Give three concrete examples of explanation based on the use of analogy in one or more areas of Cognitive Systems. For each, explain why it is interesting—i.e., how it helps with understanding.

Word Count: 400

Analogies are comparisons between two partially similar concepts, that we use to better understand and explain one of the concepts (“Analogy” 2017). They are powerful tools in Cognitive Systems, as they allow us to forgo using jargon and technical terms when explaining in favor of simpler terms. In this essay, three such analogies will be discussed.

In Java, an exception is an unwanted event that interrupts a program – generally an error (Singh, Chaitanya, et al. 2017). Exceptions can either be handled in the function they occur, or they can be “thrown” to another function to handle (Singh, Chaitanya, et al. 2017). An analogy for exception handling involves throwing a hot potato (the exception) into a group of chefs (Holmes 2001). If the catcher is wearing oven mitts, he will be able to handle the hot potato (handle the exception). Otherwise, he must pass it on to another chef, who may have oven mitts (“throwing” the exception). This analogy illustrates that an exception can be handled or “thrown” by any function. It is effective because it helps us clearly visualize an abstract idea as a common game.

In programming, “recursion” is when a function calls itself repeatedly to solve a problem (Allain 2017). To teach recursion, an analogy is often used involving nested Russian matryoshka dolls, which are porcelain dolls where each doll contains a smaller replica of itself iteratively (“Matryoshka Nesting Dolls” 2015). In this analogy, a matryoshka doll is like a recursive function, as it must be opened repeatedly, just like a recursive function. In a recursive function, a “base case” ends the recursive loop and prevents infinite loops (Allain 2017). Similarly, to avoid opening dolls infinitely, there exists a smallest doll. This analogy is useful because it helps convey the core concept behind all recursion by comparing it to common toy.

The idea of object-oriented design is common in programming (Rouse 2008). It involves planning out a system of interacting “objects”, which can all be grouped into classes (Rouse 2008). A useful analogy for this is the real world and its conceptual hierarchies. In the real world, Phil the human will have the properties and behaviours of a human, and will clearly not be classified as a dog. Likewise, Rex the dog will be classified as dog, not a human. In object-oriented design, these “objects” will have classes like “human” or “dog”, and can be sorted according to their properties (Rouse 2008). This analogy is effective because it compares a new idea to a real-world concept of categories we are very familiar with.

References:


Rouse, Margaret. “What is object-Oriented programming (OOP)? - Definition from WhatIs.Com.” SearchMicroservices, Aug. 2008, searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP.