Extending the Reach of Cognitive Architectures

Dario Salvucci
Drexel University

with support from

Cognitive Architectures

- Many major successes…
  - several major architectures
  - leaps in scientific understanding of cognition
  - models of hundreds of domains
  - real-world applications (tutoring, gaming, driving)
- … but still a limited user base.

- How can we extend their reach?

Extending Cognitive Architectures

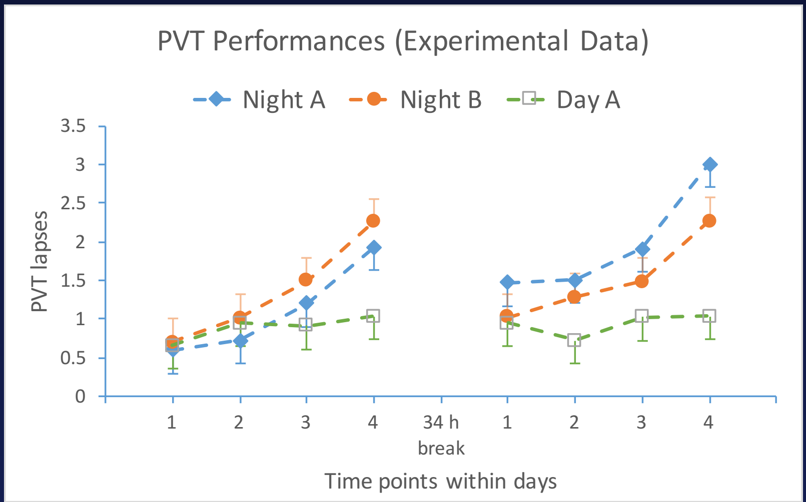
- Extend functionality with new models
  - General processes (e.g., list memory & recall, analogy)
  - Large-scale knowledge bases
  - Natural language processing rules
  - Fake news filter :)
- Extend functionality with architectural components
  - Production-rule learning (compilation, utility learning, …)
  - Metacognition
  - Threaded cognition
  - Physiological moderators (fatigue, caffeine, affect, …)
  - Fatigue modeling …

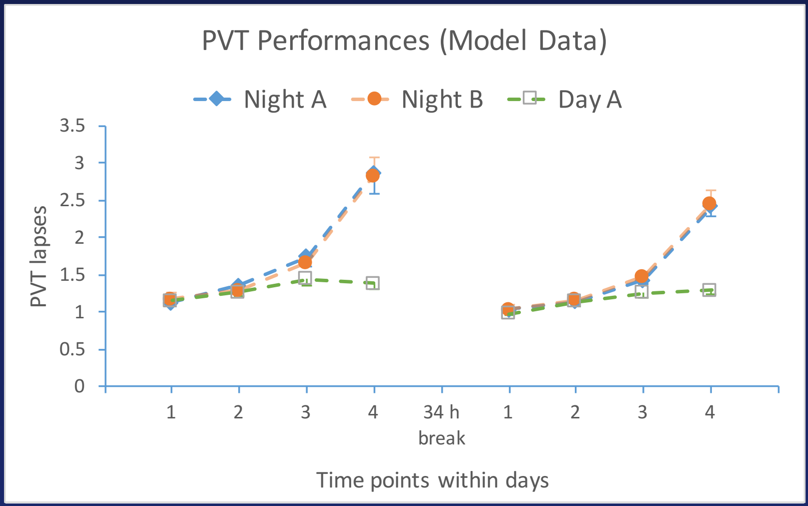
Fatigue & Driving

- Experiment (Forsman et al., 2013; Van Dongen et al., 2010, 2011)
  - 41 drivers in a driving simulator
  - Conditions comparing Day- vs. Night-shift driving

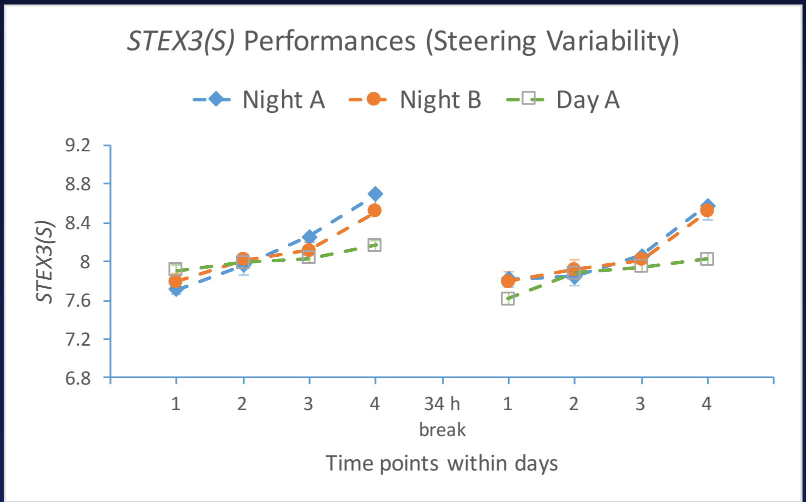
Fatigue & Driving

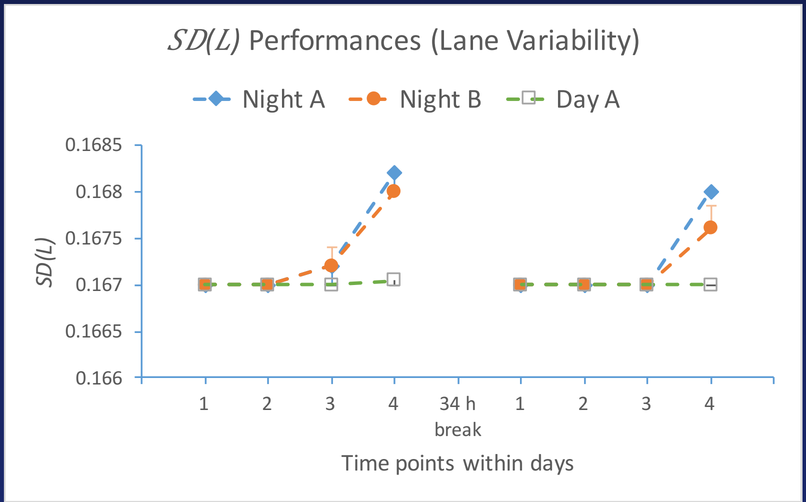
PVT





Driving





(Khosroshahi, Salvucci, Veksler, & Gunzelmann, 2016)

Extending Cognitive Architectures

- Extend computational paradigms for modeling
  - Production systems have been the dominant paradigm
    - primary benefits: flexible for learning, modular for integration
  - But in other places, production systems have largely disappeared, especially from CS curricula
    - and thus, we have a generation of students/programmers who have never learned or seen them
  - Their proposed benefits have largely not been realized
    - flexible for learning? — somewhat
    - good for integration? — hasn't really happened

Production Systems

- ACT-R code for…
  - storing information about Whiskers the cat
  - recalling this information
- Someone new to cognitive modeling will need hours (maybe days) to learn enough to understand this

```
(p rule1
    …
==>
    +imaginal>
        isa cat
        name Whiskers
        owner Jane
    -imaginal>
)

(p rule2
    …
    ?retrieval>
        state free
        buffer empty
==>
    +retrieval>
        isa cat
        owner Jane
)

(p rule3
    =retrieval>
        isa cat
        name =name
==>
   …
)
```

Cognitive Code

- Instead, here's a cognitive code version…

Java

```
memory.store(new Chunk("cat").set("owner", “Jane”).set("name", "Whiskers"));
Chunk chunk = memory.recall(new Query(“cat").add("owner", "Jane"));
```

Python

```
memory.store(m.Chunk({‘isa’: 'cat', 'name': 'Whiskers', 'owner': 'Jane'}))
chunk = memory.recall(core.Query('cat').eq('owner', 'Jane'))
```

- All college CS students (and many younger students) could read a tutorial and understand this in minutes
- They already know the syntax/semantics of the language, and can focus on understanding the cognition part
- (e.g., what happens if the chunk isn't recalled? — hard to guess in ACT-R, easy to guess in Java)

Cognitive Code

- Core simulation system
  - central Clock maintained across threads
  - threads can delay simulated time as needed:

```
agent.wait(1.0);
memory.store(new Chunk("cat"));
agent.wait(2.0);
Chunk chunk = memory.recall(new Query("cat"));
```

    - (as usual, simulation time ≠ real time)
  - system also includes various helper classes (modules, workers, buffers, items, …)

Cognitive Code

- Easy to use just one module
  - e.g., what's the recall probability/time if… ?

```
Agent agent = new Agent();
Memory memory = new Memory(agent, decay);
memory.setActivationNoise(0.5);
Chunk chunk = new Chunk("cat").set("name", "W
memory.store(chunk);
agent.wait(.5);
memory.rehearse(chunk);
agent.wait(2.5);
memory.rehearse(chunk);
agent.wait(3.7);
memory.rehearse(chunk);
agent.wait(6.2);
…  memory.computeProbabilityOfRecall(chunk)
…  memory.computeTimeToRecall(chunk)
```

Create memory and chunk

Simulate timing of rehearsal, waiting

Calculate results

Cognitive Code

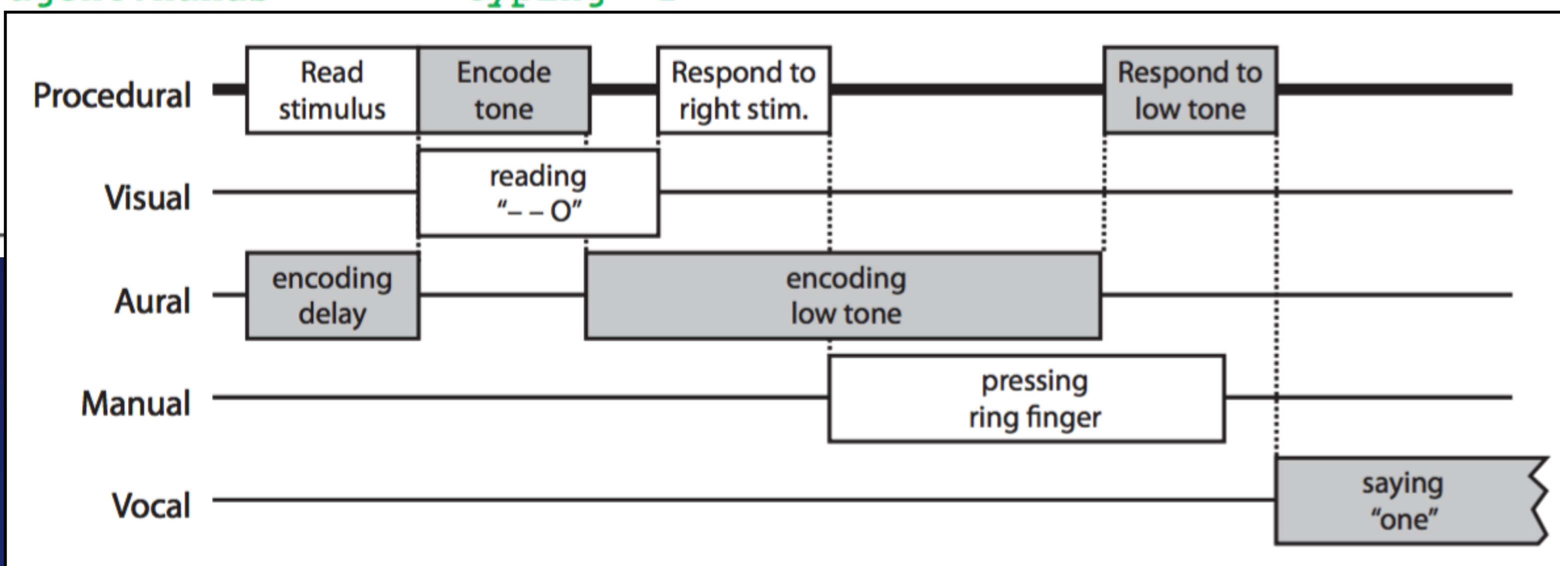
- Dual-choice task

```
agent.run(() -> {
    Object tone = audition.encode(audition.waitFor(new Query("tone")));
    if (tone.equals("low"))
        speech.say("low");
});
agent.run(() -> {
    Object stimulus = vision.encode(vision.waitFor(new Query("stimulus")));
    if (stimulus.equals("O--"))
        typing.type("1");
});
agent.wait(1.0);
vision.add(new Visual("stimulus", 10, 10, 10, 10), "O--");
audition.add(new Aural("tone"), "low");
agent.waitForAll();
```

Cognitive Code

- Dual-choice task

```
0.000   agent             wait for {isa=stimulus}
0.050   agent             wait for {isa=tone}
1.000   agent.vision      found {isa:"stimulus" x:10 y:10 w:10 h:10 seen:false}
1.000   agent             encode {isa:"stimulus" x:10 y:10 w:10 h:10 seen:false}
1.050   agent.audition    found {isa:"tone" heard:false}
1.050   agent             encode {isa:"tone" heard:false}
1.185   agent.vision      encoded O--
1.185   agent             type "1"
1.235   agent.hands       typing "1"
1.385
1.385
1.435
1.444
1.785
```

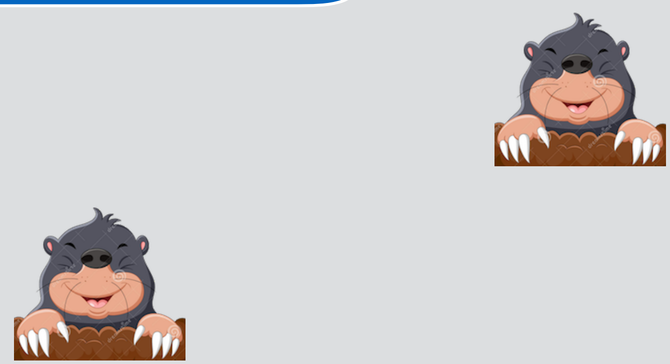


Click-a-Mole

Click-a-Mole

Game parameters

- Mole size (larger = easier)
- Mole time (longer = easier)



Click-a-Mole

```
agent = new Agent().setLog(OUTPUT);
vision = new Vision(agent);
hands = new Hands(agent);
mouse = new Mouse(hands, vision).setRandomizeTime(true).addClickListener((visu
	clickMole();
});

…

agent.run(() -> {
	while (agent.getTime() < RUN_TIME) {
		Visual visual = vision.waitFor(new Query("mole"));
		mouse.pointAndClickMouse(visual);
	}
});

agent.run(() -> {
	while (agent.getTime() < RUN_TIME + 3.0) {
		moleVisible = true;
		int x = Utilities.random.nextInt(500);
		int y = Utilities.random.nextInt(500);
		Visual mole = new Visual("mole", x, y, moleSize, moleSize);
		vision.add(mole, "X");
		agent.wait(moleTime);
		moleVisible = false;
		vision.clear();
		agent.wait(1.5 - moleTime);
	}
});

agent.waitForAll();
```

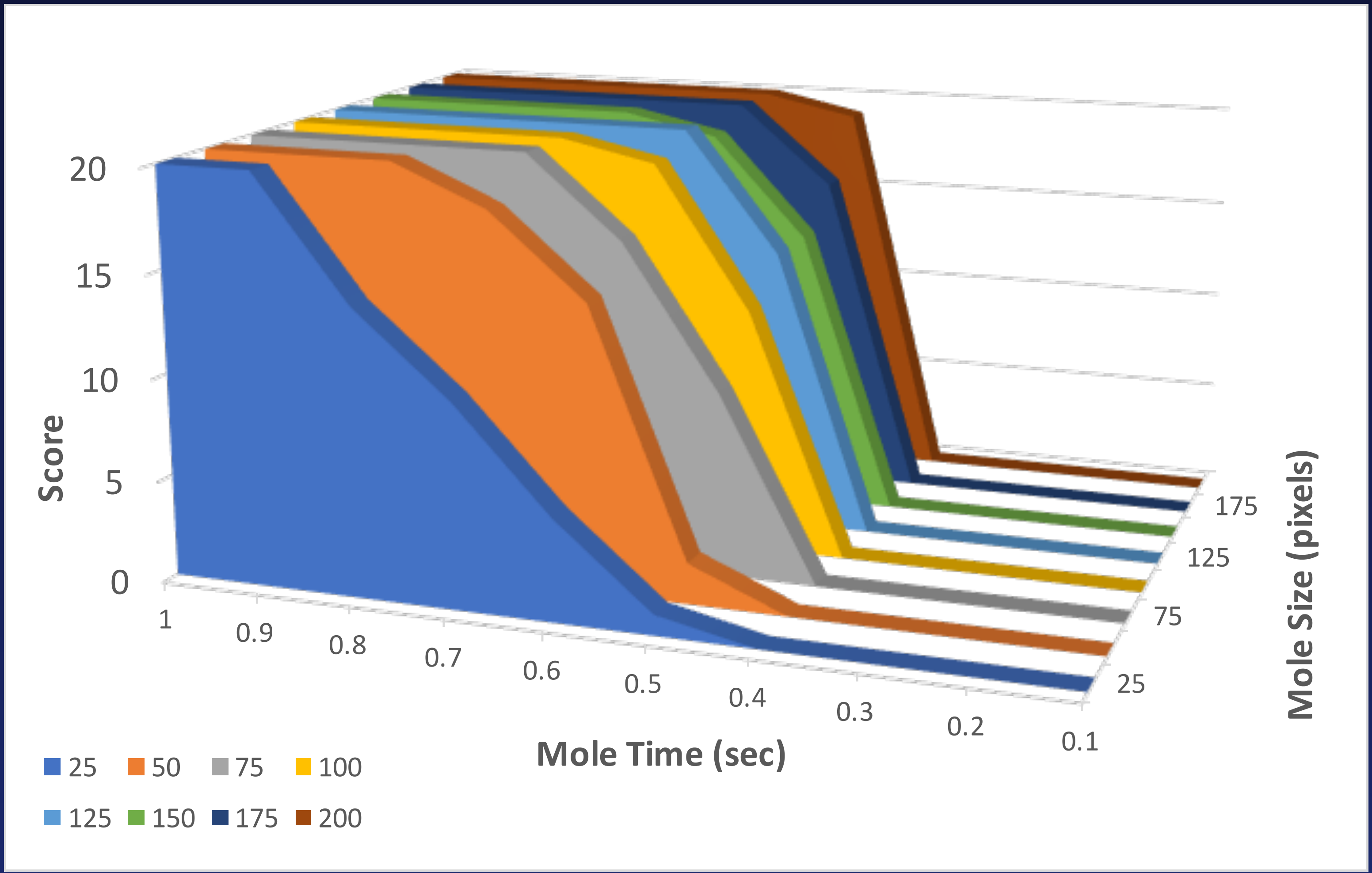
Setup

Model

Task

Click-a-Mole

- Score vs. mole size & time in a 30-second game…



Extending the Reach…

- (1) Production rules are a huge limiting factor.
  - Can we maintain 90% of the theory without production rules, and make ACT-R much more accessible?
  - Current "cognitive code" prototypes in Java and Python are heading in this direction
- (2) We need to focus more on out-of-the-box predictions.
  - Most people don't have data and won't do model fitting
  - What can we give them such that, within minutes, they can make predictions about their domain of interest?
    - Give the model a {game board, essay, car interface, …} the model will give you predicted behaviors + analysis