Formal Verification in Scala

Ramon Boss, Anna Doukmak

2019-06-13

Table of Contents

Table of Contents

Stainless

- Pure Scala
- Pre- and Postcondition
- Outcome: valid, invalid, unknown

Example: max

```scala
def max(x: Int, y: Int): Int = {
  val d = x - y
  if (d > 0) x
  else y
}
```

---

https://epfl-lara.github.io/stainless/tutorial.html#warm-up-max

Example: max

```
1  def max(x: Int, y: Int): Int = {
2    require(0 <= x && 0 <= y)
3    val d = x - y
4    if (d > 0) x
5    else y
6  } ensuring (res =>
7    x <= res && y <= res && (res == x || res == y))
```

---

https://epfl-lara.github.io/stainless/tutorial.html#warm-up-max

Example: max

```
[...     ]
[Warning ]  => INVALID
[Warning ] Found counter-example:[Warning ]   x: Int -> 1
[Warning ]   y: Int -> -2147483648
[  Info  ]  - Result for 'postcondition' VC for max @2:3:
[Warning ]  => INVALID
[Warning ] Found counter-example:
[Warning ]   x: Int -> -2147483648
[Warning ]   y: Int -> 2147483647
[...     ]
```

Properties

- Bitcoin-S
- No-Inflation Property
- Addition-with-Zero Property

Table of Contents

Rewrite Code: Turning Object into Case Object

This:

```
1 object Satoshis extends BaseNumbers[Satoshis] {
2   val zero = Satoshis(Int64.zero)
3   val one = Satoshis(Int64.one)
4 }
```

Becomes this:

```
1 case object Satoshis extends BaseNumbers[Satoshis] {
2   val zero = Satoshis(Int64.zero)
3   val one = Satoshis(Int64.one)
4 }
```

Rewrite Code: Getting Rid of Abstract Type Member

This:

```
1  sealed abstract class CurrencyUnit {
2    type A
3
4    protected def underlying: A
5  }
6
7  sealed abstract class Satoshis extends CurrencyUnit {
8    override type A = Int64
9  }
```

Becomes this:

```
1  sealed abstract class CurrencyUnit {
2    protected def underlying: Int64
3  }
4
5  sealed abstract class Satoshis extends CurrencyUnit
```

Rewrite Code: BigInt &-Function to Bound Check

This:

```
1  sealed abstract class Number {
2    def andMask: BigInt
3    def checkResult(result: BigInt): BigInt = {
4      require((result & andMask) == result)
5      result
6    }
7  }
```

Becomes this:

```
1  sealed abstract class Number {
2    def checkResult(result: BigInt): BigInt = {
3      require(
4        result <= BigInt("9223372036854775807") &&
5        result >= BigInt("-9223372036854775808")
6      )
7      result
8    }
9  }
```

Formal Specification

```
1  def +(c: CurrencyUnit): CurrencyUnit = {
2    require(c.satoshis == Satoshis.zero)
3    Satoshis(
4      satoshis.underlying + c.satoshis.underlying
5    )
6  } ensuring(res => res.satoshis == this.satoshis)
```

Output of Stainless

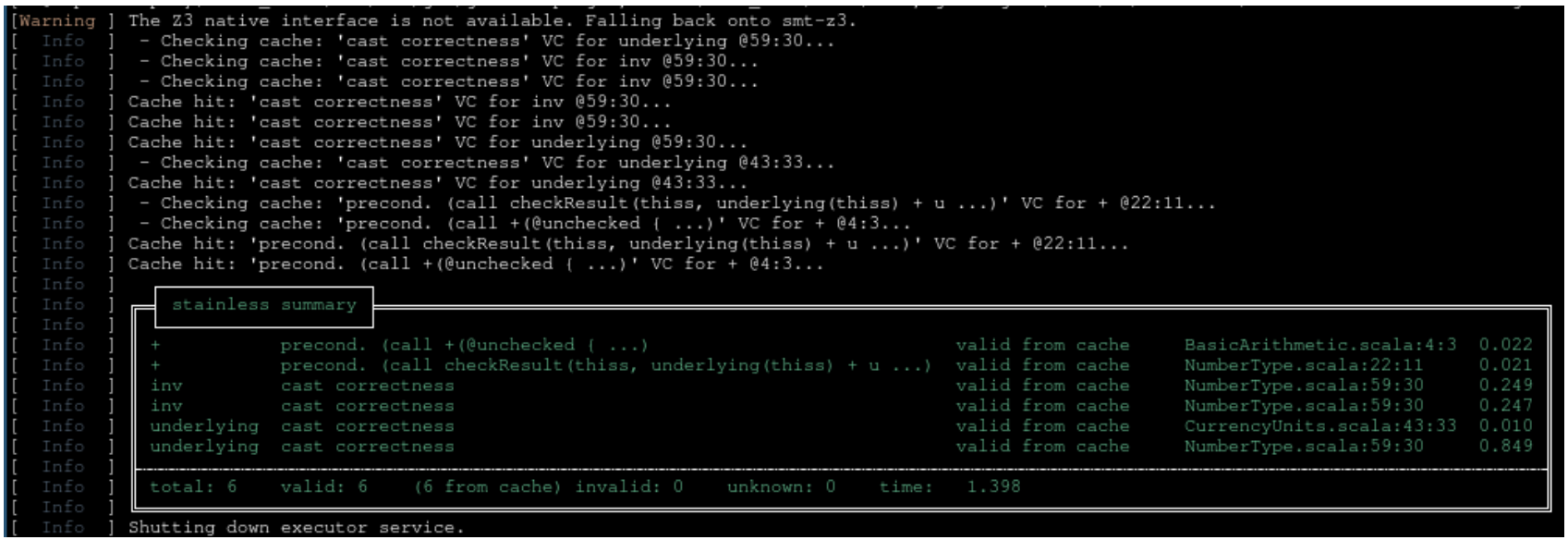

Table of Contents

Conclusion

- Write verifiable code
- We verified not original Bitcoin-S code
- Provided feedback to Stainless developers
- Found a bug in Bitcoin-S

Found Bug in Bitcoin-S

6 core-test/src/test/scala/org/bitcoins/core/protocol/transaction/TransactionTest.scala

```
@@ -294,6 +294,12 @@ class TransactionTest extends FlatSpec with MustMatchers {
294 294         }
295 295       }
296 296
297     +   it must "check transaction with two out point referencing the same tx with different indexes" in {
298     +     val hex = "0200000002924942b0b7c12ece0dc8100d74a1cd29acd6cfc60698bfc3f07d83890eec20b6000000006a47304402202831
299     +     val btx = BaseTransaction.fromHex(hex)
300     +     ScriptInterpreter.checkTransaction(btx) must be(true)
301     +   }
302     +
297 303     private def findInput(
298 304         tx: Transaction,
299 305         outPoint: TransactionOutPoint): Option[(TransactionInput, Int)] = {
```

4 core/src/main/scala/org/bitcoins/core/script/interpreter/ScriptInterpreter.scala

```
@@ -779,8 +779,8 @@ sealed abstract class ScriptInterpreter {
779 779       val totalSpentByOutputs: CurrencyUnit =
780 780         outputValues.fold(CurrencyUnits.zero)(_ + _)
781 781       val allOutputsValidMoneyRange = validMoneyRange(totalSpentByOutputs)
782     -     val prevOutputTxIds = transaction.inputs.map(_.previousOutput.txId)
783     -     val noDuplicateInputs = prevOutputTxIds.distinct.size == prevOutputTxIds.size
    782 +     val prevOutputs = transaction.inputs.map(_.previousOutput)
    783 +     val noDuplicateInputs = prevOutputs.distinct.size == prevOutputs.size
784 784
785 785       val isValidScriptSigForCoinbaseTx = transaction.isCoinbase match {
786 786         case true =>
```