We will treat the remaining operation specifications briefly, pausing only to take note of newly introduced features of Z.

RemoveEntry

ΔPhoneDB
oldnumber?: Phone
name?: Person

---

name? $\mapsto$ oldnumber? $\in$ telephones
telephones' = telephones $\setminus$ {name? $\mapsto$ oldnumber?}
members' = members

The pre-condition for removing an entry requires its presence, so we again have an exceptional condition to treat. The post-condition is expressed using a new pre-defined Z operation — set difference. **Set difference** denotes the set consisting of all elements in the first set but not in the second.

UnknownEntry

ΔPhoneDB
oldnumber?: Phone
name?: Person
rep!: Report

---

name? $\mapsto$ oldnumber? $\in$ telephones
rep! = 'Unknown entry'

Then the ultimate operation specification is given as usual.

$$\text{DoRemoveEntry} \triangleq \text{RemoveEntry} \gg \text{Success} \lor \text{UnknownEntry}$$

The comparison with the Miranda animation again reveals a close match.

```
removeEntry n e (mem, tel) = (mem, tel -- [(n,e)])
doRemoveEntry (mem, tel) (n,e)          ||  correction added
        = write "Okay\n" (phdb (removeEntry n e (mem, tel))),if member tel (n,e)
        = write "Unknown entry\n" (phdb (mem, tel)), otherwise
```

AddMember

```
ΔPhoneDB
name?: Person
─────────────────────────
name? ∈ members
members' = members ∪ {name?}
telephones' = telephones
```

AlreadyMember

```
ΔPhoneDB
name?: Person
rep!: Report
─────────────────────────
name? ∈ members
rep! = 'Already a member'
```

The complete operation specification is then

DoAddMember ≙ AddMember ⨾ Success
∨ AlreadyMember

The Miranda animation code directly follows this specification too.

```
addMember n (mem, tel) = (mem ++ [n], tel)
doAddMember (mem, tel) n                ||  correction added
        = write "Okay\n" (phdb (addMember n (mem, tel))),if ~member mem n
        = write "Already member\n" (phdb (mem, tel)), otherwise
```

The next PhoneDB operation brings us to other new pre-defined operations in Z. The first pertains to a relation R ⊆ X × Y. Given a subset W ⊆ X, the **domain restriction** of R to W is the relation W ◁ R defined by

W ◁ R = {(x,y) | x∈W ∧ (x,y)∈R}.

This permits the identification of the subdomain W of the relation to be *included*.

The second pre-defined Z operation needed here is **anti-restriction** (or **domain subtraction**). It also pertains to a relation R ⊆ X × Y. Given a subset W ⊆ X, the anti-restriction of R to W is the relation W ⩤ R defined by

W ⩤ R = {(x,y) | x∉W ∧ (x,y)∈R}.

This permits the identification of a subdomain W of the relation to be *excluded*. Domain subtraction is used in the last operation specification.

RemoveMember
ΔPhoneDB
name?: Person

name? ∈ members
members' = members \ {name?}
telephones' = {name?} ⩤ telephones

And finally,

DoRemoveMember ≙ RemoveMember ⨾ Success
NotMember

This completes the specification of the PhoneDB operations, and its Miranda counterpart is also routine.

```
removeMember n (mem, tel) = (mem -- [n], ndres [n] tel)
doRemoveMember (mem, tel) n          ll correction added
     = write "Okay\n" (phdb (removeMember n (mem, tel))), if member mem n
     = write "Not a member\n" (phdb (mem, tel)), otherwise

ndres u f = [(x,y) | (x,y) <-f; ~member u x]
```