Chapter 1
What is a Computable Model?

As a first approximation, a *computable model* is a mathematical model constructed from data types using operations and relations that are computable relative to those types. But what do we understand by the terms *mathematical model* and *data type* and what is it to be *computable* relative to the latter? As a preliminary to formalization, in this chapter we aim to clarify how we intend to use these notions. The rest of the book will provide mathematical substance to these more informal considerations.

1.1 Mathematical Models

The term *mathematical model* is often used to mean a model of a system built from mathematical structures and their associated notions. Such models are employed throughout engineering, the natural sciences, and the social ones. Typical examples range from models constructed from sets, number systems of various kinds, algebraic structures, especially categories, topological spaces through to probabilistic and statistical models, etc. Very common examples employ the real and complex number systems and, in particular, consist of differential equations such as the following.

\[
m \frac{d^2 f(t)}{dt^2} = - \nabla(g(f(t)))
\]

\[
\frac{dS}{S} = \mu dt + \sigma dX
\]

The first is the model of a particle in a potential field and the second is the Black–Scholes partial differential equation for a derivative price. The exact meaning of the terms involved need not detain us; our only concern is that they illustrate how mathematical notions (in these cases differential equations) are used to model natural or artificial phenomena. This very general notion of modeling partly illustrates how we intend to use the term *mathematical modeling*.

However, our primary use of the term is closer to that found in logic and set theory [5, 11, 13] where sets, relations, and functions (conceived of as sets of
tuples) are the basic building blocks for the construction of mathematical models of axiomatic systems. While this kind of modeling may be seen as a special case of the more general notion, it is distinguished by the central role it gives to sets. For example, in the denotational semantics [19] of programming languages, programs are modeled as set-theoretic functions acting upon some set-theoretic representation of the underlying abstract machine. In formal ontology, a typical representation of time will model instants as certain sets of events [22]. And modal notions such as necessity and possibility are unpacked in terms of sets of possible worlds [4]. Such modeling is ubiquitous in mathematical logic and theoretical computer science. Indeed, if one takes set theory as a foundation for mathematics, to which everything can be reduced, then ultimately all mathematical models are set-theoretic.

In very general terms, we shall often follow the structure of these set-theoretic models. However, we shall not build our models from sets. Instead, we shall employ data types and computable relations and functions operating on them. And in their fundamental guise, these are not to be interpreted as sets but taken as sui generis. In particular, our notion of type has its origins in computer science [16] and our notion of relation/function has its origin in computability theory [6], intensional logic [20, 21], and specification [23].

This is the general picture of our enterprise. We now look at matters in more detail.

1.2 Specifications, Programs, and Models

While our models will not be set-theoretic, neither will they be programming models, where the latter consists solely of programs written in some programming language. There is a crucial distinction between mathematical models and programming ones. For while it is true that the process of programming results in the construction of models from programs and data types, and so fits our desiderata for being computable, they are not, by themselves, mathematical models. In isolation, a programming language is just that, i.e., a language. And without some mathematical interpretation of its constructs, aside from the formal nature of its grammar, it has no mathematical content. And neither do the programs written in it. Such content has to be imposed upon it via a semantic account (of some kind) and it is this that renders it amenable to mathematical analysis.

In fact, computable models are closer to specifications [23, 12] than programs. Indeed, specifications form the building blocks of computable models, which take the form of suites of interrelated individual relation and function specifications. However, our models are taken to include a much broader class of phenomena than the usual description of a software engineering system. While we are not aiming to exclude such systems, quite the contrary, eventually we shall take matters somewhat further and consider the construction of computable models that have more theoretical interest. In particular, many are best seen as providing a computational makeover of some standard set-theoretic models such as the modelling of events and time.