More Reasoning about Z specifications

The example state invariant proofs do not reflect one of the technical difficulties in formal Z proofs. State variables will normally be changed repeatedly as operations are carried out. When we are interested in establishing properties of a sequence of operations, then the post-states for one become the pre-states of the next. The decorating notations become the only way to refer to these various intermediate values, and in some situations this is awkward at best.

Note that when decorating notations are applied to schema names, we normally apply post-state notation only to state variables and output variables. The conventions about what to do about applying schema name decoration to input/output variables is one of the potential trouble spots of Z.

For this example, we consider the assertion for Diller's telephone database that expresses an obvious expectation of this system. Namely, after performing an AddEntry operation, a FindPhones operation with the same name will return the new number. Formally (ignoring exceptions),

Claim:

AddEntry(name?,newnumber?) ⇒ FindPhones'(name?)
⇒ newnumber? ∈ numbers!'.

The first step is to expand the schemas into the appropriate logic formulas. The expanded logic formula is

(name? ∈ members
∧ name? ↦ newnumber? ∈ telephones
∧ telephones' = telephones ∪ {name? ↦ newnumber?}
∧ members' = members)
⇒ (name? ∈ dom telephones'
∧ numbers!' = telephones'(|{name?}|))
⇒ newnumber? ∈ numbers!'.

To justify the validity of this implication, we just notice that from the hypothesis, telephones' includes the pair name? ↦ newnumber?, and hence numbers!' = telephones'(|{name?}|)) includes name?.

Note that if we wish to analyze any subsequent operations on these results, we would need to carry forward the conditions and conclusions implicit in ΞFindPhones', namely

members'' = members'
telephones'' = telephones'