Symbols of Code

by Sven Nilsen, 2022

In this paper I introduce four properties of systems that makes them programmable.

In the paper "Symbols of Power"[1], I described four properties of systems that have great influence. Using the same approach, I used GPT-3[2] to come up with four properties of systems that could explain the non-trivial relationships symbols/semantics, genes/language and physical laws/universe.

A symbol of code is a system which has the following properties:

1. Can be combined to create new systems (combinatory)
2. Can be modified to change their properties (mutable defineable)
3. Can be changed by replacement with other symbols (replaceable)
4. Can be affected by the contex in which it appears (context dependent)

This is the first time I have used to AI technology to generate descriptions of system properties. The original text needed some adjustement and clarification, but the overall idea was great. The properties above are listed in the same order that GPT-3 generated.

GPT-3 was able to arrive at these properties using only the three examples, listed in the first paragraph, of non-trivial relationships. The challenge was to find a way to describe properties of such systems when the type of relationship was not understood. I believe that the four properties generated captures the essence of programmable systems well.

The idea of defining symbols of code is to generalise the notion of a programmable system. Like symbols of power, symbols of code are intended as a building block for thinking about AGI[3].

Since such systems are defined by their properties, it is easy to check whether some system satisfies the definition by making simplifying assumptions of each property. I go into more detail about this process in the "Symbols of Power"[1] paper.

The most obvious example of a symbol of code, which the name suggests, is computer code. Although computer code is abundant in modern technology, there is some ambiguity between various use of terms for it like "software" or "programming language" etc. These various terms might not capture the essence of what a symbol of code does. By explicitly defining the properties one is reasoning about plus coining a term "symbol of code" that only uses these properties, one avoids the ambiguity that comes with a term like "software".

A symbol of code focuses on the particular properties that makes a system programmable, instead of what programming means in a general sense. For example, written sentences in a text editor might be thought of as a symbol of code. You can combine sentences to create new ones (combinatory), you can change the sentence directly (mutable defineable) or replace it with another sentence (replaceable) and the meaning of a sentence is affected by the context in which it appears (context dependent). With other words, written sentences satisfied all properties that makes a system programmable. However, it is unusual to think about written sentences as a form of programming. This gets clearer when thinking about programming languages, because the source code in a programming language is often constructed out of written sentences. With other words, a programming language "adds something" to written sentences that e.g. compiles and executes. Thus, one can think about programming languages as extending written sentences as a system.

References:

[1] “Symbols of Power”
Sven Nilsen, 2021
https://github.com/advancedresearch/path_semantics/blob/master/papers-wip2/symbols-of-power.pdf

[2] “GPT-3”
OpenAI – Engine APIs
https://beta.openai.com/docs/engines/gpt-3

[3] “Artificial general intelligence”
Wikipedia
https://en.wikipedia.org/wiki/Artificial_general_intelligence