Scrify: an online tool to validate Bitcoin script

Stefano Bistarelli, Andrea Bracciali, Ivan Mercanti
HOW TRANSACTIONS WORK

1. Joe's public key
2. Hash
3. Previous owner's sign
4. Joe's private key

5. Alice's public key
6. Hash
7. Joe's sign
8. Alice's private key

9. Bob's public key
10. Hash
11. Alice's sign
12. Bob's private key
How Transactions Work

1. Transaction 1
   - Locking script
   - Unlocking script

2. Transaction 2
   - Locking script
   - Unlocking script

3. Transaction 3
   - Locking script
   - Unlocking script

Verify the transactions.
Add 2 to 8 and check if the result is equal.
BITCOIN SCRIPT

SCRIPT

Locking Script

Unlocking Script
ScriFY

Input Script (write or select Example P2PKH)

☐ Redeem Script  Run
OUR TOOL

SCRIFY

ScriFy

Input Script (write or select Example equation)

OP_2 OP_ADD OP_8 OP_EQUAL

☐ Redeem Script    Run
Program Output

---------- AST ----------
0:     OP_2;
1:     OP_ADD;
2:     OP_8;
3:     OP_EQUAL;

---------- Inferred ----------

--- Symbolic evaluation report of execution branch 0 ---

Branch's decision points:

Required initial main stack:
|-------------------------|
head -> | X_(0)

Required initial alternative stack:
head -> |-------------------------|

Inferred constraints:
Constraint: ((Int 2) + (X_(0))) == (Int 8)

Resulting symbolic stack:
[ ]

---------- Verdict ----------
types correct, 1 branch(es) viable
Locking Script

2  ADD  8  EQUAL

Unlocking Script

6
PAY TO SCRIPT HASH (P2SH)

**Locking Script**
- HASH160
- \(<\text{redeemSH}>\)
- EQUAL

**Unlocking Script**
- \(<\text{redeem script}>\)
STANDARD TRANSACTIONS

PAY TO SCRIPT HASH (P2SH)

Unlocking Script

<redeem script>

Locking Script

HASH160 <redeemSH> EQUAL

2 <PubK1> <PubK2> <PubK3> 3 CHECKMULTISIG
PAY TO SCRIPT HASH (P2SH)

**Locking Script**

- HASH160
- \(<\text{redeemSH}>\)
- EQUAL
- \(<\text{PubK1}>\)
- \(<\text{PubK2}>\)
- \(<\text{PubK3}>\)
- \(2\)
- \(3\)
- CHECKMULTISIG

**Unlocking Script**

- OP_0
- \(<\text{sig1}>\)
- \(<\text{sig2}>\)
- \(<\text{redeem script}>\)
EXAMPLES

ONLINE TOOL

"SPECIAL EXECUTION"
---------------------
--------- AST --------
0: OP_P2SH;
1: OP_HASH160:

OP_HASH160 PUSH a7690b6478b372940e63794204a7690b6478b372 OP_EQUAL

Required initial main stack:
|--------------|
head -> | X(0)
| X(-1)

Required initial alternative stack:
head -> |--------------|

Inferred constraints:
Constraint: (Hash_20 (X(0))) == (BS_20 "a7690b6478b372940e63794204a7690b6478b372")
Resulting symbolic stack:
[X(-1) = ExeVerifier(X(0))]

---------------------
---------- Verdict ------
types correct, 1 branch(es) viable
"SPECIAL EXECUTION"
----------------------
-------------- AST --------------
0: OP_P2SH;
1: OP_HASH160;
2: BS_20 "a7690b6478b372940e63794204a7690b6478b372";
3: OP_EQUAL;
----------------------
----- Inferred ------
---
--- Symbolic evaluation report of execution branch 0 ---
---
Branch's decision points:

Required initial main stack:
|-----------------------|
head -> | X_(0) |
| X_(-1) |

Required initial alternative stack:
head -> |-----------------------|

Inferred constraints:
Constraint: (Hash_20 (X_(0))) == (BS_20 "a7690b6478b372940e63794204a7690b6478b372")
Resulting symbolic stack:
[X_(-1) = ExeVerifier_(X_(0))]

-------- Verdict --------
types correct, 1 branch(es) viable
PAY TO WITNESS SCRIPT HASH (P2WSH)

Locking Script

Unlocking Script

Witness script
PAY TO SCRIPT HASH (P2SH)

**Locking Script**

```
HASH160 <redeemSH> EQUAL
```

**Unlocking Script**

```
OP_0 <sig1> <sig2> <redeem script>
```
"SPECIAL EXECUTION"
------------------------
--------- AST ---------
0:    OP_P2SH;
1:    OP_SHA256;

OP_0 PUSH 701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d

Required initial main stack:
|-------------------------|
head -> | X_ (0)
| X_ (-1)
|-------------------------|

Required initial alternative stack:
head -> |-------------------------|

Inferred constraints:
Constraint: (Hash_32 (X_ (0))) == (BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d")

Resulting symbolic stack:
[X_ (-1) = ExeVerifier_ (X_ (0))]

--------- Verdict ---------
types correct, 1 branch(es) viable
"SPECIAL EXECUTION"
------------------------
-------- AST --------
0:    OP_P2SH;
1:    OP_SHA256;
2:    BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d";
3:    OP_EQUAL;
------------------------
-------- Inferred ------
--- Symbolic evaluation report of execution branch 0
---
Branch's decision points:

Required initial main stack:
    |------------------------|
head ->  | X_(0) |
         | X_(-1) |
    |------------------------|
Required initial alternative stack:
head ->  |------------------------|
Inferred constraints:
    Constraint: (Hash_32 (X_(0))) == (BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d")
Resulting symbolic stack:
    [X_(-1) = ExeVerifier_(X_(0))]

---------- Verdict ----------
types correct, 1 branch(es) viable
REDEEM SCRIPT

ScriFY

Input Script (write or select Example P2WSH)

OP_0  PUSH 701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d

 Redeem Script

52210375e00eb7e29da82b89367947f29ef34afb75e8654f6ea368e0eccfd92976b7c2103a1b26313f430c4b15bb1fdce663207659d8cac749a0e53d70eff01874496eff2103c96d495bfdd5ba4145e3e046fee45e84a8a48ad05bd8dbb395c011a32cf0f88653ae
EXAMPLES

-------------- AST --------------
0: OP_2;
1: BS_33 "0375e00eb72e29da82b89367947f29ef34afb75e8654f6ea368e0acdfd92976b7c";
2: BS_33 "03a1b26313f430c4b15bb1fdce663207659d8cac749a0e53d70eff01874496eff";
3: BS_33 "03c96d495bfdd5ba4145e3e046fee45e84a8a48ad05bd8dbb395c011a32cf9f880";
4: OP_3;
5: OP_CHECKMULTISIG;
6: BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d";
7: OP_SHA256;
8: BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d";
9: OP_EQUAL;

-------------- Inferred --------------
--- Symbolic evaluation report of execution branch 0 ---
---
Branch's decision points:

Required initial main stack:

	[--------------------------------]
head -> | X_(0)
| X_(-1)
| X_(-2)
[--------------------------------]

Required initial alternative stack:

head -> [------------------]

Inferred constraints:

Constraint: (Hash_32 (BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d")) == (BS_32 "701a8d401c84fb13e6baf169d59684e17abd9fa216c8cc5b9fc63d622ff8c58d")
Constraint: (X_(-2)) == (Int 0)

Resulting symbolic stack:

[MultiSig [X_(-1),X_(0)]] [BS_33 "0375e00eb72e29da82b89367947f29ef34afb75e8654f6ea368e0acdfd92976b7c",BS_33 "03a1b26313f430c4b15bb1fdce663207659d8cac749a0e53d70eff01874496eff",BS_33 "03c96d495bfdd5ba4145e3e046fee45e84a8a48ad05bd8dbb395c011a32cf9f880"]

-------------- Verdict --------------
types correct, 1 branch(es) viable
CONCLUSION AND FUTURE WORKS
CONCLUSION AND FUTURE WORKS

- We design a online tool for symbolic model checker script

scrify.dmi.unipg.it
CONCLUSION AND FUTURE WORKS

- We design an online tool for symbolic model checker script
- We include several special cases
CONCLUSION AND FUTURE WORKS

- We design an online tool for symbolic model checker script
- We include several special cases
- Extend to multi-step protocols
Scrify: an online tool to validate Bitcoin script

Stefano Bistarelli, Andrea Bracciali, Ivan Mercanti

THANKS FOR THE ATTENTION. QUESTIONS?

DLT 23 - 26/05/2023