Rate +
	3. Wonder Woman (2017)	7.5	☆ Rate +
	4. Jumanji: Welcome to the Jungle (2017)	7	☆ Rate +

PBD takeaway:

To add loops efficiently, first find objects that should be treated together.

One loop per relation, start before cell use

load <https://www.imdb.com/se...> into p1

for movie in movie_list:

scrape [Star Wars: The Last Jedi](#) in p1 and call it movie_title movie cell

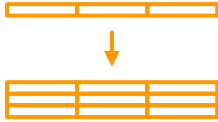
scrape [\(2017\)](#) in p1 and call it movie_year movie cell

click [Star Wars: The Last Jedi](#) in p1 movie cell

for actor in actor_list:

scrape [Mark Hamill](#) in p2 and call it actor_name actor cell

scrape [Luke Skywalker](#) in p2 and call it actor_role actor cell



Problem 2: Relation Ambiguity

IMDb Find Movies, TV shows, Celebrities and more.. All

scrape

Top-US-Grossing Feature Films Released 2017-01-01 to 2017-12-31

1 to 50 of 11,607 titles | Next » View Mode: Compact | Detailed

Sort by: Popularity | Alphabetical | IMDb Rating | Number of Votes | US Box Office | Runtime | Year | Release Date

1. **Star Wars: The Force Awakens** (2017) scrape

PG-13 | 152 min | Action, Adventure, Fantasy

★ 7.2 Rate this 85 Metascore

Rey develops her newly discovered abilities with the guidance of Luke Skywalker, who is unsettled by the strength of her powers. Meanwhile, the Resistance prepares for battle with the First Order.

Director: Rian Johnson | Stars: Daisy Ridley, John Boyega, Mark Hamill, Carrie Fisher

Votes: 419,970 | Gross: \$620.18M

2. **Beauty and the Beast** (2017)

PG | 129 min | Family, Fantasy, Musical

★ 7.2 Rate this 65 Metascore

A selfish prince is cursed to become a monster for the rest of his life, unless he learns to fall in love with a beautiful young woman he keeps prisoner.

Director: Bill Condon | Stars: Emma Watson, Dan Stevens, Luke Evans, Josh Gad

Votes: 218,261 | Gross: \$504.01M

3. **Wonder Woman** (2017)

PG-13 | 141 min | Action, Adventure, Fantasy

★ 7.5 Rate this 76 Metascore

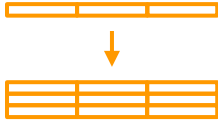
Given this demo, what's the right relation? Is node 1 included? If not, do we want purple or orange cells in rows 2 and 3? Maybe purple + orange + unhighlighted?

The issue:

Can extract many relations from one page. Set of interacted nodes → 1 chosen relation?

Past solutions:

Have user label multiple rows.



Problem 2: Relation Ambiguity

Top-US-Grossing Feature Films Released 2017-01-01 to 2017-12-31

1 to 50 of 11,607 titles | Next » View Mode: Compact | Detailed

Sort by: Popularity | Alphabetical | IMDb Rating | Number of Votes | **US Box Office** | Runtime | Year | Release Date

- Star Wars: The Last Jedi** (2017)
 PG-13 | 152 min | Action, Adventure, Fantasy
 ★ 7.2 Rate this 85 Metascore
 Rey develops her newly discovered abilities with the guidance of Luke Skywalker, who is unsettled by the strength of her powers. Meanwhile, the Resistance prepares for battle with the First Order.
 Director: Rian Johnson | Stars: Daisy Ridley, John Boyega, Mark Hamill, Carrie Fisher
 Votes: 419,970 | Gross: \$620.18M
- Beauty and the Beast** (2017)
 PG | 129 min | Family, Fantasy, Musical
 ★ 7.2 Rate this 65 Metascore
 A selfish prince is cursed to become a monster for the rest of his life, unless he learns to fall in love with a beautiful young woman he keeps prisoner.
 Director: Bill Condon | Stars: Emma Watson, Dan Stevens, Luke Evans, Josh Gad
 Votes: 218,261 | Gross: \$504.01M
- Wonder Woman** (2017)
 PG-13 | 141 min | Action, Adventure, Fantasy
 ★ 7.5 Rate this 76 Metascore

Solution:

Take advantage of domain-specific patterns (e.g, web design best practices) to find objects we should treat together

$\text{siblingWithShape}([n1, n2], s) \rightarrow \emptyset$

$\text{siblingWithShape}([n2], s) \rightarrow n3$

$\text{relation} \rightarrow [n2, n3, n4]$

User Study:

PBD vs. traditional programming

Setup:

Within-subject study, 15 CS PhD students

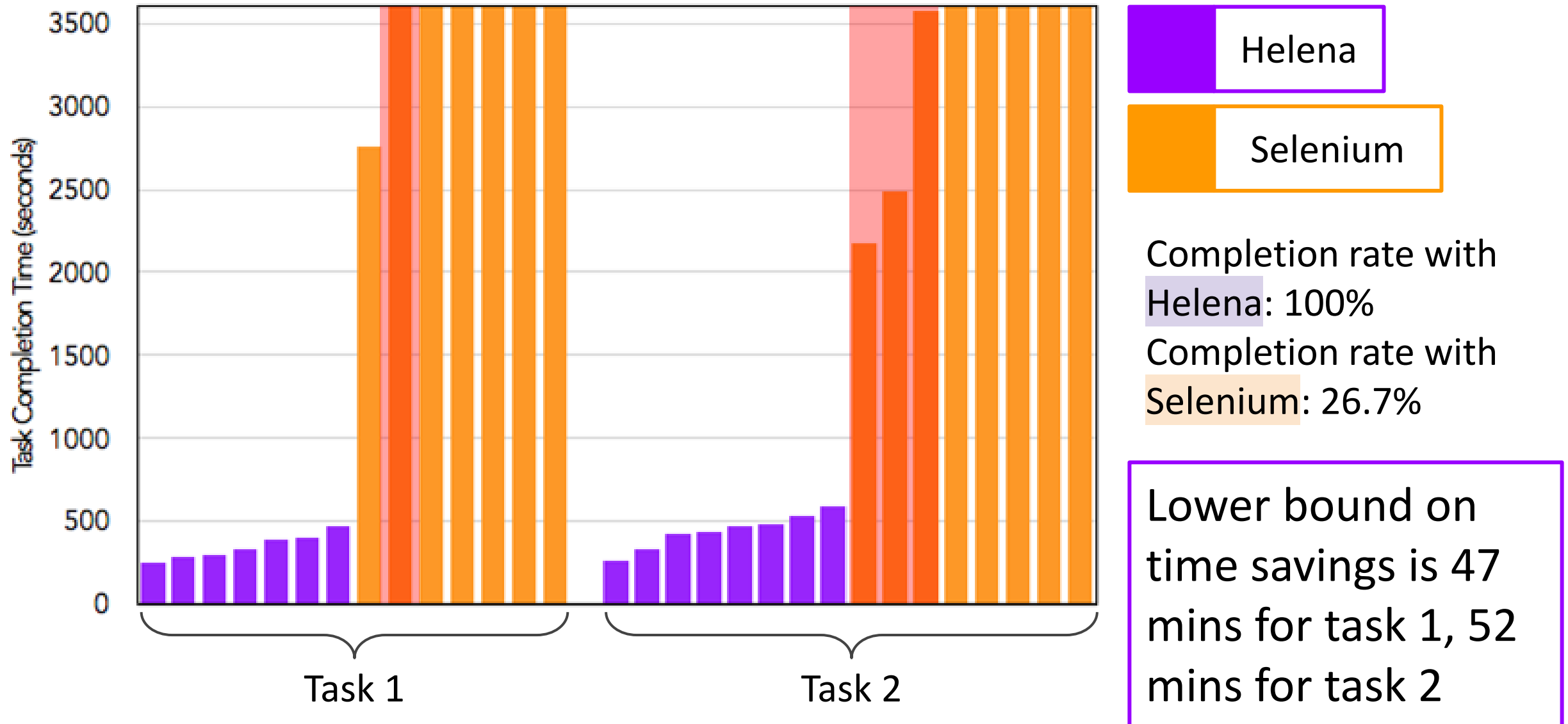
2 tasks: Authors-Papers and Foundations-Tweets

2 tools: Helena then Selenium OR Selenium then Helena

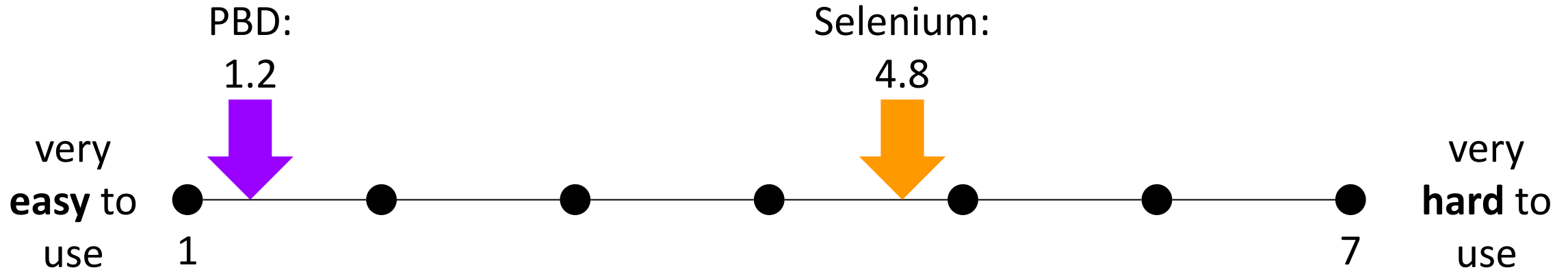
9/15 prior scraping experience

4/15 prior Selenium experience

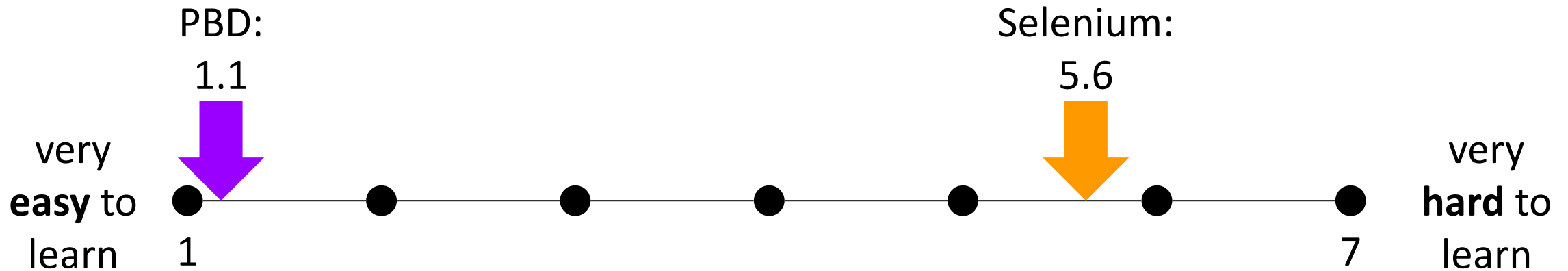
Q1: Can users learn PBD faster?



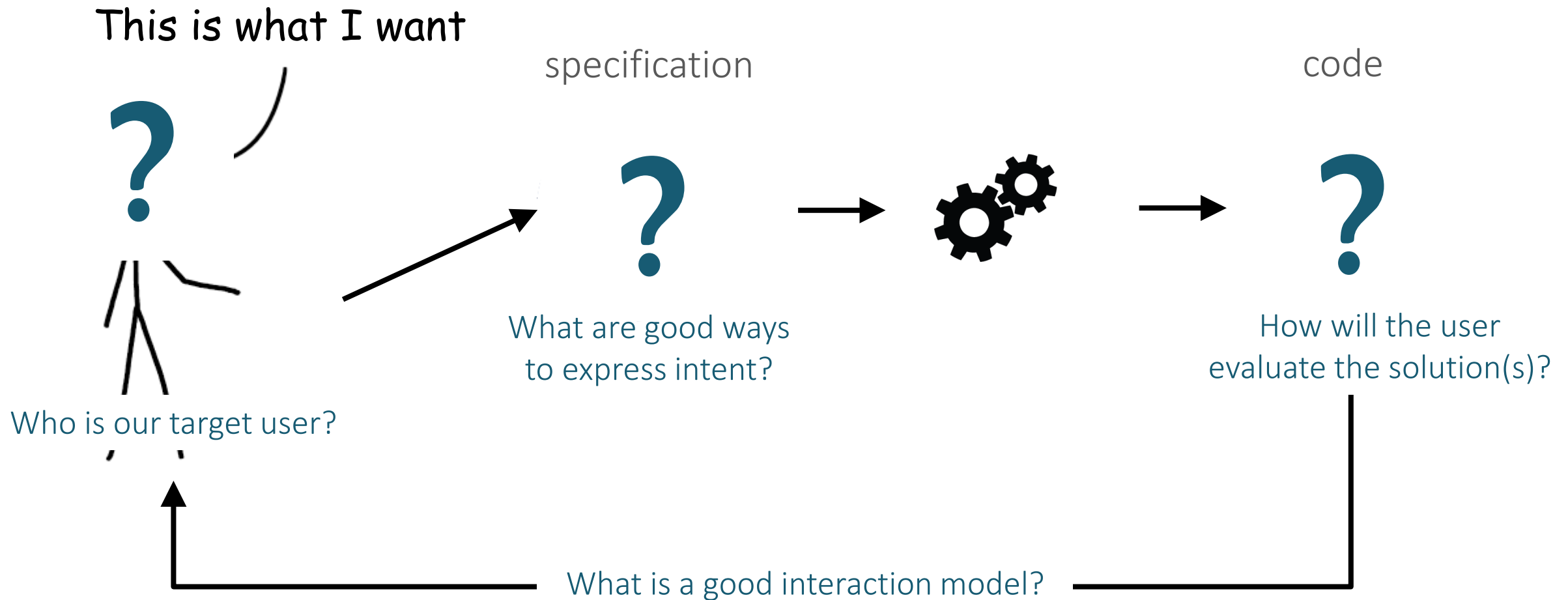
Q2: Do users perceive PBD as more usable?



Q3: Do users perceive PBD as more learnable?



The big picture



Data wrangling for data scientists

Design goals

Formative study with data scientists identified following goals:

1. Must be available where the data scientist works—within their notebooks
2. Must generate Python or R code they can inspect/modify

JupyterLab

http://localhost:8888/lab

File Edit View Run Kernel Tabs Settings Help

ds.ipynb Python 3

[2]: `df = pd.read_csv("montcoalert.zip")`
`df.head(10)[["latlng", "title"]]`

Derive Column by Example

	latlng	title	Column
0	(40.2978759, -75.5812935)	EMS: BACK PAINS/INJURY	EMS
1	(40.2580614, -75.2646799)	EMS: DIABETIC EMERGENCY	EMS
2	(40.1211818, -75.3519752)	Fire: GAS-ODOR/LEAK	Fire
3	(40.1161530, -75.3435130)	EMS: CARDIAC EMERGENCY	EMS
4	(40.2514920, -75.6033497)	EMS: DIZZINESS	EMS
5	(40.2534732, -75.2832450)	EMS: HEAD INJURY	EMS
6	(40.1821111, -75.1277951)	EMS: NAUSEA/VOMITING	EMS
7	(40.2172859, -75.4051820)	EMS: RESPIRATORY EMERGENCY	EMS
8	(40.2890267, -75.3995896)	EMS: SYNCOPAL EPISODE	EMS
9	(40.1023985, -75.2914577)	Traffic: VEHICLE ACCIDENT -	Traffic

Python ▶ Insert as cell below and run + Insert as cell below Copy

[3]: `def derive_value_program(s):`
`return s.split(":")[0]`

[4]: `df["type"] = df.title.apply(derive_value_program)`
`df["type"].value_counts().plot.barh();`

Emergency Type	Count
Fire	~65,000
Traffic	~155,000
EMS	~205,000

A. Users create a data frame with their dataset and sample it.

B. WREX's interactive grid where users can derive a new column and give data transformation examples.

C. WREX's code window containing synthesized code generated from grid interactions.

D. Synthesized code inserted into a new input cell.

E. Applying synthesized code to full data frame and plotting results.

User study

12 participants (data scientists)

6 tasks with two datasets

- string extractions, transformation, formatting

Study results

Task	Manual		WREX		Frequency	
	<i>n</i>	%	<i>n</i>	%	<i>n</i>	Dist.
A1	3	50%	6	100%	12	3
A2	0	0%	6	100%	12	2
A3	2	33%	6	100%	12	2
B1	0	0%	6	100%	12	3
B2	3	50%	6	100%	12	2
B3	4	67%	6	100%	12	2

Table 2: Participant task completion under WREX and manual data wrangling conditions. Participant reported frequency of tasks in day-to-day work. Participants were given five minutes to complete each task. Rating scale for task frequency from left-to-right: Never (1), Rarely (2), Occasionally (3), Moderately (4), A great deal (5). Median values precede each distribution.

Task	Acceptability			
	<i>n</i>	Grid	Code ₁	Code ₂
A1	6	5	3	5
A2	6	5	2	5
A3	6	5	2	5
B1	6	4	2	4
B2	6	4	3	5
B3	6	5	3	5

Table 4: How acceptable was the grid experience and the corresponding synthesized code snippet? Rating scale from left-to-right: Unacceptable (1), Slightly unacceptable (2), Neutral (3), Slightly acceptable (4), and Acceptable (5). Code₁ are the ratings from the code synthesized in the in-lab study. Code₂ are the ratings after incorporating the participants' feedback. Median values precede each distribution.

Regae

[Zhang, Lowmanstone, Wang, Glassman, UIST'20]

Better UI for a regex synthesizer

Regae: contributions

Novel way to express intent: semantics augmentation

Novel way to explain synthesis results to user: data augmentation

Automata-theoretic algorithms to generate explanatory examples

- familiar examples with different output, corner cases, distinguishing examples

Usability confirmed by user study

- Completion rate: 12/12 vs 4/12; twice more confident; less cognitive load

Regae: limitations

Limited to regexes

Which parts are generalizable and which not?

Marking as general affects completeness

Not tolerant to user mistakes

User study participants might not be representative

Regae: questions

Behavioral constraints? Structural constraints? Search strategy?

- IO examples
- Built-in DSL
- Top-down enumerative search

Regae: questions

Does semantic augmentation contribute to behavioral or structural constraints, or something else?

- Structural because it affects the search space

What about data augmentation?

- Directly contributes only to result comprehension
- Indirectly to behavioral because users can use those examples as input

Regae: questions

When can we soundly reject the sketch `concat(<num>, e)`?

- If e.g. `<num>` is marked excluded [that's not what I meant]
- When there is a positive example that doesn't start with a number
- More generally, replace `e` with `star(<any>)` and check whether all positives can be parsed!
 - if under **not**, then replace with an empty-language regex
- Another idea is define equivalence on regexes, e.g. `optional(star(e))` is equivalent to `star(e)`

Regae: questions

Why is it important to randomize the order of control vs treatment?