Reasoning about Z specifications

A primary purpose of formal specification methods is to be able to make deductions about the behavior of any implementation that realizes a formal Z specification. In this episode, we will examine an example of reasoning from a Z specification. In particular, we explore a significant aspect of the coherence (i.e., consistency) of Z specifications.

One thing we certainly wish to be confident about a specification is that a state invariant actually *is* invariant. That is, for each □ operation, we should be able to deduce from the pre/post-conditions that if the invariant is true before the operation is performed, it is still true after the operation is performed. For our first instance of proving from a Z specification, we shall again refer to Diller's telephone database example. We will not formally establish the invariant for all operations, but will explore a couple of instances.

The first operation schema we pursue is AddEntry. We prove that
AddEntry □ dom telephones □ members □ dom telephones' □ members'.

The first step is to expand the schema into the appropriate logical formulas to obtain
(name? □ members
□ name? |□ newnumber? □ telephones
□ telephones' = telephones □ {name? |□ newnumber?}
□ members' = members)
□ dom telephones □ members
□ dom telephones' □ members'.

The implication follows in four simple steps.
dom telephones'
= dom telephones □ {name?}, since telephones' = telephones □ {name? |□ newnumber?}
□ members □ {name?}, since dom telephones □ members
= members, since name? □ members
= members', since members' = members.

Next we examine the state invariant for an apparently more interesting case, the RemoveMember schema — this schema changes *both* state variables.

We prove that
  RemoveMember □ dom telephones □ members □ dom telephones' □ members'.

Again, the first step is to expand the schema into the appropriate logic formulas to obtain

(name? □ members
  □ members' = members \ {name?}
  □ telephones' = {name?} ⩤ telephones)
□ dom telephones □ members
□ dom telephones' □ members'.

This is easily proven by

dom telephones'	
= dom telephones \ {name?}	since telephones' = {name?} ⩤ telephones)
□ members \ {name?}	since dom telephones □ members
= members'.	

Proofs of the state invariant for the other □ operation schemas are similar. Internal inconsistency is a fatal flaw for a formal specification, but may be difficult to detect. Verifying that the written operation specifications logically imply all invariants are preserved is therefore a useful check to perform.