Introducing specification in Z

Although Z is a specification language, not a programming language, it involves directly similar components:

- syntax,
- semantics,
- a pre-defined “library” of specification elements.

We will discuss and illustrate these components in the context of Diller’s running example in chapter 4 — a telephone directory system. Note that chapter 19 presents an animation of this specification in Miranda. This program (with a few small corrections to make it compile) appears in our class directory. We will use the program to augment our discussion, and you may find it useful for experimentation.

Each Z specification begins with **basic type definitions**, written

$[\text{Ident}_1, \ldots, \text{Ident}_n]$.

This declares *names* (only) of the atomic types of the specification. These identifiers play a role completely analogous to sort identifiers in algebraic specification. Z is strongly typed, and each variable must have its type declared.

There are no strict rules in Z with respect to upper/lower-case use. One common convention is to write basic type identifiers all in upper-case, but Diller does not follow this convention.

A **word** is a sequence of letters, digits, and underscores, starting with a letter (or in a few cases covered later, a special symbol). An **identifier** is a word followed by a (possibly empty) **decoration** — a sequence of symbols from {', ?, !}.

A common type in Z is a *binary relation*. If X and Y are sets, then X ⇸ Y denotes the collection of all binary relations over X × Y. A **variable declaration**

telephones: Person ⇸ Phone

prescribes that the variable 'telephones' is a member of the collection of binary relations over Person × Phone. No particular set of ordered pairs is identified as the values of the variable, this is just a type declaration. For instance, it's possible that (diller, 4794) ∈ telephones, and in Z we write diller ↦ 4794 ∈ telephones. The symbol ↦ is known as **maplet**, and is used to construct ordered pairs; from this construction, we can infer that diller ∈ Person and 4794 ∈ Phone.

The first specification component we encounter is a schema definition. A **schema definition** includes:

- a name for the schema,
- declarations that introduce constituent variables and their types, and
- predicates (i.e., relations) that describe properties the variables are guaranteed to possess.