B weth.mch

```
1
 2
   MACHINE
 3
        B weth
 4
  SEES Solidity_Types
 5
   INCLUDES
 6
        Platform, account, allowance
 7
   ABSTRACT_CONSTANTS
 8
        threshold
9
   PROPERTIES
10
        threshold = 2
11
   VARIABLES
12
        manager, depositors, donated
13
   INVARIANT
14
       depositors <: ADDRESS &
15
       manager : USERS &
       donated : BOOL &
16
17
       balanceOf(THIS) >=
       (SIGMA(ct).(ct : dom(accountOf) | accountOf(ct))) &
18
19
       card(depositors)<= threshold</pre>
20
   INITIALISATION manager := init_msg_sender || depositors := {} || donated := FALSE
21
22
    OPERATIONS
23
        deposit(msg_sender, msg_value) =
24
25
            msg_sender : USERS & msg_value : NAT1
26
        THEN
            IF balanceOf(msg_sender) - msg_value : NAT & accountOf(msg_sender) + msg_value
27
    : NAT &
28
            balanceOf(THIS) + msg_value: NAT THEN
29
                transfer(msg_sender, THIS, msg_value)||
                set_accountOf_abstract({msg_sender
30
31
                     |-> accountOf(msg_sender) + msg_value}) ||
                IF (accountOf(msg_sender) + msg_value >= threshold )
32
                & msg_sender /: depositors & card(depositors) < threshold
33
                THEN
34
35
                    depositors := depositors \/ {msg_sender}
36
                END
37
            END
        END
38
39
40
        withdraw(msg sender, amount) =
        PRF
41
42
            msg_sender : USERS & amount: NAT1
43
        THEN
44
            IF accountOf(msg_sender) >= amount & balanceOf(msg_sender) + amount: NAT &
            balanceOf(THIS) - amount : NAT THEN
45
                transfer(THIS , msg_sender, amount)||
46
47
                set_accountOf_abstract({msg_sender |-> accountOf(msg_sender) - amount})
48
            END
        END
49
50
51
        transferTo(msg sender, dst, amount) =
52
        PRF
53
            msg_sender : USERS & dst: USERS & amount: NAT1
54
        THEN
55
            IF accountOf(msg_sender) > amount &
```

```
56
                // accountOf(msg_sender) - amount : NAT & //pas utile
57
                 msg_sender /= dst &
58
                 accountOf(dst) + amount : NAT
59
             THEN
                 set_accountOf_abstract({msg_sender |-> accountOf(msg_sender)
60
                 - amount, dst |-> accountOf(dst) + amount })
61
62
             END
         END
63
64
65
         approve(msg_sender, dst, amount)=
66
67
             msg_sender : USERS & dst : USERS & amount : NAT1
68
         THEN
69
             IF dst /= msg_sender THEN
70
                 set_allowanceOf_abstract(msg_sender, { dst |->amount})
71
             END
         END
72
73
74
         transferFrom(msg_sender, sender, recipient, amount ) =
75
76
             msg_sender : USERS & sender : USERS &
77
             recipient : USERS & amount : NAT1
78
         THEN
79
             IF sender /= recipient &
80
                 allowanceOf(sender)(msg_sender) >= amount &
81
                 accountOf(sender) >= amount &
                 accountOf(recipient) + amount : NAT &
82
83
                 allowanceOf(sender)(msg_sender) - amount : NAT
             THEN
84
85
                 set_accountOf_abstract({recipient |->
86
                   accountOf(recipient) + amount, sender
                     |-> accountOf(sender) - amount})
87
88
                 set_allowanceOf_abstract(sender, {msg_sender
89
                      |-> allowanceOf(sender)(msg_sender) - amount})
90
             END
91
         END
92
93
94
         rewardTopDepositors(msg_sender, msg_value) =
95
96
             msg_sender : USERS & msg_value : NAT
97
         THEN
98
             IF msg_value = threshold &
99
                 msg_sender = manager &
100
                 card(depositors) = threshold &
101
                 donated = FALSE &
102
                 balanceOf(THIS) + msg_value : NAT &
103
                 balanceOf(manager) - msg_value : NAT &
                 !xx.(xx : depositors => accountOf(xx) + 1 : NAT)
104
105
             THEN
                 transfer(manager, THIS, msg_value) ||
106
                 set_accountOf_abstract(%xx. (xx : depositors | accountOf(xx) + 1)) ||
107
                 donated := TRUE
108
109
             END
110
          END
     END
111
```

2 of 2 12/14/2023, 5:08 PM

B weth i.imp

```
1 /* B_weth_i
   * Author: ASUS
 3
   * Creation date: 8/31/2023
 4
   */
 5
 6 IMPLEMENTATION B_weth_i
 7
   REFINES B_weth
 8 SEES Solidity Types
   IMPORTS Platform, account, allowance, depositedOver100
 9
10 CONCRETE_CONSTANTS
11
        threshold i
12 PROPERTIES
13
        threshold_i : NAT & threshold_i = threshold
14 VALUES
15
        threshold_i = 2
    CONCRETE_VARIABLES manager_i, depositors_i, index, donated_i
16
17
    INVARIANT
18
        index : NAT & index>=0 & donated_i : BOOL &
19
        depositors_i : 0..threshold_i --> ADDRESS &
        manager_i : USERS & manager_i = manager & donated_i = donated &
20
21
        index = card(depositors) &
22
        depositors_i[0..index-1] = depositors &
23
        (0..index-1) < | depositors_i : 0..index-1 >-> depositors &
        depositedOver_100~[{TRUE}] = depositors
24
25
26
    INITIALISATION
27
        index := 0;
28
        depositors_i := (0..threshold_i) * {addr_0} ;
29
        manager_i := init_msg_sender;
30
        donated i := FALSE
31
32
    OPERATIONS
33
        deposit(msg_sender, msg_value) =
34
        BEGIN
35
            VAR senderBalance, senderAccount, thisBalance IN
36
                senderAccount <-- get_accountOf(msg_sender);</pre>
                senderBalance <-- get_balanceOf(msg_sender);</pre>
37
38
                thisBalance <-- get_balanceOf(THIS);</pre>
39
40
                IF thisBalance + msg_value <= MAXINT & senderBalance - msg_value >= 0 &
41
                     senderAccount + msg_value <= MAXINT</pre>
42
                THEN
43
                     set_accountOf(msg_sender, senderAccount + msg_value);
44
                    transfer(msg_sender, THIS, msg_value);
45
                    VAR distinct IN
                         distinct <-- get_depositedOver_100(msg_sender);</pre>
46
                         IF senderAccount + msg_value >= threshold_i & distinct = FALSE &
47
    index < threshold i
48
                         THEN
49
                             depositors_i(index) := msg_sender;
                             set depositedOver_100(msg_sender, TRUE);
50
                             index := index +1
51
52
                         END
53
                     END
                END
54
55
            END
```

```
56
         END
 57
 58
 59
         withdraw(msg_sender, amount) =
         BEGIN
 60
              VAR senderAccount, senderBalance, thisBalance IN
 61
                  senderAccount <-- get_accountOf(msg_sender);</pre>
 62
                  senderBalance <-- get_balanceOf(msg_sender);</pre>
 63
 64
                  thisBalance <--get_balanceOf(THIS);</pre>
                  IF senderAccount >= amount & senderBalance + amount <= MAXINT &</pre>
 65
     thisBalance>=amount
 66
                  THEN
 67
                      transfer(THIS , msg_sender, amount);
                      set accountOf(msg sender, senderAccount - amount)
 68
 69
                  END
 70
              END
 71
         END
 72
 73
 74
         transferTo(msg_sender, dst, amount) =
 75
         BEGIN
 76
              VAR senderBalance, receiverBalance IN
 77
                  senderBalance <-- get_accountOf(msg_sender);</pre>
 78
                  receiverBalance <-- get_accountOf(dst);</pre>
 79
                  IF senderBalance > amount & receiverBalance + amount <= MAXINT & msg_sender</pre>
     /= dst
 80
                  THEN
 81
                      set_accountOf(msg_sender, senderBalance - amount);
                      set accountOf(dst, receiverBalance + amount)
 82
 83
                  END
              END
 84
         END
 85
 86
         ;
 87
 88
         approve(msg_sender, dst, amount)=
 89
 90
              IF msg_sender /= dst THEN
 91
                  set_allowanceOf(msg_sender, dst, amount)
 92
              END
         END
 93
 94
 95
         transferFrom(msg_sender, sender, recipient, amount) =
 96
         BEGIN
 97
              VAR senderBalance, recipientBalance, allowance IN
                  senderBalance <-- get_accountOf(sender);</pre>
 98
 99
                  recipientBalance <-- get_accountOf(recipient);</pre>
100
                  allowance <-- get_allowanceOf(sender, msg_sender);</pre>
                  IF sender /= recipient & allowance >= amount & senderBalance >= amount &
101
102
                      recipientBalance + amount <= MAXINT</pre>
103
                  THEN
104
                      set_accountOf(sender, senderBalance - amount);
                      set_accountOf(recipient, recipientBalance + amount);
105
                      set_allowanceOf(sender, msg_sender, allowance - amount)
106
107
                  END
108
              END
         END
109
110
111
         rewardTopDepositors(msg_sender, msg_value) =
112
         BEGIN
```

```
113
              VAR thisBalance, managerBalance IN
114
                 thisBalance <-- get_balanceOf(THIS);</pre>
115
                  managerBalance <-- get_balanceOf(manager_i);</pre>
116
                  IF msg_value = threshold_i &
                      msg_sender = manager_i &
117
118
                      index = threshold i &
                      donated_i = FALSE &
119
120
                      thisBalance + msg_value <= MAXINT &
121
                      managerBalance - msg_value >= 0
                  THEN
122
                      //* jj : NAT;
123
124
                      //* safe : BOOL;
125
                      VAR jj, safe IN
126
                          jj := 0;
127
                          safe := TRUE;
128
                          WHILE jj < index & safe = TRUE DO
129
                              VAR depositorBalance IN
                                   depositorBalance <-- get_accountOf(depositors_i(jj));</pre>
130
131
                                   safe := bool(depositorBalance + 1 <= MAXINT);</pre>
132
                                   jj := jj+ 1
                               END
133
134
                          INVARIANT 0<=index & jj<=index & jj>=0 &
135
                               safe = bool(!xx.(xx : ran((0..jj-1) < | depositors i) = >
     accountOf(xx) + 1 : NAT)) &
136
                               donated i= FALSE &
                               !xx.(xx : ran((0..jj-2) < | depositors_i) => accountOf(xx) + 1 :
137
     NAT)
138
                          VARIANT index - jj
139
                          END;
140
                          IF (safe=TRUE) THEN
141
142
                               transfer(msg_sender, THIS, msg_value);
143
                              donated_i := TRUE;
144
                               //* ii : NAT;
145
                              VAR ii, depositorBalance IN
146
                                   ii := 0;
147
                                   WHILE ii < index DO
148
                                       depositorBalance <-- get_accountOf(depositors_i(ii));</pre>
149
                                       set_accountOf(depositors_i (ii), depositorBalance + 1);
150
                                       ii := ii+ 1
151
                                   INVARIANT ii=threshold i or ii: dom(depositors i) &
152
                                       accountOf =
153
                                       accountOf$0<+(%xx. (xx : depositors_i[0..(ii-1)] |</pre>
     accountOf\$0(xx) + 1)) &
154
                                       threshold_i = threshold &
155
                                       donated_i = TRUE & safe= TRUE &
                                       depositors i[0..(ii-1)]<: depositors &
156
157
                                       !xx.(xx : ran((ii+1..index-1) < | depositors_i) =>
158
     accountOf(xx) + 1 : NAT)
159
                                   VARIANT index - ii
160
                                   END
                               END
161
                          END
162
163
                      END
                  END
164
              END
165
166
         END
     END
167
```

Platform.mch

```
1
 2
    MACHINE
 3
        Platform
 4
    SEES
 5
        Solidity_Types
 6
    ABSTRACT_VARIABLES
 7
        balanceOf
 8
    INVARIANT
 9
        balanceOf : ADDRESS --> NAT
10
    INITIALISATION
11
       balanceOf :: ADDRESS --> NAT
12
    OPERATIONS
13
14
        // Used in animating model in proB.
15
        addRandomAmountToBalance =
16
        BEGIN
17
            ANY amount_, xx WHERE xx : USERS & amount_ : 1..3 & balanceOf(xx) + amount_ :
    NAT THEN
18
                balanceOf(xx) := balanceOf(xx) + amount_
19
            END
20
        END
21
22
        transfer ( from , to , amount ) =
23
            from : ADDRESS &
24
25
            to: ADDRESS &
26
            to /= from &
27
            amount : NAT &
28
            ( balanceOf ( to ) + amount ) : NAT &
            ( balanceOf ( from ) - amount ) : NAT
29
30
31
            balanceOf := balanceOf <+ { from |-> ( balanceOf ( from ) - amount ) , to |-> (
    balanceOf ( to ) + amount ) }
32
        END
33
34
        transfer_abstract(updates) =
35
36
           updates : ADDRESS +-> NAT
37
        THEN
38
            balanceOf := balanceOf <+ updates</pre>
        END
39
40
        ret <-- get_balanceOf ( adr ) =
41
42
        PRE
43
            adr : ADDRESS
44
        THEN
45
        ret := balanceOf ( adr )
        END
46
47
        // Test if model is vulnerable to ForceFeeding
48
49
        ForceFeeding (amount) =
        PRE amount : NAT & balanceOf(THIS) + amount : NAT
50
51
            balanceOf := balanceOf <+ {THIS |-> ( balanceOf ( THIS ) + amount )}
52
53
        END
54
        ;
```

```
transfer_(balanceUpdates) =
pre
balanceUpdates : ADDRESS +-> NAT

THEN
balanceOf := balanceOf <+ balanceUpdates
balanceOf := balanceOf <+ balanceUpdates
balanceOf := balanceOf <+ balanceUpdates</pre>
```

Solidity_Types.mch

```
1
 2
   MACHINE
 3
        Solidity_Types
 4
   SETS
 5
        ADDRESS = {addr_0, THIS, addr_1, addr_2, addr_3}; BYTES
 6 CONSTANTS
 7
        init_msg_sender, init_msg_value, USERS, init_block_timestamp
    PROPERTIES
    \label{eq:USERS} $$ - {THIS, addr_0} \& init_msg\_sender : USERS \& init_msg\_value : NAT \& init\_block\_timestamp : NAT \\
 9
10
11
12
   END
```

1 of 1 12/14/2023, 5:10 PM

account.mch

```
1
 2
   MACHINE
 3
        account
 4
   SEES
 5
       Solidity_Types
 6 VARIABLES
 7
       account0f
   INVARIANT
 9
        accountOf : ADDRESS --> NAT
10
   INITIALISATION
        accountOf := ADDRESS * {0}
11
12
13
   OPERATIONS
14
15
        ret <-- get_accountOf(key) =</pre>
16
        PRE
17
            key : ADDRESS
18
        THEN
            ret := accountOf(key)
19
20
        END
21
22
        set_accountOf_abstract(updates) =
23
        PRE
24
            updates : ADDRESS +-> NAT
25
26
            accountOf := accountOf <+ updates</pre>
27
28
        END
29
        ;
30
        set_accountOf(key, value) =
31
32
            key : ADDRESS & value : NAT
33
        THEN
            accountOf(key) := value
34
35
        END
36
        ;
37
38
        ret <-- get_account = ret := accountOf
39
40
    END
41
```

1 of 1 12/14/2023, 5:08 PM

allowance.mch

```
1
 2
    MACHINE
 3
        allowance
 4
 5
   SEES
 6
        Solidity_Types
 7
 8
   VARIABLES
 9
        allowanceOf
10
    INVARIANT
        allowanceOf : ADDRESS --> ( ADDRESS--> NAT )
11
12
13
    INITIALISATION
14
        allowanceOf :: (ADDRESS --> ( ADDRESS --> NAT ))
15
16
17
    OPERATIONS
18
        set_allowanceOf_abstract( key, updates ) =
19
20
            key : ADDRESS & updates : ADDRESS +-> NAT
21
22
        THEN
            allowanceOf(key) := allowanceOf(key) <+ updates</pre>
23
24
        END
25
        ;
26
27
28
        ret <-- get_allowanceOf ( key1, key2 ) =</pre>
29
        PRE
30
            key1 : ADDRESS & key2 : ADDRESS
31
        THEN
32
            ret := allowanceOf ( key1 )( key2 )
33
        END
34
35
36
        set_allowanceOf(key1, key2, value) =
37
            key1: ADDRESS & key2 : ADDRESS & value: NAT
38
39
        THEN
            allowanceOf(key1)(key2) := value
40
        END
41
42
43
44 END
```

1 of 1 12/14/2023, 5:09 PM

depositedOver100.mch

```
1
 2
   MACHINE
 3
        depositedOver100
 4
   SEES
 5
        Solidity_Types
   VARIABLES
 6
 7
        depositedOver_100
   INVARIANT
 9
        depositedOver_100: ADDRESS --> BOOL
10
    INITIALISATION
        depositedOver_100 := ADDRESS * {FALSE}
11
12
13
    OPERATIONS
14
15
         set_depositedOver_100_abstract(updates) =
         PRE updates : ADDRESS +-> BOOL
16
17
            depositedOver_100 := depositedOver_100 <+ updates</pre>
18
         END
19
20
21
22
         set_depositedOver_100(key, value) =
23
         PRE key: ADDRESS & value: BOOL
24
25
            depositedOver_100(key) := value
26
         END
27
28
29
        ret <-- get_depositedOver_100(key) =</pre>
30
        PRE key : ADDRESS
31
32
            ret := depositedOver_100(key)
33
        END
34
    END
35
```

1 of 1 12/14/2023, 5:09 PM