Explorations of ACT-R, Cognitive Code, & Teachable Agents

Dario Salvucci & Jennifer Engimann

Drexel University

Teachable Agents

- We have various ways to get computers to do stuff
  - Traditional programming
    - Experts writing code to tell a machine exactly what to do
  - Machine learning
    - Training a machine using large data sets of information/behavior
  - Scripting (in many forms)
    - Apple Shortcuts, Google Routines, *If-This-Then-That*

Teachable Agents

- Is there a better way?

Teachable Agents

- There have been many nice building blocks in the ICCM / CogArch community in recent years…
  - Learning from Instructions (Anderson, Taatgen, others)
  - Interactive Task Learning (Laird, others)
  - Cognitive Twin (Somers, Oltramari, Lebiere)
- These efforts complement programming & machine learning in that they offer promise for…
  - Learning from one or just a few examples
  - Flexibility in transferring skills across domains
  - Accounting for realistic human performance

Teachable Agents

- In the past year, we've been exploring how to apply some of our past work on instruction learning to building teachable agents w/ cognitive architectures
- Caveat: This is all very half-baked at the moment
- At least two dimensions to this effort:
  - Theoretical questions
    - Centrally: Can an architecture like ACT-R help us build a better teachable agent?
  - Technological questions
    - Centrally: Can we build a teachable agent that can be applied to, and is usable for, common real-world tasks?

Web-Based Task Agents

- Web-based UIs have become the dominant form of user interface
  - Both within standard web browsers (e.g., Chrome, Safari) and within embedded apps (e.g., Electron-based apps)
- So, an agent that can perform web-based tasks can perform (or help perform) a multitude of tasks…
  - Searching for information
  - Online shopping
  - Managing a calendar/schedule
  - Playing games
  - etc.

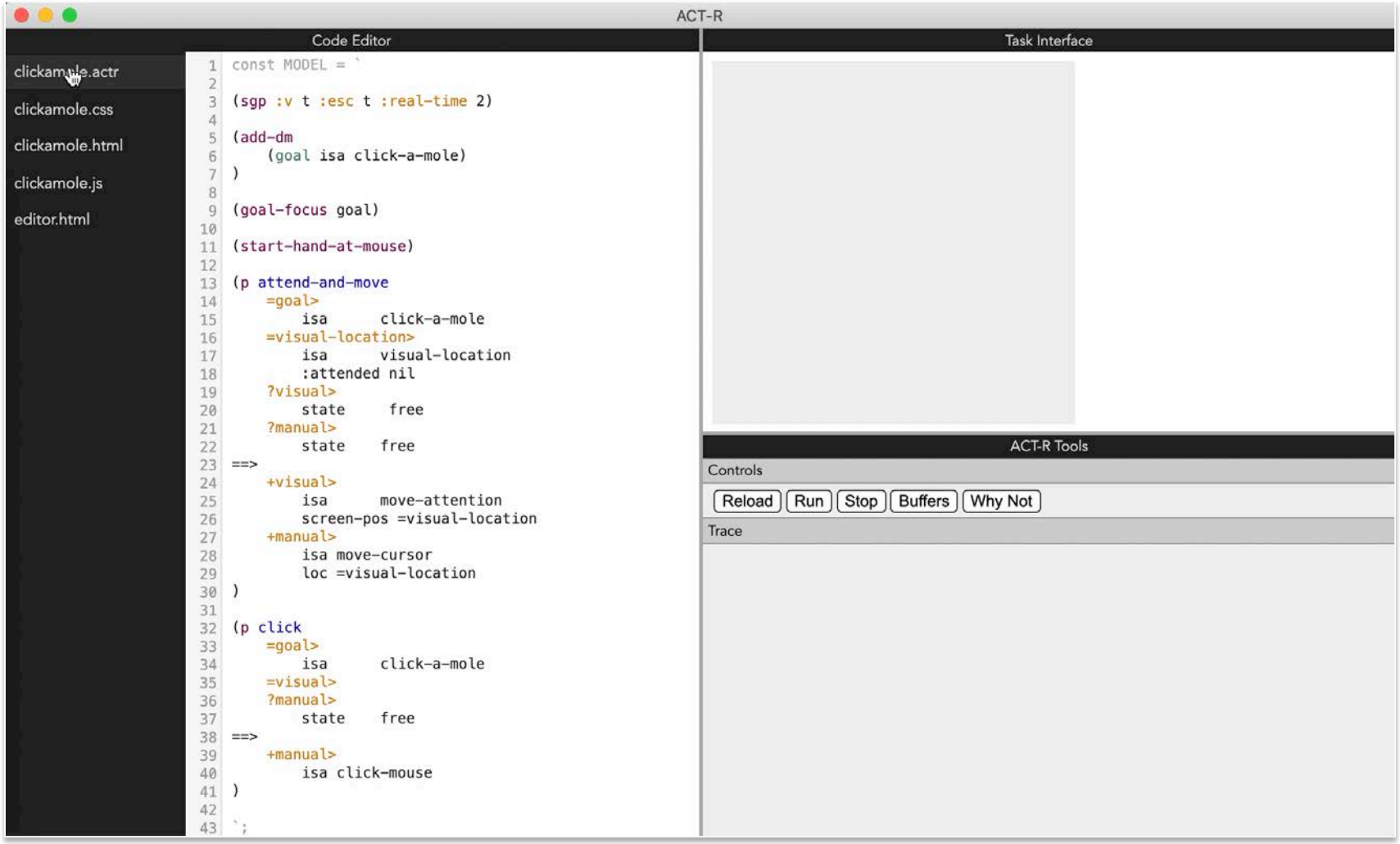
ACT-R + Web-Based Tasks

- We've just started developing a JavaScript version of ACT-R for two reasons…
- (1) To facilitate a cognitive modeling course
  - CS students are very comfortable with web development, and JavaScript runs on any machine
  - (This will likely replace the Java version we've been using)
- (2) To facilitate web-based ACT-R agents…
  at least, potentially
  - It can run natively within the browser without relying on communication with an external server

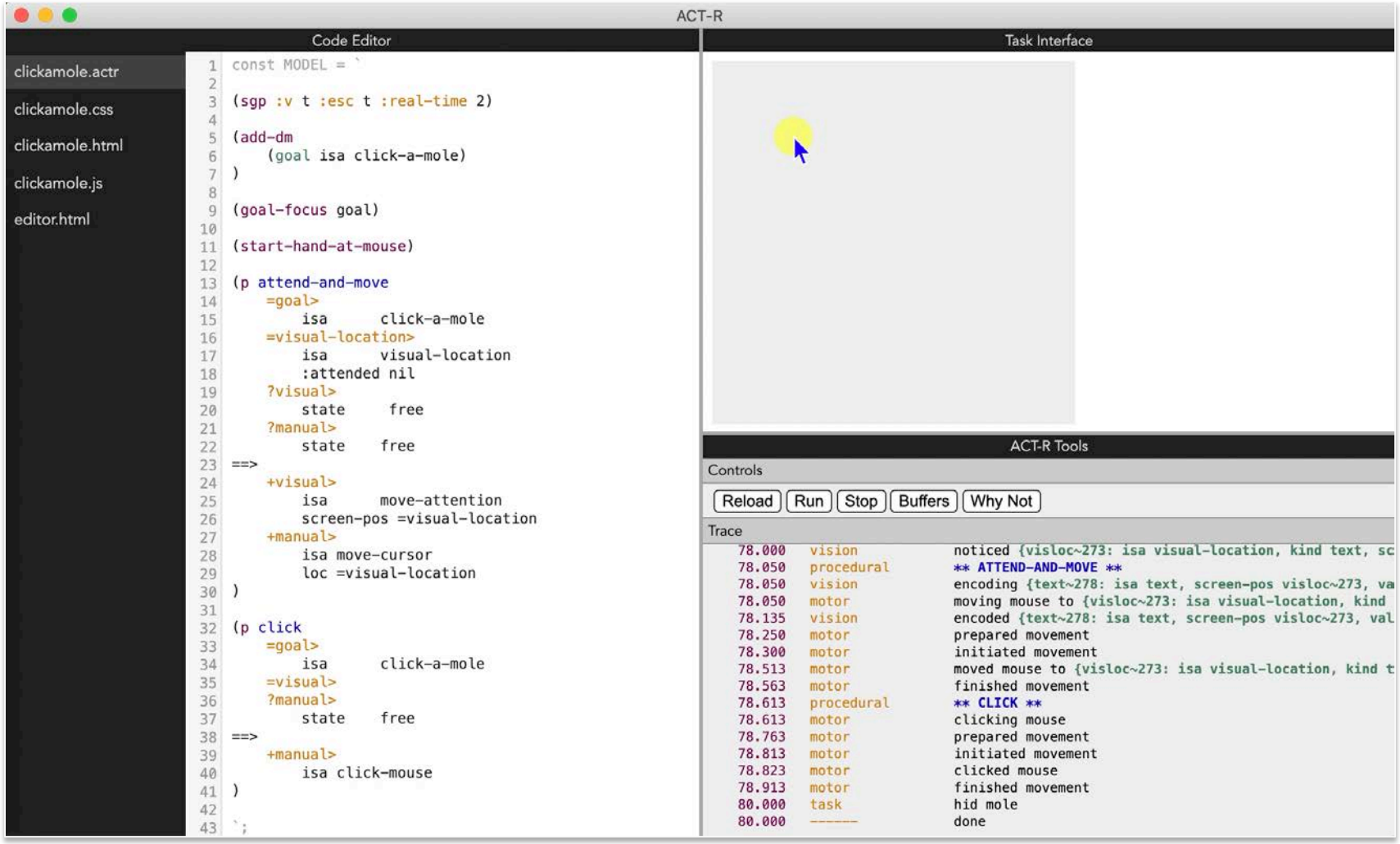
ACT-R + Web-Based Tasks

- The code is 1 month old now and still a mess…
- But we can do some interesting things with it…
- Especially when embedded into a full desktop app using Electron

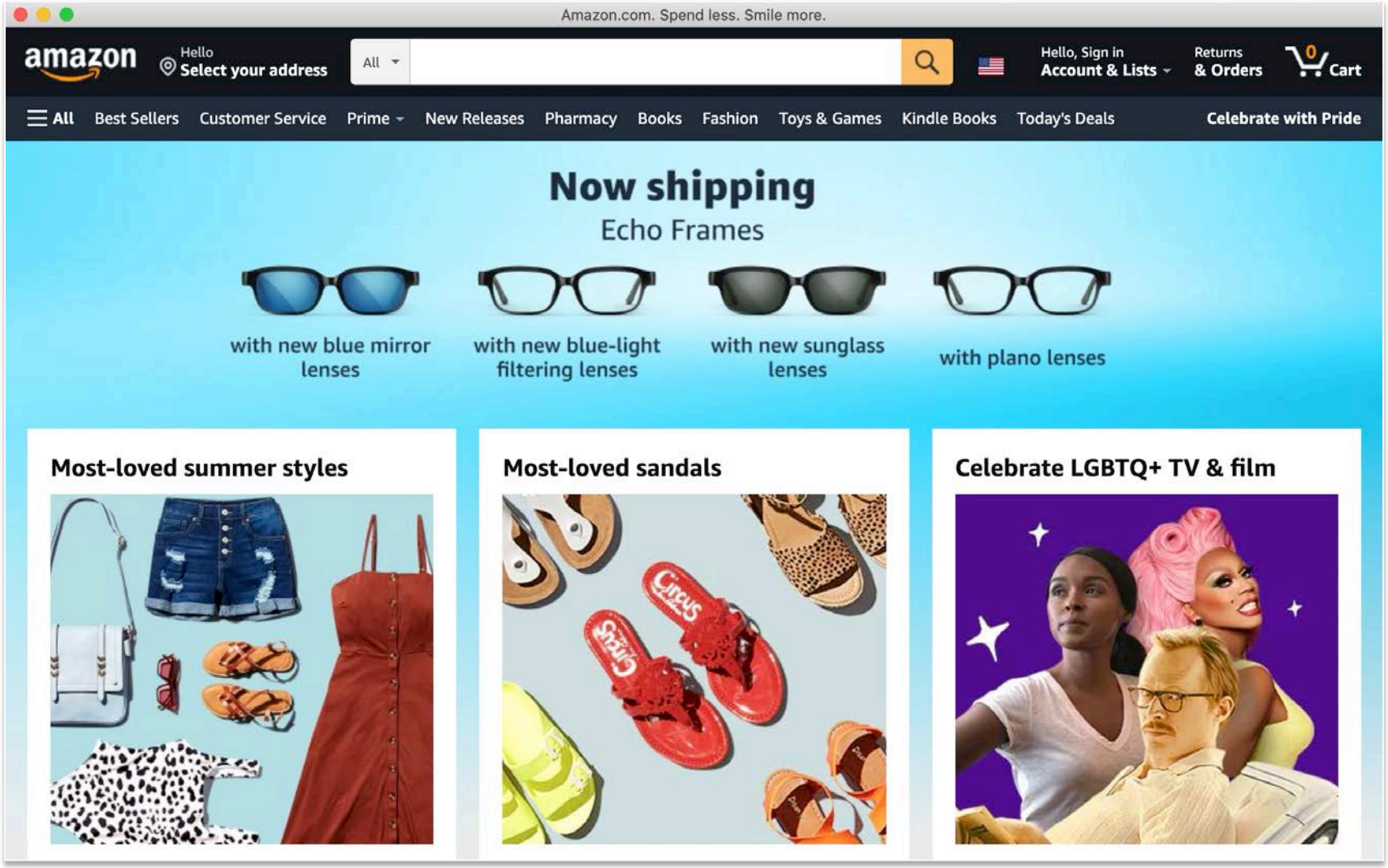
Demo: Click-a-Mole



Demo: Paired Associates



Demo: Amazon


Amazon.com. Spend less. Smile more.
amazon
Hello
Select your address
All
Hello, Sign in
Account & Lists
Returns
& Orders
0
Cart
All
Best Sellers
Customer Service
Prime
New Releases
Pharmacy
Books
Fashion
Toys & Games
Kindle Books
Today's Deals
Celebrate with Pride
Now shipping
Echo Frames
with new blue mirror lenses
with new blue-light filtering lenses
with new sunglass lenses
with plano lenses
Most-loved summer styles
Most-loved sandals
Celebrate LGBTQ+ TV & film

ACT-R + Web-Based Tasks

- If only these "real site" models were so easy… but, as you might suspect, there are many challenges with real web sites
  - Overloaded pages (e.g., JavaScript-heavy)
  - Clock synchronization (or lack thereof)
  - Inference of visual areas and types (e.g., what's a button vs. text vs. input vs. other?)
  - Security constraints for reading/modifying site pages
- For now, we're taking baby steps on a need-to-solve basis

Cognitive Code + Web-Based Tasks

- We've also been exploring an alternative modeling approach of "*cognitive code*"
  - Instead of a specialized production-system language, allow models to be built in common programming languages
- Latest instantiation: *Think*
  - Modules inspired primarily by ACT-R (and working in similar ways – e.g., same memory equations)
  - Code written in Python – no production rules
  - Idea: can we can get most of the benefits of modeling with this scheme, without the difficulty of a specialized language

Cognitive Code + Web-Based Tasks

- For example, here's a *Think* model of behavior in the paired-associates task…

```
visual = self.vision.wait_for(isa='word')
word = self.vision.encode(visual)
chunk = self.memory.recall(word=word)
if chunk:
    self.motor.type(chunk.get('digit'))
visual = self.vision.wait_for(isa='digit')
digit = self.vision.encode(visual)
self.memory.store(word=word, digit=digit)
```

Wait for & encode the word

Try to recall the associated digit & respond if recalled

Wait for & encode the digit

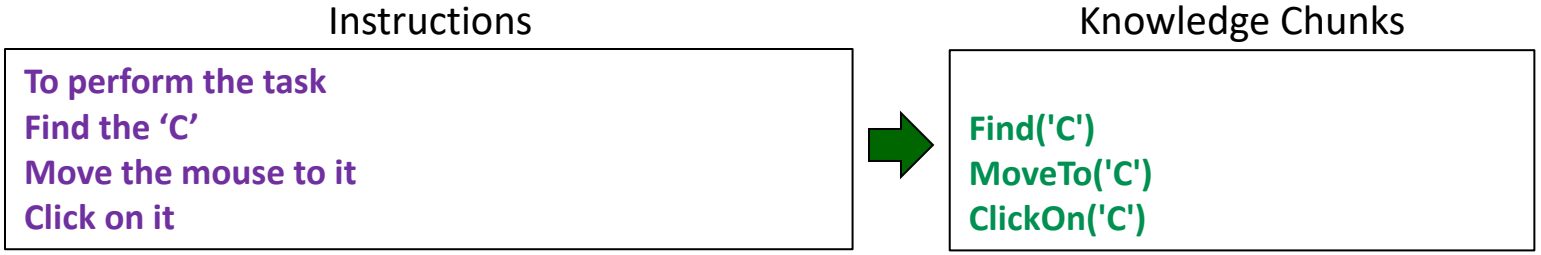
Store the word-digit association

Learning from Instructions

- We have been working on efforts to extend instruction learning in cognitive code…

Learning from Instructions

- 1. Instruction Grounding
  - Grounding similar words to the same object
    - "**Wait for a digit**" … "**Type the number**"
    - The model doesn't know what "**number**" refers to…
      so checks memory for related words, like "**digit**"
  - Grounding pronouns (or generally, anaphora resolution)



  - (see Toth, Taatgen, Hendriks, & van Rij, ICCM '21, for related complexities)

Learning from Instructions

- 2. Instruction Inference
  - Inference of unclear or missing steps ("gap filling")
  - For many actions, execution has implicit preconditions — we can infer these

Instructions v1

To perform the task
Find the 'C'
Move the mouse to it
Click on it
Repeat

Instructions v2

To perform the task
Find the 'C'
Click on it
Repeat

Instructions v3

To perform the task
Click on the 'C'

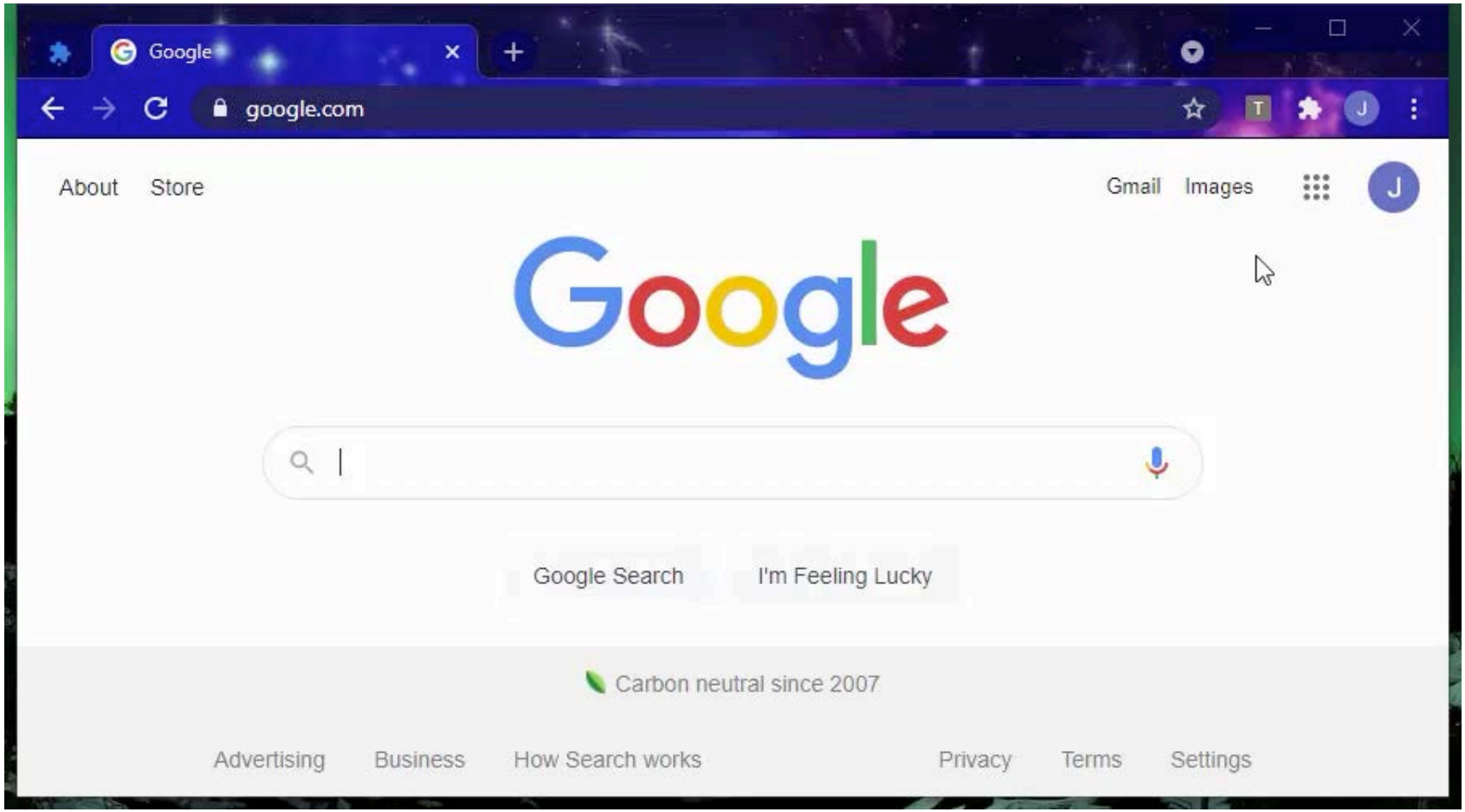
Learning from Instructions

- 3. Interactive Learning & Execution
  - Interpretation: "**Wait for a stimulus**" … "**Type the digit**"
    - "**stimulus**" and "**digit**" are not synonyms/related
    - Learner: "**What's the digit**?" → Teacher: "**the stimulus**"
      → Learner proceeds
  - Execution: ClickOn(button), but there's no button
    - visual find operation fails to find the expected object
    - Learner: "**Where's the button**?" → Teacher: "**this**" <*pointing*>
      → Learner proceeds

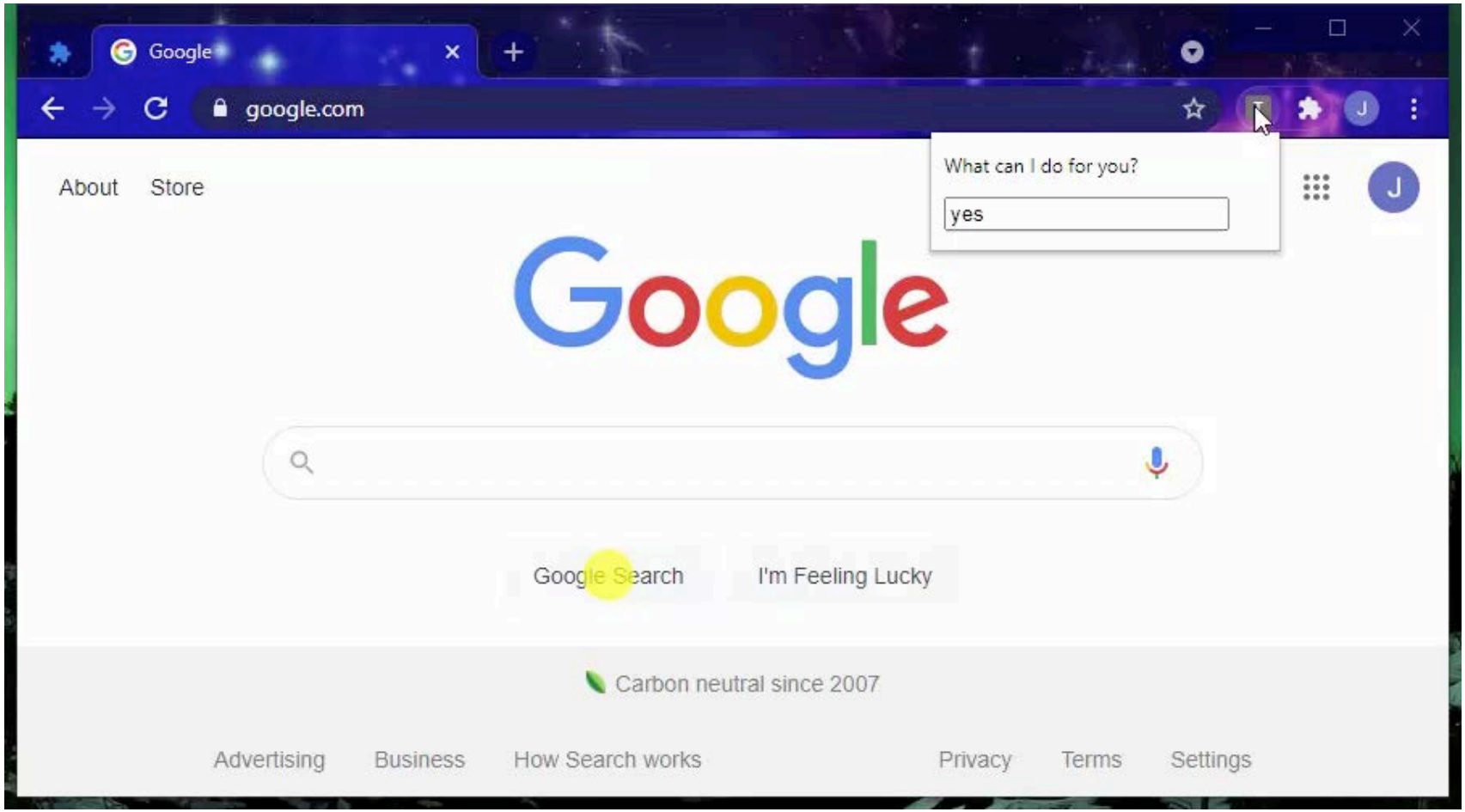
Teachable Agent Prototype

- We (mostly Jenn) are embedding our prior instruction-learning code into a web-based agent with a few new layers…
  - Chrome browser
    - Handles any possible web page
  - Chrome extension within the browser
    - Communicates information between the browser page and the Think architecture (e.g., visual objects, user actions)
  - Think model running as a Python server
    - Communicates with the extension via WebSockets

Teachable Agent Prototype



Teachable Agent Prototype

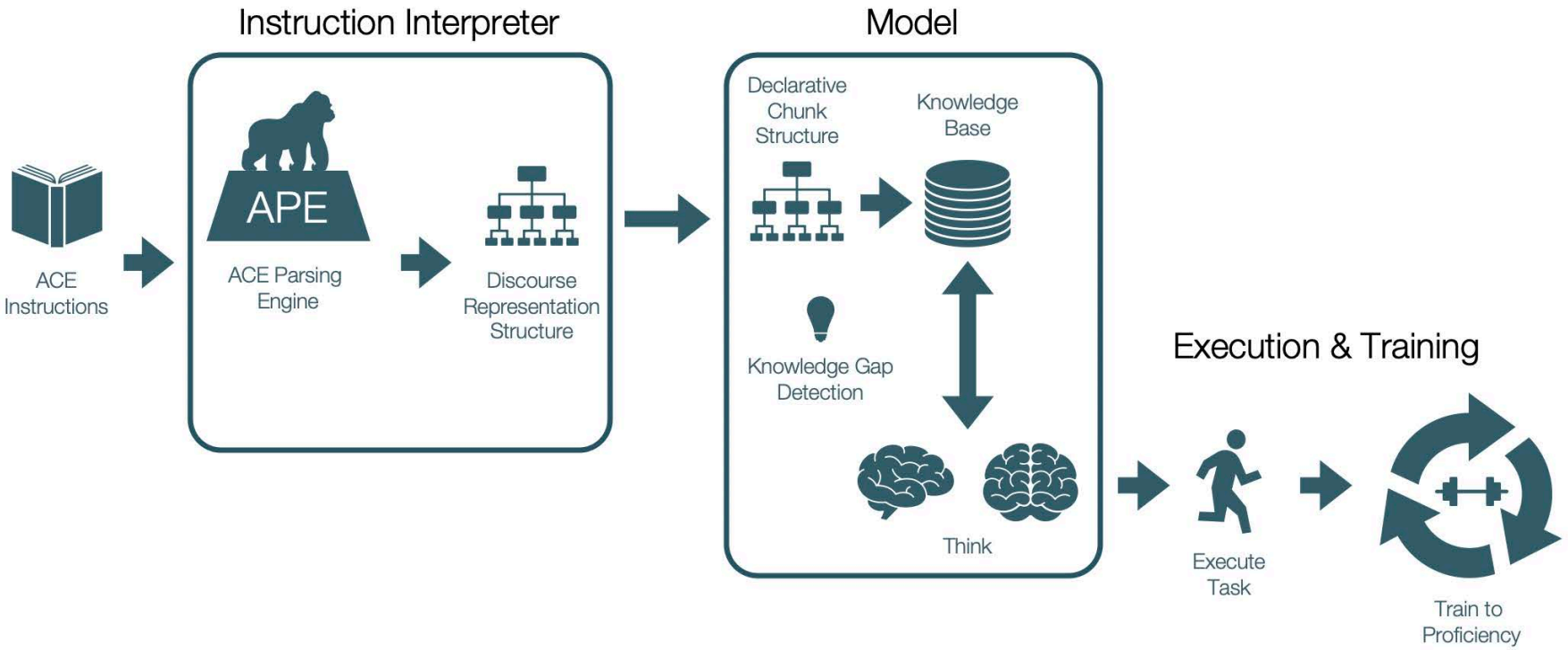


The Big Question

- Can building such an agent with a cognitive architecture really make for a better agent?
  - We think so, but we don't know yet
- Specifically: Is it important for a teachable agent to act in a human-like way?
  - Possible links just to this year's ICCM…
    - Toth, Taatgen, Hendriks, van Rij, 2021: language references
    - Chiarelli, 2021: self-explanation for users and/or agent
    - Halverson, Myers, Gearhart, Linakis, Gunzelmann, 2021: stressors
- We're looking into possible experiments to explore this
- We've also started discussions with Chris MacLellan @ Drexel to fuse ideas from Soar, intelligent tutoring, etc.

Related Work

- This is all part of an ongoing project on "Undifferentiated Agents" (w/ AFRL, Kansas State)



Thank you!

- Thanks for listening
- And thanks to our research sponsor:

This work was funded by a grant from the
Air Force Office of Scientific Research (#FA9550-18-1-0371).