Chapter 2

Preliminaries

In this chapter, we present a background on topics relevant to the remaining chapters. First, we will introduce a cognitive architecture, ACT-R, and provide information on how this framework uses computational techniques to model cognition and behavior. We will also provide context on heuristics in decision making from the cognitive science literature.

2.1 ACT-R

ACT-R (Adaptive Control of Thought-Rationale) is an architecture that provides a framework for developing agents that mimic human perception, learning, and memory [12]. It includes modules that represent various cognitive functions such as declarative memory, procedural memory, visual attention, and more [27]. Modules store information that is known to the cognitive agent inside of buffers. For example, the declarative memory module contains a retrieval buffer that stores memories recently retrieved. The visual buffer stores information about what the agent is currently attending to visually. Modules can communicate a limited amount of information with one another through the buffers[12].

Using the ACT-R framework, researchers have successfully modeled and predicted

human behavior for a range of applications in psychology, human-computer interaction, education, and neuroscience [28, 29]. The architecture has also been used in many human-machine interaction tasks, such as developing interactive virtual tutors [30] and teaching robots about human behavior [31].

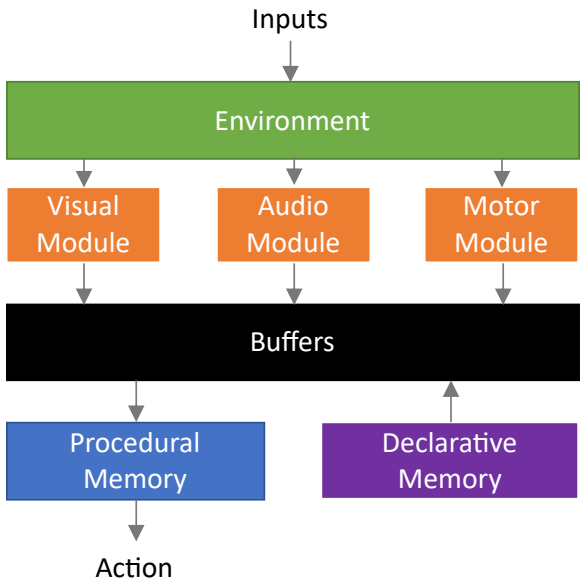

Figure 2.1: Components of the ACT-R architecture. Input is gathered through various modules, while information known to the agent is stored in declarative memory. Information that is currently being attended to is stored in the buffers. Procedural memory uses the information available in the buffers to choose a specific action.

2.1.1 ACT-R Components

ACT-R is made up of several components (see Figure 2.1). Modules such as the Visual, Audio, and Motor modules collect information from the environment. We discuss the Audio module in more detail in Section 2.1.4. When a sight or sound is actively being perceived and attended to, this item is stored in a buffer associated with its module. Other buffers are also available for storing information related to the current situation. For example, a Goal Buffer stores information about the agent's current goals. The Imaginal Buffer stores other task-relevant details that

are important for the agent to choose its actions. The Declarative Memory and Procedural Memory modules handle how agents store information and learn from it. These modules are described in more detail in the following sections.

2.1.2 Declarative Memory

ACT-R's declarative memory module stores facts that the model knows as instances in memory. Each instance includes the data associated with a particular fact and an activation value that represents how difficult it is to remember. The activation value is calculated based on how often the fact has been retrieved and how recently. More details about this calculation are provided below. When the activation value associated with a particular instance is high, it is more likely to be retrieved. If the activation value is below the amount specified by a *retrieval threshold* parameter, then it cannot be retrieved at all.

As an example, consider a cognitive agent that is asked to recall the sum of 2 + 2. The agent will make a retrieval request to the declarative memory module. If a chunk exists in memory that contains this fact, there is a possibility it will be retrieved. However, if the activation value is too low (if the fact hasn't been retrieved recently or often), then the fact will not be retrieved. However, if the activation value is high enough, then the model will successfully retrieve the sum of $2+2$ and provide a response. Now we discuss how the activation value is calculated.

Activation The activation value of a memory instance i is made up of four components, including base-level learning (B_i), partial matching (P_i), spreading activation (S_i), and noise (ϵ_i).

$$A_i = B_i + P_i + S_i + \epsilon_i \tag{2.1}$$

Below, we briefly describe the components of activation.

Base-Level Learning Base-level learning (B_i) affects the final activation value based on the recency of a particular memory and how frequently it is retrieved.

$$B_i = \ln \sum_{j=1}^{\infty} t_j^{-d} \tag{2.2}$$

Here, n refers to the number of past references to instance i, t_j refers to the time since the jth reference, and d refers to the decay rate and is set to a default value of 0.4. This equation causes instances that have been recently retrieved or frequently retrieved to have a higher activation value.

Partial Matching Partial matching controls how similar instance i must be to the current retrieval request in order to be retrieved.

$$P_i = \sum P M_{li} \tag{2.3}$$

Here, l represents each attribute in instance i. P is the mismatch parameter, M_{li} refers to the distance between l and the corresponding attribute instance i. If i and l are the same, then this value is 1; otherwise, it is 0. For example, consider that we have stored in memory an instance with the following attributes: entertainment=videogames, hair=blonde, hobby=drawing. We could then make a retrieval request for an instance with the attributes: entertainment=sports, hair=blonde, hobby=drawing, and the P_i value would be 2 since two of the three attributes are the same. The more attributes in common, the higher the activation value, and the more likely it is to be retrieved.

Spreading Activation Spreading activation is used to model the effect of context on memory retrieval [32] and is an important component in the models described in this chapter. When the instance has attributes in common with the current situation, the S_i weights are applied to the shared attributes increase, as described in Equation

2.4 below, raising the instance's activation and likelihood that memories with these will be retrieved [12].

$$S_i = \sum_{j=1}^{n} W_j S_{ji} \text{ where } W_j = 1/n \text{ and } S_{ji} = S - \ln(fan_j) \tag{2.4}$$

In the above equations, W_j refers to attentional weights applied to each attribute in a given instance. By default, these weights are spread evenly among all n attributes. The variable fan_j refers to one more than the number of chunks in declarative memory that contain an attribute that is the same as j. The variable S_{ji} represents the spreading activation between an attribute in declarative memory and a matching attribute in the retrieval request.

Finally, since human memory retrieval does not always lead to remembering the same memory every time, noise is added to the model to simulate memory's stochastic nature. Noise is generated according to a logistic distribution with a mean of 0 and variance, σ^2, relating to the parameter s as follows.

$$\sigma^2 = \frac{\pi^2}{3} s^2 \tag{2.5}$$

The four components of activation work together to simulate the memory retrieval process that is important in modeling many learning and memory tasks.

2.1.3 Procedural Memory and Utility Learning

The procedural memory module represents ingrained or learned rules and heuristics that are known to the agent and describe how it will respond to the environment. The procedural module uses *procedural rules* to compare what is in the ACT-R buffers with the conditions defined by the modeler. If they match, then that rule is valid in the given context. If multiple production rules match the information in the buffers, then any of the rules are valid, and ACT-R must decide which rule to follow. The

probability P_i that ACT-R will select a rule from among a set of matching ones can be calculated using the Boltzmann equation (see Equation 2.6). By using a probabilistic approach to memory retrieval, ACT-R ensures that the agent can explore the decision space and possibly find other, more preferable paths to success [33].

$$P_i = \frac{e^{U_i/\sqrt{2}s}}{\sum_j e^{U_j/\sqrt{2}s}} \tag{2.6}$$

Utility values (U_i) are assigned to production rules to represent whether or not they are useful for the current task. After receiving feedback about a decision, the decision maker updates the utility of the rules that led to that outcome [8]. At iteration n, these values are

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \tag{2.7}$$

Here, α refers to the learning rate and has a default value of 0.2. When the model arrives at a state where it receives an external reward (specified by the reward parameter), then each production rule i will receive a time-discounted reward $R_i(n)$, which is the total external reward minus the amount of time that passed since production i was selected. This results in lower rewards applied to production rules that occurred long before the reward was received.

2.1.4 ACT-R Audio Module

The Audio Module provides basic functionality for responding to sounds in the environment. Figure 2.2 illustrates how the module integrates into ACT-R. When a sound becomes available to the agent's environment, it appears as a sound event in the audio module's Audicon, which is a data representation of all sounds that an agent can attend to. Sounds in the Audicon exist in the environment, but have not been perceived or attended to by the agent. Sound events are represented in the Audicon

as a set of attributes, including the type of sound, its content, and if its internal or external to the agent. Once the sound is available in the Audicon, the ACT-R agent can perceive and attend to it by retrieving it from the Audicon and storing it in the Aural-Location buffer. Unless specified by the modeler, the Aural-Location buffer can only store one sound at a time. At this stage, the agent can be said to have perceived the sound, but not yet encoded it. The information is encoded as a symbolic representation of the sound stored in the Aural Buffer.

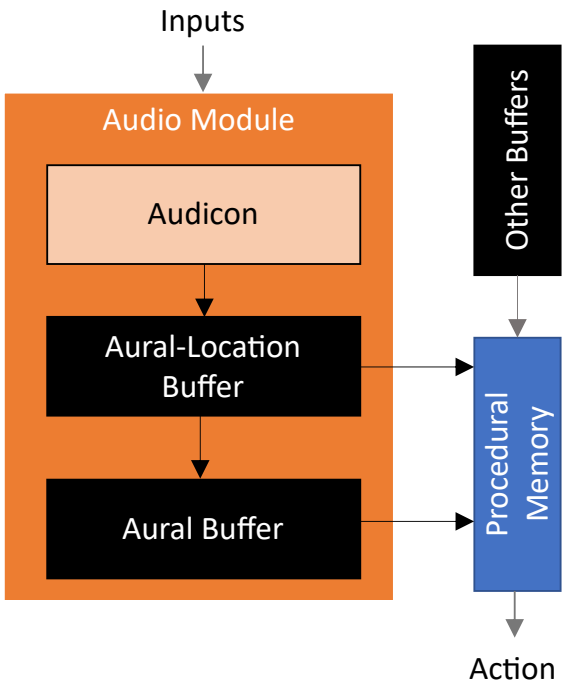

Figure 2.2: Visual depiction of how the audio module communicates with the ACT-R buffers.

2.2 Heuristics in Decision Making

We will now provide some background on heuristics in decision making. Heuristics are strategies, or adaptive shortcuts, that humans use to make decisions. This idea originated from the study of bounded rationality [34], or the observation that both the human mind and the environment make application of normative decision models

impossible. Heuristics have been studied in-depth in many fields; see Hilbig [35] for an overview. Though sometimes seen as second-best alternatives when maximization is not possible, other research views heuristics as ideally adapted strategies that capitalize on the structure of the environment to provide solutions when optimization is not an option, e.g., NP-hard problems, ill-defined problems, or unfamiliar/time-sensitive problems. In many situations, heuristics have been shown to outperform solutions that use more complex algorithms. For example, in DeMiguel et al. [36], researchers found that 3000 months of stock market data would be required before a sample-based mean-variance strategy would outperform a heuristic strategy to evenly divide funds across all stocks in a small 25-asset portfolio.

Key to the idea of heuristics is that they are fast, composed typically of three steps: search, stop, and apply decision rules. They are also frugal, ignoring some of the casually relevant information. Proponents of fast and frugal heuristics have promoted the idea of ecological rationality, which examines the rationality of a decision strategy in its environment [37]. Many heuristics have been identified in decision-making. For example, *satisficing* is one such heuristic that is when a set of alternatives are presented sequentially, and not much is known about future alternatives. In this scenario, the decision-maker chooses the first one that surpasses some aspiration level [1]. *Take the best* is another heuristic strategy that decision-makers may employ. When using *take the best*, a set of cues is ordered, and then each one is considered in turn until finding one that differentiates between two choices [1].