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 \( \square \) 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
\[
\text{AddEntry} \land \text{dom telephones} \land \text{members} \land \text{dom telephones}' \land \text{members}'.
\]

The first step is to expand the schema into the appropriate logical formulas to obtain
\[
\begin{align*}
( \text{name?} & \land \text{members} \\
\square \text{name?} & \lor \text{newnumber?} \land \text{telephones} \\
\square \text{telephones}' & = \text{telephones} \land \{\text{name?} \lor \text{newnumber?}\} \\
\square \text{members}' & = \text{members}) \\
\square \text{dom telephones} & \land \text{members} \\
\square \text{dom telephones}' & \land \text{members}'.
\end{align*}
\]

The implication follows in four simple steps. dom telephones'
\[
= \text{dom telephones} \land \{\text{name?}\}, \quad \text{since telephones}' = \text{telephones} \\
\land \{\text{name?} \lor \text{newnumber?}\}
\]
\[
\square \text{members} \land \{\text{name?}\}, \quad \text{since dom telephones} \land \text{members} \\
= \text{members}, \quad \text{since name?} \land \text{members} \\
= \text{members}', \quad \text{since members}' = \text{members}.
\]
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 \[\text{dom telephones} \cap \text{members} \subseteq \text{dom telephones}' \cap \text{members}'.\]

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

\[
(\text{name}? \in \text{members} \\
\text{members}' = \text{members} \setminus \{\text{name}?) \\
\text{telephones}' = \{\text{name}?) \subseteq \text{telephones}) \\
\text{dom telephones} \cap \text{members} \\
\text{dom telephones}' \cap \text{members}'.
\]

This is easily proven by

dom telephones'  \\
= \text{dom telephones} \setminus \{\text{name}?) \quad \text{since} \quad \text{telephones}' = \{\text{name}?) \subseteq \text{telephones) \}
\text{members} \setminus \{\text{name}?) \quad \text{since} \quad \text{dom telephones} \cap \text{members} \\
= \text{members}'.

Proofs of the state invariant for the other \[\text{D}\] 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.