Formal Verification in Scala

Ramon Boss, Anna Doukmak

2019-06-13

Table of Contents

Formal Verification

Problems Verifying Bitcoin-S

Conclusion

Table of Contents

Formal Verification

Problems Verifying Bitcoin-S

Conclusion

Stainless

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

Example: max

```scala
1  def max(x: Int, y: Int): Int = {
2    val d = x - y
3    if (d > 0) x
4    else y
5  }
```

---

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

Example: max

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

Example: max

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

Example: max

```scala
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

Properties

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

Table of Contents

Rewrite Code: Turning Object into Case Object

This:

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

Becomes this:

```scala
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:

```scala
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:

```scala
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:

```scala
sealed abstract class Number {
  def andMask: BigInt
  def checkResult(result: BigInt): BigInt = {
    require((result & andMask) == result)
    result
  }
}
```

Becomes this:

```scala
sealed abstract class Number {
  def checkResult(result: BigInt): BigInt = {
    require(
      result <= BigInt("9223372036854775807") &&
      result >= BigInt("-9223372036854775808")
    )
    result
  }
}
```

Formal Specification

```scala
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 =>
```