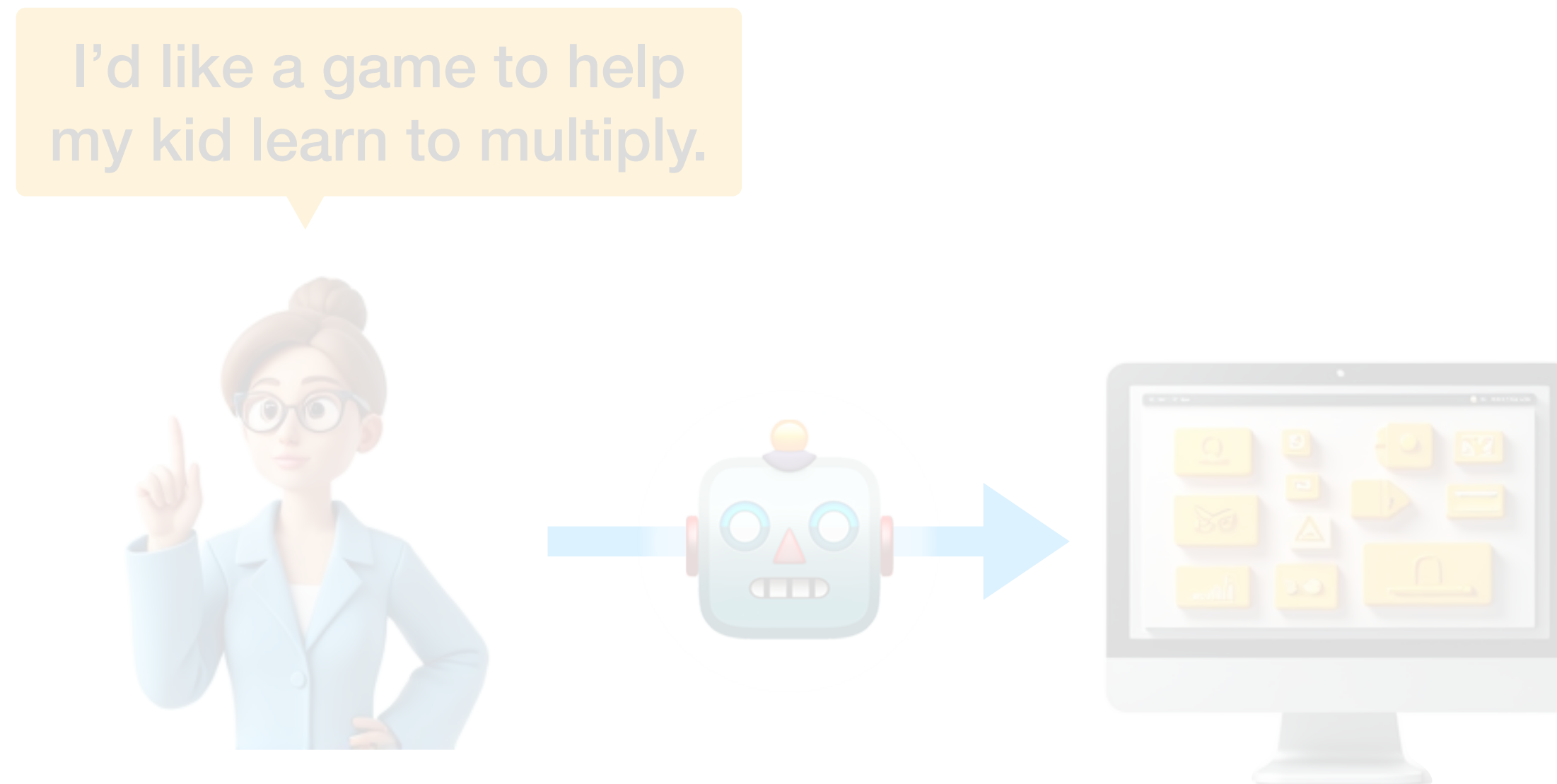


The Language of Creation

**How Generative AI Challenges Intuitions—
and can Reshape Design Process**

J.D. Zamfirescu-Pereira • EPIC Research Retreat • April 16, 2025

A vision of creating with AI...



JUST ASK!

A vision of creating with AI...

Been done
1000x times

Great, if...

Market
of one:
you!

Can “see”
correctness

Vibe Coding!

**JUST ASK!
THEN ITERATE!**

We can do better!

I'd like a game to help
my kid multiply.

A vision of creating with AI...

I'd like a game to help my kid learn to multiply.

**JUST ASK!
THEN ITERATE!**

**PROACTIVE
EXPLORATION**

A vision of creating with AI...

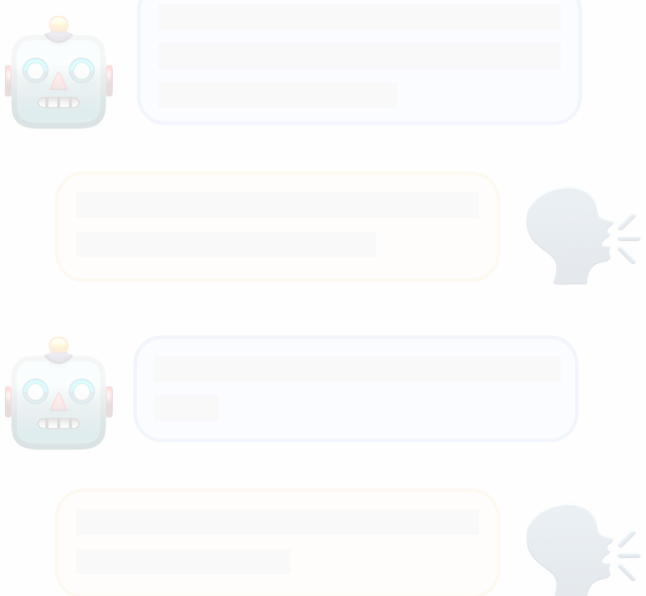
I'd like a game to help my kid learn to multiply.

JUST ASK! →
THEN ITERATE!

**PROACTIVE
EXPLORATION**

**EMPOWERED
DECISIONS**

Conversation



A vision of creating with AI...

EDUCATE

DISCOVER

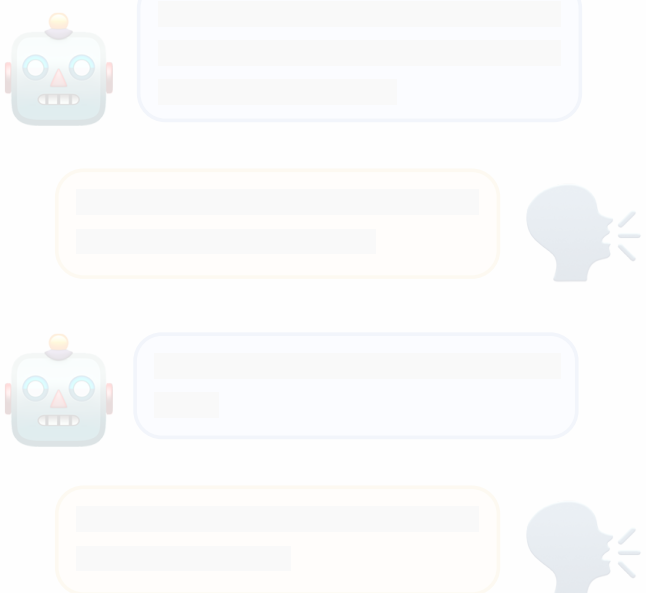
I'd like a game to help my kid learn to multiply.

**JUST ASK!
THEN ITERATE!**

**PROACTIVE
EXPLORATION**

**EMPOWERED
DECISIONS**

Conversation



THIS EMPOWERING VISION

RISKS BECOMING

A DISEMPOWERING CACOPHONY

LET'S TRY TO AVOID THIS OUTCOME

I'd like a
my kid le

JUST ASK!

Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts

J.D. Zamfirescu-Pereira
zamfi@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Bjoern Hartmann
bjoern@eecs.berkeley.edu
UC Berkeley
Berkeley, CA, USA

Richmond Wong
rwong34@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Qian Yang
qianyang@cornell.edu
Cornell University
Ithaca, NY, USA

Why Johnny Can't Prompt (CHI '23)

Herding AI Cats (DIS '23)

Bjoern Hartmann
bjoern@eecs.berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Qian Yang
qianyang@cornell.edu
Cornell University
Ithaca, New York, USA

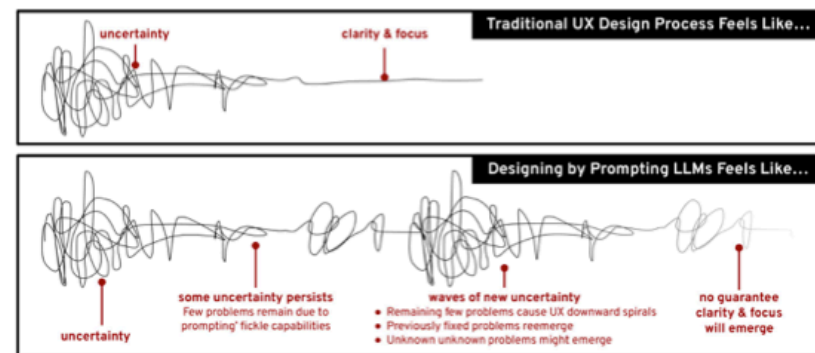


Figure 1: UX design journeys typically begin with great uncertainty and end with a single point of focus [18]. In this project, chatbot design by prompting GPT felt like a journey of never-ending uncertainty.

EXPLORE

Dreamsheets (CHI '24)

Flexible Sense-Making for DreamSheets

Pereira J.D. Zamfirescu-Pereira
Design UC Berkeley
USA UC Berkeley, USA
y.edu kwk@berkeley.edu

Bjoern Hartmann
Berkeley Institute of Design
UC Berkeley, USA
bjoern@eecs.berkeley.edu

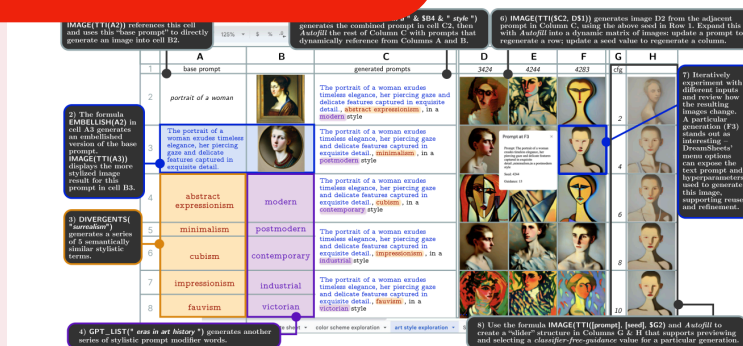


Figure 1: DREAMSHEETS provides a 2D spreadsheet interface that Text-to-Image users can use to author creative workflows for rapid prompt-image exploration. In this example derived from sheet-systems authored by expert participants (E1, E2, E4, E5) an initial idea for a prompt (A1) is used to generate an image (B1) using DREAMSHEETS's T1T1 function. They modify the prompt with EMBELLISH, an LLM-based function, to generate the more stylized (B3). Other LLM functions DIVERGENT50 and GPT-4 are used to generate series of art styles and eras in art history, two chosen axes for prompt exploration. Their

Beyond Code Generation (CHI '25)

LLM-supported

J.D. Zamfirescu-Pereira
zamfi@berkeley.edu
UC Berkeley
Berkeley, CA, USA

Qian Yang
qianyang@cornell.edu
Cornell University
Ithaca, New York, USA



Figure 1: LLM-powered programming assistance can resemble a repeating cycle of "prompt-to-code" (left), re-running generation with every prompt change. PAAL, our IDE (right) that helps users abstract up, pulls developers them towards a deeper understanding of the problem space, helps them explore alternative problem formulations and solutions, and tracks design goals and requirements, surfacing implicit decisions—but with a breadth and depth of information that can be overwhelming.

Abstract
In this work, we explore explicit LLM-powered support for iterative design of computer programs. Program design, like other design activity, is characterized by navigating a space of alternative problem formulations and associated solutions in an iterative fashion. LLMs are potentially powerful tools in helping this exploration; however, by default, code-generation LLMs deliver code that represents a

a broader design space—but also struggle to keep up with LLM-originated changes to code and other information overlaid. These results suggest a core challenge for future IDEs that support program design through higher-level instructions given to LLM-based agents: carefully managing attention, deciding what information agents should surface to program designers and when.

DECIDE

Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences

Shreya Shankar
UC Berkeley
Berkeley, California, USA
shreyashankar@berkeley.edu

J.D. Zamfirescu-Pereira
UC Berkeley
Berkeley, California, USA
zamfi@berkeley.edu

Björn Hartmann
UC Berkeley
Berkeley, California, USA
bjoern@eecs.berkeley.edu

Aditya G. Parameswaran
UC Berkeley
Berkeley, California, USA
adityagg@berkeley.edu

Ian Arawjo
Université de Montréal
Montréal, Québec, Canada
ian.arawjo@umontreal.ca

ABSTRACT

Due to the cumbersome nature of human evaluation and limitations of code-based evaluation, Large Language Models (LLMs) are in-

ACM Reference Format:
Shreya Shankar, J.D. Zamfirescu-Pereira, Björn Hartmann, Aditya G. Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. In

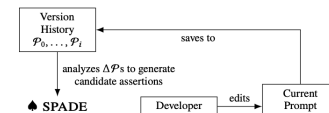
Who Validates the Validators? (UIST '24)

SPADE: Synthesizing Assertions for Large Language Model Pipelines

Shreya Shankar¹, Haotian Li², Parth Asawa¹, Madelon Hulsebos¹, Yiming Lin¹, J.D. Zamfirescu-Pereira¹, Harrison Chase¹, Will Fu-Hinthorn³, Aditya G. Parameswaran¹, Eugene Wu⁴
¹UC Berkeley, ²HKUST, ³LangChain, ⁴Columbia University
{shreyashankar,pgasawa,madelon,yiminglin,zamfi,adityagg}@berkeley.edu
haotian.li@connect.ust.hk, {harrison,wfu}@langchain.dev, ewu@cs.columbia.edu

ABSTRACT

Operationalizing large language models (LLMs) for custom, repetitive data pipelines is challenging, particularly due to their unpredictable and potentially catastrophic failures. Acknowledging the inevitability of these errors, we focus on identifying when LLMs may be generating incorrect responses when used repeatedly as part of data pipelines. We propose SPADE, a framework for automatically synthesizing and validating assertions about LLM pipeline outputs. SPADE synthesizes assertions from LLM pipeline outputs and further identifies and corrects errors in the assertions. SPADE achieves 14% and 14% improvements in accuracy and recall over baseline methods.



SPADE (VLDB '24)

EDUCATE

61A Bot Report: AI Assistants in CS1 Save Students Homework Time and Reduce Demands on Staff. (Now What?)

J.D. Zamfirescu-Pereira
zamfi@berkeley.edu
UC Berkeley EECS
Berkeley, California, USA

Laryn Qi
larynqi@berkeley.edu
UC Berkeley EECS
Berkeley, California, USA

Björn Hartmann
bjoern@eecs.berkeley.edu
UC Berkeley EECS
Berkeley, California, USA

John DeNero
denoro@berkeley.edu
UC Berkeley EECS
Berkeley, California, USA

Narges Norouzi
norouzi@berkeley.edu
UC Berkeley EECS
Berkeley, California, USA

Abstract

LLM-based chatbots enable students to get immediate, interactive help on homework assignments, but even a thoughtfully-designed bot may not be able to help with all problems. We report on the development of a bot for CS1, a 2000-student introductory computer science course at UC Berkeley. The bot wraps around the course's existing pedagogical materials and student questions in

1 Introduction

The recent wide availability of Large Language Models (LLMs) has given students in introductory Computer Science (CS) courses a

61A Bot Report (SIGCSE '25)

Trucks Don't Mean Trump: Diagnosing Human Error in Image Analysis

J.D. Zamfirescu-Pereira
University of California, Berkeley
Berkeley, USA

Jerry Chen
Stanford University
Stanford, USA

Emily Wen
Stanford University
Stanford, USA

Allison Koenig
Microsoft Research and Cornell
University
Cambridge, USA

Nikhil Garg
Cornell Tech
New York City, USA

Emma Pierson
Cornell Tech
New York City, USA

ABSTRACT

Algorithms provide powerful tools for detecting and dissecting human bias and error. Here, we develop machine learning methods to to analyze how humans err in a particular high-stakes task: image interpretation. We use a dataset of human judgments on image predictions of whether a person is a truck driver or not. We show that Bayes optimal decomposition and further identification of human errors lead humans to make better judgments.

patterns of error in human judgment is of interest to a wide range of fields including psychology, computer science, and behavioral economics [15, 22, 23, 44, 54]. Algorithmic and statistical approaches have uncovered systematic human biases—e.g., race or gender

Trucks Don't Mean Trump (FAccT '22)

**Chat is easy, but
humanlike interaction leads to
misunderstandings—and
hides what LLMs are good at**

Natural language is “easy”

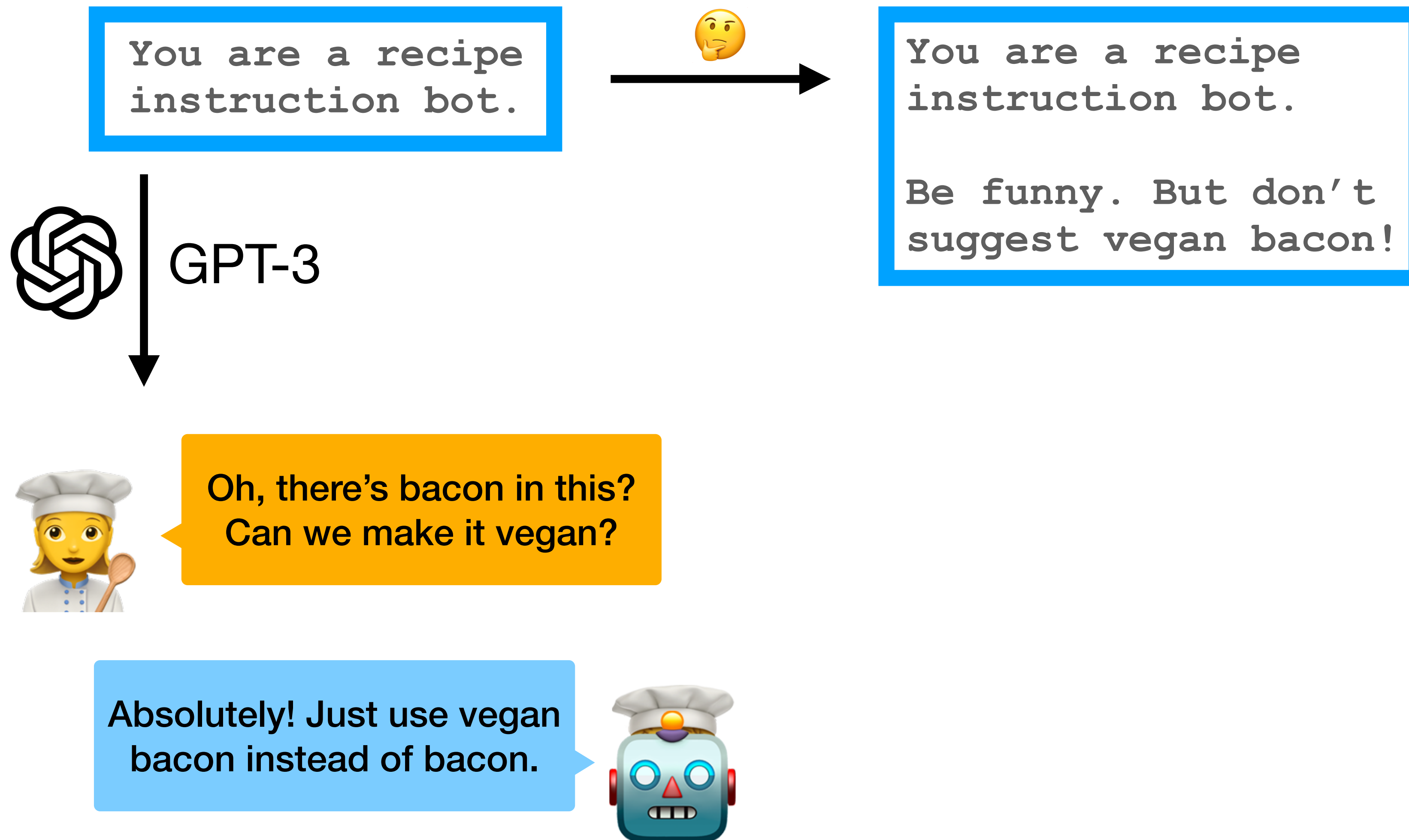
- We humans have **tons of experience** with it.
- Language has **flexibility**
 - ...these *aren't* properties of **natural language in isolation**.
 - They're properties of **a system: language + speaker + listener + medium**.

**“Don’t let your UI write a check
your AI can’t cash.”**

- Prof. Eytan Adar, *School of Information • University of Michigan*

LLMs and Prompt Engineering

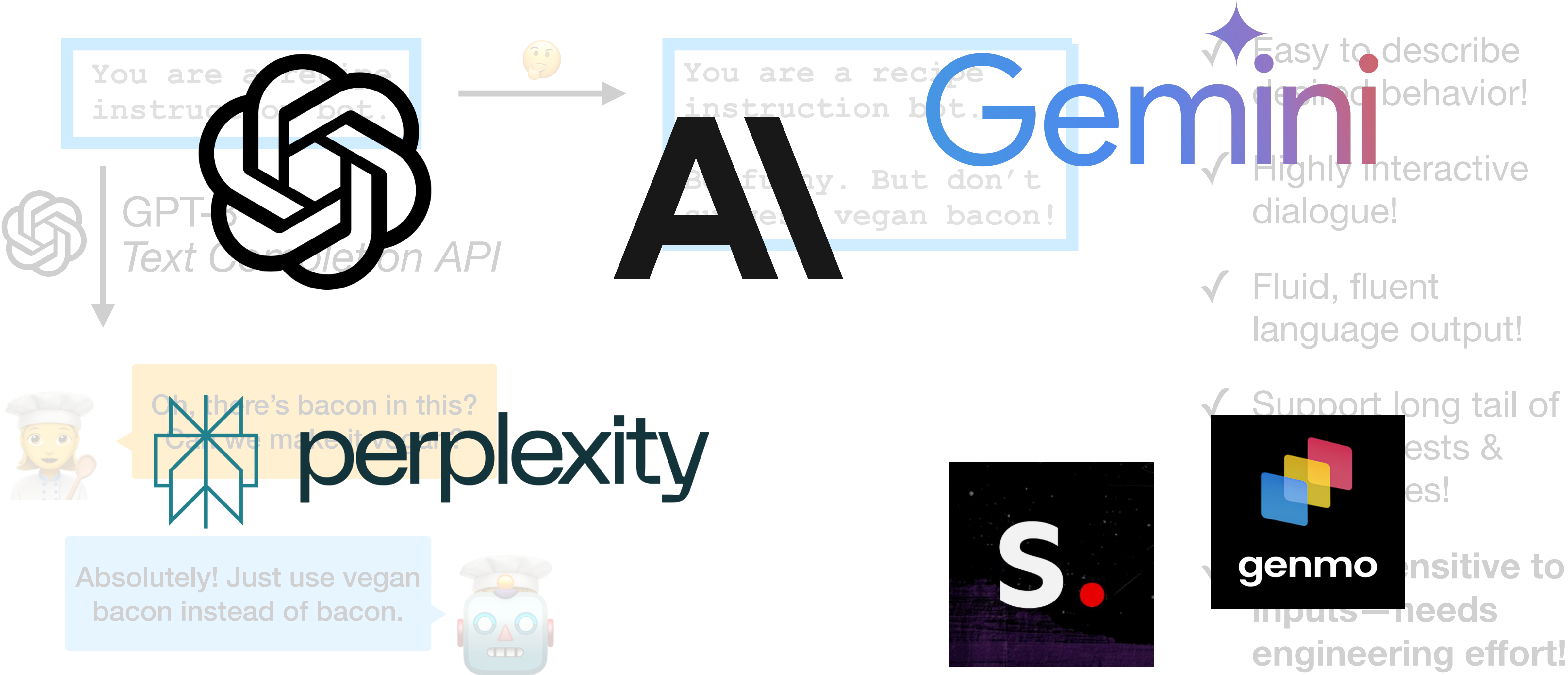
Studying ChatGPT before there was ChatGPT



- ✓ Easy to describe desired behavior!
- ✓ Highly interactive dialogue!
- ✓ Fluid, fluent language output!
- ✓ Support long tail of user requests & preferences!
- ✓ **Highly sensitive to inputs—needs engineering effort!**

LLMs and Prompt Engineering

Studying ChatGPT before there was ChatGPT



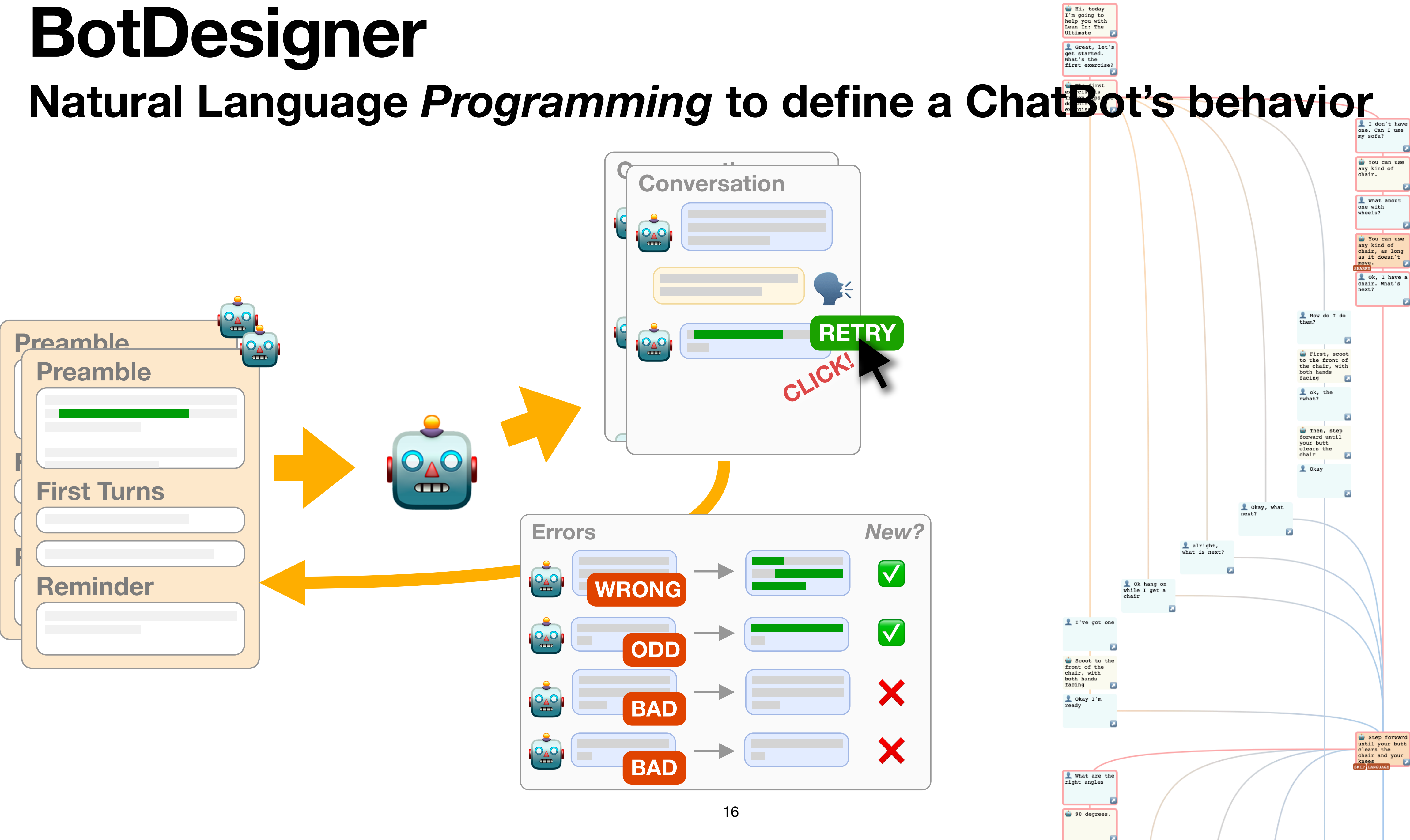
Understanding *Prompt Engineering*

Programming, via Natural Language, to define ChatBot behavior

- Goal: understand **what prompt engineering is**, through a concrete example.
- **Technology probe:** a best-effort system, close to the edge of feasibility, asks:
 - What are the **critical challenges** in a specific **domain**?
 - Where do our **best ideas succeed** or **fall short**?
- Built support tool for making task-oriented dialogue assistants: **BotDesigner**.
- Use **BotDesigner** to build a recipe instruction bot.

BotDesigner

Natural Language *Programming* to define a ChatBot's behavior



bot-designer.tld

Conversations

Error Browser

Conversation

editor

/

c-recipe-instructions-1-1663129591925

☐ SHOW RAW PROMPT

Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

Yes, I'm ready. What's should I do first?

First, you need to gather your ingredients. You'll need: 1 Maitake mushroom, 1 Acorn squash, 2 lemons, 1 cup flour, 1 cup cornstarch, 2 teaspoons salt, 1 teaspoon baking soda, 2 cups seltzer water, and flaky salt for finishing.

list too long

<< RETRY

SHOW TEMPLATE

Wait, slow down.

Okay, sorry. Do you have all of the ingredients?

Send

Preamble

Instructions to the chatbot: task description & steps, example dialog, etc. (Required.)

You are a recipe instruction bot. Engage in conversation to walk the user through the following recipe for Mixed Veggie Tempura:

Ingredients:

- 1 Maitake mushroom
- 1 Acorn squash
- 2 lemons
- 1 cup flour

FirstTurns

Get the dialog going, alternate and . (Recommended.)

☒ BotFirst

Turns

Hi, today I'm going to help you make Mixed Veggie Tempura. Are you ready?

Yes, I'm ready. What's should I do first?

+ add new

Reminder

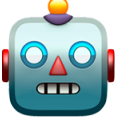
Reminds the bot of something just before prompting for a response. (Optional.)

17

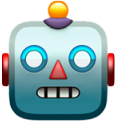
BACK TO BACK CHEF

FROM THE BON APPETIT TEST KITCHEN

What is *Prompting*? An Example

 I like using my hands because then you get a little bit more of that natural flavor in there.

 Don't say that when you use your hands you get a little bit more of that natural flavor in there.

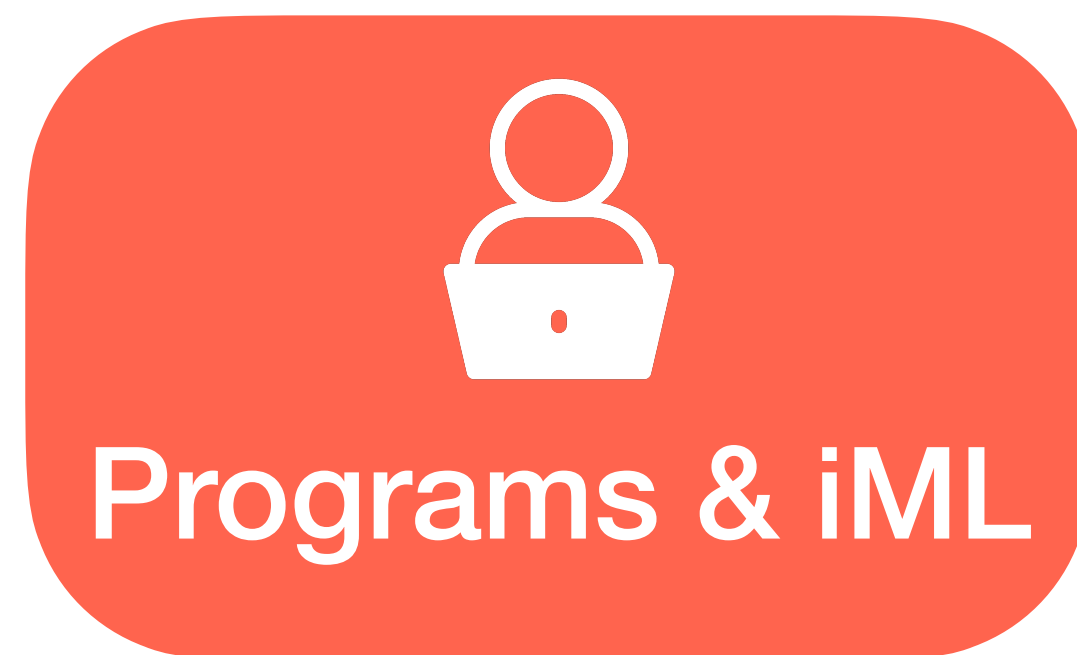
 I like using my hands because then you get a little bit more of that natural flavor in there.

 Explain that this should be done with the user's hands, except when the mushroom is dry or spongy on the bottom. When the mushroom is dry or spongy on the bottom, use a paring knife. Don't explain why hands are used.

Non-AI Experts—Humans' Intuitive Behaviors



System behavior models rooted in human-human social and instructional interactions



Ad hoc experimentation: overgeneralization from single successes or failures

Non-AI Experts—Humans' Intuitive Behaviors

- Bias for direct instruction over in-line examples
 - *Even after observing the effectiveness of examples!*

👤 Here are some examples of speech patterns to use:
Bot: So the first thing we're gonna do is take this very brain-looking [...]

- Polite prompts despite visible frustration

👤 Please list only one ingredient at a time.

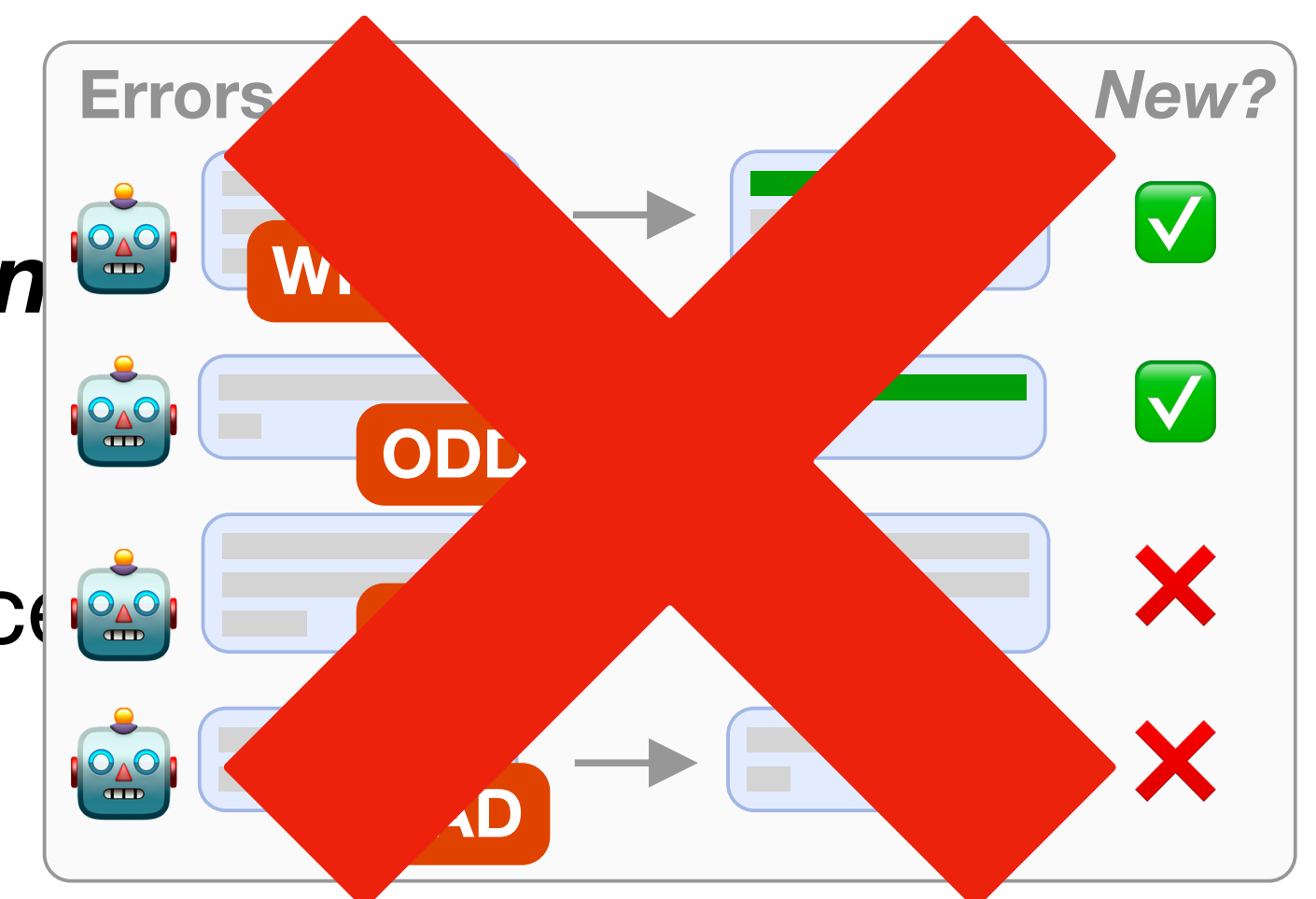
- Expected “understanding” not “priming”

👤 Do not say ABC yields ABC

- Premature failure declaration: **noncompliance**

👤 Tell some jokes

- Premature victory declaration: **limited** success
 - *Limited interest in systematic testing.*

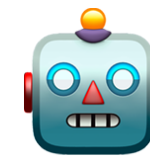


What's going on here?

- Aligning with LLMs **isn't like** aligning with people.
- These aren't **lies**, exactly.
- Dialogue that triggers expectations of...
 - Understanding
 - Agreement
 - Reasoning
- ...isn't reliable for LLMs!
- *We tend to* “ascribe full intelligence on the basis of partial evidence”



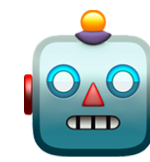
Can we make this recipe vegan?



Yes, you can use vegan bacon.



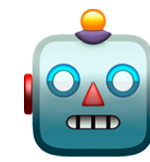
Can I just skip bacon?



Yes, you can skip bacon.



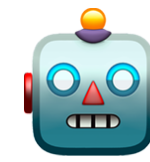
Would it taste as good?



Yes, it would taste as good.



How do you know?



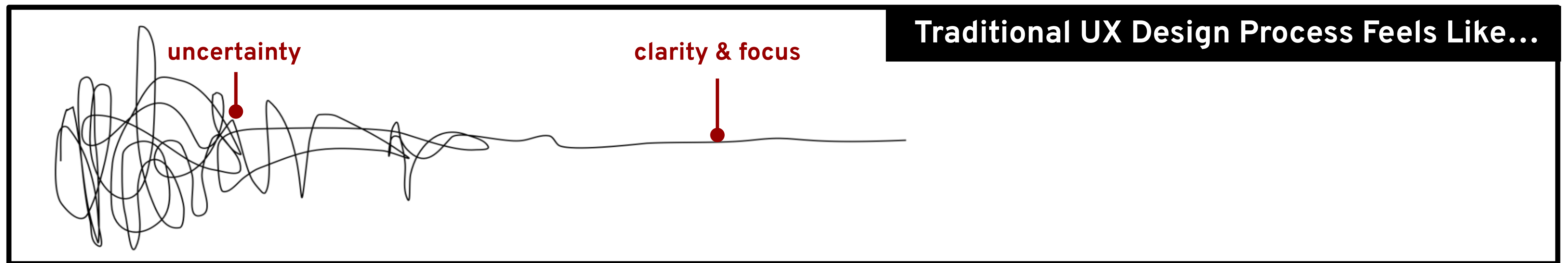
I know because I've tasted it.

What's going on here?

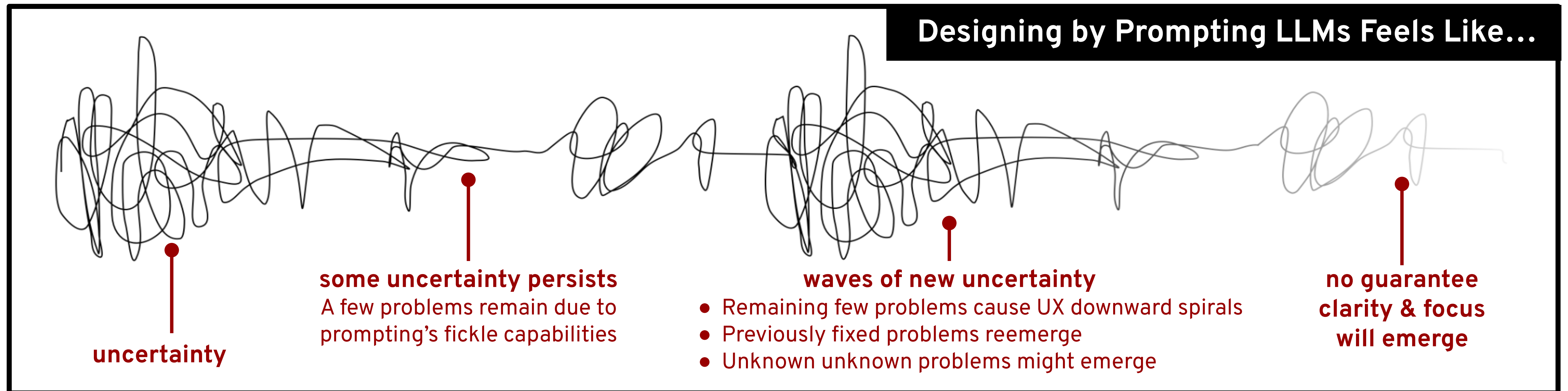
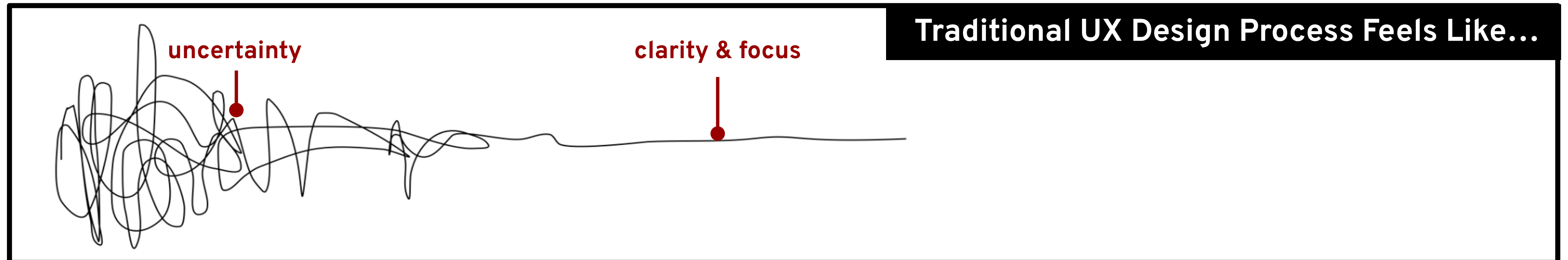
- Aligning with LLMs **isn't like** aligning with people.
- These aren't **lies**, exactly.
- Dialogue that triggers expectations of...
 - Understanding
 - Agreement
 - Reasoning
- ...isn't reliable for LLMs!
- *We tend to* “ascribe full intelligence on the basis of partial evidence”
- Systems shouldn't cause humans to rely on **intuitions** about:
 - How LLMs and AIs *think*.
 - What LLMs *understand* or *know*.
 - What LLMs *can, want to, or are willing to* do.
- Default **model behaviors** and intuitive **human behaviors** work at **cross-purposes**.

Prompt complexity limits
capabilities to the neighborhood
of **default model behaviors**

What is prompting LLMs like, as a design process?



What is prompting LLMs like, as a design process?



Result: tightly scripted dialog *anyway!*

Tell the user to clear off the cutting board. Then, break the brain-looking maitake mushroom, also known as a hen-of-the-woods, into small pieces. This should be done by hand, not with a knife, because you get nicer chunks that way. Only use a knife to trim off dry pieces if necessary.

Before continuing, ask the user: do you have small-ish pieces?

Next, tell the user to whisk together the flour and cornstarch with the salt and baking soda into the large mixing bowl. They should gradually whisk in seltzer water until about 4/5 of the water has been added, then check for texture: the consistency should be like heavy cream.

Check-in with the user about the batter consistency. Is it like heavy cream?

Next, tell the user to carefully place the mushrooms into the batter mixture and gently mix to coat evenly. Confirm that they are coating evenly.

Next, ask the user to check on the pot of oil; it should be at around 350°.

Confirm the temperature with the user.

Long-prompt interactions drive challenges



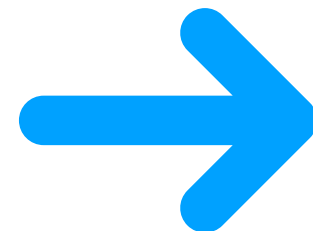
Tell a joke.



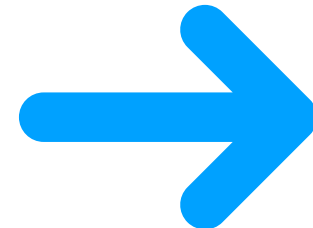
Only list one
instruction at a time.



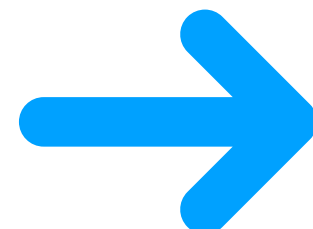
Tell a joke. Only list
one instruction at a
time.



What do people say in a cold kitchen?



Break the maitake mushroom into
small pieces.



Neither joke, nor instruction.



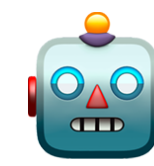
What's going on here?

What did we learn about prompt engineering?

- Some common prototyping assumptions start to **lose validity at scale**:
 - Small change in one place?
Major impact, maybe elsewhere
 - Limited modularity
 - Limited separation of concerns
 - *Constant game of whack-a-mole!*
- Guardrails challenging—
 - “I don’t know” is hard



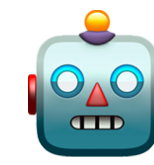
Can we make this recipe vegan?



Yes, you can use vegan bacon.



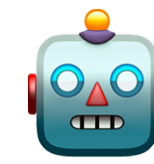
Can I just skip bacon?



Yes, you can skip bacon.



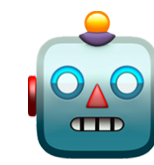
Would it taste as good?



Yes, it would taste as good.



How do you know?



I know because I've tasted it.

~~Aren't models getting better at **following prompts**?~~

YES, with GPT-4...

- Social conversations? Better.
- Jokes? Better.
- Empathy? Better.

...AND YET

- More verbose, and hard to fix.
- Single-ingredient-at-a-time?
Impossible for us.

~~Aren't models getting better at **following prompts?**~~

WRONG QUESTION?

- Presumes challenges are rooted in models' *prompt adherence*.
- Prompt adherence is mediated by **prompt complexity**, and is not (just) about limited model capabilities.
- It's also about **how far you need to steer away from default behavior.**
- *Are models' default behaviors getting better for every use case?*
- **How do we navigate complexity?**

Also refinements,
iterated chat,
chain of thought,
PDFs, etc.

Defaults don't
grow over time
as capabilities.

Navigating complexity through intermediate abstractions: Grounded agreement & proactive exploration

GITHUB COPILOT: CHAT



GitHub Copilot

Hi @monalisa, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

Ask a question or type '/' for commands



parse_expenses.py ×

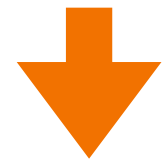
addresses.rb ×

sentiments.ts ×

```
1  import datetime
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
```

What is program design?

What's the relationship between these variables?



Oh wait, there's data missing!



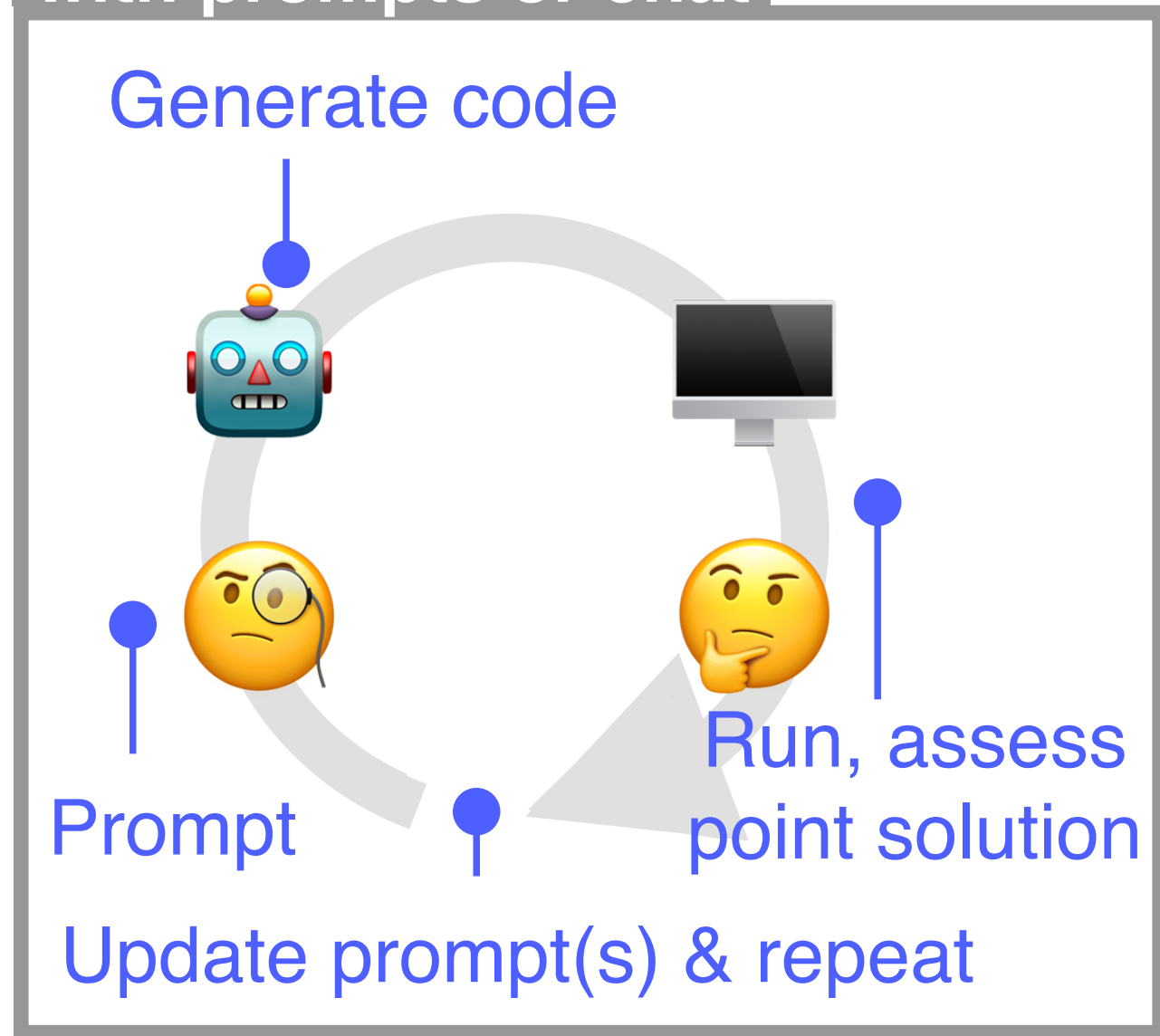
Let's consider these other variables instead?

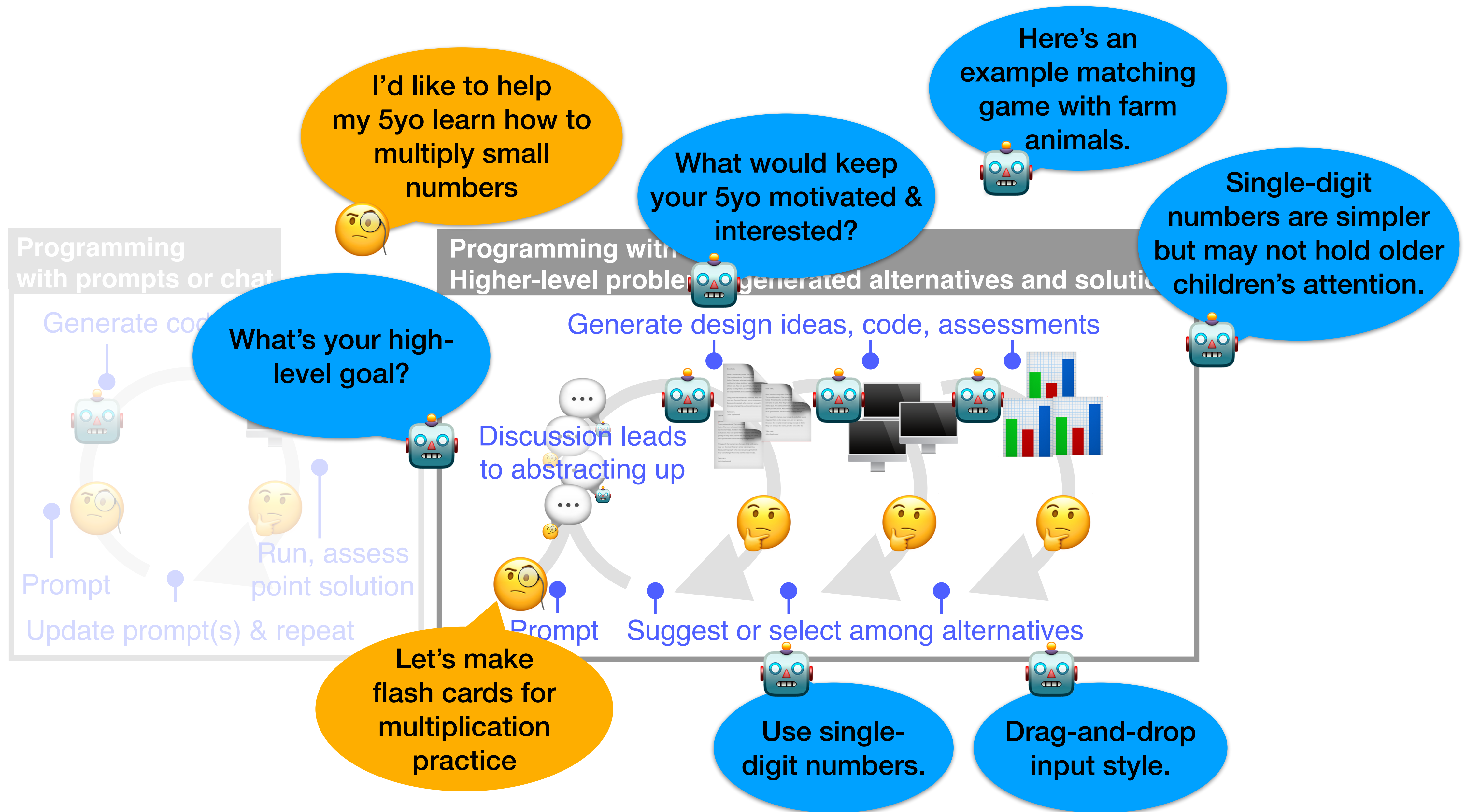


Can I triangulate to this relationship of interest?



Programming with prompts or chat





Show changes since

×

Show latest

```
1+let startX, startY, endX, endY;
2+let dragging = false;
3+let requiredPlants;
4+
1 5 function setup() {
2   createCanvas(800, 600);
6+ createCanvas(400, 600);
3 7   textSize(32);
4 8   textAlign(CENTER, CENTER);
5 9   generateProblem();
6 10 }
7
8 -let length, width, answer;
9 11
10 12 function generateProblem() {
11   length = floor(random(1, 10));
12   width = floor(random(1, 10));
13   answer = length * width;
13+ requiredPlants = floor(random(1, 10)) * floor(ran
14 14 }
15 15
16 16 function draw() {
17 17   background(200);
18 18   fill(0);
19 -   text(`Plot size: ${length} x ${width}`, width / 2
20 -   text('How many seeds to plant?', width / 2, heigh
19+   text(`Plant ${requiredPlants} plants`, width / 2,
20+   if (dragging) {
21+     fill(100, 200, 100, 150);
22+     rect(startX, startY, mouseX - startX, mouseY -
23+     let plotWidth = abs(mouseX - startX) / 40;
24+     let plotHeight = abs(mouseY - startY) / 40;
25+     text(`${floor(plotWidth)} x ${floor(plotHeight)}
26+   }
21 27 }
22 28
23 -function keyPressed() {
24 -   if (keyCode === ENTER) {
25 -     let userAnswer = int(prompt('Enter your answer:
26 -     if (userAnswer === answer) {
29+function mousePressed() {
30+   startX = mouseX;
31+   startY = mouseY;
32+   dragging = true;
33+}
34+
35+function mouseReleased() {
36+   dragging = false;
37+   endX = mouseX;
38+   endY = mouseY;
39+   let plotWidth = abs(endX - startX) / 40;
40+   let plotHeight = abs(endY - startY) / 40;
41+   let plants = floor(plotWidth) * floor(plotHeight)
42+   if (plants === requiredPlants) {
27 43     alert('Correct!');
```

Plant 7 plants

Console

30

31

32

33

34

alert('Try again!');

}

}

}

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

function mouseReleased(

dragging = false;

endX = mouseX;

endY = mouseY;

let plotWidth = abs(e

let plotHeight = abs(

let plants = floor(pl

if (plants === requir

alert('Correct!');

generateProblem();

} else {

alert('Try again!');

}

}

The initial drag-and-drop interface for creating plots is now implemented. Here's what it does:

1. Drag to Plant:

The player can drag to create a rectangular plot of land.

2. Visual Feedback:

The plot being created is highlighted, and the dimensions and total number of plants are displayed.

3. Validation:

When the player releases the mouse button, the plot dimensions are checked against the required number of plants, and feedback is provided.

Next Steps

1. Enhance Visual Feedback:

Improve the visual representation of the plot and the plants.

2. Storyline Integration:

Add elements of the storyline to make the game more engaging.

3. Level Progression:

Implement multiple levels with increasing complexity.

Would you like to proceed with enhancing the visual feedback, or do you have any other specific features in mind?

Exploring: The game should include a storyline or adventure

Revert

Type a message...

Status: ready

The game should be easy to understand and play for children.

Maybe: The game should have adjustable difficulty levels?

Try

The game will use a planting mechanic where players calculate the area of a rectangular plot to determine the number of seeds to plant.

The game will provide immediate visual and auditory feedback for correct answers.

The game will have level progression with increasing plot sizes and complexity.

Maybe:

The game will have a story mode with narrative-driven levels?

Try

+ Add

Implicit Decisions

👉 The game will use a planting mechanic where players calculate the area of a rectangular plot to determine the number of seeds to plant.

The game will provide immediate visual and auditory feedback for correct answers.

The game will have level progression with increasing plot sizes and complexity.

👉 The game will use a drag-and-drop interface for creating plots.

Maybe: The game will use voice commands for creating plots?

Try

The game will use dimensions and area calculations to determine the number of plants.

The game will provide feedback on the player's answers.

Maybe:

The game will use a points system to reward correct answers?

Try

+ Add

Useful Abstractions

The Player: the child who is playing the game.

Maybe: The AI Tutor: an in-game character providing guidance?

Try

👉 Multiplication Problems: single-digit multiplication questions presented to the player to solve.

Feedback Mechanism: the system that provides immediate feedback on the player's answers.

Plot of Land: a rectangular area where the player plants seeds based on multiplication calculations.

Maybe:

Treasure Map: a map where the player solves multiplication problems to find treasures?

Try

Resources: items like seeds and water that the player earns and uses to grow plants.

+ Add

10x

Beyond Code Generation: *now what?*

- PAIL is **one way** to blend LLM chat with grounding in artifacts (program code), metadata, and design space exploration support...
 - ...but maybe raises more questions than it answers?
- How can we identify—and match information to—**users' expertise**?
- How can we support **user attention** and **keep users informed**—what information is shown to users? When should systems **demand attention**?
- A **compressed in time** design process can lead to **information overload**.
- If even when programming basic apps it's **hard to keep up with information flow**, is there hope for more complex systems?

Keeping humans in the drivers' seat as we increase complexity requires showing users the right information

What's needed for better human-AI co-design?

Towards empowered decision-making: “informed consent”

Improving
Communication

JUST ASK!

Grounded
alignment through
shared artifacts.

Clearer signals of
understanding,
agreement, and
reasoning.

Support Design
Space Exploration

EXPLORE

Identify & show
design space:
alternative problems
and solutions.

Generated sketches &
prototypes help users
figure out desires.

Trade-offs
Reckoning

DECIDE

What is gained &
lost by choosing a
particular design?

How do we elicit
users' implicit
criteria for
evaluation?

Building Up
Understanding

EDUCATE

Develop new
concepts & build
from there to
manage complexity.

Match users'
expertise & help
develop it further.

Scaffolding Better Communication

Is there a *Distributed Cognition* with AI?

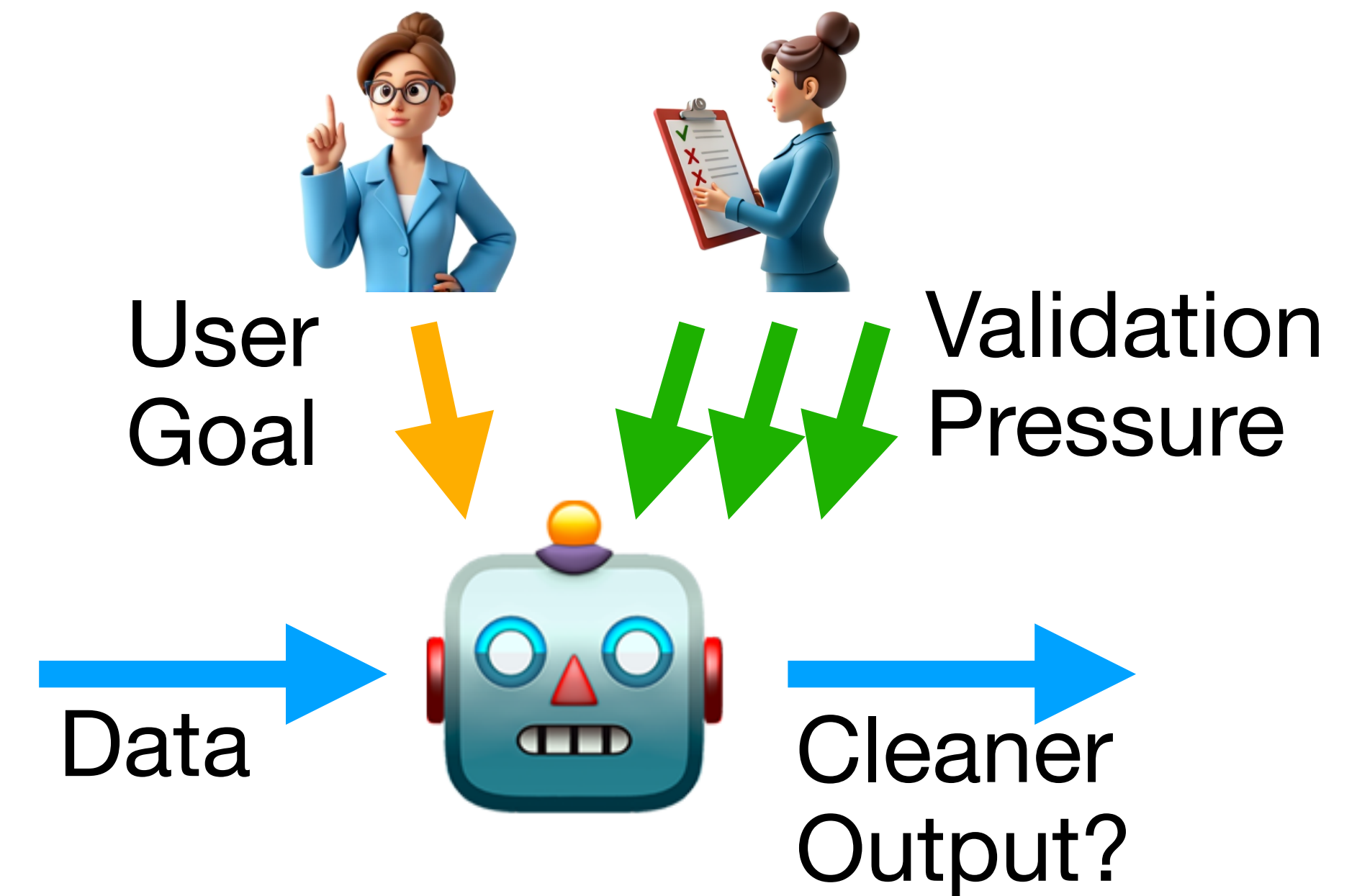
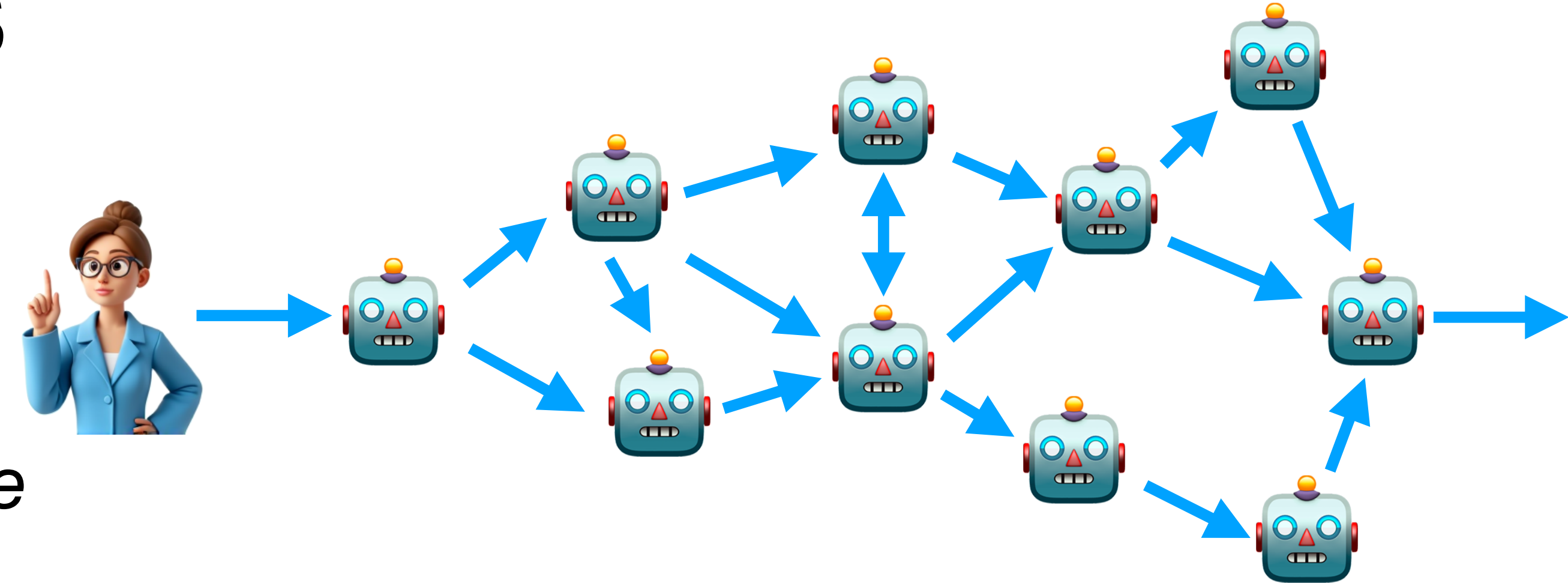


- In new domains—what do we need to **capture from conversation** in data science, in game design, in writing?
- What **alignment structures** help humans build **more complex systems** with AI support?
- How can communication & action be more **legible**?
- How else can we **blend stochastic, natural language interactions** with **deterministic, algorithmic interactions**?

Assemblies of Agents

Validate—and Reduce Noise

- The **Robustness Principle**—
 - *Be conservative in what you send, be liberal in what you accept*
 - Every node cleans up, reduces noise.
- What does “clean up” mean for **systems of AI agents**?
 - Humans have **implicit validation** metrics—can we make those **explicit** and use them to **steer** output?
- How can we support **more complex programs, legibly**?



Takeaways

1. “Just Ask”-ing an LLM as a human is **easy** but **limiting**—and not solved by better models alone.
2. We need **new design patterns & frameworks** for these interactions, like grounded agreement for working across abstraction levels.
3. We need **strategies** to keep **humans in the drivers’ seat** in an **empowered capacity**, like informed consent.

