Chapter 2

From intuitionistic type theory to dependently typed programming

We start with an introduction to intuitionistic type theory [Martin-Löf, 1984] and dependently typed programming [Altenkirch et al., 2005; McBride, 2004] using the Agda language [Norell, 2007, 2009; Bove and Dybjer, 2009]. Intuitionistic type theory was developed by Martin-Löf to serve as a foundation of intuitionistic mathematics like Bishop's renowned work on constructive analysis [Bishop and Bridges, 1985]. While originated from intuitionistic type theory, dependently typed programming is more concerned with mechanisation and practicalities, and is influenced by the program construction movement. It has thus departed from the mathematical traditions considerably, and deviations can be found from syntactic presentations to the underlying philosophy.

```
data Nat : Set where
  zero : Nat
  suc  : Nat → Nat
```

Bibliography

Altenkirch, Thorsten, McBride, Conor, and McKinna, James (2005). Why dependent types matter. Available at `http://www.cs.nott.ac.uk/~txa/publ/ydtm.pdf`.

Bishop, Errett and Bridges, Douglas (1985). *Constructive Analysis*. Springer-Verlag.

Bove, Ana and Dybjer, Peter (2009). Dependent types at work. In *Language Engineering and Rigorous Software Development*, volume 5520 of *Lecture Notes in Computer Science*, pages 57–99. Springer-Verlag.

Martin-Löf, Per (1984). *Intuitionistic Type Theory*. Bibliopolis, Napoli.

McBride, Conor (2004). Epigram: Practical programming with dependent types. In *Advanced Functional Programming*, volume 3622 of *Lecture Notes in Computer Science*, pages 130–170.

Norell, Ulf (2007). *Towards a practical programming language based on dependent type theory*. Ph.D. thesis, Chalmers University of Technology.

Norell, Ulf (2009). Dependently typed programming in Agda. In *Advanced Functional Programming*, volume 5832 of *Lecture Notes in Computer Science*, pages 230–266. Springer-Verlag.