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

```
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))
```

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: 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  }
```

Rewrite Code: Getting Rid of Abstract Type Member

This:

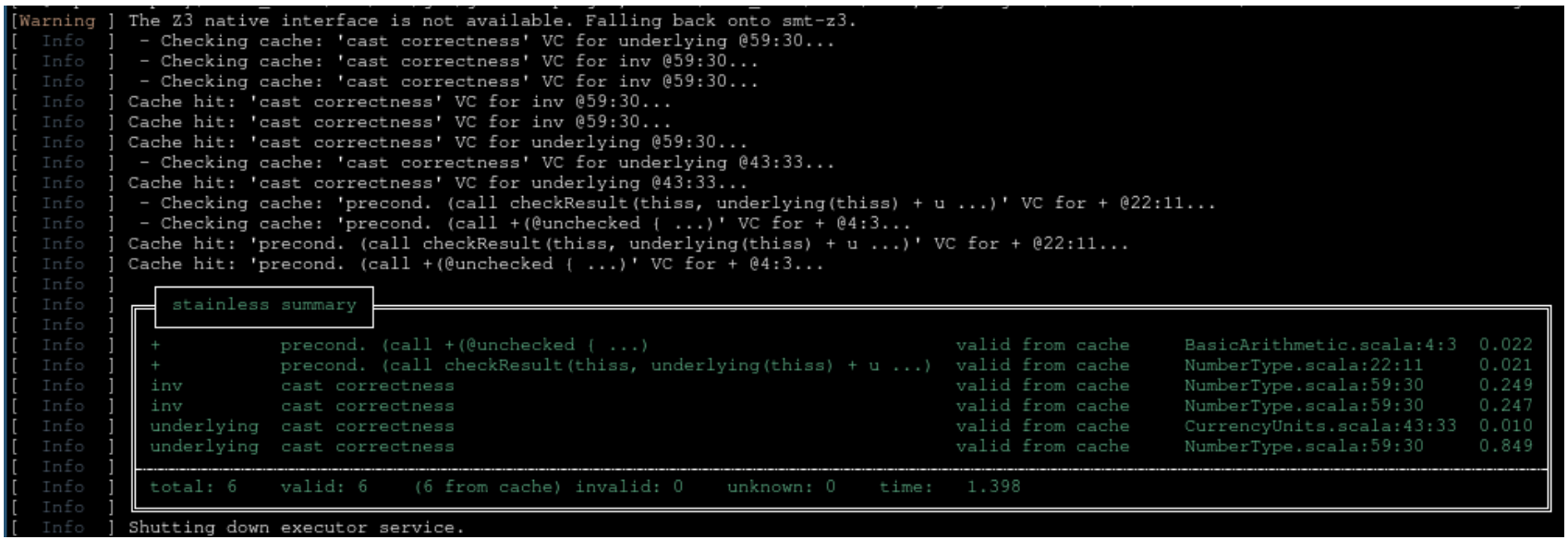
```
1  sealed abstract class CurrencyUnit {
2    type A
3
4    protected def underlying: A
5  }
6
7  sealed abstract class Satoshis extends CurrencyUnit
```

Becomes this:

```
1    ???
```

Table of Contents

Output of Stainless



Lessons Learned

- Write Verifiable Code
- We Verified Not Original Bitcoin-S Code