CHAPTER 9 DEFINING NEW UNITS OF MEASURE

Occasionally units of measure are needed that do not come predefined in the ASCEND system. You can define a new unit of measure by defining the conversion factor. In this chapter, we examine how to do this easily for an individual user and on a system-wide basis.

9.1 *CAVEATS*

Order matters!

Order matters for defining units of measure in three ways.

- a unit of measure must be defined before it is used anywhere.
- the first definition ASCEND reads for a unit of measure is the only definition ASCEND sees.
- new units can be defined only from already defined units.

Measuring units are absolutely global in the ASCEND environment—they are not deleted when the Library of types is deleted. Once you define a unit's conversion factor, you are stuck with it until you shut down and restart ASCEND. For any unit conversion definition, only the first conversion factor seen is accepted. Redefinitions of the same unit are ignored.

Multiplicative unit conversions only!

The various units ASCEND uses are all obtained by conversion factors (multiplication only) from the SI units. So, for example, temperatures may be in degrees Rankine but not in Fahrenheit. In this chapter we address creating new conversion factors. For handling non-multiplicative conversions (such as the Fahrenheit or Celsius offsets) see Section 8.2.

9.2 INDIVIDUALIZED UNITS

There are two scenarios for individualized units of measure. One in which you need a measure defined only for a specific model and another in which you want to define a measure that you will use throughout your modeling activities in the future. The syntax for both is the same, but where best to put the UNITS statement differs.

9.2.1 UNITS OF MEASURE FOR A SPECIFIC MODEL

Units of measure which are used in only one model can be defined at the beginning of the model itself or before the model, but not the units appear in the model definition. Let us suppose you want to measure speed in {furlong/fortnight} in a model. ASCEND does not define furlong, fortnight, or furlong/fortnight. (We cannot find standard definitions for them!).

```
MODEL mock_turtle;
   d IS_A distance;
   delta_t IS_A time;
   s IS_A speed
   s = d/delta_t;
   (* We really should write s * delta_t = d;
    * to avoid division by zero.
    *)
UNITS
   furlong = {3.17*kilometer};
   fortnight = {10*day};
END UNITS;
METHODS
METHOD default_self;
   d := 1 {furlong};
   t := 5 {hours};
END default_self;
(* other standard methods omitted *)
END mock_turtle;
```

In mock_turtle we define *furlong* and *fortnight* conversions before they are used in the methods and before any equations which use them. Also, notice that even though ASCEND rejects this model `mock_turtle`, as it will because of the missing ";" after "`speed`" in the fourth line, *furlong* and *fortnight* still get defined. The UNITS statement can appear in any context and gets processed regardless of any other errors in that context.

9.2.2 UNITS OF MEASURE FOR ALL YOUR PERSONAL MODELS

If you commonly use a set of units that is not in the default ASCEND library measures.a4l, you can create your own personal library of units in the user data directory ascdata. The location of this directory is given by ASCEND at the end of all the start-up spew it prints to the Console window (or xterm under UNIX) as shown below. You will see a path other than `/usr0/ballan/` of course.

```
---------------------------------
User data directory is /usr0/ballan/ascdata
---------------------------------
```

Create the library file myunits.a4l in your ascdata directory. This file should contain a UNITS statement and any comments or NOTES you wish to make. This file should contain any conversions that you change often. For example:

```
UNITS (* Units for Norway, maybe?*)
euro = {1*currency};
(* currency is the fundamental financial unit *)
kroner = {0.00314*euro};
nk = {kroner};
USdollar = {0.9*euro};
CANdollar = {0.65*USdollar};
END UNITS;
```

Note that this file contains a definition of USdollar different from that given in the standard library measures.a4l. ASCEND will warn you about the conflict. You must load myunits.a4l into ASCEND before atoms.a4l or any of our higher level libraries. You can ensure that this happens by putting the statement

```
REQUIRE “myunits.a4l”;
```

on the very first line in all your model definition files.

9.3 NEW SYSTEM-WIDE UNITS

Suppose you are maintaining ASCEND on a network of computers with many users. You have a standard set of models stored in a centrally located directory, and you want to define units for use by everyone on the network. In this case, just edit models/measures.a4l, the default units of measure library. ASCEND is an open system.

Make the new unit conversion definition statement(s) of the form

```
newunit = {combination of old units};
```

as described in Section 9.2. In the file measures.a4l, add your statement(s) anywhere inside the block of definitions that starts with UNITS and ends with “END UNITS.” The existing definitions are divided up into groups by comment statements. If your conversion

belongs to one of the groups, it is best to put the conversion in that group. The groups are given in Table 9-1.

Table 9-1 Groups of units in the current measures library

distance
mass
time
molecular quantities
money
reciprocal time (frequency)
area
volume
force
pressure
energy
power
absolute viscosity
electric charge
miscellaneous electromagnetic
swiped from C math.h
constant based conversions
subtly dimensionless measures
light quantities
miscellaneous rates
time variant conversions

9.4 SEND THEM IN

We are always on the lookout for useful unit conversions to add to measures.a4l. If you create a myunits.a4l containing unit conversion definitions of general use (i.e. not currency exchange rates and other time-varying conversions), please mail us a copy and include your name in a comment. Thank you very much.