Structured Data

Michael P. Fourman

February 2, 2010

This book has two parts, each divided into chapters. These chapters are, in turn, divided into sections, subsections and paragraphs. The structure of the book is summarised in the following diagram.

The book is designed to be read through from beginning to end. However, it is possible to omit some sections, or to read the book in a different order. If you choose to do this, you should be aware that some sections depend on others. Here is a diagram that summarises these dependencies.

Our subject is structured data; the diagrams describing the structure of this book are good examples of two kinds of structure that we will study. We will be mainly concerned with the representation and manipulation of such structures by computer programs, but we begin with an informal discussion of these two examples.

Trees

A hierarchical structure, such as the structure shown in diagram 1, is called a *tree.*[1] When talking about trees we use the following vocabulary: A tree consists of a collection of *nodes* — each of which may have some finite number of *children*. A node with no children (zero is a perfectly good finite number) is called a *leaf;* other nodes are called *internal* nodes. One distinguished node, called the *root,* has no parents; every other node is a descendant of the root (in just one way).

The name, *tree,* comes from the branching structure, typical of a botanic tree. But, perversely, draw our trees upside-down. If we make a drawing of the tree, with each node one unit higher than each of its children, we say the *height* of a node is the maximum height difference between it and any of its descendants. The height of an internal node is one more than the maximum of the heights of its children. (The height of a leaf is zero.)

[1] Strictly speaking, the trees we'll mention here are *rooted* trees.

This note is unfinished.