What is Computation?

Luke Hunsberger and the CMPU-181 Class

April 15, 2013

Computation involves the transformation of input data into output data. To describe computation or to cause a computer (or another person) to carry out some computation requires using a language. In particular, the language must describe the data manipulated by the computation and the computation itself. Syntax and semantics are essential ingredients of any language, whether a spoken language (e.g., English or German) or a programming language (e.g., Scheme or Java). For a programming language, syntax rules specify the legal expressions or statements that constitute a program; semantics rules specify what the legal expressions or statements mean, including the kinds of data or computations they represent. Together, the syntax and semantics of a programming language constitute the computational model of that language.

The Scheme programming language is a *functional* programming language in the sense that the main thing that happens during the execution of any Scheme program is that functions get applied to inputs. The syntax of Scheme specifies the character sequences that form legal *expressions.* For example, the character sequences, `3`, `(+ 2 3)`, and `#t`, are legal pieces of Scheme syntax. The semantics of Scheme specifies the datum that each legal expression denotes, as well as what it evaluates to. For example, the above pieces of syntax respectively denote the number three, a non-empty list containing several items, and the *true* truth value. These data respectively evaluate to the number three, the number five, and *true*. The evaluation of expressions is carried out by the built-in *evaluation function*, which plays a central role in Scheme's computational model. The evaluation of non-empty lists enable Scheme programmers to cause arbitrary functions to be applied to any inputs. Lambda expressions in Scheme enable the programmer to specify functions of his or her own design.