Cognitive Architecture in Mobile Music Interactions

MichiganEngineering
Computer Science & Engineering

Nate Derbinsky (nlderbin@umich.edu) Georg Essl (gessl@eecs.umich.edu)

University of Michigan School of Music, Theatre & Dance
department of PERFORMING ARTS TECHNOLOGY

Abstract

How can cognitive architecture facilitate the development and exploration of interactive music interfaces on mobile platforms?

We…
- integrated Soar with UrMus
- implemented four demonstration agents that use diverse learning mechanisms in two mobile music interfaces
- evaluated real-time interactivity on modern mobile devices

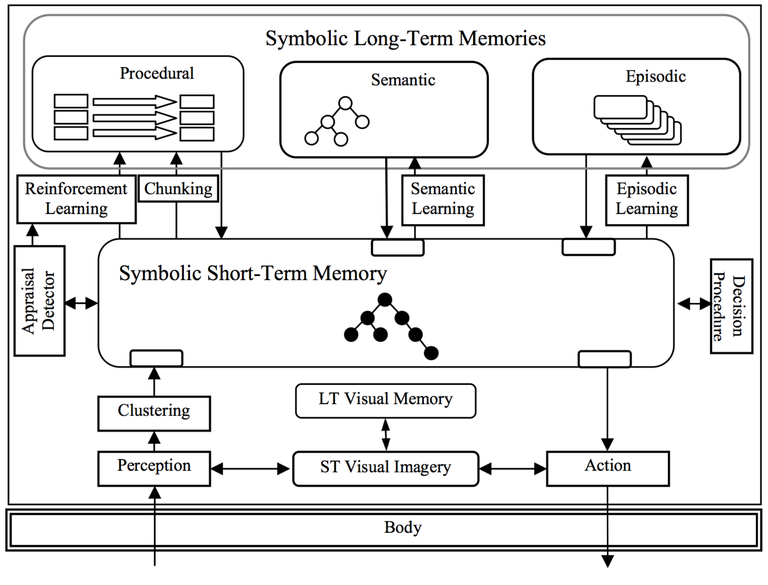
Cognitive Architecture

Aims to develop and understand human-level intelligence across a diverse set of tasks and domains

Specification of fixed aspects of cognition
- Short/Long-term memories of agent beliefs, goals, and experience
- Representation of agent knowledge
- Functional processes that apply knowledge to produce behavior
- Learning mechanisms that adapt knowledge over time

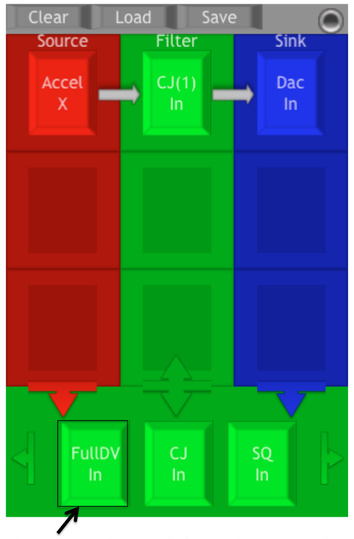
Potential benefits for music performance
- Improved interactions with a learning system ala cognitive intuition
- Flexible source of integrated, optimized machine learning

Soar (Laird 2008)



- Used extensively for AI systems and modeling human cognition
- Efficiently brings large knowledge sources to bear on decisions
- Supports many programming languages (C++, Java, Python, …) on numerous software/hardware platforms (Windows, Mac, Linux; games, robotics, mobile devices, …)

UrMus (Essl 2010)



- Multi-layered environment on multi-touch mobile devices
  - interface design
  - interaction design
  - Interactive performance
- Flexible system to receive input and organize visual and other content in response
- Supported on iOS and Android platforms

Region. Organizing element for touch input and visual output

Integration

- Each region *can* have an instance of Soar
- Potential for multiple, independently actuating objects and cognitive decision models
- Minimal Lua interface to C++ Soar kernel:

Loading Rules

```
r = Region()
r:SoarLoadRules("simon-rl", "soar")
```

Managing Perception & Run Control

```
t = r:SoarCreateConstant(0, "time",
                         clicks)
r:SoarExec("step "..delayDecisions)
r:SoarDelete(t)
```

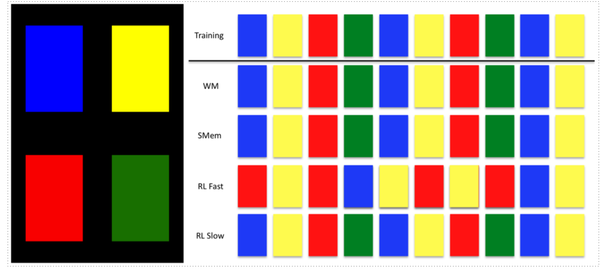
Processing Actions & Proprioceptive Feedback

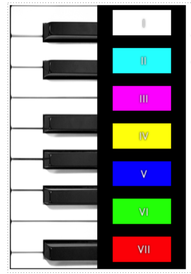
```
name, params = r:SoarGetOutput()
result = params.output
r:SoarSetOutputStatus(1)
```

Simulation Conclusion & Reinitialization

```
r:SoarFinish()
r:SoarInit()
```

Demonstration Interfaces





Training		Output 1		Output 2	
Chord	Melody	Chord	Melody	Chord	Melody
I	EGGE	I	GGGC	I	GGGC
V	DGGD	V	CGGG	V	CGGG
VI	CEEC	VI	CCEC	VI	CCEC
III	BEEB	V	DCDD	V	DCDD
IV	ACCA	VI	ECEC	VI	ECEC
I	GCCG				
IV	ACCE				
V	DCDD				

Soar Run-time Evaluation on iPod Touch
- 0.199 seconds total CPU time (maximum)
- 1 millisecond per decision (maximum)

Diverse Learning Methods

Example instances of the three memory models we used.

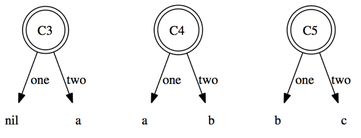
1. Working Memory Model

Agent directly encodes exact musical input sequence.

Blue -> Yellow -> Red -> Green -> Blue …

2. Semantic Memory Model

Agent encodes musical bi-grams; retrievals are biased by recency.



3. Procedural Memory Reinforcement Model

Agent iteratively learns musical bi-gram utility from user interaction.

```
If
  the current chord is C-E-G AND
  the last note played was G AND
  the agent is considering note C
Then
  the expected value is  0,-8…   -16,…
```

Agent Rehearsal

User Interaction