CertiK Audit Report For BitMovio

Request Date: 2019-06-25
Revision Date: 2019-07-10
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and BitMovio(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

For more information: https://certik.org/

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by BitMovio. This audit was conducted to discover issues and vulnerabilities in the source code of BitMovio's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issue found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

Low

The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

Testing Summary

PASS

CERTIK *believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

Jul 10, 2019

Score
100

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No address check for transfer functions, which may lead to possible value loss.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **MoviBits.sol** (rinkeby $0xe7c4c20ab7ba26a464b2f6519ffe309acdaf0cad$)
  `a4ddfb590b139205921c9e05c3e562273189b598ba8c5d0b322f3e807889ec95`

Summary

CertiK was chosen by BitMovio to audit the design and implementation of its MoviBits smart contract. To ensure comprehensive protection,the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Discussion

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

MoviBit.sol (rinkeby $0x25fae2c33648fcc080d1078f8bebaae074a3b4df$, previous)

- IMPORTANT Recommend adding zero address check in `transfer()`, `transferFrom()` to prevent possible asset loss, since the total supply is fixed and there is no way to mint tokens after initialization.
  - (BitMovio - Resolved in latest update rinkeby $0xe7c4c20ab7ba26a464b2f6519ffe309acdaf0cad$)
- INFO Recommend supplementing error message for `require()` checks.
  - (BitMovio - Resolved in latest update rinkeby $0xe7c4c20ab7ba26a464b2f6519ffe309acdaf0cad$)

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 5 in File MoviBits.sol

```
5 pragma solidity 0.5.9;
```

Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 5 in File MoviBit.sol

```
5 pragma solidity 0.5.9;
```

Only these compiler versions are safe to compile your code: 0.5.9

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30  /*@CTK FAIL "transferFrom to same address"
31      @tag assume_completion
32      @pre from == to
33      @post __post.allowed[from][msg.sender] ==
34  */
```

Raw code location	Line 35-41 in File howtoread.sol

Raw code

```
35  function transferFrom(address from, address to
        ) {
36      balances[from] = balances[from].sub(tokens
37      allowed[from][msg.sender] = allowed[from][
38      balances[to] = balances[to].add(tokens);
39      emit Transfer(from, to, tokens);
40      return true;
41  }
```

Counterexample This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
53         balance: 0x0
54     }
55 }
```

Post environment

```
56
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

SafeMath add

10, Jul 2019
12.71 ms

Line 18-26 in File MoviBits.sol

```
18  /*@CTK "SafeMath add"
19    @tag spec
20    @tag is_pure
21    @post (a + b < a || a + b < b) == __reverted
22    @post !__reverted -> c == a + b
23    @post !__reverted -> !__has_overflow
24    @post !__reverted -> !__has_assertion_failure
25    @post !(__has_buf_overflow)
26  */
```

Line 27-30 in File MoviBits.sol

```
27  function add(uint a, uint b) internal pure returns (uint c) {
28      c = a + b;
29      require(c >= a);
30  }
```

The code meets the specification.

Formal Verification Request 2

SafeMath sub

10, Jul 2019
9.72 ms

Line 31-39 in File MoviBits.sol

```
31  /*@CTK "SafeMath sub"
32    @tag spec
33    @tag is_pure
34    @post (b > a) == __reverted
35    @post !__reverted -> c == a - b
36    @post !__reverted -> !__has_overflow
37    @post !__reverted -> !__has_assertion_failure
38    @post !(__has_buf_overflow)
39  */
```

Line 40-43 in File MoviBits.sol

```
40  function sub(uint a, uint b) internal pure returns (uint c) {
41      require(b <= a);
42      c = a - b;
43  }
```

The code meets the specification.

Formal Verification Request 3

SafeMath mul zero

10, Jul 2019
11.62 ms

Line 44-49 in File MoviBits.sol

```
44  /*@CTK "SafeMath mul zero"
45    @tag spec
46    @tag is_pure
47    @pre (a == 0)
48    @post c == 0
49  */
```

Line 60-63 in File MoviBits.sol

```
60  function mul(uint a, uint b) internal pure returns (uint c) {
61      c = a * b;
62      require(a == 0 || c / a == b);
63  }
```

The code meets the specification.

Formal Verification Request 4

SafeMath mul nonzero

10, Jul 2019
3.51 ms

Line 50-59 in File MoviBits.sol

```
50  /*@CTK "SafeMath mul nonzero"
51    @tag spec
52    @tag is_pure
53    @pre (a != 0)
54    @post (a * b / a != b) == __reverted
55    @post !__reverted -> c == a * b
56    @post !__reverted -> !__has_overflow
57    @post !__reverted -> !__has_assertion_failure
58    @post !(__has_buf_overflow)
59  */
```

Line 60-63 in File MoviBits.sol

```
60  function mul(uint a, uint b) internal pure returns (uint c) {
61      c = a * b;
62      require(a == 0 || c / a == b);
63  }
```

The code meets the specification.

Formal Verification Request 5

SafeMath div

10, Jul 2019
11.76 ms

Line 64-72 in File MoviBits.sol

```
64  /*@CTK "SafeMath div"
65    @tag spec
66    @tag is_pure
67    @post (b == 0) == __reverted
68    @post !__reverted -> c == a / b
69    @post !__reverted -> !__has_overflow
70    @post !__reverted -> !__has_assertion_failure
71    @post !(__has_buf_overflow)
72  */
```

Line 73-76 in File MoviBits.sol

```
73  function div(uint a, uint b) internal pure returns (uint c) {
74      require(b > 0);
75      c = a / b;
76  }
```

The code meets the specification.

Formal Verification Request 6

transferOwnership

10, Jul 2019
17.51 ms

Line 119-124 in File MoviBits.sol

```
119  /*@CTK transferOwnership
120    @tag assume_completion
121    @pre msg.sender == owner
122    @pre _newOwner != address(0)
123    @post __post.newOwner == _newOwner
124   */
```

Line 125-127 in File MoviBits.sol

```
125  function transferOwnership(address payable _newOwner) public onlyOwner {
126      newOwner = _newOwner;
127  }
```

The code meets the specification.

Formal Verification Request 7

Owned_transferOwnership

10, Jul 2019
1.28 ms

Line 110-113 in File MoviBits.sol

```
110 /*@CTK Owned
111   @tag assume_completion
112   @post __post.owner == msg.sender
113  */
```

Line 125-127 in File MoviBits.sol

```
125 function transferOwnership(address payable _newOwner) public onlyOwner {
126     newOwner = _newOwner;
127 }
```

✅ The code meets the specification.

Formal Verification Request 8

acceptOwnership

📅 10, Jul 2019
⏱ 20.77 ms

Line 129-134 in File MoviBits.sol

```
129 /*@CTK acceptOwnership
130   @tag assume_completion
131   @pre msg.sender == newOwner
132   @post __post.owner == newOwner
133   @post __post.newOwner == address(0)
134  */
```

Line 135-140 in File MoviBits.sol

```
135 function acceptOwnership() public {
136     require(msg.sender == newOwner, "Calling address is not the new contract owner"
            );
137     emit OwnershipTransferred(owner, newOwner);
138     owner = newOwner;
139     newOwner = address(0);
140 }
```

✅ The code meets the specification.

Formal Verification Request 9

MoviBit

📅 10, Jul 2019
⏱ 25.48 ms

Line 161-164 in File MoviBits.sol

```
161 /*@CTK MoviBit
162   @tag assume_completion
163   @post __post.balances[__post.owner] == __post._totalSupply
164 */
```

Line 165-172 in File MoviBits.sol

```
165    constructor() public {
166        symbol = "MVBIT";
167        name = "MoviBits Token";
168        decimals = 18;
169        _totalSupply = 1000000000 * 10**uint(decimals);
170        balances[owner] = _totalSupply;
171        emit Transfer(address(0), owner, _totalSupply);
172    }
```

The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

10, Jul 2019
15.82 ms

Line 178 in File MoviBits.sol

```
178    //@CTK NO_OVERFLOW
```

Line 184-186 in File MoviBits.sol

```
184    function totalSupply() public view returns (uint) {
185        return _totalSupply.sub(balances[address(0)]);
186    }
```

The code meets the specification.

Formal Verification Request 11

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.45 ms

Line 179 in File MoviBits.sol

```
179    //@CTK NO_BUF_OVERFLOW
```

Line 184-186 in File MoviBits.sol

```
184    function totalSupply() public view returns (uint) {
185        return _totalSupply.sub(balances[address(0)]);
186    }
```

The code meets the specification.

Formal Verification Request 12

totalSupply correctness

10, Jul 2019
7.38 ms

Line 180-183 in File MoviBits.sol

```
180 /*@CTK "totalSupply correctness"
181   @tag assume_completion
182   @post __return <= _totalSupply
183  */
```

Line 184-186 in File MoviBits.sol

```
184 function totalSupply() public view returns (uint) {
185     return _totalSupply.sub(balances[address(0)]);
186 }
```

The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

10, Jul 2019
5.9 ms

Line 192 in File MoviBits.sol

```
192 //@CTK NO_OVERFLOW
```

Line 198-200 in File MoviBits.sol

```
198 function balanceOf(address tokenOwner) public view returns (uint balance) {
199     return balances[tokenOwner];
200 }
```

The code meets the specification.

Formal Verification Request 14

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.32 ms

Line 193 in File MoviBits.sol

```
193 //@CTK NO_BUF_OVERFLOW
```

Line 198-200 in File MoviBits.sol

```
198 function balanceOf(address tokenOwner) public view returns (uint balance) {
199     return balances[tokenOwner];
200 }
```

The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

10, Jul 2019
0.32 ms

Line 194 in File MoviBits.sol

```
194    //@CTK NO_ASF
```

Line 198-200 in File MoviBits.sol

```
198    function balanceOf(address tokenOwner) public view returns (uint balance) {
199        return balances[tokenOwner];
200    }
```

The code meets the specification.

Formal Verification Request 16

balanceOf correctness

10, Jul 2019
0.33 ms

Line 195-197 in File MoviBits.sol

```
195    /*@CTK "balanceOf correctness"
196      @post balance == __post.balances[tokenOwner]
197     */
```

Line 198-200 in File MoviBits.sol

```
198    function balanceOf(address tokenOwner) public view returns (uint balance) {
199        return balances[tokenOwner];
200    }
```

The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

10, Jul 2019
45.48 ms

Line 208 in File MoviBits.sol

```
208    //@CTK NO_OVERFLOW
```

Line 226-232 in File MoviBits.sol

```
226    function transfer(address to, uint tokens) public returns (bool success) {
227        require(to != address(0), "Cannot send to 0x0 address");
228        balances[msg.sender] = balances[msg.sender].sub(tokens);
229        balances[to] = balances[to].add(tokens);
230        emit Transfer(msg.sender, to, tokens);
231        return true;
232    }
```

The code meets the specification.

Formal Verification Request 18

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
9.44 ms

Line 209 in File MoviBits.sol

```
209    //@CTK NO_BUF_OVERFLOW
```

Line 226-232 in File MoviBits.sol

```
226    function transfer(address to, uint tokens) public returns (bool success) {
227        require(to != address(0), "Cannot send to 0x0 address");
228        balances[msg.sender] = balances[msg.sender].sub(tokens);
229        balances[to] = balances[to].add(tokens);
230        emit Transfer(msg.sender, to, tokens);
231        return true;
232    }
```

The code meets the specification.

Formal Verification Request 19

transfer correctness

10, Jul 2019
46.44 ms

Line 210-217 in File MoviBits.sol

```
210    /*@CTK "transfer correctness"
211      @tag assume_completion
212      @pre msg.sender != to
213      @post to != address(0)
214      @post tokens <= balances[msg.sender]
215      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
216      @post __post.balances[to] == balances[to] + tokens
217     */
```

Line 226-232 in File MoviBits.sol

```
226    function transfer(address to, uint tokens) public returns (bool success) {
227        require(to != address(0), "Cannot send to 0x0 address");
228        balances[msg.sender] = balances[msg.sender].sub(tokens);
229        balances[to] = balances[to].add(tokens);
230        emit Transfer(msg.sender, to, tokens);
231        return true;
232    }
```

The code meets the specification.

Formal Verification Request 20

transfer self correctness

10, Jul 2019
37.74 ms

Line 218-225 in File MoviBits.sol

```
218   /*@CTK "transfer self correctness"
219     @tag assume_completion
220     @pre msg.sender == to
221     @post to != address(0)
222     @post tokens <= balances[msg.sender]
223     @post __post.balances[msg.sender] == balances[msg.sender]
224     @post __post.balances[to] == balances[to]
225    */
```

Line 226-232 in File MoviBits.sol

```
226   function transfer(address to, uint tokens) public returns (bool success) {
227       require(to != address(0), "Cannot send to 0x0 address");
228       balances[msg.sender] = balances[msg.sender].sub(tokens);
229       balances[to] = balances[to].add(tokens);
230       emit Transfer(msg.sender, to, tokens);
231       return true;
232   }
```

The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

10, Jul 2019
10.47 ms

Line 243 in File MoviBits.sol

```
243   //@CTK NO_OVERFLOW
```

Line 249-253 in File MoviBits.sol

```
249   function approve(address spender, uint tokens) public returns (bool success) {
250       allowed[msg.sender][spender] = tokens;
251       emit Approval(msg.sender, spender, tokens);
252       return true;
253   }
```

The code meets the specification.

Formal Verification Request 22

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.44 ms

Line 244 in File MoviBits.sol

```
244    //@CTK NO_BUF_OVERFLOW
```

Line 249-253 in File MoviBits.sol

```
249    function approve(address spender, uint tokens) public returns (bool success) {
250        allowed[msg.sender][spender] = tokens;
251        emit Approval(msg.sender, spender, tokens);
252        return true;
253    }
```

The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

10, Jul 2019
0.44 ms

Line 245 in File MoviBits.sol

```
245    //@CTK NO_ASF
```

Line 249-253 in File MoviBits.sol

```
249    function approve(address spender, uint tokens) public returns (bool success) {
250        allowed[msg.sender][spender] = tokens;
251        emit Approval(msg.sender, spender, tokens);
252        return true;
253    }
```

The code meets the specification.

Formal Verification Request 24

approve correctness

10, Jul 2019
1.7 ms

Line 246-248 in File MoviBits.sol

```
246    /*@CTK "approve correctness"
247      @post __post.allowed[msg.sender][spender] == tokens
248     */
```

Line 249-253 in File MoviBits.sol

```
249    function approve(address spender, uint tokens) public returns (bool success) {
250        allowed[msg.sender][spender] = tokens;
251        emit Approval(msg.sender, spender, tokens);
252        return true;
253    }
```

The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

10, Jul 2019
53.85 ms

Line 265 in File MoviBits.sol

```
265    //@CTK NO_OVERFLOW
```

Line 276-283 in File MoviBits.sol

```
276    function transferFrom(address from, address to, uint tokens) public returns (bool
           success) {
277        require(to != address(0), "Cannot send to 0x0 address");
278        balances[from] = balances[from].sub(tokens);
279        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280        balances[to] = balances[to].add(tokens);
281        emit Transfer(from, to, tokens);
282        return true;
283    }
```

The code meets the specification.

Formal Verification Request 26

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
14.88 ms

Line 266 in File MoviBits.sol

```
266    //@CTK NO_BUF_OVERFLOW
```

Line 276-283 in File MoviBits.sol

```
276    function transferFrom(address from, address to, uint tokens) public returns (bool
           success) {
277        require(to != address(0), "Cannot send to 0x0 address");
278        balances[from] = balances[from].sub(tokens);
279        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280        balances[to] = balances[to].add(tokens);
281        emit Transfer(from, to, tokens);
282        return true;
283    }
```

The code meets the specification.

Formal Verification Request 27

transferFrom correctness

10, Jul 2019
159.08 ms

Line 267-275 in File MoviBits.sol

```
267 /*@CTK "transferFrom correctness"
268   @tag assume_completion
269   @pre to != 0x0
270   @pre tokens <= balances[from] && tokens <= allowed[from][msg.sender]
271   @post to != from -> __post.balances[from] == balances[from] - tokens
272   @post to != from -> __post.balances[to] == balances[to] + tokens
273   @post to == from -> __post.balances[from] == balances[from]
274   @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
275  */
```

Line 276-283 in File MoviBits.sol

```
276 function transferFrom(address from, address to, uint tokens) public returns (bool
        success) {
277     require(to != address(0), "Cannot send to 0x0 address");
278     balances[from] = balances[from].sub(tokens);
279     allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280     balances[to] = balances[to].add(tokens);
281     emit Transfer(from, to, tokens);
282     return true;
283 }
```

The code meets the specification.

Formal Verification Request 28

If method completes, integer overflow would not happen.

10, Jul 2019
5.5 ms

Line 290 in File MoviBits.sol

```
290 //@CTK NO_OVERFLOW
```

Line 296-298 in File MoviBits.sol

```
296 function allowance(address tokenOwner, address spender) public view returns (uint
        remaining) {
297     return allowed[tokenOwner][spender];
298 }
```

The code meets the specification.

Formal Verification Request 29

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.33 ms

Line 291 in File MoviBits.sol

```
291 //@CTK NO_BUF_OVERFLOW
```

Line 296-298 in File MoviBits.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 30

Method will not encounter an assertion failure.

10, Jul 2019
0.32 ms

Line 292 in File MoviBits.sol

```
292     //@CTK NO_ASF
```

Line 296-298 in File MoviBits.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 31

allowance correctness

10, Jul 2019
0.35 ms

Line 293-295 in File MoviBits.sol

```
293     /*@CTK "allowance correctness"
294       @post remaining == allowed[tokenOwner][spender]
295      */
```

Line 296-298 in File MoviBits.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 32

If method completes, integer overflow would not happen.

10, Jul 2019
31.66 ms

Line 313 in File MoviBits.sol

```
313     //@CTK NO_OVERFLOW
```

Line 319-322 in File MoviBits.sol

```
319     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
320         approve(spender, allowed[msg.sender][spender].add(addedValue));
321         return true;
322     }
```

The code meets the specification.

Formal Verification Request 33

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.54 ms

Line 314 in File MoviBits.sol

```
314     //@CTK NO_BUF_OVERFLOW
```

Line 319-322 in File MoviBits.sol

```
319     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
320         approve(spender, allowed[msg.sender][spender].add(addedValue));
321         return true;
322     }
```

The code meets the specification.

Formal Verification Request 34

increaseAllowance correctness

10, Jul 2019
1.92 ms

Line 315-318 in File MoviBits.sol

```
315     /*@CTK "increaseAllowance correctness"
316       @tag assume_completion
317       @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] +
              addedValue
318      */
```

Line 319-322 in File MoviBits.sol

```
319     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
320         approve(spender, allowed[msg.sender][spender].add(addedValue));
321         return true;
322     }
```

The code meets the specification.

Formal Verification Request 35

If method completes, integer overflow would not happen.

10, Jul 2019
29.55 ms

Line 339 in File MoviBits.sol

```
339    //@CTK NO_OVERFLOW
```

Line 351-354 in File MoviBits.sol

```
351    function decreaseAllowance(address spender, uint256 subtractedValue) public
           returns (bool) {
352        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353        return true;
354    }
```

The code meets the specification.

Formal Verification Request 36

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.55 ms

Line 340 in File MoviBits.sol

```
340    //@CTK NO_BUF_OVERFLOW
```

Line 351-354 in File MoviBits.sol

```
351    function decreaseAllowance(address spender, uint256 subtractedValue) public
           returns (bool) {
352        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353        return true;
354    }
```

The code meets the specification.

Formal Verification Request 37

decreaseApproval0

10, Jul 2019
13.27 ms

Line 341-345 in File MoviBits.sol

```
341    /*@CTK decreaseApproval0
342      @pre __return == true
343      @pre allowed[msg.sender][spender] <= subtractedValue
344      @post __post.allowed[msg.sender][spender] == 0
345     */
```

Line 351-354 in File MoviBits.sol

```
351 function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
352     approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353     return true;
354 }
```

The code meets the specification.

Formal Verification Request 38

decreaseApproval

10, Jul 2019
2.36 ms

Line 346-350 in File MoviBits.sol

```
346 /*@CTK decreaseApproval
347   @pre __return == true
348   @pre allowed[msg.sender][spender] > subtractedValue
349   @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
        subtractedValue
350  */
```

Line 351-354 in File MoviBits.sol

```
351 function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
352     approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353     return true;
354 }
```

The code meets the specification.

Formal Verification Request 39

Owned_Owned__transferOwnership

10, Jul 2019
19.09 ms

Line 110-113 in File MoviBits.sol

```
110 /*@CTK Owned
111   @tag assume_completion
112   @post __post.owner == msg.sender
113  */
```

(Inheritance) Line 125-127 in File MoviBits.sol

```
125 function transferOwnership(address payable _newOwner) public onlyOwner {
126     newOwner = _newOwner;
127 }
```

The code meets the specification.

Formal Verification Request 40

Owned_transferOwnership

10, Jul 2019
19.0 ms

Line 110-113 in File MoviBits.sol

```
110   /*@CTK Owned
111     @tag assume_completion
112     @post __post.owner == msg.sender
113    */
```

(Inheritance) Line 125-127 in File MoviBits.sol

```
125   function transferOwnership(address payable _newOwner) public onlyOwner {
126       newOwner = _newOwner;
127   }
```

The code meets the specification.

Formal Verification Request 41

SafeMath add

10, Jul 2019
12.52 ms

Line 18-26 in File MoviBit.sol

```
18   /*@CTK "SafeMath add"
19     @tag spec
20     @tag is_pure
21     @post (a + b < a || a + b < b) == __reverted
22     @post !__reverted -> c == a + b
23     @post !__reverted -> !__has_overflow
24     @post !__reverted -> !__has_assertion_failure
25     @post !(__has_buf_overflow)
26   */
```

Line 27-30 in File MoviBit.sol

```
27   function add(uint a, uint b) internal pure returns (uint c) {
28       c = a + b;
29       require(c >= a);
30   }
```

The code meets the specification.

Formal Verification Request 42

SafeMath sub

10, Jul 2019
9.5 ms

Line 31-39 in File MoviBit.sol

```
31  /*@CTK "SafeMath sub"
32    @tag spec
33    @tag is_pure
34    @post (b > a) == __reverted
35    @post !__reverted -> c == a - b
36    @post !__reverted -> !__has_overflow
37    @post !__reverted -> !__has_assertion_failure
38    @post !(__has_buf_overflow)
39  */
```

Line 40-43 in File MoviBit.sol

```
40  function sub(uint a, uint b) internal pure returns (uint c) {
41      require(b <= a);
42      c = a - b;
43  }
```

The code meets the specification.

Formal Verification Request 43

SafeMath mul zero

10, Jul 2019
11.42 ms

Line 44-49 in File MoviBit.sol

```
44  /*@CTK "SafeMath mul zero"
45    @tag spec
46    @tag is_pure
47    @pre (a == 0)
48    @post c == 0
49  */
```

Line 60-63 in File MoviBit.sol

```
60  function mul(uint a, uint b) internal pure returns (uint c) {
61      c = a * b;
62      require(a == 0 || c / a == b);
63  }
```

The code meets the specification.

Formal Verification Request 44

SafeMath mul nonzero

10, Jul 2019
3.7 ms

Line 50-59 in File MoviBit.sol

```
50  /*@CTK "SafeMath mul nonzero"
51    @tag spec
52    @tag is_pure
53    @pre (a != 0)
```

```
54      @post (a * b / a != b) == __reverted
55      @post !__reverted -> c == a * b
56      @post !__reverted -> !__has_overflow
57      @post !__reverted -> !__has_assertion_failure
58      @post !(__has_buf_overflow)
59    */
```

Line 60-63 in File MoviBit.sol

```
60    function mul(uint a, uint b) internal pure returns (uint c) {
61        c = a * b;
62        require(a == 0 || c / a == b);
63    }
```

The code meets the specification.

Formal Verification Request 45

SafeMath div

10, Jul 2019
10.71 ms

Line 64-72 in File MoviBit.sol

```
64    /*@CTK "SafeMath div"
65      @tag spec
66      @tag is_pure
67      @post (b == 0) == __reverted
68      @post !__reverted -> c == a / b
69      @post !__reverted -> !__has_overflow
70      @post !__reverted -> !__has_assertion_failure
71      @post !(__has_buf_overflow)
72    */
```

Line 73-76 in File MoviBit.sol

```
73    function div(uint a, uint b) internal pure returns (uint c) {
74        require(b > 0);
75        c = a / b;
76    }
```

The code meets the specification.

Formal Verification Request 46

transferOwnership

10, Jul 2019
15.18 ms

Line 119-124 in File MoviBit.sol

```
119   /*@CTK transferOwnership
120     @tag assume_completion
121     @pre msg.sender == owner
122     @pre _newOwner != address(0)
```

```
123     @post __post.newOwner == _newOwner
124    */
```

Line 125-127 in File MoviBit.sol

```
125    function transferOwnership(address payable _newOwner) public onlyOwner {
126        newOwner = _newOwner;
127    }
```

The code meets the specification.

Formal Verification Request 47

Owned_transferOwnership

10, Jul 2019
1.09 ms

Line 110-113 in File MoviBit.sol

```
110    /*@CTK Owned
111      @tag assume_completion
112      @post __post.owner == msg.sender
113     */
```

Line 125-127 in File MoviBit.sol

```
125    function transferOwnership(address payable _newOwner) public onlyOwner {
126        newOwner = _newOwner;
127    }
```

The code meets the specification.

Formal Verification Request 48

acceptOwnership

10, Jul 2019
16.92 ms

Line 129-134 in File MoviBit.sol

```
129    /*@CTK acceptOwnership
130      @tag assume_completion
131      @pre msg.sender == newOwner
132      @post __post.owner == newOwner
133      @post __post.newOwner == address(0)
134     */
```

Line 135-140 in File MoviBit.sol

```
135    function acceptOwnership() public {
136        require(msg.sender == newOwner, "Calling address is not the new contract owner"
               );
137        emit OwnershipTransferred(owner, newOwner);
138        owner = newOwner;
139        newOwner = address(0);
140    }
```

The code meets the specification.

Formal Verification Request 49

MoviBit

10, Jul 2019
21.47 ms

Line 161-164 in File MoviBit.sol

```
161 /*@CTK MoviBit
162   @tag assume_completion
163   @post __post.balances[__post.owner] == __post._totalSupply
164 */
```

Line 165-172 in File MoviBit.sol

```
165 constructor() public {
166     symbol = "MVBIT";
167     name = "MoviBits Token";
168     decimals = 18;
169     _totalSupply = 1000000000 * 10**uint(decimals);
170     balances[owner] = _totalSupply;
171     emit Transfer(address(0), owner, _totalSupply);
172 }
```

The code meets the specification.

Formal Verification Request 50

If method completes, integer overflow would not happen.

10, Jul 2019
12.82 ms

Line 178 in File MoviBit.sol

```
178 //@CTK NO_OVERFLOW
```

Line 184-186 in File MoviBit.sol

```
184 function totalSupply() public view returns (uint) {
185     return _totalSupply.sub(balances[address(0)]);
186 }
```

The code meets the specification.

Formal Verification Request 51

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.37 ms

Line 179 in File MoviBit.sol

```
179     //@CTK NO_BUF_OVERFLOW
```

Line 184-186 in File MoviBit.sol

```
184     function totalSupply() public view returns (uint) {
185         return _totalSupply.sub(balances[address(0)]);
186     }
```

The code meets the specification.

Formal Verification Request 52

totalSupply correctness

10, Jul 2019
5.87 ms

Line 180-183 in File MoviBit.sol

```
180     /*@CTK "totalSupply correctness"
181       @tag assume_completion
182       @post __return <= _totalSupply
183      */
```

Line 184-186 in File MoviBit.sol

```
184     function totalSupply() public view returns (uint) {
185         return _totalSupply.sub(balances[address(0)]);
186     }
```

The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

10, Jul 2019
4.72 ms

Line 192 in File MoviBit.sol

```
192     //@CTK NO_OVERFLOW
```

Line 198-200 in File MoviBit.sol

```
198     function balanceOf(address tokenOwner) public view returns (uint balance) {
199         return balances[tokenOwner];
200     }
```

The code meets the specification.

Formal Verification Request 54

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.27 ms

Line 193 in File MoviBit.sol

```
193    //@CTK NO_BUF_OVERFLOW
```

Line 198-200 in File MoviBit.sol

```
198    function balanceOf(address tokenOwner) public view returns (uint balance) {
199        return balances[tokenOwner];
200    }
```

The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

10, Jul 2019
0.35 ms

Line 194 in File MoviBit.sol

```
194    //@CTK NO_ASF
```

Line 198-200 in File MoviBit.sol

```
198    function balanceOf(address tokenOwner) public view returns (uint balance) {
199        return balances[tokenOwner];
200    }
```

The code meets the specification.

Formal Verification Request 56

balanceOf correctness

10, Jul 2019
0.3 ms

Line 195-197 in File MoviBit.sol

```
195    /*@CTK "balanceOf correctness"
196      @post balance == __post.balances[tokenOwner]
197     */
```

Line 198-200 in File MoviBit.sol

```
198    function balanceOf(address tokenOwner) public view returns (uint balance) {
199        return balances[tokenOwner];
200    }
```

The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

10, Jul 2019
37.81 ms

Line 208 in File MoviBit.sol

```
208    //@CTK NO_OVERFLOW
```

Line 226-232 in File MoviBit.sol

```
226    function transfer(address to, uint tokens) public returns (bool success) {
227        require(to != address(0), "Cannot send to 0x0 address");
228        balances[msg.sender] = balances[msg.sender].sub(tokens);
229        balances[to] = balances[to].add(tokens);
230        emit Transfer(msg.sender, to, tokens);
231        return true;
232    }
```

The code meets the specification.

Formal Verification Request 58

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
8.3 ms

Line 209 in File MoviBit.sol

```
209    //@CTK NO_BUF_OVERFLOW
```

Line 226-232 in File MoviBit.sol

```
226    function transfer(address to, uint tokens) public returns (bool success) {
227        require(to != address(0), "Cannot send to 0x0 address");
228        balances[msg.sender] = balances[msg.sender].sub(tokens);
229        balances[to] = balances[to].add(tokens);
230        emit Transfer(msg.sender, to, tokens);
231        return true;
232    }
```

The code meets the specification.

Formal Verification Request 59

transfer correctness

10, Jul 2019
38.78 ms

Line 210-217 in File MoviBit.sol

```
/*@CTK "transfer correctness"
  @tag assume_completion
  @pre msg.sender != to
  @post to != address(0)
  @post tokens <= balances[msg.sender]
  @post __post.balances[msg.sender] == balances[msg.sender] - tokens
  @post __post.balances[to] == balances[to] + tokens
 */
```

Line 226-232 in File MoviBit.sol

```
function transfer(address to, uint tokens) public returns (bool success) {
    require(to != address(0), "Cannot send to 0x0 address");
    balances[msg.sender] = balances[msg.sender].sub(tokens);
    balances[to] = balances[to].add(tokens);
    emit Transfer(msg.sender, to, tokens);
    return true;
}
```

The code meets the specification.

Formal Verification Request 60

transfer self correctness

10, Jul 2019
34.7 ms

Line 218-225 in File MoviBit.sol

```
/*@CTK "transfer self correctness"
  @tag assume_completion
  @pre msg.sender == to
  @post to != address(0)
  @post tokens <= balances[msg.sender]
  @post __post.balances[msg.sender] == balances[msg.sender]
  @post __post.balances[to] == balances[to]
 */
```

Line 226-232 in File MoviBit.sol

```
function transfer(address to, uint tokens) public returns (bool success) {
    require(to != address(0), "Cannot send to 0x0 address");
    balances[msg.sender] = balances[msg.sender].sub(tokens);
    balances[to] = balances[to].add(tokens);
    emit Transfer(msg.sender, to, tokens);
    return true;
}
```

The code meets the specification.

Formal Verification Request 61

If method completes, integer overflow would not happen.

10, Jul 2019
8.96 ms

Line 243 in File MoviBit.sol

```
243 //@CTK NO_OVERFLOW
```

Line 249-253 in File MoviBit.sol

```
249 function approve(address spender, uint tokens) public returns (bool success) {
250     allowed[msg.sender][spender] = tokens;
251     emit Approval(msg.sender, spender, tokens);
252     return true;
253 }
```

The code meets the specification.

Formal Verification Request 62

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.31 ms

Line 244 in File MoviBit.sol

```
244 //@CTK NO_BUF_OVERFLOW
```

Line 249-253 in File MoviBit.sol

```
249 function approve(address spender, uint tokens) public returns (bool success) {
250     allowed[msg.sender][spender] = tokens;
251     emit Approval(msg.sender, spender, tokens);
252     return true;
253 }
```

The code meets the specification.

Formal Verification Request 63

Method will not encounter an assertion failure.

10, Jul 2019
0.31 ms

Line 245 in File MoviBit.sol

```
245 //@CTK NO_ASF
```

Line 249-253 in File MoviBit.sol

```
249 function approve(address spender, uint tokens) public returns (bool success) {
250     allowed[msg.sender][spender] = tokens;
251     emit Approval(msg.sender, spender, tokens);
252     return true;
253 }
```

The code meets the specification.

Formal Verification Request 64

approve correctness

10, Jul 2019
1.65 ms

Line 246-248 in File MoviBit.sol

```
246 /*@CTK "approve correctness"
247   @post __post.allowed[msg.sender][spender] == tokens
248  */
```

Line 249-253 in File MoviBit.sol

```
249 function approve(address spender, uint tokens) public returns (bool success) {
250     allowed[msg.sender][spender] = tokens;
251     emit Approval(msg.sender, spender, tokens);
252     return true;
253 }
```

The code meets the specification.

Formal Verification Request 65

If method completes, integer overflow would not happen.

10, Jul 2019
49.44 ms

Line 265 in File MoviBit.sol

```
265 //@CTK NO_OVERFLOW
```

Line 276-283 in File MoviBit.sol

```
276 function transferFrom(address from, address to, uint tokens) public returns (bool
        success) {
277     require(to != address(0), "Cannot send to 0x0 address");
278     balances[from] = balances[from].sub(tokens);
279     allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280     balances[to] = balances[to].add(tokens);
281     emit Transfer(from, to, tokens);
282     return true;
283 }
```

The code meets the specification.

Formal Verification Request 66

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
16.52 ms

Line 266 in File MoviBit.sol

```
266     //@CTK NO_BUF_OVERFLOW
```

Line 276-283 in File MoviBit.sol

```
276     function transferFrom(address from, address to, uint tokens) public returns (bool
            success) {
277         require(to != address(0), "Cannot send to 0x0 address");
278         balances[from] = balances[from].sub(tokens);
279         allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280         balances[to] = balances[to].add(tokens);
281         emit Transfer(from, to, tokens);
282         return true;
283     }
```

The code meets the specification.

Formal Verification Request 67

transferFrom correctness

10, Jul 2019
143.77 ms

Line 267-275 in File MoviBit.sol

```
267     /*@CTK "transferFrom correctness"
268       @tag assume_completion
269       @pre to != 0x0
270       @pre tokens <= balances[from] && tokens <= allowed[from][msg.sender]
271       @post to != from -> __post.balances[from] == balances[from] - tokens
272       @post to != from -> __post.balances[to] == balances[to] + tokens
273       @post to == from -> __post.balances[from] == balances[from]
274       @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
275      */
```

Line 276-283 in File MoviBit.sol

```
276     function transferFrom(address from, address to, uint tokens) public returns (bool
            success) {
277         require(to != address(0), "Cannot send to 0x0 address");
278         balances[from] = balances[from].sub(tokens);
279         allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
280         balances[to] = balances[to].add(tokens);
281         emit Transfer(from, to, tokens);
282         return true;
283     }
```

The code meets the specification.

Formal Verification Request 68

If method completes, integer overflow would not happen.

10, Jul 2019
4.62 ms

Line 290 in File MoviBit.sol

```
290     //@CTK NO_OVERFLOW
```

Line 296-298 in File MoviBit.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 69

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.32 ms

Line 291 in File MoviBit.sol

```
291     //@CTK NO_BUF_OVERFLOW
```

Line 296-298 in File MoviBit.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 70

Method will not encounter an assertion failure.

10, Jul 2019
0.3 ms

Line 292 in File MoviBit.sol

```
292     //@CTK NO_ASF
```

Line 296-298 in File MoviBit.sol

```
296     function allowance(address tokenOwner, address spender) public view returns (uint
            remaining) {
297         return allowed[tokenOwner][spender];
298     }
```

The code meets the specification.

Formal Verification Request 71

allowance correctness

10, Jul 2019
0.32 ms

Line 293-295 in File MoviBit.sol

```
293 /*@CTK "allowance correctness"
294   @post remaining == allowed[tokenOwner][spender]
295  */
```

Line 296-298 in File MoviBit.sol

```
296 function allowance(address tokenOwner, address spender) public view returns (uint
        remaining) {
297     return allowed[tokenOwner][spender];
298 }
```

The code meets the specification.

Formal Verification Request 72

If method completes, integer overflow would not happen.

10, Jul 2019
25.43 ms

Line 313 in File MoviBit.sol

```
313 //@CTK NO_OVERFLOW
```

Line 319-322 in File MoviBit.sol

```
319 function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
320     approve(spender, allowed[msg.sender][spender].add(addedValue));
321     return true;
322 }
```

The code meets the specification.

Formal Verification Request 73

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.47 ms

Line 314 in File MoviBit.sol

```
314 //@CTK NO_BUF_OVERFLOW
```

Line 319-322 in File MoviBit.sol

```
319 function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
320     approve(spender, allowed[msg.sender][spender].add(addedValue));
321     return true;
322 }
```

The code meets the specification.

Formal Verification Request 74

increaseAllowance correctness

10, Jul 2019
1.67 ms

Line 315-318 in File MoviBit.sol

```
315 /*@CTK "increaseAllowance correctness"
316   @tag assume_completion
317   @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] +
          addedValue
318  */
```

Line 319-322 in File MoviBit.sol

```
319 function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
320     approve(spender, allowed[msg.sender][spender].add(addedValue));
321     return true;
322 }
```

The code meets the specification.

Formal Verification Request 75

If method completes, integer overflow would not happen.

10, Jul 2019
24.6 ms

Line 339 in File MoviBit.sol

```
339 //@CTK NO_OVERFLOW
```

Line 351-354 in File MoviBit.sol

```
351 function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
352     approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353     return true;
354 }
```

The code meets the specification.

Formal Verification Request 76

Buffer overflow / array index out of bound would never happen.

10, Jul 2019
0.48 ms

Line 340 in File MoviBit.sol

```
340    //@CTK NO_BUF_OVERFLOW
```

Line 351-354 in File MoviBit.sol

```
351    function decreaseAllowance(address spender, uint256 subtractedValue) public
           returns (bool) {
352        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353        return true;
354    }
```

The code meets the specification.

Formal Verification Request 77

decreaseApproval0

10, Jul 2019
11.32 ms

Line 341-345 in File MoviBit.sol

```
341    /*@CTK decreaseApproval0
342      @pre __return == true
343      @pre allowed[msg.sender][spender] <= subtractedValue
344      @post __post.allowed[msg.sender][spender] == 0
345     */
```

Line 351-354 in File MoviBit.sol

```
351    function decreaseAllowance(address spender, uint256 subtractedValue) public
           returns (bool) {
352        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353        return true;
354    }
```

The code meets the specification.

Formal Verification Request 78

decreaseApproval

10, Jul 2019
2.28 ms

Line 346-350 in File MoviBit.sol

```
346     /*@CTK decreaseApproval
347       @pre __return == true
348       @pre allowed[msg.sender][spender] > subtractedValue
349       @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
              subtractedValue
350      */
```

Line 351-354 in File MoviBit.sol

```
351     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
352         approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
353         return true;
354     }
```

The code meets the specification.

Formal Verification Request 79

Owned_Owned__transferOwnership

10, Jul 2019
18.69 ms

Line 110-113 in File MoviBit.sol

```
110     /*@CTK Owned
111       @tag assume_completion
112       @post __post.owner == msg.sender
113      */
```

(Inheritance) Line 125-127 in File MoviBit.sol

```
125     function transferOwnership(address payable _newOwner) public onlyOwner {
126         newOwner = _newOwner;
127     }
```

The code meets the specification.

Formal Verification Request 80

Owned_transferOwnership

10, Jul 2019
18.54 ms

Line 110-113 in File MoviBit.sol

```
110     /*@CTK Owned
111       @tag assume_completion
112       @post __post.owner == msg.sender
113      */
```

(Inheritance) Line 125-127 in File MoviBit.sol

```
125     function transferOwnership(address payable _newOwner) public onlyOwner {
126         newOwner = _newOwner;
127     }
```

The code meets the specification.

Source Code with CertiK Labels

File MoviBits.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-07-10
*/

pragma solidity 0.5.9;

// ----------------------------------------------------------------------------
// Symbol      : MVBIT
// Name        : MoviBits Token
// Total supply: 1,000,000,000
// Decimals    : 18
// Reference_1 : https://theethereum.wiki/w/index.php/ERC20_Token_Standard
// Reference_2 : https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/
    contracts/token/ERC20/ERC20.sol
// ----------------------------------------------------------------------------


library SafeMath {
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == a + b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function add(uint a, uint b) internal pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    /*@CTK "SafeMath sub"
      @tag spec
      @tag is_pure
      @post (b > a) == __reverted
      @post !__reverted -> c == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function sub(uint a, uint b) internal pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    /*@CTK "SafeMath mul zero"
      @tag spec
      @tag is_pure
      @pre (a == 0)
      @post c == 0
    */
    /*@CTK "SafeMath mul nonzero"
      @tag spec
      @tag is_pure
      @pre (a != 0)
```

```
      @post (a * b / a != b) == __reverted
      @post !__reverted -> c == a * b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function mul(uint a, uint b) internal pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    /*@CTK "SafeMath div"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> c == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function div(uint a, uint b) internal pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}


// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
    function totalSupply() public view returns (uint);
    function balanceOf(address tokenOwner) public view returns (uint balance);
    function allowance(address tokenOwner, address spender) public view returns (uint
        remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool
        success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address payable public owner;
    address payable public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public {
        owner = msg.sender;
    }

```

```
    /*@CTK Owned
      @tag assume_completion
      @post __post.owner == msg.sender
     */
    modifier onlyOwner {
        require(msg.sender == owner, "Calling address is not the contract owner");
        _;
    }

    /*@CTK transferOwnership
      @tag assume_completion
      @pre msg.sender == owner
      @pre _newOwner != address(0)
      @post __post.newOwner == _newOwner
     */
    function transferOwnership(address payable _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }

    /*@CTK acceptOwnership
      @tag assume_completion
      @pre msg.sender == newOwner
      @post __post.owner == newOwner
      @post __post.newOwner == address(0)
     */
    function acceptOwnership() public {
        require(msg.sender == newOwner, "Calling address is not the new contract owner"
            );
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}


// ----------------------------------------------------------------------------
// Movibit Token Contract
// ----------------------------------------------------------------------------
contract MoviBits is ERC20Interface, Owned {
    using SafeMath for uint;

    string public symbol;
    string public name;
    uint8 public decimals;
    uint _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;

    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    /*@CTK MoviBit
      @tag assume_completion
      @post __post.balances[__post.owner] == __post._totalSupply
    */
    constructor() public {
        symbol = "MVBIT";
```

```
        name = "MoviBits Token";
        decimals = 18;
        _totalSupply = 1000000000 * 10**uint(decimals);
        balances[owner] = _totalSupply;
        emit Transfer(address(0), owner, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "totalSupply correctness"
      @tag assume_completion
      @post __return <= _totalSupply
     */
    function totalSupply() public view returns (uint) {
        return _totalSupply.sub(balances[address(0)]);
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account `tokenOwner`
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "balanceOf correctness"
      @post balance == __post.balances[tokenOwner]
     */
    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to `to` account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "transfer correctness"
      @tag assume_completion
      @pre msg.sender != to
      @post to != address(0)
      @post tokens <= balances[msg.sender]
      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
      @post __post.balances[to] == balances[to] + tokens
     */
    /*@CTK "transfer self correctness"
      @tag assume_completion
      @pre msg.sender == to
      @post to != address(0)
      @post tokens <= balances[msg.sender]
      @post __post.balances[msg.sender] == balances[msg.sender]
      @post __post.balances[to] == balances[to]
```

```
     */
    function transfer(address to, uint tokens) public returns (bool success) {
        require(to != address(0), "Cannot send to 0x0 address");
        balances[msg.sender] = balances[msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        emit Transfer(msg.sender, to, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve correctness"
      @post __post.allowed[msg.sender][spender] == tokens
     */
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Transfer `tokens` from the `from` account to the `to` account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the `from` account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "transferFrom correctness"
      @tag assume_completion
      @pre to != 0x0
      @pre tokens <= balances[from] && tokens <= allowed[from][msg.sender]
      @post to != from -> __post.balances[from] == balances[from] - tokens
      @post to != from -> __post.balances[to] == balances[to] + tokens
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
     */
    function transferFrom(address from, address to, uint tokens) public returns (bool
        success) {
        require(to != address(0), "Cannot send to 0x0 address");
        balances[from] = balances[from].sub(tokens);
        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        emit Transfer(from, to, tokens);
```

```
        return true;
    }


    // ---------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ---------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
      @post remaining == allowed[tokenOwner][spender]
     */
    function allowance(address tokenOwner, address spender) public view returns (uint remaining) {
        return allowed[tokenOwner][spender];
    }


    /**
     * @dev Atomically increases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to `approve` that can be used as a mitigation for
     * problems described in `IERC20.approve`.
     *
     * Emits an `Approval` event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "increaseAllowance correctness"
      @tag assume_completion
      @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] + addedValue
     */
    function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
        approve(spender, allowed[msg.sender][spender].add(addedValue));
        return true;
    }


    /**
     * @dev Atomically decreases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to `approve` that can be used as a mitigation for
     * problems described in `IERC20.approve`.
     *
     * Emits an `Approval` event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
```

```
     * `subtractedValue`.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK decreaseApproval0
      @pre __return == true
      @pre allowed[msg.sender][spender] <= subtractedValue
      @post __post.allowed[msg.sender][spender] == 0
     */
    /*@CTK decreaseApproval
      @pre __return == true
      @pre allowed[msg.sender][spender] > subtractedValue
      @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
          subtractedValue
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
        return true;
    }


    // ----------------------------------------------------------------------
    // Reject any incoming ETH deposit
    // ----------------------------------------------------------------------
    function () external payable {
        revert();
    }

}
```

File MoviBit.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-07-10
*/

pragma solidity 0.5.9;

// ------------------------------------------------------------------------
// Symbol      : MVBIT
// Name        : MoviBits Token
// Total supply: 1,000,000,000
// Decimals    : 18
// Reference_1 : https://theethereum.wiki/w/index.php/ERC20_Token_Standard
// Reference_2 : https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/
    contracts/token/ERC20/ERC20.sol
// ------------------------------------------------------------------------


library SafeMath {
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == a + b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
```

```
    */
    function add(uint a, uint b) internal pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    /*@CTK "SafeMath sub"
      @tag spec
      @tag is_pure
      @post (b > a) == __reverted
      @post !__reverted -> c == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function sub(uint a, uint b) internal pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    /*@CTK "SafeMath mul zero"
      @tag spec
      @tag is_pure
      @pre (a == 0)
      @post c == 0
    */
    /*@CTK "SafeMath mul nonzero"
      @tag spec
      @tag is_pure
      @pre (a != 0)
      @post (a * b / a != b) == __reverted
      @post !__reverted -> c == a * b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function mul(uint a, uint b) internal pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    /*@CTK "SafeMath div"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> c == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function div(uint a, uint b) internal pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}


// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
// ----------------------------------------------------------------------------
```

```
contract ERC20Interface {
    function totalSupply() public view returns (uint);
    function balanceOf(address tokenOwner) public view returns (uint balance);
    function allowance(address tokenOwner, address spender) public view returns (uint
        remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool
        success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address payable public owner;
    address payable public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public {
        owner = msg.sender;
    }

    /*@CTK Owned
      @tag assume_completion
      @post __post.owner == msg.sender
     */
    modifier onlyOwner {
        require(msg.sender == owner, "Calling address is not the contract owner");
        _;
    }

    /*@CTK transferOwnership
      @tag assume_completion
      @pre msg.sender == owner
      @pre _newOwner != address(0)
      @post __post.newOwner == _newOwner
     */
    function transferOwnership(address payable _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }

    /*@CTK acceptOwnership
      @tag assume_completion
      @pre msg.sender == newOwner
      @post __post.owner == newOwner
      @post __post.newOwner == address(0)
     */
    function acceptOwnership() public {
        require(msg.sender == newOwner, "Calling address is not the new contract owner"
            );
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
```

```
        newOwner = address(0);
    }
}


// ------------------------------------------------------------------------
// Movibit Token Contract
// ------------------------------------------------------------------------
contract MoviBits is ERC20Interface, Owned {
    using SafeMath for uint;

    string public symbol;
    string public name;
    uint8 public decimals;
    uint _totalSupply;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;

    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    /*@CTK MoviBit
      @tag assume_completion
      @post __post.balances[__post.owner] == __post._totalSupply
    */
    constructor() public {
        symbol = "MVBIT";
        name = "MoviBits Token";
        decimals = 18;
        _totalSupply = 1000000000 * 10**uint(decimals);
        balances[owner] = _totalSupply;
        emit Transfer(address(0), owner, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "totalSupply correctness"
      @tag assume_completion
      @post __return <= _totalSupply
     */
    function totalSupply() public view returns (uint) {
        return _totalSupply.sub(balances[address(0)]);
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account `tokenOwner`
    // ------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "balanceOf correctness"
      @post balance == __post.balances[tokenOwner]
```

```
     */
    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }


    // ----------------------------------------------------------------------
    // Transfer the balance from token owner's account to `to` account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ----------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "transfer correctness"
      @tag assume_completion
      @pre msg.sender != to
      @post to != address(0)
      @post tokens <= balances[msg.sender]
      @post __post.balances[msg.sender] == balances[msg.sender] - tokens
      @post __post.balances[to] == balances[to] + tokens
     */
    /*@CTK "transfer self correctness"
      @tag assume_completion
      @pre msg.sender == to
      @post to != address(0)
      @post tokens <= balances[msg.sender]
      @post __post.balances[msg.sender] == balances[msg.sender]
      @post __post.balances[to] == balances[to]
     */
    function transfer(address to, uint tokens) public returns (bool success) {
        require(to != address(0), "Cannot send to 0x0 address");
        balances[msg.sender] = balances[msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        emit Transfer(msg.sender, to, tokens);
        return true;
    }


    // ----------------------------------------------------------------------
    // Token owner can approve for `spender` to transferFrom(...) `tokens`
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ----------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve correctness"
      @post __post.allowed[msg.sender][spender] == tokens
     */
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }

```

```

    // ---------------------------------------------------------------------------
    // Transfer ‘tokens‘ from the ‘from‘ account to the ‘to‘ account
    //
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the ‘from‘ account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ---------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "transferFrom correctness"
      @tag assume_completion
      @pre to != 0x0
      @pre tokens <= balances[from] && tokens <= allowed[from][msg.sender]
      @post to != from -> __post.balances[from] == balances[from] - tokens
      @post to != from -> __post.balances[to] == balances[to] + tokens
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
     */
    function transferFrom(address from, address to, uint tokens) public returns (bool
        success) {
        require(to != address(0), "Cannot send to 0x0 address");
        balances[from] = balances[from].sub(tokens);
        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        emit Transfer(from, to, tokens);
        return true;
    }


    // ---------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender’s account
    // ---------------------------------------------------------------------------
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
      @post remaining == allowed[tokenOwner][spender]
     */
    function allowance(address tokenOwner, address spender) public view returns (uint
        remaining) {
        return allowed[tokenOwner][spender];
    }


    /**
     * @dev Atomically increases the allowance granted to ‘spender‘ by the caller.
     *
     * This is an alternative to ‘approve‘ that can be used as a mitigation for
     * problems described in ‘IERC20.approve‘.
     *
     * Emits an ‘Approval‘ event indicating the updated allowance.
     *
     * Requirements:
     *
```

```
     * - `spender` cannot be the zero address.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "increaseAllowance correctness"
      @tag assume_completion
      @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] +
          addedValue
     */
    function increaseAllowance(address spender, uint256 addedValue) public returns (
        bool) {
        approve(spender, allowed[msg.sender][spender].add(addedValue));
        return true;
    }


    /**
     * @dev Atomically decreases the allowance granted to `spender` by the caller.
     *
     * This is an alternative to `approve` that can be used as a mitigation for
     * problems described in `IERC20.approve`.
     *
     * Emits an `Approval` event indicating the updated allowance.
     *
     * Requirements:
     *
     * - `spender` cannot be the zero address.
     * - `spender` must have allowance for the caller of at least
     * `subtractedValue`.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK decreaseApproval0
      @pre __return == true
      @pre allowed[msg.sender][spender] <= subtractedValue
      @post __post.allowed[msg.sender][spender] == 0
     */
    /*@CTK decreaseApproval
      @pre __return == true
      @pre allowed[msg.sender][spender] > subtractedValue
      @post __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
          subtractedValue
     */
    function decreaseAllowance(address spender, uint256 subtractedValue) public
        returns (bool) {
        approve(spender, allowed[msg.sender][spender].sub(subtractedValue));
        return true;
    }


    // ----------------------------------------------------------------------
    // Reject any incoming ETH deposit
    // ----------------------------------------------------------------------
    function () external payable {
        revert();
    }

}
```