Animating Abstract Data Types in Miranda

Miranda provides a mechanism that allows ADT definitions to be integrated into scripts. The details of this facility are explained in the in-line Miranda manual, Section 20: Algebraic type definitions. The short story is that this provides for the addition of programmer defined types that behave for most purposes as ADTs. The type definition introduces the “constructors” of the new type. These constructors are functions that create values of the new type. To signify their special role, their syntax is distinguished from other function identifiers — they must be written with a leading upper-case letter. Other functions that are defined to operate on values of this new type have normal identifiers (i.e., leading lower-case letter).

For instance, the ADT queue (of numbers) could be added with the Miranda type definition:

```
queue ::= New | Add queue num
```

This declares the new type identifier 'queue' with the two constructor functions 'New' and 'Add'. Miranda algebraic type definitions deliberately resemble BNF as one of their results is to validate a new collection of legal expressions. Since 'New' and 'Add' are constructors for the new type 'queue', their result types are 'queue' and this is not explicitly written. 'New' is a constant of the type (a function with no arguments), and 'Add' is a function taking two arguments, the first of type 'queue' and the second of type 'num'. In ADT notation we would write:

New: queue and Add: queue x num $\rightarrow$ queue.

We can subsequently define functions operating on these new values using the ADT equational specification. For instance,

```
isNew(New) = True
isNew(Add q n) = False
front(Add q n)  = n, if isNew(q)
                = front(q), otherwise
```

A complete Miranda version of this ADT appears in our class directory. Miranda will evaluate any expression of type queue, reducing it to its simplest form — for instance, entering del (Add (del (Add (Add (Add New 1) 2) 3)) 4) yields the expected result Add (Add New 3) 4, and front (del (Add (del (Add (Add (Add New 1) 2) 3)) 4)) yields 3.

Note that Miranda also provides for polymorphic ADTs. Parametric types in Miranda are written with the notations *, **, ***, … If we wish to have a polymorphic queue (i.e., the element type is a parameter) we could define

```
queue * ::= New | Add queue *
```