|  |  |  |  |
| --- | --- | --- | --- |
| **用例编号** | **用例名称** | **测试内容** | **测试方法** |
| 7.1.3 智能合约安全形式化验证（solidity语言智能合约） | | | |
| 3-1 | function\_precondition1.sol | 检测function\_precondition1.sol智能合约，验证通过  （precondition指调用函数前应当满足的条件，对于function\_precondition1.sol智能合约，x和y在调用add\_to\_x前是相等的，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择function\_precondition1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-2 | function\_precondition2.sol | 检测function\_precondition2.sol智能合约，验证不通过  （precondition指调用函数前应当满足的条件，对于function\_precondition1.sol智能合约，x和y在调用add\_to\_x前是相等的，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择function\_precondition2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-3 | function\_postcondition1.sol | 检测function\_postcondition1.sol智能合约，验证通过  （postcondition指调用函数后应当满足的条件，对于function\_postcondition1.sol智能合约，x和y在调用add\_to\_x后满足x == (y + 1)，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择function\_postcondition1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now  在智能合约安全形式化验证原型系统，选择solidity语言形式化验证，提交该测试用例并进行分析，“postcondition x == (y + 1)”应当验证通过 |
| 3-4 | function\_postcondition2.sol | 检测function\_postcondition2.sol智能合约，验证不通过  （postcondition指调用函数后应当满足的条件，对于function\_postcondition2.sol智能合约，x和y在调用add\_to\_x后不满足x == (y + 1)，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择function\_postcondition2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-5 | contract\_invariant1.sol | 检测contract\_invariant1.sol智能合约，验证通过  （invariant指调用函数前后应当满足的条件，对于contract\_invariant1.sol智能合约，x和y在调用add函数前后满足x == y，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择contract\_invariant1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-6 | contract\_invariant2.sol | 检测contract\_invariant2.sol智能合约，验证不通过  （invariant指调用函数前后应当满足的条件，对于contract\_invariant2.sol智能合约，当x为奇数，y为偶数时，调用add函数后不满足x == y，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择contract\_invariant2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-7 | loop\_invariant1.sol | 检测loop\_invariant1.sol智能合约，验证通过  （loop invarian指使用循环过程中应当满足的条件，对于loop\_invariant1.sol智能合约，使用循环后满足y <= x，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择loop\_invariant1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-8 | loop\_invariant2.sol | 检测loop\_invariant2.sol智能合约，验证不通过  （loop invarian指使用循环过程中应当满足的条件，对于loop\_invariant2.sol智能合约，当y为偶数，x为奇数时，使用循环后不满足y <= x，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择loop\_invariant2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-9 | modification\_specifier1.sol | 检测modification\_specifier1.sol智能合约，验证通过  （modifies指允许在函数中修改对应的变量，对函数add，对x和y的修改均满足要求，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modification\_specifier1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-10 | modification\_specifier2.sol | 检测modification\_specifier2.sol智能合约，验证不通过  （modifies指允许在函数中修改对应的变量，对函数add，由于没有modifies y约束，对y的修改不满足要求，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modification\_specifier2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-11 | modification\_specifier\_condition1.sol | 检测modification\_specifier\_condition1.sol智能合约，验证通过  （modifies x if n >= 0指当n>=0时才能修改x，函数add在修改x和y时均满足n>=0，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modification\_specifier\_condition1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-12 | modification\_specifier\_condition2.sol | 检测modification\_specifier\_condition2.sol智能合约，验证不通过  （modifies y if n < 0指当n < 0时才能修改y，函数add在修改y时n>=0，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择modification\_specifier\_condition2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-13 | uint\_sum\_function1.sol | 检测uint\_sum\_function1.sol智能合约，验证通过  （\_\_verifier\_sum\_uint(balances) == totalSupply指balances数据结构的uint类型的值的总和应该和totalSupply保持相等，transfer函数是对balances组内的转账，所以balances总数不变，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择uint\_sum\_function1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now  在智能合约安全形式化验证原型系统，选择solidity语言形式化验证，提交该测试用例并进行分析，“invariant \_\_verifier\_sum\_uint(balances) == totalSupply”应当验证通过 |
| 3-14 | uint\_sum\_function2.sol | 检测uint\_sum\_function2.sol智能合约，验证不通过  （\_\_verifier\_sum\_uint(balances) == totalSupply指balances数据结构的uint类型的值的总和应该和totalSupply保持相等，transfer函数是对balances组内的转账，但是仅有付款的语句没有收款的语句，balances总数减少，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择uint\_sum\_function2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-15 | int\_sum\_function1.sol | 检测int\_sum\_function1.sol智能合约，验证通过  （\_\_verifier\_sum\_int(balances) == totalSupply指balances数据结构的int类型的值的总和应该和totalSupply保持相等，transfer函数是对balances组内的转账，所以balances总数不变，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择int\_sum\_function1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-16 | int\_sum\_function2.sol | 检测int\_sum\_function2.sol智能合约，验证不通过  （\_\_verifier\_sum\_uint(balances) == totalSupply指balances数据结构的int类型的值的总和应该和totalSupply保持相等，transfer函数是对balances组内的转账，但是仅有付款的语句没有收款的语句，balances总数减少，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择int\_sum\_function2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-17 | uint\_old\_value\_function1.sol | 检测uint\_old\_value\_function1.sol智能合约，验证通过  （x == \_\_verifier\_old\_uint(y) + 1指x和y都是uint类型的值，x应当等于y修改前的值加1，对于函数add\_to\_x，x等于“y = y - 3”之前的y的值加1，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择uint\_old\_value\_function1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-18 | uint\_old\_value\_function2.sol | 检测uint\_old\_value\_function2.sol智能合约，验证不通过  （x == \_\_verifier\_old\_uint(y) + 2指x和y都是uint类型的值，x应当等于y修改前的值加2，对于函数add\_to\_x，x等于“y = y - 3”之前的y的值加1，不满足条件，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择uint\_old\_value\_function2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-19 | int\_old\_value\_function1.sol | 检测int\_old\_value\_function1.sol智能合约，验证通过  （x == \_\_verifier\_old\_uint(y) + 1指x和y都是int类型的值，x应当等于y修改前的值加1，对于函数add\_to\_x，x等于“y = y - 3”之前的y的值加1，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择int\_old\_value\_function1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-20 | int\_old\_value\_function2.sol | 检测int\_old\_value\_function2.sol智能合约，验证不通过  （x == \_\_verifier\_old\_uint(y) + 2指x和y都是int类型的值，x应当等于y修改前的值加2，对于函数add\_to\_x，x等于“y = y - 3”之前的y的值加1，不满足条件，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择int\_old\_value\_function2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-21 | emit\_specifier1.sol | 检测emit\_specifier1.sol智能合约，验证通过  （emits e指函数必须产生日志e，对于函数f，其调用了emit产生了e，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择emit\_specifier1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-22 | emit\_specifier2.sol | 检测emit\_specifier2.sol智能合约，验证不通过  （emits e指函数必须产生日志e，对于函数f，其没有调用emit功能，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择emit\_specifier2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-23 | ERC20\_event1.sol | 检测ERC20\_event1.sol智能合约，验证通过  （该智能合约中插入了多条emits约束均被满足，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_event1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-24 | ERC20\_event2.sol | 检测ERC20\_event2.sol智能合约，验证不通过  （该智能合约中插入了多条emits约束，其中一条加在transfer函数上的emits Transfer约束指transfer函数必须产生Transfer日志，但是该函数并没有调用emit功能产生日志，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_event2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-25 | ERC20\_totalsuppply1.sol | 检测ERC20\_totalsuppply1.sol智能合约，验证通过  （\_\_verifier\_sum\_uint(balances) == totalSupply指balances数据结构的int类型的值的总和应该和totalSupply保持相等，该约束被放在了合约边外，对合约内所有函数有效，任意一个transfer函数均是对balances组内的转账，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_totalsuppply1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-26 | ERC20\_totalsuppply2.sol | 检测ERC20\_totalsuppply2.sol智能合约，验证不通过  （\_\_verifier\_sum\_uint(balances) == totalSupply指balances数据结构的int类型的值的总和应该和totalSupply保持相等，该约束被放在了合约边外，对合约内所有函数有效，其中一个transfer函数是对balances组内的转账，但是仅有付款的语句没有收款的语句，balances总数减少，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_totalsuppply2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-27 | ERC20\_modifies1.sol | 检测ERC20\_modifies1.sol智能合约，验证通过  （该智能合约中插入了多条modifies约束均被满足，验证通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_modifies1.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |
| 3-28 | ERC20\_modifies2.sol | 检测ERC20\_modifies2.sol智能合约，验证不通过  （该智能合约中插入了多条modifie条件，其中一条modifies balanceOf if balanceOf[msg.sender] >= amount指只有当balanceOf[msg.sender] >= amount时才能修改balanceOf，但是transfer函数中没有对balanceOf[msg.sender] >= amount的require要求，导致可以对balanceOf无条件修改，该约束条件失效，验证不通过） | 1. 打开网页  https://github.com/my-code-cloud/EvaluationCases/tree/main/solidity%E5%90%88%E7%BA%A6%E5%BD%A2%E5%BC%8F%E5%8C%96%E9%AA%8C%E8%AF%81  2. 选择ERC20\_modifies2.sol将其代码复制到<http://39.103.152.161/>的合约代码中  3. 选择Solidity，形式化验证  4. 点击Analyze Now |