# Parameterized Algebraic Specifications

It is often desirable to parameterize our specifications. For instance, when we describe the Stack ADT, the behavior is not dependent on the type of items placed on the stack — the essential stack behavior is the same regardless of whether the items are integers, strings, queues or any other kind of item. We therefore prefer to describe Stack[Item] and leave Item completely abstract.

However, it's important to realize that when we introduce parameters that are regarded as other abstract data types, the sense of our specification changes. We are no longer describing "data algebras", but instead are describing a mapping from data algebras to data algebras. Once the Item data algebra is known, then and only then, can we determine the Stack-of-Item data algebra, and it is this correspondence that has been described rather than any one data algebra.

We intend the mapping that we describe with a parameterized specification to preserve the equalities in any specification used to instantiate the parameter(s). For instance, in Stack[Integer], we still expect that i*(j+k) = i*j + i*k for all integers i, j, and k. Since we do not include inequalities, this is a safe conclusion. We also wish to know that there is no collapsing (i.e., inconsistency) in an instantiating specification. However, this is far from assured. We routinely include additional equations for the argument types in our parameterized specifications. For instance, TOP(PUSH(s,x)) = x is an equation for integers in Stack[Integer]. Conceptually it is much more difficult to resolve questions of consistency and sufficient completeness of parameterized specifications since they pertain to *all* conceivable instantiations of the parameters.

The analysis of a parameterized type is pursued in a way analogous to that which we have used for ordinary specifications. Namely, identify the equivalence classes of the ground terms, and regard the data algebra obtained by applying the operations to these classes as the essence of the specification. This requires the following adjustments:

1. take as "ground terms" those terms that have no variables, except for those of the parameter(s) sorts, and
2. sufficient completeness means proving any "ground term" is equivalent to one whose only operations are in pre-defined and parameter ADTs