1 Address Book — v1

1.1 The Specification

MODULE *addressbook*

CONSTANTS $Names$, $Emails$
VARIABLE $address$

$Invariant \triangleq address \in [Names \to Emails]$

$Init \triangleq address = \langle\rangle$ A different ideom for an empty function...

$Add(name,\ email) \triangleq$
$address' = [n \in \text{DOMAIN } address \cup \{name\} \mapsto$
IF $n \in \text{DOMAIN } address$
THEN $address[n]$
ELSE $email$
$]$

$Remove(name) \triangleq$
$address' = [n \in \text{DOMAIN } address \setminus \{name\} \mapsto address[n]]$

$Find(name) \triangleq$
$\land\ name \in \text{DOMAIN } address$
$\land\ \text{UNCHANGED } address$

$Next \triangleq$
$\exists\, n \in Names,\ e \in Emails :$
$\lor\ Add(n,\ e)$
$\lor\ Remove(n)$
$\lor\ Find(n)$

Modification History
Last modified Sat Feb 13 10:50:44 GMT 2021 by alunm
Created Wed Feb 10 21:49:46 GMT 2021 by alunm

2 The Model

2.1 Model Overview

The Behaviour specification is an *Initial-predicate and next-state relation*

Initial Predicate *Init*

Next-state relation *Next*

The Model values assigned to declared constants

Set of Names is set to

$Names \leftarrow \{\text{“a”}, \text{“b”}\}$

Set of email addresses is set to

$Emails \leftarrow \{\text{“1”}, \text{“2”}\}$

2.2 Checks and verifications

No invariants were checked.

2.3 Results

A summary of the results

Statistics a summaries of the actions and number of states found.

States found	97
Distinct states	9

Action	Location	States Found	**Distinct states**
Init	Line 7	1	1
Add	Line 9	36	6
Remove	Line 16	36	2
Find	Line 19	24	0

2.4 Discussion

2.4.1 Model description

The state of the system is . A function that maps from names to addresses.

The initial conditions are an empty function, which can be also written as an empty sequence.

The Next relation is that a name and address can be added or removed from the function, or an address found.

2.4.2 Comments

Using a function we run into some of it's limits. With the invariant stated, the domain is a set of names; an initial value of an empty function does not have its domain in the set described by the invariant.

Using a function to model the look-up does work and is the right thing. The problem with this case is that the initial condition of an empty function is not stricktly in the set of functions stated in the invariant. The problem here is stating the right invariant.

2.4.3 Interpretation of results

The number of distict states found for the *Remove* operation is initially surprising, however it is correct, the state of the system after an entry has been removed is the same as the state before the entry was added.

3 Address book — v2

3.1 The Specification

MODULE *addressbook*

CONSTANTS $Names$, $Emails$
VARIABLE $address$

$Invariant \triangleq address \in \text{SUBSET } [name : Names,\ email : Emails]$

$Init \triangleq address = \{\}$

$Find(name) \triangleq$
$\quad \exists\, a \in address : a.name = name$

$Add(name,\ email) \triangleq$
$\quad \land \neg Find(name)$ not in address book
$\quad \land address' = address \cup \{[name \mapsto name,\ email \mapsto email]\}$

$Remove(name) \triangleq$
$\quad \land Find(name)$
$\quad \land address' = address \setminus \{\text{CHOOSE } a \in address : a.name = name\}$

$Next \triangleq$
$\quad \exists\, n \in Names,\ e \in Emails :$
$\quad\quad \lor Add(n,\ e)$
$\quad\quad \lor Remove(n)$
$\quad\quad \lor Find(n) \land \text{UNCHANGED } address$

Modification History
Last modified Sat Feb 13 11:31:50 GMT 2021 by alunm
Created Wed Feb 10 21:49:46 GMT 2021 by alunm

4 The Model

4.1 Model Overview

The same model was used as for version 1

4.2 Checks and verifications

Invariants We now have an invariant that can be checked. *Invariant*

4.3 Results

A summary of the results

Statistics a summaries of the actions and number of states found.

States found	61
Distinct states	9

Action	Location	States Found	**Distinct states**
Init	Line 7	1	1
Add	Line 13	12	8
Remove	Line 17	24	0
Next	Line 25	24	0

4.4 Discussion

4.4.1 Model description

The state of the system is . now modelled as a set of records, which an reflection seems a good description of a database. The initial conditions are now just a simple empty set. The invariant is now a subset of all possible records.

The Find operation is only marginally more complex. The operation has to be a predicate to count as an action that can be performed. It tests *if* an entry is found in the set. It is also useful as a guard condition on the add and remove operations.

i This illustrates how there are more than one way of specifying a system. But as how both can satisfy the model and requirements *both* are correct specifications.