PART II

Definition by Choices

Chapter 12

Defining new types: a bird's-eye view

If you're the sort of person who likes to look at the map before heading out on a trip, this brief chapter should give you an idea of the road we'll be travelling for the next fifteen or twenty chapters. If, on the other hand, you're the sort of person to whom the map doesn't make sense until you've seen some of the road itself, ignore this chapter for now, but come back to it after a few more chapters.

We've learned a bunch of things you can do with images, a bunch of things you can do with numbers, and a bunch of things you can do with strings. These are three important data types that happen to be built into Racket, but for many problems they're not enough and we'll need to *define our own data types*. There are several ways to do this, each with a corresponding new way of writing functions.

First, a new data type can be a *choice* among two or more existing data types, and functions will have to choose among two or more simpler expressions or functions.

Second, a new data type can have *parts* drawn from existing data types, in which case functions will have to dissect data into their parts, and put together parts into new data.

Third, a new data type can (by combining the notions of "definition by choices" and "definition by parts") be defined in terms of itself; functions will likewise be defined in terms of themselves.

Fourth, a new data type can be constructed by "abstraction": observing similarities and intentionally overlooking differences among several existing data types to create a *general* data type that can be specialized to do the job of any of the original types, and more; likewise, from several similar functions one can often construct a single *general* function that does the job of all the original functions and more.

Over 400 years ago, the English philosopher John Locke wrote something eerily close to this, although the match isn't perfect:

> The acts of the mind, wherein it exerts its power over simple ideas, are chiefly these three:
>
> 1. Combining several simple ideas into one compound one, and thus all complex ideas are made.
> 2. The second is bringing two ideas, whether simple or complex, together, and setting them by one another so as to take a view of them at once, without uniting them into one, by which it gets all its ideas of relations.

> 3. The third is separating them from all other ideas that accompany them in their real existence: this is called abstraction, and thus all its general ideas are made.

I might add a fourth, "applying an idea to itself," which has proven to be a powerful — if easily misused — technique in mathematics, computer science, and philosophy.