

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 &
16    donated : BOOL &
17    balanceOf(THIS) >=
18    (SIGMA(ct).(ct : dom(accountOf)| accountOf(ct))) &
19    card(depositors) <= threshold
20
21 INITIALISATION manager := init_msg_sender || depositors := {} || donated := FALSE
22 OPERATIONS
23     deposit(msg_sender, msg_value) =
24     PRE
25         msg_sender : USERS & msg_value : NAT1
26     THEN
27         IF balanceOf(msg_sender) - msg_value : NAT & accountOf(msg_sender) + msg_value
: NAT &
28         balanceOf(THIS) + msg_value: NAT THEN
29             transfer(msg_sender, THIS, msg_value)||
30             set_accountOf_abstract({msg_sender
31                 |-> accountOf(msg_sender) + msg_value}) ||
32             IF (accountOf(msg_sender) + msg_value >= threshold )
33             & msg_sender /: depositors & card(depositors) < threshold
34             THEN
35                 depositors := depositors \ / {msg_sender}
36             END
37         END
38     END
39 ;
40 withdraw(msg_sender, amount) =
41 PRE
42     msg_sender : USERS & amount: NAT1
43 THEN
44     IF accountOf(msg_sender) >= amount & balanceOf(msg_sender) + amount: NAT &
45     balanceOf(THIS) - amount : NAT THEN
46         transfer(THIS , msg_sender, amount)||
47         set_accountOf_abstract({msg_sender |-> accountOf(msg_sender) - amount})
48     END
49 END
50 ;
51 transferTo(msg_sender, dst, amount) =
52 PRE
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
60         set_accountOf_abstract({msg_sender |-> accountOf(msg_sender)
61         - amount, dst |-> accountOf(dst) + amount })
62     END
63 END
64 ;
65 approve(msg_sender, dst, amount)=
66 PRE
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
72 END
73 ;
74 transferFrom(msg_sender, sender, recipient, amount ) =
75 PRE
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 &
82     accountOf(recipient) + amount : NAT &
83     allowanceOf(sender)(msg_sender) - amount : NAT
84 THEN
85     set_accountOf_abstract({recipient |->
86     accountOf(recipient) + amount, sender
87     |-> accountOf(sender) - amount})
88     ||
89     set_allowanceOf_abstract(sender, {msg_sender
90     |-> allowanceOf(sender)(msg_sender) - amount})
91 END
92 END
93 ;
94 rewardTopDepositors(msg_sender, msg_value) =
95 PRE
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 &
104     !xx.(xx : depositors => accountOf(xx) + 1 : NAT)
105 THEN
106     transfer(manager, THIS, msg_value) ||
107     set_accountOf_abstract(%xx. (xx : depositors | accountOf(xx) + 1)) ||
108     donated := TRUE
109 END
110 END
111 END

```

B_weth_i.imp

```
1  /* B_weth_i
2  * Author: ASUS
3  * Creation date: 8/31/2023
4  */
5
6  IMPLEMENTATION B_weth_i
7  REFINES B_weth
8  SEES Solidity_Types
9  IMPORTS Platform, account, allowance, depositedOver100
10 CONCRETE_CONSTANTS
11     threshold_i
12 PROPERTIES
13     threshold_i : NAT & threshold_i = threshold
14 VALUES
15     threshold_i = 2
16 CONCRETE_VARIABLES manager_i, depositors_i, index, donated_i
17 INVARIANT
18     index : NAT & index >= 0 & donated_i : BOOL &
19     depositors_i : 0..threshold_i --> ADDRESS &
20     manager_i : USERS & manager_i = manager & donated_i = donated &
21     index = card(depositors) &
22     depositors_i[0..index-1] = depositors &
23     (0..index-1) <| depositors_i : 0..index-1 --> depositors &
24     depositedOver_100~[{TRUE}] = depositors
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);
37             senderBalance <-- get_balanceOf(msg_sender);
38             thisBalance <-- get_balanceOf(THIS);
39
40             IF thisBalance + msg_value <= MAXINT & senderBalance - msg_value >= 0 &
41                 senderAccount + msg_value <= MAXINT
42             THEN
43                 set_accountOf(msg_sender, senderAccount + msg_value);
44                 transfer(msg_sender, THIS, msg_value);
45                 VAR distinct IN
46                     distinct <-- get_depositedOver_100(msg_sender);
47                     IF senderAccount + msg_value >= threshold_i & distinct = FALSE &
index < threshold_i
48                 THEN
49                     depositors_i(index) := msg_sender;
50                     set_depositedOver_100(msg_sender, TRUE);
51                     index := index + 1
52                 END
53             END
54         END
55     END
```

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

```

113     VAR thisBalance, managerBalance IN
114     thisBalance <-- get_balanceOf(THIS);
115     managerBalance <-- get_balanceOf(manager_i);
116     IF msg_value = threshold_i &
117     msg_sender = manager_i &
118     index = threshold_i &
119     donated_i = FALSE &
120     thisBalance + msg_value <= MAXINT &
121     managerBalance - msg_value >= 0
122     THEN
123         /* jj : NAT;
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
130             depositorBalance <-- get_accountOf(depositors_i(jj));
131             safe := bool(depositorBalance + 1 <= MAXINT);
132             jj := jj+ 1
133         END
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 &
137         !xx.(xx : ran((0..jj-2) <| depositors_i) => accountOf(xx) + 1 :
NAT)
138         VARIANT index - jj
139         END;
140
141         IF (safe=TRUE) THEN
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));
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)] |
accountOf$0(xx) + 1)) &
154             threshold_i = threshold &
155             donated_i = TRUE & safe= TRUE &
156             depositors_i[0..(ii-1)]<: depositors &
157             jj=index &
158             !xx.(xx : ran((ii+1..index-1) <| depositors_i) =>
accountOf(xx) + 1 : NAT)
159             VARIANT index - ii
160             END
161         END
162     END
163 END
164 END
165 END
166 END
167 END

```

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   PRE
24     from : ADDRESS &
25     to : ADDRESS &
26     to /= from &
27     amount : NAT &
28     ( balanceOf ( to ) + amount ) : NAT &
29     ( balanceOf ( from ) - amount ) : NAT
30   THEN
31     balanceOf := balanceOf <+ { from |-> ( balanceOf ( from ) - amount ) , to |-> (
balanceOf ( to ) + amount ) }
32   END
33 ;
34 transfer_abstract(updates) =
35   PRE
36     updates : ADDRESS +-> NAT
37   THEN
38     balanceOf := balanceOf <+ updates
39   END
40 ;
41 ret <-- get_balanceOf ( adr ) =
42   PRE
43     adr : ADDRESS
44   THEN
45     ret := balanceOf ( adr )
46   END
47 ;
48 // Test if model is vulnerable to ForceFeeding
49 ForceFeeding (amount) =
50   PRE amount : NAT & balanceOf(THIS) + amount : NAT
51   THEN
52     balanceOf := balanceOf <+ {THIS |-> ( balanceOf ( THIS ) + amount )}
53   END
54 ;

```

```
55     transfer_(balanceUpdates) =  
56     PRE  
57         balanceUpdates : ADDRESS +-> NAT  
58     THEN  
59         balanceOf := balanceOf <+ balanceUpdates  
60     END  
61 END
```

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
8 | PROPERTIES
9 |     USERS = ADDRESS - {THIS, addr_0} & init_msg_sender : USERS & init_msg_value : NAT &
    init_block_timestamp : NAT
10 |
11 |
12 | END
```


account.mch

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

allowance.mch

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

depositedOver100.mch

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