Fundação Getúlio Vargas

Modelagem Matemática

Programming a criptocurrency in Agda

Student:
Guilherme Horta Alvares da Silva
Professor:
Doctor Flávio Codeço Coelho

Contents

1 Introduction **2**
1.1 The bitcoin 2
1.2 Agda Introduction 2
1.3 UTXO Bitcoin 3
1.4 TXTree in Agda 3

2 Methods **3**

3 Conclusion **3**

References **4**

1 Introduction

The bitcoin (Nakamoto et al., 2008) was one of the first cripto currencies created in the world. Satoshi Nakamoto created it in 2008 using many ideas from cypherpunk community.

Before bitcoin, there were a lot of cryptograph cash ideas based on ecash protocol. Adam Back developed a proof-of-work scheme for spam control. Wei Dai propose b-money for the first proposal for distributed digital scarcity. And Hal Finney created a reusable proof of work for hashcash for its algorithm of proof of work.

Since 2008, the total market of Bitcoin came to 330 billions dollars in 17 of December of 2018 and his value has a historic record of 20 thousands dollars. Other criptocurrencies like Ethereum and Monero were created and they both have a good market value. Cripto currencies will be a great finance instrument for a near future.

Cripto currencies are used for smart contracts too. For example, it is possible to reserve some part of money to the seller in the blockchain. To unlock the money, it is necessary to get a signature of the buyer. If the buyer receive the product, he will sign the contract. If the product does not come in time, the buyer receive his money back. Smart contracts are widely adopted today the big funds and these contracts are fully governed by algorithms.

1.1 The bitcoin

1.2 Agda Introduction

Agda is a dependently typed functional language developed by Norell at Chalmers University of Technology as his PhD Thesis. The current version of Agda is Agda 2.

Agda is also a proof assistance based on intensional Martin-Löf type theory. It looks like CoQ, but does not have tatics. Agda is a total language, so it is garanteed that the code always terminal and coverage all inputs. Agda needs it to be a consistent language.

Agda has inductive data types that are similar to algebric data types in non-depently typed programming language. The definition of Peano numbers in Agda:

```
data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ
```

Definitions in Agda are done using induction. For example, the sum of two numbers in Agda:

```
_+_ : ℕ → ℕ → ℕ
zero + m = m
suc n + m = suc (n + m)
```

In Agda, because of dependent types, it is possible to make some restrictions in types that is not possible in other language. For example, get the first element of a vector. For it, it is necessary to specify in the type that the vector should have at size greater or equal tha than one.

```
head : {A : Set} {n : ℕ} (vec : Vector A (suc n)) → A
head (x :: vec) = x
```

Another good example is that in sum of two matrices, they should have the same dimentions.

```
_+m_ : {m n : ℕ} (P Q : Matrix ℕ m n) → Matrix ℕ m n
[] +m [] = []
(vx :: P) +m (vy :: Q) = (vx +v vy) :: (P +m Q)
```

1.3 UTXO Bitcoin

1.4 TXTree in Agda

2 Methods

3 Conclusion

References

Nakamoto, S., et al. (2008). Bitcoin: A peer-to-peer electronic cash system.