Goal

In this thesis we experiment in formal verification and try to verify Scala code.
We verify a fragment of Bitcoin-S. Bitcoin-S is a Scala implementation of the Bitcoin protocol.
To verify the code we use Stainless.

Summary

Formal verification is a method to check the correctness of a program based on the formal specification. Using a verification tool, all possible inputs can be explored, in contrast to unit testing, where the inputs must be specified separately.

Stainless

We use Stainless as our verification tool.

- It takes Scala code, written in functional style allowing some imperative features, as input
- Explores all possible inputs
- Reports inputs for which a program fails
- Gives counterexamples which violate the specification
- or confirms the correctness of a program

Bitcoin-S

First, the property we want to verify:

- a non-coinbase transaction cannot generate new coins

We transform the code to functional code so Stainless can verify it.
Our research has shown that the code implementing this property has many dependencies. Thus, a large part of the code must be rewritten which needs a lot of time.

So, we turn our analysis to another functionality of Bitcoin-S:

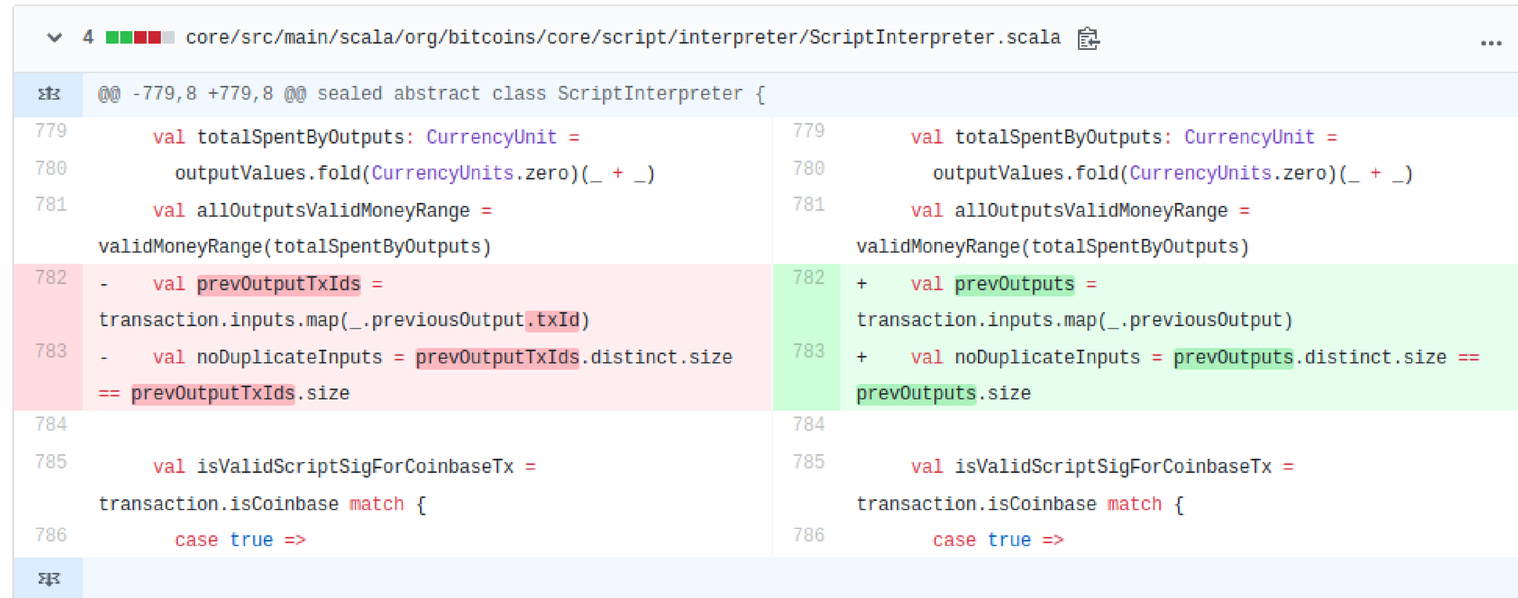
- coin addition with zero results in the same value

To verify this functionality we rewrite the code needed for the addition and define the formal specification with Stainless functions *require* and *ensuring*.

```scala
def +(c: CurrencyUnit): CurrencyUnit = {
  require(c.satoshis == Satoshis.zero)
  Satoshis(
    satoshis.underlying + c.satoshis.underlying
  )
} ensuring(
   res => res.satoshis == this.satoshis
)
```

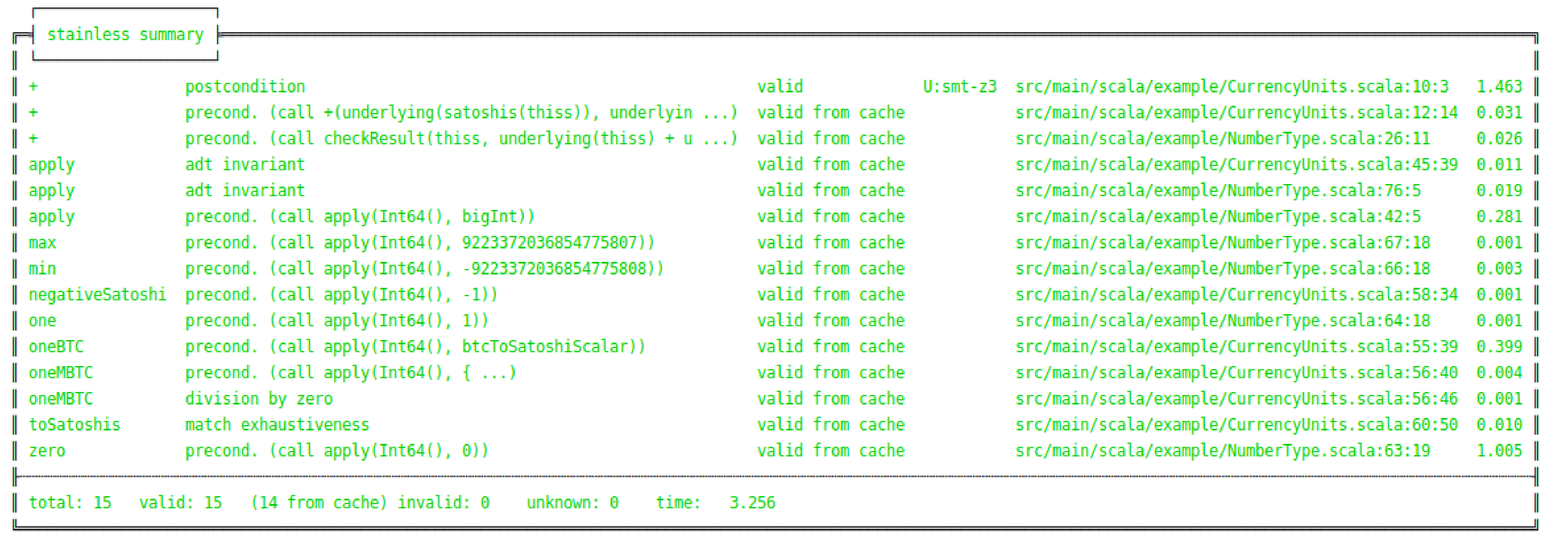
Results

During the work we found a bug in Bitcoin-S in the function checking the correctness of a transaction. Its implementation didn't allow transactions that reference two or more outputs of the same previous transaction. We fixed it and made a pull request which has been merged by developers of the Bitcoin-S project.



Furthermore, we verified the coin addition with zero. For this purpose we extracted two classes and rewrote their code.

Here the output of the verified code:



Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Experiments in Formal Verification of Scala Code

Bachelor Thesis 2019 Degree Program Computer Science

Graduates: Ramon Boss
Anna Doukmak
Professor: Kai Brünnler