CLEARSY
Safety Solutions Designer

Hackathon
JUL2025

AIX
LYON
PARIS
STRASBOURG

WWW.CLEARSY.COM

Airlock
Access Control

PART I

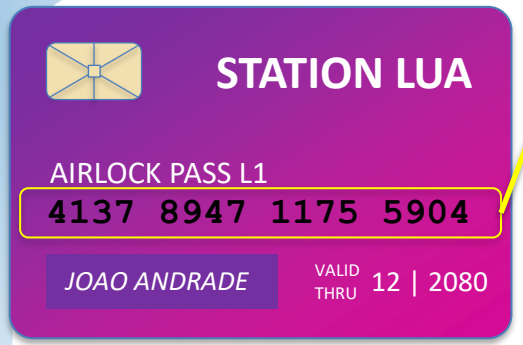
Thierry Lecomte
R&D Director

THIERRY.LECOMTE@CLEARSY.COM

Access Control (security)

- *Airlocks can be manipulated by smartcard holders*
- *Single validation method based on smartcard number*
  - *16 digits (0-9)*
  - *Ex: 1234 5678 9012 3456*

The algorithm



4	1	3	7	8	9	4	7	1	1	7	5	5	9	0	4

Step 1: double 1st, 3rd, 5th, … digits

8	1	6	7	16	9	8	7	2	1	14	5	10	9	0	4

Step 2: add the digits if the result is greater than 9

8	1	6	7	7	9	8	7	2	1	5	5	1	9	0	4

Step 3: sum the value of all digits

8+1+6+7+7+9+8+7+2+1+5+5+1+9+0+4=80

Step 4: if the total ends with 0, the number is valid.
If not, it is invalid

Your turn

- *Develop a B model of validation function [8 pts]*
  - *Software development project used for the whole hackathon*
  - *MACHINE only, no IMPLEMENTATION*
  - *One OPERATION is_valid with one parameter (a table of 16 digits) and one return value (Boolean)*
  - *No need of VARIABLES*
  - *Check syntax, generate proof obligation, prove automatically with Atelier B*

  - **Save a screenshot of the project status**

```
MACHINE
    ACCESS_CARD
ABSTRACT_CONSTANTS
PROPERTIES
OPERATIONS
    ok <-- is_valid(tab) =
    PRE
        tab: 0..15 --> 0..9
    THEN
    END

END
```

Your turn

- *Check the numbers with ProB [2 pts]*
  - 4137 8947 1175 5904
  - 1234 5678 9012 3456
  - 0018 2634 4259 6775

  - Save the probtrace file with the value of oks

```
MACHINE
    test_ACCESS_CARD
OPERATIONS
    test_is_valid = skip
END
```

```
IMPLEMENTATION test_ACCESS_CARD_i
REFINES test_ACCESS_CARD
IMPORTS ACCESS_CARD
CONCRETE_VARIABLES
    oks
INVARIANT
    oks : 0..2 --> BOOL
INITIALISATION
    oks := (0..2) * {FALSE}
OPERATIONS
    test_is_valid =
    VAR ok IN
    END
END
```

```
IMPLEMENTATION test_ACCESS_CARD_i
REFINES test_ACCESS_CARD
IMPORTS ACCESS_CARD
CONCRETE_VARIABLES
    oks
INVARIANT
    oks : 0..2 --> BOOL
INITIALISATION
    oks := (0..2) * {FALSE}
OPERATIONS
    test_is_valid =
    VAR ok IN
        ok <-- is_valid({ /* 4137 8947 1175 5904 */
            });
        oks(0) := ok;

        ok <-- is_valid({ /* 1234 5678 9012 3456 */
            });
        oks(1) := ok;

        ok <-- is_valid({ /* 0018 2634 4259 6775  */
            });
        oks(2) := ok
    END
END
```

Hints

- *No VARIABLES, only CONSTANTS*
- *No IF THEN ELSE in specification [one-liner]*
  - *ok : (ok: BOOL & <predicates>)*
  - *ok := bool(<predicate>)*
- *SIGMA(xx).(xx: SET | VALUE(xx)) to calculate VALUE(xx) over a set*

- *Sets on indexes (odd, even) are fixed*
- *Doubling values is fixed*
- *Possibility to use constant functions and composition*

Your turn

- ***Optional**: Is there any trivial (but suspect) number validated with the algorithm? [1 pt]*
- ***Optional**: Can you design a simple method to quickly generate some valid numbers without paper and computer? [2 pt]*