Chapter 2

Modelling the World

How can we model the real world? Here are some examples of objects, and concepts one might want to work with formally in a computer.

- Practical applications
  - Different fruits
  - Lego pieces
  - Hand of cards
  - Atoms
  - Puzzle pieces
  - Pegs
  - Coins
- a little more theoretical
  - Geometric forms
  - Numbers
  - Lists
  - Trees
  - Boolean Circuits
- and even more abstract
  - Functions
  - Machine language programs
  - Turing machine programs
  - Strategies

As a running example for this lecture consider the game "Torres". It consists of a game board and game pieces, stackable. How can we represent this in a computer? Note that we are not interested right now in how to learn this game, merely how we can bring it into a computer.

What does a computer offer?

- Bits and bytes at the lowest level
- Memory cells
- A programming language
  - Low level languages: Machine language,
  - Imperative languages: C, C++, Java.
  - Functional languages: ML, Scheme, Haskell
  - Logic programming languages: Prolog, Forum, Lolli
- Different levels of abstraction

Finding the right level of abstraction is enormously important.

- The concepts of the programming language help you
- We use Standard ML of New Jersey

2.1 Objects and Types

The main idea of abstraction is provided by the concept of an object or, as it sometimes called: term.

Definition 2.1.1 (Objects) *An object is something that represents something from the real world, an entity to which we assign meaning. Objects are build from constructors that can be freely named (avoiding duplicates of course).*

Example 2.1.2 (Fruit) Imagine a situation where we have to work with fruits. In the best of all cases, we would like to write `apple` for an apple, or `pear` for a pear.

That we can do this in ML is thanks to its concept of datatype. In general, a type describes a set of objects that have a certain, specific property. In this class, types will be omnipresent, so it is good to get used to them early, in theory and in practice. At the beginning they may be difficult to grasp, and when programming they might get in your way (at least so you think), but hopefully you will grow to appreciate them at the end of this class.

In ML we distinguish to kinds of datatypes. Types that are built-in to ML (we will see many examples of this kind in the next few lectures), or it can be defined by the user which we will learn about in the first few lectures. User defined types are called datatypes.

Definition 2.1.3 (Datatype) *A datatype is a user defined type, whose objects can be created by composing a user defined set of constructors in a type correct fashion (what this means will become clear soon).*

Here are two examples of datatypes. One is the one from above about fruit, and the other about how to represent a man in Torres.

Example 2.1.4 (Fruit again)

```
datatype Fruit
  = Apple
  | Pear
  | Banana
  | Orange
```

In this example `Fruit` is the name of the type, and `Apple`, `Pear`, `Banana`, or `Orange` are its constructors.

Example 2.1.5 (Torres)

```
datatype Color
  = King
  | Green
  | Blue
  | Red
  | Orange
  | Black
```

2.2 Values and Computations

The predominant task a computer is used for is computation. Computer programs manipulate the objects of our intent in prespecified ways. Consider for example the Torres game. We might want to compute how many points a player scores during on of scoring rounds. (The score is defined as the product of the surface times the height of a man of the players color in a castle.)

One thing is clear after you have experimented with and studied the functional programming language ML for a while. What ever computation you do in ML, if your program can be typed before the computation is actually started, it will compute a value (surely only if the program terminates). In our setting values are special objects, they are objects that do not evaluate further (or mathematically speaking, they only evaluate to themselves). Mathematics is very important in computer science. For ML, for example, we can show that if a program is well-typed it can never get stuck. It will never crash when running. ML might reject type incorrect prorams but it will never crash.

In summary this means that we will distinguish between two kinds of objects in the programming language. The first are values as described above. The other are expressions, objects that do something. Each type in ML, that is every user defined datatype and every built in type, can be seen as a set of values and expressions. In the next chapter we go into some depth, show many examples for different types, define their values, and discuss the different expressions that are offered by ML.