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 Performances (Experimental Data)

<table>
<thead>
<tr>
<th>Time points within days</th>
<th>PVT lapses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

STEX3(S) Performances (Steering Variability)

<table>
<thead>
<tr>
<th>Time points within days</th>
<th>STEX3(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.8</td>
</tr>
<tr>
<td>2</td>
<td>7.2</td>
</tr>
<tr>
<td>3</td>
<td>7.6</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>8.4</td>
</tr>
<tr>
<td>6</td>
<td>8.8</td>
</tr>
<tr>
<td>7</td>
<td>9.2</td>
</tr>
</tbody>
</table>

PVT Performances (Model Data)

<table>
<thead>
<tr>
<th>Time points within days</th>
<th>PVT lapses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.166</td>
</tr>
<tr>
<td>2</td>
<td>0.1665</td>
</tr>
<tr>
<td>3</td>
<td>0.167</td>
</tr>
<tr>
<td>4</td>
<td>0.1675</td>
</tr>
<tr>
<td>5</td>
<td>0.168</td>
</tr>
<tr>
<td>6</td>
<td>0.1685</td>
</tr>
<tr>
<td>7</td>
<td>0.17</td>
</tr>
</tbody>
</table>

SD(L) Performances (Lane Variability)

<table>
<thead>
<tr>
<th>Time points within days</th>
<th>SD(L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.166</td>
</tr>
<tr>
<td>2</td>
<td>0.1665</td>
</tr>
<tr>
<td>3</td>
<td>0.167</td>
</tr>
<tr>
<td>4</td>
<td>0.1675</td>
</tr>
<tr>
<td>5</td>
<td>0.168</td>
</tr>
<tr>
<td>6</td>
<td>0.1685</td>
</tr>
<tr>
<td>7</td>
<td>0.17</td>
</tr>
</tbody>
</table>

(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

```r
(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

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

Python

```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:

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

- (as usual, simulation time $\neq$ 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...?

```java
Agent agent = new Agent();
Memory memory = new Memory(agent, decay);
memory.setActivationNoise(0.5);
Chunk chunk = new Chunk("cat").set("name", "Whiskers");
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

```java
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
Click-a-Mole
Click-a-Mole

Game parameters
- Mole size (larger = easier)
- Mole time (longer = easier)
Click-a-Mole

```java
agent = new Agent().setLog(OUTPUT);
vision = new Vision(agent);
hands = new Hands(agent);
mouse = new Mouse(hands, vision).setRandomizeTime(true).addClickListener((visual) -> {
    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();
```
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