Jon Bowen

CS3210

Exercise #1

A clear definition of “programming language” is elusive. Sebesta’s *Concepts of Programming Languages* omits a definition. All other texts I found on the topic similarly omit a direct definition. Sources that do define the term are generally vague or colloquial. The clearest ones, collected here, generally make reference to instructions or behavior.

PC Magazine defines a programming language as "a language used to write instructions for the computer. It lets the programmer express data processing in a symbolic manner" (https://www.pcmag.com/encyclopedia/term/49829/programming-language). This is a good definition in that it makes reference to both instructions and data, which is essentially all a program is. Importantly, a programming language under this definition cannot consist of just data. The lack of instructions is what distinguished markup languages such as HTML from programming languages. Programming languages must provide a means of causing computer to act on data.

In an ACM article, Andrew Ko begins with the definition of a programming language as a “formal means of specifying a computer behavior.” They “are notations for telling computers what to do in the future” (https://dl.acm.org/citation.cfm?doid=3001878.3001880). This definition is nice because it recognizes that programming languages are “mere” notation. No computer “speaks” or “understands” any programming language. The notation has no agency until the formal specification becomes binary instructions and data that can move through a CPU. In addition, this definition uses the term “computer behavior” whereas the previous definition used “instructions.” This is a higher level view. At the micro level what a program is “doing” at any given moment is executing the instruction that is currently loaded. In most programming languages, the author does not intentionally specify this level of detail. The term “computer behavior” is closer to the intentionality expressed by the author in the language they are writing.

The Encyclopedia Britannica defines a computer programming language as “any of various languages for expressing a set of detailed instructions for a digital computer.” Their definition further distinguishes between low-level language as those “requiring a programmer to manage explicitly all of a computer’s idiosyncratic features of data storage and operation,” and high-level languages as those that “shield a programmer from worrying about such considerations and provide a notation that is more easily written and read by programmers” (https://www.britannica.com/technology/computer-programming-language). Like the previous definitions, the primary component of this one is that the language is a means of specifying the operation of a computer. The distinction between low-level and high-level languages is an interesting one. Assembly language is considered a programming language just as Java is. However, suppose the programmer directly creates manufacturer-specific machine language. Is machine language considered a programming language in a formal sense? It must be because programs written in high-level languages are deterministic and are ultimately converted to this form. There must, then, be a mapping that is 1-1 and from any high-level language to machine language. The alphabet, grammar, and set of legal constructions is certainly harder for a human to use correctly, but it is no less a programming language.

To provide our own definition of a programming language, we resort to formal definitions unlike other sources. A language is a set of strings and a grammar, a set of rule for how those strings can be combined. A programming language is a language, in the above sense, for which there exists a 1-1 and onto map to the formal language, in the same sense, defined by the manufacturer of a computer which causes the computer to take inputs and produce outputs. This definition encompasses the previous definitions mentioned and explicitly recognizes that the computer manufacturer defines the only language that is actually efficacious in producing predictable effects from the action of a CPU.