

CERTIK AUDIT REPORT FOR BENEPITOKEN



Request Date: 2019-06-12
Revision Date: 2019-06-20
Platform Name: Ethereum



Contents

| | |
|---------------------------------------|-----------|
| Disclaimer | 1 |
| About CertiK | 2 |
| Exective Summary | 3 |
| Vulnerability Classification | 3 |
| Testing Summary | 4 |
| Audit Score | 4 |
| Type of Issues | 4 |
| Vulnerability Details | 5 |
| Manual Review Notes | 6 |
| Static Analysis Results | 7 |
| Formal Verification Results | 8 |
| How to read | 8 |
| Source Code with CertiK Labels | 41 |

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 BenepitToken(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/>

Executive Summary

This report has been prepared as product of the Smart Contract Audit request by Benepit-Token. This audit was conducted to discover issues and vulnerabilities in the source code of BenepitToken'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 issues 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 condition, 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 vulnerability, but no concern found yet.

Testing Summary

PASS

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

Jun 20, 2019



Type of Issues

CertiK smart label engine applied 100% covered 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. | 1 | 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 issue found.

Low

No issue found.

(Note: Violations in formal verification result section are for internal evaluation only and are not indication of security issue in the client code.)

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **BenepitToken.sol**

9cfe778cc77ca1f1cb8c86b6513c455b8df8c0006fb0e84e431332a679891653

Summary

CertiK was chosen by Benepit to audit the design and implementation of its BenepitToken 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.

Recommendations

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.

Ownable

- **transferOwnership()** — Recommend using pull model instead of push model when changing the owner of the contract to further reduce the risk manual error.

```
address owner;
address proposedOwner;
function proposeNewOwner(address newOwner) isOwner public {
    require(newOwner != address(0), ...);
    require(newOwner != address(0), ...);
    proposedOwner = newOwner;
    // emit LogOwnerTransferProposed ...
}
function claimOwnership() public {
    require(msg.sender == proposedOwner, ...);
    owner = proposedOwner;
    proposedOwner = address(0);
    // emit LogOwnerTransferred ...
}
```


Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File BenepitToken.sol

```
1 pragma solidity 0.4.24;
```

⚠ Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClipped-ABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData

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 | <pre> 30 /*@CTK FAIL "transferFrom to same address" 31 @tag assume_completion 32 @pre from == to 33 @post __post.allowed[from][msg.sender] == 34 */ </pre> |
| Raw code location | Line 35-41 in File howtoread.sol |
| Raw code | <pre> 35 function transferFrom(address from, address to 36) { 37 balances[from] = balances[from].sub(tokens 38 allowed[from][msg.sender] = allowed[from][39 balances[to] = balances[to].add(tokens); 40 emit Transfer(from, to, tokens); 41 return true; </pre> |
| Counterexample | <div>  This code violates the specification </div> <div> <pre> 1 Counter Example: 2 Before Execution: 3 Input = { 4 from = 0x0 5 to = 0x0 6 tokens = 0x6c 7 } 8 This = 0 </pre> </div> <div> <pre> 52 } 53 balance: 0x0 54 } 55 } </pre> </div> <div> <pre> 57 After Execution: 58 Input = { 59 from = 0x0 60 to = 0x0 61 tokens = 0x6c </pre> </div> |

Formal Verification Request 1

Method will not encounter an assertion failure.

 20, Jun 2019

 19.94 ms

Line 26 in File BenepitToken.sol

26 `//@CTK FAIL NO_ASF`

Line 34-44 in File BenepitToken.sol

```

34 function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
35     // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
36     // benefit is lost if 'b' is also tested.
37     if (_a == 0) {
38         return 0;
39     }
40
41     c = _a * _b;
42     assert(c / _a == _b);
43     return c;
44 }
```

 This code violates the specification.


```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _a = 2
5     _b = 156
6   }
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24    c = 0
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": "EmptyAddress"
30    }
31  ]
32
33 Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

 20, Jun 2019

 306.19 ms

Line 27-33 in File BenepitToken.sol

```

27  /*@CTK "SafeMath mul"
28     @post ((_a > 0) && (((_a * _b) / _a) != _b)) == (__reverted)
29     @post !__reverted -> c == _a * _b
30     @post !__reverted == !__has_overflow
31     @post !__reverted -> !(__has_assertion_failure)
32     @post !(__has_buf_overflow)
33  */

```

Line 34-44 in File BenepitToken.sol

```

34  function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
35      // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
36      // benefit is lost if 'b' is also tested.
37      if (_a == 0) {
38          return 0;
39      }
40
41      c = _a * _b;
42      assert(c / _a == _b);
43      return c;
44  }


```

 The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

 20, Jun 2019

 5.41 ms

Line 49 in File BenepitToken.sol

```

49  //@CTK FAIL NO_ASF

```

Line 57-62 in File BenepitToken.sol

```

57  function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
58      // assert(_b > 0); // Solidity automatically throws when dividing by 0
59      // uint256 c = _a / _b;
60      // assert(_a == _b * c + _a % _b); // There is no case in which this doesn't hold
61      return _a / _b;
62  }

```

 This code violates the specification.

```

1  Counter Example:
2  Before Execution:
3      Input = {
4      _a = 0

```

```

5      _b = 0
6  }
7  Internal = {
8      __has_assertion_failure = false
9      __has_buf_overflow = false
10     __has_overflow = false
11     __has_returned = false
12     __reverted = false
13     msg = {
14         "gas": 0,
15         "sender": 0,
16         "value": 0
17     }
18 }
19 Other = {
20     __return = 0
21     block = {
22         "number": 0,
23         "timestamp": 0
24     }
25 }
26 Address_Map = [
27     {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30     }
31 ]
32
33 Function invocation is reverted.

```

Formal Verification Request 4

SafeMath div



20, Jun 2019



0.32 ms

Line 50-56 in File BenepitToken.sol

```

50  /*@CTK "SafeMath div"
51      @post _b != 0 -> !__reverted
52      @post !__reverted -> __return == _a / _b
53      @post !__reverted -> !__has_overflow
54      @post !__reverted -> !(__has_assertion_failure)
55      @post !(__has_buf_overflow)
56  */

```

Line 57-62 in File BenepitToken.sol

```

57  function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
58      // assert(_b > 0); // Solidity automatically throws when dividing by 0
59      // uint256 c = _a / _b;
60      // assert(_a == _b * c + _a % _b); // There is no case in which this doesn't hold
61      return _a / _b;
62  }


```




The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

 20, Jun 2019

 10.69 ms

Line 67 in File BenepitToken.sol

```
67  //@CTK FAIL NO_ASF
```

Line 75-78 in File BenepitToken.sol

```
75  function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
76      assert(_b <= _a);
77      return _a - _b;
78  }
```


 This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _a = 0
5          _b = 1
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14             "gas": 0,
15             "sender": 0,
16             "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22             "number": 0,
23             "timestamp": 0
24         }
25     }
26     Address_Map = [
27         {
28             "key": "ALL_OTHERS",
29             "value": "EmptyAddress"
30         }
31     ]
32
33  Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

 20, Jun 2019

 0.84 ms

Line 68-74 in File BenepitToken.sol

```
68  /*@CTK "SafeMath sub"
69      @post (_a < _b) == __reverted
70      @post !__reverted -> __return == _a - _b
71      @post !__reverted -> !__has_overflow
72      @post !__reverted -> !(__has_assertion_failure)
73      @post !(__has_buf_overflow)
74  */
```

Line 75-78 in File BenepitToken.sol


```
75  function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
76      assert(_b <= _a);
77      return _a - _b;
78  }
```

 The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

 20, Jun 2019

 11.26 ms

Line 83 in File BenepitToken.sol

```
83  //@CTK FAIL NO_ASF
```

Line 91-95 in File BenepitToken.sol

```
91  function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
92      c = _a + _b;
93      assert(c >= _a);
94      return c;
95  }
```

 This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _a = 191
5          _b = 65
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14             "gas": 0,
15             "sender": 0,
16             "value": 0
17         }
```

```


18     }
19     Other = {
20         block = {
21             "number": 0,
22             "timestamp": 0
23         }
24         c = 0
25     }
26     Address_Map = [
27         {
28             "key": "ALL_OTHERS",
29             "value": "EmptyAddress"
30         }
31     ]
32
33 Function invocation is reverted.

```

Formal Verification Request 8

SafeMath add

 20, Jun 2019

 2.56 ms

Line 84-90 in File BenepitToken.sol

```

84  /*@CTK "SafeMath add"
85      @post (_a + _b < _a || _a + _b < _b) == __reverted
86      @post !__reverted -> c == _a + _b
87      @post !__reverted -> !__has_overflow
88      @post !__reverted -> !(__has_assertion_failure)
89      @post !(__has_buf_overflow)
90  */

```

Line 91-95 in File BenepitToken.sol

```

91  function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
92      c = _a + _b;
93      assert(c >= _a);
94      return c;
95  }


```

 The code meets the specification.

Formal Verification Request 9

If method completes, integer overflow would not happen.

 20, Jun 2019

 5.34 ms

Line 114 in File BenepitToken.sol

```

114  //@CTK NO_OVERFLOW

```

Line 121-123 in File BenepitToken.sol


```
121 function totalSupply() public view returns (uint256) {  
122     return totalSupply_;  
123 }
```

✓ The code meets the specification.

Formal Verification Request 10

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

📅 20, Jun 2019

🕒 0.39 ms

Line 115 in File BenepitToken.sol

```
115 //CTK NO_BUF_OVERFLOW
```

Line 121-123 in File BenepitToken.sol

```
121 function totalSupply() public view returns (uint256) {  
122     return totalSupply_;  
123 }
```

✓ The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

📅 20, Jun 2019

🕒 0.35 ms

Line 116 in File BenepitToken.sol

```
116 //CTK NO_ASF
```

Line 121-123 in File BenepitToken.sol

```
121 function totalSupply() public view returns (uint256) {  
122     return totalSupply_;  
123 }
```

✓ The code meets the specification.

Formal Verification Request 12

totalSupply

📅 20, Jun 2019

🕒 0.32 ms

Line 117-120 in File BenepitToken.sol

```
117 /*CTK totalSupply  
118     @tag assume_completion  
119     @post (__return) == (totalSupply_)  
120 */
```

Line 121-123 in File BenepitToken.sol


```
121 function totalSupply() public view returns (uint256) {  
122     return totalSupply_;  
123 }
```

✓ The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

 20, Jun 2019

 79.35 ms

Line 130 in File BenepitToken.sol

```
130 // @CTK_NO_OVERFLOW
```

Line 143-151 in File BenepitToken.sol


```
143 function transfer(address _to, uint256 _value) public returns (bool) {  
144     require(_value <= balances[msg.sender]);  
145     require(_to != address(0));  
146  
147     balances[msg.sender] = balances[msg.sender].sub(_value);  
148     balances[_to] = balances[_to].add(_value);  
149     emit Transfer(msg.sender, _to, _value);  
150     return true;  
151 }
```

✓ The code meets the specification.

Formal Verification Request 14

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

 20, Jun 2019

 10.48 ms

Line 131 in File BenepitToken.sol

```
131 // @CTK_NO_BUF_OVERFLOW
```


Line 143-151 in File BenepitToken.sol


```
143 function transfer(address _to, uint256 _value) public returns (bool) {  
144     require(_value <= balances[msg.sender]);  
145     require(_to != address(0));  
146  
147     balances[msg.sender] = balances[msg.sender].sub(_value);  
148     balances[_to] = balances[_to].add(_value);  
149     emit Transfer(msg.sender, _to, _value);  
150     return true;  
151 }
```

✓ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

 20, Jun 2019

 42.72 ms

Line 132 in File BenepitToken.sol

132 `//@CTK FAIL NO_ASF`

Line 143-151 in File BenepitToken.sol

```

143 function transfer(address _to, uint256 _value) public returns (bool) {
144     require(_value <= balances[msg.sender]);
145     require(_to != address(0));
146
147     balances[msg.sender] = balances[msg.sender].sub(_value);
148     balances[_to] = balances[_to].add(_value);
149     emit Transfer(msg.sender, _to, _value);
150     return true;
151 }
```

 This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _to = 4
5     _value = 4
6   }
7   This = 0
8   Internal = {
9     __has_assertion_failure = false
10    __has_buf_overflow = false
11    __has_overflow = false
12    __has_returned = false
13    __reverted = false
14    msg = {
15      "gas": 0,
16      "sender": 0,
17      "value": 0
18    }
19  }
20  Other = {
21    __return = false
22    block = {
23      "number": 0,
24      "timestamp": 0
25    }
26  }
27  Address_Map = [
28    {
29      "key": 0,
30      "value": {
31        "contract_name": "BasicToken",
32        "balance": 0,
33        "contract": {
34          "balances": [
35
```

```


36         "key": 4,
37         "value": 252
38     },
39     {
40         "key": 0,
41         "value": 4
42     },
43     {
44         "key": "ALL_OTHERS",
45         "value": 255
46     }
47 ],
48 "totalSupply_": 0
49 }
50 }
51 },
52 {
53     "key": "ALL_OTHERS",
54     "value": "EmptyAddress"
55 }
56 ]
57
58 Function invocation is reverted.

```

Formal Verification Request 16

transfer

 20, Jun 2019

 149.97 ms

Line 133-142 in File BenepitToken.sol

```

133  /*@CTK transfer
134     @tag assume_completion
135     @pre _to != address(0)
136     @pre _value <= balances[msg.sender]
137     @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
138     @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
139         - _value)
139     @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
140     @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
141     @post __return == true
142 */

```

Line 143-151 in File BenepitToken.sol

```

143  function transfer(address _to, uint256 _value) public returns (bool) {
144      require(_value <= balances[msg.sender]);
145      require(_to != address(0));
146
147      balances[msg.sender] = balances[msg.sender].sub(_value);
148      balances[_to] = balances[_to].add(_value);
149      emit Transfer(msg.sender, _to, _value);
150      return true;
151  }


```

 The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

 20, Jun 2019

 4.44 ms

Line 158 in File BenepitToken.sol

158 `//@CTK NO_OVERFLOW`

Line 165-167 in File BenepitToken.sol


```
165 function balanceOf(address _owner) public view returns (uint256) {  
166 return balances[_owner];  
167 }
```

 The code meets the specification.

Formal Verification Request 18

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

 20, Jun 2019

 0.5 ms

Line 159 in File BenepitToken.sol

159 `//@CTK NO_BUF_OVERFLOW`

Line 165-167 in File BenepitToken.sol


```
165 function balanceOf(address _owner) public view returns (uint256) {  
166 return balances[_owner];  
167 }
```

 The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

 20, Jun 2019

 0.32 ms

Line 160 in File BenepitToken.sol

160 `//@CTK NO_ASF`


Line 165-167 in File BenepitToken.sol


```
165 function balanceOf(address _owner) public view returns (uint256) {  
166 return balances[_owner];  
167 }
```

 The code meets the specification.

Formal Verification Request 20

balanceOf

 20, Jun 2019

 0.34 ms

Line 161-164 in File BenepitToken.sol

```
161  /*@CTK balanceOf
162     @tag assume_completion
163     @post (__return) == (balances[_owner])
164  */
```


Line 165-167 in File BenepitToken.sol


```
165  function balanceOf(address _owner) public view returns (uint256) {
166      return balances[_owner];
167  }
```

 The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

 20, Jun 2019

 100.36 ms

Line 212 in File BenepitToken.sol

```
212  //@CTK NO_OVERFLOW
```


Line 227-244 in File BenepitToken.sol


```
227  function transferFrom(
228      address _from,
229      address _to,
230      uint256 _value
231  )
232      public
233      returns (bool)
234  {
235      require(_value <= balances[_from]);
236      require(_value <= allowed[_from][msg.sender]);
237      require(_to != address(0));
238
239      balances[_from] = balances[_from].sub(_value);
240      balances[_to] = balances[_to].add(_value);
241      allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242      emit Transfer(_from, _to, _value);
243      return true;
244  }
```

 The code meets the specification.

Formal Verification Request 22

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

 20, Jun 2019

 10.56 ms

Line 213 in File BenepitToken.sol

213 `//@CTK NO_BUF_OVERFLOW`

Line 227-244 in File BenepitToken.sol


```
227 function transferFrom(  
228     address _from,  
229     address _to,  
230     uint256 _value  
231 )  
232     public  
233     returns (bool)  
234 {  
235     require(_value <= balances[_from]);  
236     require(_value <= allowed[_from][msg.sender]);  
237     require(_to != address(0));  
238  
239     balances[_from] = balances[_from].sub(_value);  
240     balances[_to] = balances[_to].add(_value);  
241     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);  
242     emit Transfer(_from, _to, _value);  
243     return true;  
244 }
```

 The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

 20, Jun 2019

 63.24 ms

Line 214 in File BenepitToken.sol

214 `//@CTK FAIL NO_ASF`

Line 227-244 in File BenepitToken.sol

```
227 function transferFrom(  
228     address _from,  
229     address _to,  
230     uint256 _value  
231 )  
232     public  
233     returns (bool)  
234 {  
235     require(_value <= balances[_from]);  
236     require(_value <= allowed[_from][msg.sender]);  
237     require(_to != address(0));  
238 }
```

```

239 balances[_from] = balances[_from].sub(_value);
240 balances[_to] = balances[_to].add(_value);
241 allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242 emit Transfer(_from, _to, _value);
243 return true;
244 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _from = 0
5     _to = 4
6     _value = 137
7   }
8   This = 0
9   Internal = {
10     __has_assertion_failure = false
11     __has_buf_overflow = false
12     __has_overflow = false
13     __has_returned = false
14     __reverted = false
15     msg = {
16       "gas": 0,
17       "sender": 0,
18       "value": 0
19     }
20   }
21   Other = {
22     __return = false
23     block = {
24       "number": 0,
25       "timestamp": 0
26     }
27   }
28   Address_Map = [
29     {
30       "key": 0,
31       "value": {
32         "contract_name": "StandardToken",
33         "balance": 0,
34         "contract": {
35           "allowed": [
36             {
37               "key": 128,
38               "value": [
39                 {
40                   "key": 0,
41                   "value": 0
42                 },
43                 {
44                   "key": "ALL_OTHERS",
45                   "value": 137
46                 }
47               ]
48             },
49             {
50               "key": 0,

```



```

51         "value": [
52             {
53                 "key": 0,
54                 "value": 200
55             },
56             {
57                 "key": "ALL_OTHERS",
58                 "value": 137
59             }
60         ]
61     },
62     {
63         "key": 64,
64         "value": [
65             {
66                 "key": 0,
67                 "value": 0
68             },
69             {
70                 "key": "ALL_OTHERS",
71                 "value": 137
72             }
73         ]
74     },
75     {
76         "key": "ALL_OTHERS",
77         "value": [
78             {
79                 "key": "ALL_OTHERS",
80                 "value": 119
81             }
82         ]
83     }
84 ],
85 "balances": [
86     {
87         "key": 32,
88         "value": 0
89     },
90     {
91         "key": 128,
92         "value": 0
93     },
94     {
95         "key": 64,
96         "value": 32
97     },
98     {
99         "key": 0,
100        "value": 164
101    },
102    {
103        "key": 68,
104        "value": 0
105    },
106    {
107        "key": 4,
108        "value": 121

```

```

109         },
110         {
111             "key": 8,
112             "value": 4
113         },
114         {
115             "key": "ALL_OTHERS",
116             "value": 119
117         }
118     ],
119     "totalSupply_": 0
120 }
121 }
122 },
123 {
124     "key": "ALL_OTHERS",
125     "value": "EmptyAddress"
126 }
127 ]


```

129 Function invocation is reverted.

Formal Verification Request 24

transferFrom

 20, Jun 2019

 330.71 ms

Line 215-226 in File BenepitToken.sol

```

215  /*@CTK "transferFrom"
216     @tag assume_completion
217     @pre (_to) != (address(0))
218     @pre (_value) <= (balances[_from])
219     @pre (_value) <= (allowed[_from][msg.sender])
220     @post (_from != _to) -> (__post.balances[_to] == (balances[_to] + _value))
221     @post (_from != _to) -> (__post.balances[_from] == (balances[_from] - _value))
222     @post (_from == _to) -> (__post.balances[_to] == balances[_to])
223     @post (_from == _to) -> (__post.balances[_from] == balances[_from])
224     @post (__post.allowed[_from][msg.sender]) == (allowed[_from][msg.sender] - _value)
225     @post (__return) == (true)
226  */

```

Line 227-244 in File BenepitToken.sol

```

227  function transferFrom(
228      address _from,
229      address _to,
230      uint256 _value
231  )
232  public
233  returns (bool)
234  {
235      require(_value <= balances[_from]);
236      require(_value <= allowed[_from][msg.sender]);
237      require(_to != address(0));
238  }

```


```
239     balances[_from] = balances[_from].sub(_value);
240     balances[_to] = balances[_to].add(_value);
241     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242     emit Transfer(_from, _to, _value);
243     return true;
244 }
```

✓ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

 20, Jun 2019

 7.3 ms

Line 255 in File BenepitToken.sol

```
255 // @CTK_NO_OVERFLOW
```

Line 262-266 in File BenepitToken.sol


```
262 function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266 }
```

✓ The code meets the specification.

Formal Verification Request 26

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

 20, Jun 2019

 0.32 ms

Line 256 in File BenepitToken.sol

```
256 // @CTK_NO_BUF_OVERFLOW
```

Line 262-266 in File BenepitToken.sol


```
262 function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266 }
```

✓ The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

 20, Jun 2019

 0.3 ms

Line 257 in File BenepitToken.sol

257 `//@CTK NO_ASF`

Line 262-266 in File BenepitToken.sol


```
262 function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266 }
```

 The code meets the specification.

Formal Verification Request 28

approve

 20, Jun 2019

 1.13 ms

Line 258-261 in File BenepitToken.sol

```
258 /*@CTK approve
259     @tag assume_completion
260     @post (__post.allowed[msg.sender][_spender]) == (_value)
261 */
```

Line 262-266 in File BenepitToken.sol


```
262 function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266 }
```

 The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

 20, Jun 2019

 4.81 ms

Line 274 in File BenepitToken.sol

274 `//@CTK NO_OVERFLOW`

Line 281-290 in File BenepitToken.sol


```
281 function allowance(  
282     address _owner,  
283     address _spender  
284 )  
285     public  
286     view  
287     returns (uint256)  
288 {  
289     return allowed[_owner][_spender];  
290 }
```

✓ The code meets the specification.

Formal Verification Request 30

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

 20, Jun 2019

 0.3 ms

Line 275 in File BenepitToken.sol

```
275 // @CTK_NO_BUF_OVERFLOW
```

Line 281-290 in File BenepitToken.sol


```
281 function allowance(  
282     address _owner,  
283     address _spender  
284 )  
285     public  
286     view  
287     returns (uint256)  
288 {  
289     return allowed[_owner][_spender];  
290 }
```

✓ The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

 20, Jun 2019

 0.3 ms

Line 276 in File BenepitToken.sol

```
276 // @CTK_NO_ASF
```

Line 281-290 in File BenepitToken.sol

```
281 function allowance(  
282     address _owner,  
283     address _spender  
284 )  
285     public
```


```
286     view
287     returns (uint256)
288     {
289         return allowed[_owner][_spender];
290     }
```

✓ The code meets the specification.

Formal Verification Request 32

allowance

 20, Jun 2019

 0.3 ms

Line 277-280 in File BenepitToken.sol

```
277  /*@CTK allowance
278     @tag assume_completion
279     @post (__return) == (allowed[_owner][_spender])
280  */
```

Line 281-290 in File BenepitToken.sol


```
281  function allowance(
282      address _owner,
283      address _spender
284  )
285      public
286      view
287      returns (uint256)
288  {
289      return allowed[_owner][_spender];
290  }
```

✓ The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

 20, Jun 2019

 34.08 ms

Line 301 in File BenepitToken.sol

```
301  //@CTK NO_OVERFLOW
```

Line 311-322 in File BenepitToken.sol

```
311  function increaseApproval(
312      address _spender,
313      uint256 _addedValue
314  )
315      public
316      returns (bool)
317  {
```

```

318     allowed[msg.sender][_spender] = (
319         allowed[msg.sender][_spender].add(_addedValue));
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321     return true;
322 }


```

✓ The code meets the specification.

Formal Verification Request 34

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

 20, Jun 2019

 0.75 ms

Line 302 in File BenepitToken.sol

```

302     //@CTK NO_BUF_OVERFLOW

```

Line 311-322 in File BenepitToken.sol

```

311     function increaseApproval(
312         address _spender,
313         uint256 _addedValue
314     )
315     public
316     returns (bool)
317     {
318         allowed[msg.sender][_spender] = (
319             allowed[msg.sender][_spender].add(_addedValue));
320         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321         return true;
322     }


```

✓ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

 20, Jun 2019

 7.31 ms

Line 303 in File BenepitToken.sol

```

303     //@CTK FAIL NO_ASF

```

Line 311-322 in File BenepitToken.sol

```

311     function increaseApproval(
312         address _spender,
313         uint256 _addedValue
314     )
315     public
316     returns (bool)
317     {
318         allowed[msg.sender][_spender] = (

```

```

319     allowed[msg.sender][_spender].add(_addedValue));
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321     return true;
322 }

```

✗ This code violates the specification.

```

1 Counter Example:
2 Before Execution:
3   Input = {
4     _addedValue = 161
5     _spender = 0
6   }
7   This = 0
8   Internal = {
9     __has_assertion_failure = false
10    __has_buf_overflow = false
11    __has_overflow = false
12    __has_returned = false
13    __reverted = false
14    msg = {
15      "gas": 0,
16      "sender": 0,
17      "value": 0
18    }
19  }
20  Other = {
21    __return = false
22    block = {
23      "number": 0,
24      "timestamp": 0
25    }
26  }
27  Address_Map = [
28    {
29      "key": 0,
30      "value": {
31        "contract_name": "StandardToken",
32        "balance": 0,
33        "contract": {
34          "allowed": [
35            {
36              "key": 0,
37              "value": [
38                {
39                  "key": 0,
40                  "value": 95
41                },
42                {
43                  "key": 4,
44                  "value": 128
45                },
46                {
47                  "key": "ALL_OTHERS",
48                  "value": 161
49                }
50              ]
51            },
52            {

```





```

53         "key": "ALL_OTHERS",
54         "value": [
55             {
56                 "key": "ALL_OTHERS",
57                 "value": 161
58             }
59         ]
60     },
61 ],
62     "balances": [
63         {
64             "key": 4,
65             "value": 0
66         },
67         {
68             "key": "ALL_OTHERS",
69             "value": 161
70         }
71     ],
72     "totalSupply_": 0
73 }
74 }
75 },
76 {
77     "key": "ALL_OTHERS",
78     "value": "EmptyAddress"
79 }
80 ]
81
82 Function invocation is reverted.

```

Formal Verification Request 36

increaseApproval

 20, Jun 2019
 2.17 ms

Line 304-310 in File BenepitToken.sol

```

304  /*@CTK increaseApproval
305     @tag assume_completion
306     @pre _spender != 0x0
307     @pre _spender != msg.sender
308     @post (__post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
309         _addedValue)
310     @post (__return) == (true)
311 */

```

Line 311-322 in File BenepitToken.sol

```

311  function increaseApproval(
312      address _spender,
313      uint256 _addedValue
314  )
315  public
316  returns (bool)
317  {

```


```
318     allowed[msg.sender][_spender] = (  
319         allowed[msg.sender][_spender].add(_addedValue));  
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);  
321     return true;  
322 }
```

✓ The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

 20, Jun 2019

 43.37 ms

Line 333 in File BenepitToken.sol

```
333     //@CTK_NO_OVERFLOW
```

Line 343-358 in File BenepitToken.sol


```
343     function decreaseApproval(  
344         address _spender,  
345         uint256 _subtractedValue  
346     )  
347     public  
348     returns (bool)  
349     {  
350         uint256 oldValue = allowed[msg.sender][_spender];  
351         if (_subtractedValue >= oldValue) {  
352             allowed[msg.sender][_spender] = 0;  
353         } else {  
354             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);  
355         }  
356         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);  
357         return true;  
358     }
```

✓ The code meets the specification.

Formal Verification Request 38

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

 20, Jun 2019

 0.89 ms

Line 334 in File BenepitToken.sol

```
334     //@CTK_NO_BUF_OVERFLOW
```

Line 343-358 in File BenepitToken.sol

```
343     function decreaseApproval(  
344         address _spender,  
345         uint256 _subtractedValue  
346     )
```


```
347     public
348     returns (bool)
349     {
350         uint256 oldValue = allowed[msg.sender][_spender];
351         if (_subtractedValue >= oldValue) {
352             allowed[msg.sender][_spender] = 0;
353         } else {
354             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
355         }
356         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
357         return true;
358     }
```

✓ The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

 20, Jun 2019

 5.26 ms

Line 335 in File BenepitToken.sol

```
335     //@CTK NO_ASF
```

Line 343-358 in File BenepitToken.sol


```
343     function decreaseApproval(
344         address _spender,
345         uint256 _subtractedValue
346     )
347     public
348     returns (bool)
349     {
350         uint256 oldValue = allowed[msg.sender][_spender];
351         if (_subtractedValue >= oldValue) {
352             allowed[msg.sender][_spender] = 0;
353         } else {
354             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
355         }
356         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
357         return true;
358     }
```

✓ The code meets the specification.

Formal Verification Request 40

decreaseApproval

 20, Jun 2019

 28.37 ms

Line 336-342 in File BenepitToken.sol

```

336  /*@CTK decreaseApproval
337      @tag assume_completion
338      @pre _spender != 0x0
339      @pre _spender != msg.sender
340      @post (_subtractedValue > allowed[msg.sender][_spender]) -> (__post.allowed[msg.
          sender][_spender] == 0)
341      @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (__post.allowed[msg.
          sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue)
342  */

```

Line 343-358 in File BenepitToken.sol

```

343  function decreaseApproval(
344      address _spender,
345      uint256 _subtractedValue
346  )
347      public
348      returns (bool)
349  {
350      uint256 oldValue = allowed[msg.sender][_spender];
351      if (_subtractedValue >= oldValue) {
352          allowed[msg.sender][_spender] = 0;
353      } else {
354          allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
355      }
356      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
357      return true;
358  }

```

✓ The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.



20, Jun 2019



12.94 ms

Line 402 in File BenepitToken.sol

```

402  //@CTK NO_OVERFLOW

```

Line 410-413 in File BenepitToken.sol

```

410  function renounceOwnership() public onlyOwner {
411      emit OwnershipRenounced(owner);
412      owner = address(0);
413  }

```

✓ The code meets the specification.

Formal Verification Request 42

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



20, Jun 2019



0.56 ms

Line 403 in File BenepitToken.sol

```
403  // @CTK_NO_BUF_OVERFLOW
```


Line 410-413 in File BenepitToken.sol


```
410  function renounceOwnership() public onlyOwner {  
411      emit OwnershipRenounced(owner);  
412      owner = address(0);  
413  }
```

✓ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

 20, Jun 2019

 0.49 ms

Line 404 in File BenepitToken.sol

```
404  // @CTK_NO_ASF
```

Line 410-413 in File BenepitToken.sol


```
410  function renounceOwnership() public onlyOwner {  
411      emit OwnershipRenounced(owner);  
412      owner = address(0);  
413  }
```

✓ The code meets the specification.

Formal Verification Request 44

isOwner

 20, Jun 2019

 1.53 ms

Line 405-409 in File BenepitToken.sol

```
405  /* @CTK isOwner  
406      @tag assume_completion  
407      @pre msg.sender == owner  
408      @post __post.owner == address(0)  
409  */
```

Line 410-413 in File BenepitToken.sol


```
410  function renounceOwnership() public onlyOwner {  
411      emit OwnershipRenounced(owner);  
412      owner = address(0);  
413  }
```

✓ The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

 20, Jun 2019

 40.46 ms

Line 419 in File BenepitToken.sol

```
419 // @CTK_NO_OVERFLOW
```

Line 429-431 in File BenepitToken.sol


```
429 function transferOwnership(address _newOwner) public onlyOwner {  
430     _transferOwnership(_newOwner);  
431 }
```

 The code meets the specification.

Formal Verification Request 46

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

 20, Jun 2019

 0.59 ms

Line 420 in File BenepitToken.sol

```
420 // @CTK_NO_BUF_OVERFLOW
```

Line 429-431 in File BenepitToken.sol


```
429 function transferOwnership(address _newOwner) public onlyOwner {  
430     _transferOwnership(_newOwner);  
431 }
```

 The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

 20, Jun 2019

 0.57 ms

Line 421 in File BenepitToken.sol

```
421 // @CTK_NO_ASF
```


Line 429-431 in File BenepitToken.sol


```
429 function transferOwnership(address _newOwner) public onlyOwner {  
430     _transferOwnership(_newOwner);  
431 }
```

 The code meets the specification.

Formal Verification Request 48

transferOwnership

 20, Jun 2019

 2.07 ms

Line 422-428 in File BenepitToken.sol

```
422  /*@CTK transferOwnership
423     @tag assume_completion
424     @pre msg.sender == owner
425     @pre msg.sender != _newOwner
426     @pre _newOwner != address(0)
427     @post __post.owner == _newOwner
428  */
```


Line 429-431 in File BenepitToken.sol


```
429  function transferOwnership(address _newOwner) public onlyOwner {
430      _transferOwnership(_newOwner);
431  }
```

 The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

 20, Jun 2019

 0.47 ms

Line 437 in File BenepitToken.sol

```
437  //@CTK NO_OVERFLOW
```

Line 447-451 in File BenepitToken.sol


```
447  function _transferOwnership(address _newOwner) internal {
448      require(_newOwner != address(0));
449      emit OwnershipTransferred(owner, _newOwner);
450      owner = _newOwner;
451  }
```

 The code meets the specification.

Formal Verification Request 50

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

 20, Jun 2019

 0.46 ms

Line 438 in File BenepitToken.sol

```
438  //@CTK NO_BUF_OVERFLOW
```

Line 447-451 in File BenepitToken.sol

```

447 function _transferOwnership(address _newOwner) internal {
448     require(_newOwner != address(0));
449     emit OwnershipTransferred(owner, _newOwner);
450     owner = _newOwner;
451 }


```

✓ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

 20, Jun 2019

 0.46 ms

Line 439 in File BenepitToken.sol

```

439 // @CTK NO_ASF

```

Line 447-451 in File BenepitToken.sol

```

447 function _transferOwnership(address _newOwner) internal {
448     require(_newOwner != address(0));
449     emit OwnershipTransferred(owner, _newOwner);
450     owner = _newOwner;
451 }


```

✓ The code meets the specification.

Formal Verification Request 52

_transferOwnership

 20, Jun 2019

 1.47 ms

Line 440-446 in File BenepitToken.sol

```

440 /* @CTK _transferOwnership
441     @tag assume_completion
442     @pre msg.sender == owner
443     @pre msg.sender != _newOwner
444     @pre _newOwner != address(0)
445     @post __post.owner == _newOwner
446 */

```

Line 447-451 in File BenepitToken.sol

```


447 function _transferOwnership(address _newOwner) internal {
448     require(_newOwner != address(0));
449     emit OwnershipTransferred(owner, _newOwner);
450     owner = _newOwner;
451 }


```

✓ The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

 20, Jun 2019

 10.6 ms

Line 460 in File BenepitToken.sol

```
460 // @CTK NO_OVERFLOW
```

Line 467-470 in File BenepitToken.sol


```
467 constructor() public {  
468     totalSupply_ = 30000000000 * 10**uint(decimals);  
469     balances[msg.sender] = totalSupply_;  
470 }
```

 The code meets the specification.

Formal Verification Request 54

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

 20, Jun 2019

 0.41 ms

Line 461 in File BenepitToken.sol

```
461 // @CTK NO_BUF_OVERFLOW
```

Line 467-470 in File BenepitToken.sol


```
467 constructor() public {  
468     totalSupply_ = 30000000000 * 10**uint(decimals);  
469     balances[msg.sender] = totalSupply_;  
470 }
```

 The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

 20, Jun 2019

 0.43 ms

Line 462 in File BenepitToken.sol

```
462 // @CTK NO_ASF
```

Line 467-470 in File BenepitToken.sol


```
467 constructor() public {  
468     totalSupply_ = 30000000000 * 10**uint(decimals);  
469     balances[msg.sender] = totalSupply_;  
470 }
```

 The code meets the specification.

Formal Verification Request 56

BenepitToken

 20, Jun 2019

 1.89 ms

Line 463-466 in File BenepitToken.sol

```
463  /*@CTK BenepitToken
464     @tag assume_completion
465     @post __post.balances[msg.sender] == __post.totalSupply_
466  */
```

Line 467-470 in File BenepitToken.sol

```
467  constructor() public {
468      totalSupply_ = 30000000000 * 10**uint(decimals);
469      balances[msg.sender] = totalSupply_;
470  }
```

 The code meets the specification.

Source Code with CertiK Labels

File BenepitToken.sol

```

1  pragma solidity 0.4.24;
2
3
4  /**
5   * @title ERC20Basic
6   * @dev Simpler version of ERC20 interface
7   * See https://github.com/ethereum/EIPs/issues/179
8   */
9  contract ERC20Basic {
10     function totalSupply() public view returns (uint256);
11     function balanceOf(address _who) public view returns (uint256);
12     function transfer(address _to, uint256 _value) public returns (bool);
13     event Transfer(address indexed from, address indexed to, uint256 value);
14 }
15
16
17 /**
18 * @title SafeMath
19 * @dev Math operations with safety checks that throw on error
20 */
21 library SafeMath {
22
23     /**
24     * @dev Multiplies two numbers, throws on overflow.
25     */
26     //@CTK FAIL NO_ASF
27     //@CTK "SafeMath mul"
28     @post ((_a > 0) && (((_a * _b) / _a) != _b)) == (__reverted)
29     @post !__reverted -> c == _a * _b
30     @post !__reverted == !__has_overflow
31     @post !__reverted -> !(__has_assertion_failure)
32     @post !(__has_buf_overflow)
33     */
34     function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
35         // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
36         // benefit is lost if 'b' is also tested.
37         if (_a == 0) {
38             return 0;
39         }
40
41         c = _a * _b;
42         assert(c / _a == _b);
43         return c;
44     }
45
46     /**
47     * @dev Integer division of two numbers, truncating the quotient.
48     */
49     //@CTK FAIL NO_ASF
50     //@CTK "SafeMath div"
51     @post _b != 0 -> !__reverted
52     @post !__reverted -> __return == _a / _b
53     @post !__reverted -> !__has_overflow
54     @post !__reverted -> !(__has_assertion_failure)

```

```

55     @post !(__has_buf_overflow)
56     */
57     function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
58         // assert(_b > 0); // Solidity automatically throws when dividing by 0
59         // uint256 c = _a / _b;
60         // assert(_a == _b * c + _a % _b); // There is no case in which this doesn't hold
61         return _a / _b;
62     }
63
64     /**
65     * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
        minuend).
66     */
67     //@CTK FAIL NO_ASF
68     /*CTK "SafeMath sub"
69         @post (_a < _b) == __reverted
70         @post !__reverted -> __return == _a - _b
71         @post !__reverted -> !__has_overflow
72         @post !__reverted -> !(__has_assertion_failure)
73         @post !(__has_buf_overflow)
74     */
75     function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
76         assert(_b <= _a);
77         return _a - _b;
78     }
79
80     /**
81     * @dev Adds two numbers, throws on overflow.
82     */
83     //@CTK FAIL NO_ASF
84     /*CTK "SafeMath add"
85         @post (_a + _b < _a || _a + _b < _b) == __reverted
86         @post !__reverted -> c == _a + _b
87         @post !__reverted -> !__has_overflow
88         @post !__reverted -> !(__has_assertion_failure)
89         @post !(__has_buf_overflow)
90     */
91     function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
92         c = _a + _b;
93         assert(c >= _a);
94         return c;
95     }
96 }
97
98
99
100 /**
101 * @title Basic token
102 * @dev Basic version of StandardToken, with no allowances.
103 */
104 contract BasicToken is ERC20Basic {
105     using SafeMath for uint256;
106
107     mapping(address => uint256) internal balances;
108
109     uint256 internal totalSupply_;
110
111     /**

```

```

112 * @dev Total number of tokens in existence
113 */
114 //@CTK NO_OVERFLOW
115 //@CTK NO_BUF_OVERFLOW
116 //@CTK NO_ASF
117 /*CTK totalSupply
118   @tag assume_completion
119   @post (__return) == (totalSupply_)
120 */
121 function totalSupply() public view returns (uint256) {
122     return totalSupply_;
123 }
124
125 /**
126 * @dev Transfer token for a specified address
127 * @param _to The address to transfer to.
128 * @param _value The amount to be transferred.
129 */
130 //@CTK NO_OVERFLOW
131 //@CTK NO_BUF_OVERFLOW
132 //@CTK FAIL NO_ASF
133 /*CTK transfer
134   @tag assume_completion
135   @pre _to != address(0)
136   @pre _value <= balances[msg.sender]
137   @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
138   @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
139     - _value)
140   @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
141   @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
142   @post __return == true
143 */
144 function transfer(address _to, uint256 _value) public returns (bool) {
145     require(_value <= balances[msg.sender]);
146     require(_to != address(0));
147
148     balances[msg.sender] = balances[msg.sender].sub(_value);
149     balances[_to] = balances[_to].add(_value);
150     emit Transfer(msg.sender, _to, _value);
151     return true;
152 }
153
154 /**
155 * @dev Gets the balance of the specified address.
156 * @param _owner The address to query the the balance of.
157 * @return An uint256 representing the amount owned by the passed address.
158 */
159 //@CTK NO_OVERFLOW
160 //@CTK NO_BUF_OVERFLOW
161 //@CTK NO_ASF
162 /*CTK balanceOf
163   @tag assume_completion
164   @post (__return) == (balances[_owner])
165 */
166 function balanceOf(address _owner) public view returns (uint256) {
167     return balances[_owner];
168 }

```

```

169 }
170
171
172
173 /**
174  * @title ERC20 interface
175  * @dev see https://github.com/ethereum/EIPs/issues/20
176  */
177 contract ERC20 is ERC20Basic {
178     function allowance(address _owner, address _spender)
179         public view returns (uint256);
180
181     function transferFrom(address _from, address _to, uint256 _value)
182         public returns (bool);
183
184     function approve(address _spender, uint256 _value) public returns (bool);
185     event Approval(
186         address indexed owner,
187         address indexed spender,
188         uint256 value
189     );
190 }
191
192
193
194 /**
195  * @title Standard ERC20 token
196  *
197  * @dev Implementation of the basic standard token.
198  * https://github.com/ethereum/EIPs/issues/20
199  * Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/
200     smart_contract/FirstBloodToken.sol
201 */
202 contract StandardToken is ERC20, BasicToken {
203     mapping (address => mapping (address => uint256)) internal allowed;
204
205
206     /**
207      * @dev Transfer tokens from one address to another
208      * @param _from address The address which you want to send tokens from
209      * @param _to address The address which you want to transfer to
210      * @param _value uint256 the amount of tokens to be transferred
211      */
212     //@CTK NO_OVERFLOW
213     //@CTK NO_BUF_OVERFLOW
214     //@CTK FAIL_NO_ASF
215     /*CTK "transferFrom"
216         @tag assume_completion
217         @pre (_to) != (address(0))
218         @pre (_value) <= (balances[_from])
219         @pre (_value) <= (allowed[_from][msg.sender])
220         @post (_from != _to) -> (__post.balances[_to] == (balances[_to] + _value))
221         @post (_from != _to) -> (__post.balances[_from] == (balances[_from] - _value))
222         @post (_from == _to) -> (__post.balances[_to] == balances[_to])
223         @post (_from == _to) -> (__post.balances[_from] == balances[_from])
224         @post (__post.allowed[_from][msg.sender]) == (allowed[_from][msg.sender] - _value)
225         @post (__return) == (true)

```

```

226  */
227  function transferFrom(
228      address _from,
229      address _to,
230      uint256 _value
231  )
232      public
233      returns (bool)
234  {
235      require(_value <= balances[_from]);
236      require(_value <= allowed[_from][msg.sender]);
237      require(_to != address(0));
238
239      balances[_from] = balances[_from].sub(_value);
240      balances[_to] = balances[_to].add(_value);
241      allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242      emit Transfer(_from, _to, _value);
243      return true;
244  }
245
246  /**
247   * @dev Approve the passed address to spend the specified amount of tokens on behalf
248   *       of msg.sender.
249   * Beware that changing an allowance with this method brings the risk that someone
250   * may use both the old
251   * and the new allowance by unfortunate transaction ordering. One possible solution
252   * to mitigate this
253   * race condition is to first reduce the spender's allowance to 0 and set the
254   * desired value afterwards:
255   * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
256   * @param _spender The address which will spend the funds.
257   * @param _value The amount of tokens to be spent.
258   */
259  // @CTK NO_OVERFLOW
260  // @CTK NO_BUF_OVERFLOW
261  // @CTK NO_ASF
262  /* @CTK approve
263   * @tag assume_completion
264   * @post (__post.allowed[msg.sender][_spender]) == (_value)
265   */
266  function approve(address _spender, uint256 _value) public returns (bool) {
267      allowed[msg.sender][_spender] = _value;
268      emit Approval(msg.sender, _spender, _value);
269      return true;
270  }
271
272  /**
273   * @dev Function to check the amount of tokens that an owner allowed to a spender.
274   * @param _owner address The address which owns the funds.
275   * @param _spender address The address which will spend the funds.
276   * @return A uint256 specifying the amount of tokens still available for the spender
277   *
278   */
279  // @CTK NO_OVERFLOW
280  // @CTK NO_BUF_OVERFLOW
281  // @CTK NO_ASF
282  /* @CTK allowance
283   * @tag assume_completion

```

```

279     @post (__return) == (allowed[_owner][_spender])
280 */
281 function allowance(
282     address _owner,
283     address _spender
284 )
285     public
286     view
287     returns (uint256)
288 {
289     return allowed[_owner][_spender];
290 }
291
292 /**
293  * @dev Increase the amount of tokens that an owner allowed to a spender.
294  * approve should be called when allowed[_spender] == 0. To increment
295  * allowed value is better to use this function to avoid 2 calls (and wait until
296  * the first transaction is mined)
297  * From MonolithDAO Token.sol
298  * @param _spender The address which will spend the funds.
299  * @param _addedValue The amount of tokens to increase the allowance by.
300  */
301 // @CTK NO_OVERFLOW
302 // @CTK NO_BUF_OVERFLOW
303 // @CTK FAIL NO_ASF
304 /* @CTK increaseApproval
305    @tag assume_completion
306    @pre _spender != 0x0
307    @pre _spender != msg.sender
308    @post (__post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
309        _addedValue)
310    @post (__return) == (true)
311 */
312 function increaseApproval(
313     address _spender,
314     uint256 _addedValue
315 )
316     public
317     returns (bool)
318 {
319     allowed[msg.sender][_spender] = (
320         allowed[msg.sender][_spender].add(_addedValue));
321     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
322     return true;
323 }
324
325 /**
326  * @dev Decrease the amount of tokens that an owner allowed to a spender.
327  * approve should be called when allowed[_spender] == 0. To decrement
328  * allowed value is better to use this function to avoid 2 calls (and wait until
329  * the first transaction is mined)
330  * From MonolithDAO Token.sol
331  * @param _spender The address which will spend the funds.
332  * @param _subtractedValue The amount of tokens to decrease the allowance by.
333  */
334 // @CTK NO_OVERFLOW
335 // @CTK NO_BUF_OVERFLOW
336 // @CTK NO_ASF

```



```

336  /*@CTK decreaseApproval
337      @tag assume_completion
338      @pre _spender != 0x0
339      @pre _spender != msg.sender
340      @post (_subtractedValue > allowed[msg.sender][_spender]) -> (__post.allowed[msg.
          sender][_spender] == 0)
341      @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (__post.allowed[msg.
          sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue)
342  */
343  function decreaseApproval(
344      address _spender,
345      uint256 _subtractedValue
346  )
347      public
348      returns (bool)
349  {
350      uint256 oldValue = allowed[msg.sender][_spender];
351      if (_subtractedValue >= oldValue) {
352          allowed[msg.sender][_spender] = 0;
353      } else {
354          allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
355      }
356      emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
357      return true;
358  }
359
360 }
361
362
363
364 /**
365  * @title Ownable
366  * @dev The Ownable contract has an owner address, and provides basic authorization
        control
367  * functions, this simplifies the implementation of "user permissions".
368  */
369 contract Ownable {
370     address public owner;
371
372
373     event OwnershipRenounced(address indexed previousOwner);
374     event OwnershipTransferred(
375         address indexed previousOwner,
376         address indexed newOwner
377     );
378
379
380     /**
381      * @dev The Ownable constructor sets the original 'owner' of the contract to the
            sender
382      * account.
383      */
384     constructor() public {
385         owner = msg.sender;
386     }
387
388     /**
389      * @dev Throws if called by any account other than the owner.

```

```

390  */
391  modifier onlyOwner() {
392      require(msg.sender == owner);
393      _;
394  }
395
396  /**
397   * @dev Allows the current owner to relinquish control of the contract.
398   * @notice Renouncing to ownership will leave the contract without an owner.
399   * It will not be possible to call the functions with the 'onlyOwner'
400   * modifier anymore.
401   */
402   //@CTK NO_OVERFLOW
403   //@CTK NO_BUF_OVERFLOW
404   //@CTK NO_ASF
405   /*@CTK isOwner
406       @tag assume_completion
407       @pre msg.sender == owner
408       @post __post.owner == address(0)
409   */
410  function renounceOwnership() public onlyOwner {
411      emit OwnershipRenounced(owner);
412      owner = address(0);
413  }
414
415  /**
416   * @dev Allows the current owner to transfer control of the contract to a newOwner.
417   * @param _newOwner The address to transfer ownership to.
418   */
419   //@CTK NO_OVERFLOW
420   //@CTK NO_BUF_OVERFLOW
421   //@CTK NO_ASF
422   /*@CTK transferOwnership
423       @tag assume_completion
424       @pre msg.sender == owner
425       @pre msg.sender != _newOwner
426       @pre _newOwner != address(0)
427       @post __post.owner == _newOwner
428   */
429  function transferOwnership(address _newOwner) public onlyOwner {
430      _transferOwnership(_newOwner);
431  }
432
433  /**
434   * @dev Transfers control of the contract to a newOwner.
435   * @param _newOwner The address to transfer ownership to.
436   */
437   //@CTK NO_OVERFLOW
438   //@CTK NO_BUF_OVERFLOW
439   //@CTK NO_ASF
440   /*@CTK _transferOwnership
441       @tag assume_completion
442       @pre msg.sender == owner
443       @pre msg.sender != _newOwner
444       @pre _newOwner != address(0)
445       @post __post.owner == _newOwner
446   */
447  function _transferOwnership(address _newOwner) internal {

```

```

448     require(_newOwner != address(0));
449     emit OwnershipTransferred(owner, _newOwner);
450     owner = _newOwner;
451 }
452 }
453
454
455 contract BenepitToken is StandardToken, Ownable {
456     string public name = "Benepit";
457     string public symbol = "BNP";
458     uint8 public decimals = 18;
459
460     //@CTK NO_OVERFLOW
461     //@CTK NO_BUF_OVERFLOW
462     //@CTK NO_ASF
463     /*@CTK BenepitToken
464         @tag assume_completion
465         @post __post.balances[msg.sender] == __post.totalSupply_
466     */
467     constructor() public {
468         totalSupply_ = 30000000000 * 10**uint(decimals);
469         balances[msg.sender] = totalSupply_;
470     }
471 }

```