At this point, there are a number of issues to explore. First of all, the issue of **scope of identifiers** is raised by the declaration of the output variable 'rep' in several schemas. Some identifiers have a **global scope** — their definitions are associated throughout the rest of the specification. Basic type identifiers and schema identifiers have global scope. Other identifiers have a **local scope** — their definition is in effect only within a single schema. Identifiers in the declaration part of a schema are local to that schema. Variables declared in one schema are known elsewhere only if that schema is imported (or included) by another. It is implicitly understood that to obtain a consistent specification, imported declarations and conditions must not introduce conflicts.

Another collection of issues is raised in the DoAddEntry definition by the use of the familiar logical operations to combine schemas rather than logical formulas. The brief view given here is that the resulting definition is assumed to import all the declarations of the constituent schema, and the conditions of DoAddEntry are formed from those of the constituents in the manner given in the formula.

Examination of the pre-conditions of the three operation schemas reveals that if the pre-conditions of the AddEntry operation are met, then this operation determines a new state, and a Success report is issued. If the pre-conditions of AddEntry are not met, then those of one of the other schemas is met, and a corresponding report is the result with no state change.

There is a potential inconsistency in the DoAddEntry definition as three distinct values are associated with the output variable 'rep!'. However, the pre-conditions for the constituent schemas in the DoAddEntry operation are mutually exclusive. The AddEntry schema provides the preconditions

$name? \notin members$

$name? \mapsto newnumber? \notin telephones$

Hence the negation of these pre-conditions is

$name? \in members \quad name? \mapsto newnumber? \in telephones$

Now the two disjuncts in this negation are not mutually exclusive in general. However, we are also assured of the PhoneDB invariant

$dom\ telephones \subseteq members$

Therefore if $name? \in members$ is true, $name? \mapsto newnumber? \in telephones$ must be false. Hence the three pre-conditions of the constituent schemas of DoAddEntry

$name? \in members \wedge name? \mapsto newnumber? \in telephones$

$name? \in members$

name? ↦ newnumber? ⌧ telephones

in conjunction with the invariant of PhoneDB, are mutually exclusive and no inconsistency arises for the value of 'rep!.

With the behavior of the operation of adding entries now completely specified, we are now prepared to examine the Miranda animation of this operation.

```
addEntry n e (mem, tel) = (mem, tel ++ [(n,e)])
doAddEntry (mem, tel) (n,e)              ll correction added
        = write "Okay\n" (phdb (addEntry n e (mem, tel))),
            if member mem n & ~member tel (n,e)
        = write "Not a member\n" (phdb (mem, tel)), if ~member mem n
        = write "Entry already exists\n" (phdb (mem, tel)), if member tel (n,e)
```

The 'addEntry' function returns a pair that satisfies the post-condition for that operation provided the pre-condition is True. The 'doAddEntry' definition includes the three tests for its guards, and either performs the add and issues a success “report”, or issues an error report in the other cases.