This is the BirthdayBook specification, from Spivey [1].

$[NAME, DATE]$

The *BirthdayBook* schema defines the *state space* of the birthday book system.

BirthdayBook

$$known : \mathbb{P}\, NAME$$
$$birthday : NAME \pfun DATE$$

$$known = \mathrm{dom}\, birthday$$

This *InitBirthdayBook* specifies the initial state of the birthday book system. It does not say explicitly that $birthday'$ is empty, but that is implicit, because its domain is empty.

InitBirthdayBook

$$BirthdayBook'$$

$$known' = \{\}$$

Next we have several operation schemas to define the normal (non-error) behaviour of the system.

AddBirthday

$$\Delta BirthdayBook$$
$$name? : NAME$$
$$date? : DATE$$

$$name? \notin known$$
$$birthday' = birthday \cup \{name? \mapsto date?\}$$

FindBirthday

$$\Xi BirthdayBook$$
$$name? : NAME$$
$$date! : DATE$$

$$name? \in known$$
$$date! = birthday(name?)$$

Remind

$\Xi BirthdayBook$
$today? : DATE$
$cards! : \mathbb{P}\, NAME$

$cards! = \{n : known \mid birthday(n) = today?\}$

Now we strengthen the specification by adding error handling.

$REPORT ::= ok \mid already\_known \mid not\_known$

First we define auxiliary schemas that capture various success and error cases.

Success

$result! : REPORT$

$result! = ok$

AlreadyKnown

$\Xi BirthdayBook$
$name? : NAME$
$result! : REPORT$

$name? \in known$
$result! = already\_known$

NotKnown

$\Xi BirthdayBook$
$name? : NAME$
$result! : REPORT$

$name? \notin known$
$result! = not\_known$

Finally, we define robust versions of all the operations by specifying how errors are handled.

$RAddBirthday == (AddBirthday \land Success) \lor AlreadyKnown$
$RFindBirthday == (FindBirthday \land Success) \lor NotKnown$
$RRemind == Remind \land Success$

References

[1] J. Michael Spivey. *The Z Notation: A Reference Manual*. International Series in Computer Science. Prentice-Hall International (UK) Ltd, second edition, 1992.