

# Qualificação

Guilherme

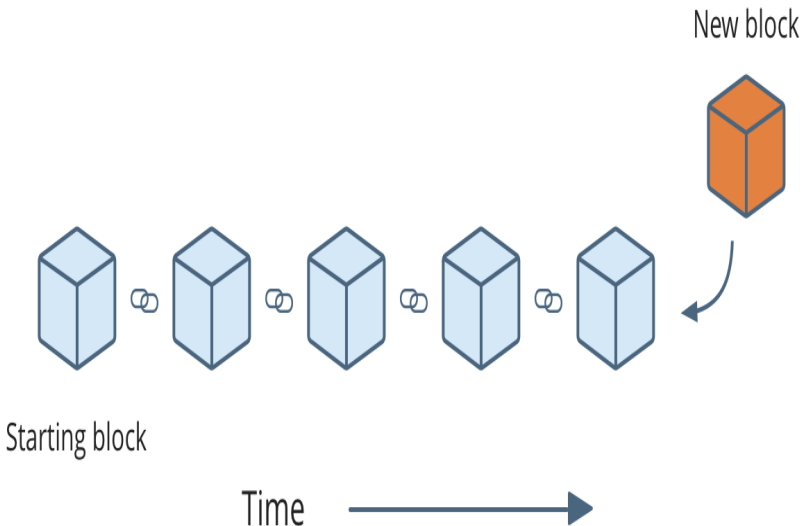
FGV

2019

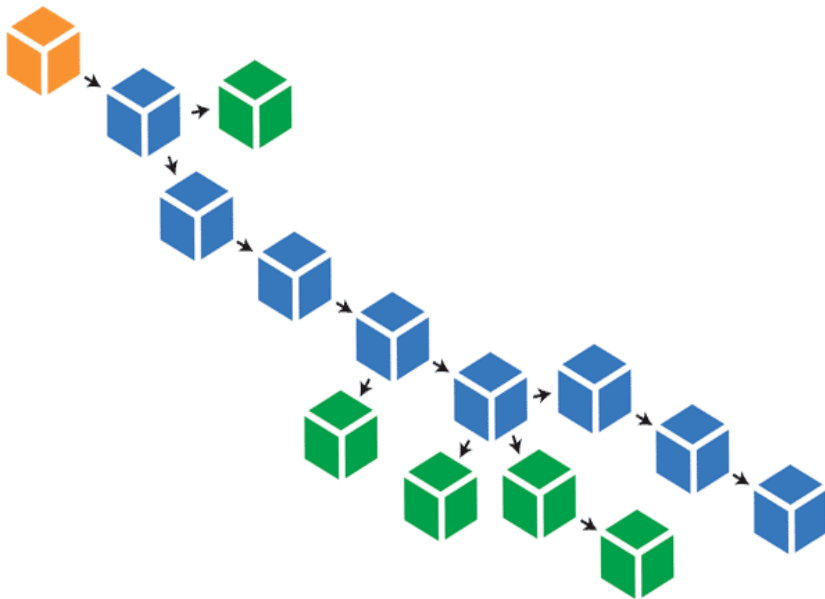
# Bitcoin

- É uma cripto-moeda descentralizada, sendo uma forma de dinheiro eletrônico.
- É considerada a primeira moeda digital mundial descentralizada, constituindo um sistema econômico alternativo (peer-to-peer electronic cash system), e responsável pelo ressurgimento do sistema bancário livre.
- O Bitcoin permite transações financeiras sem intermediários, mas verificadas por todos usuários (nodos) da rede, que são gravadas em um banco de dados distribuídos, chamado de blockchain, uma rede descentralizada.

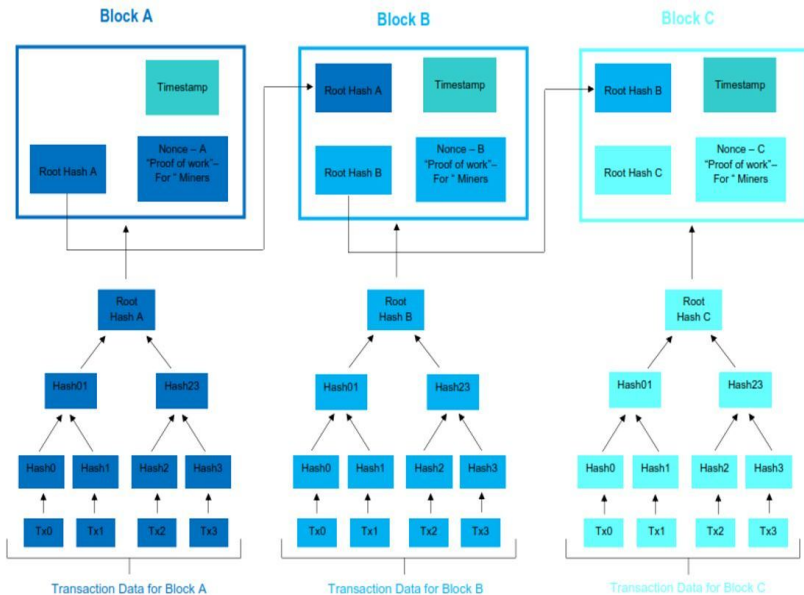
# Blockchain



# Blockchain



# Blockchain



# Agda

- Linguagem funcional e com sistema de tipos expressivo, capaz de representar especificações.
- Capaz de especificar e programar em um único lugar. Processo de verificação acontece no compilador.

# Agda — II

- Linguagem não possui Built-in como em Python. Tipos como inteiros, ponto flutuantes, strings, vetores deve, ser definidos pelo próprio usuário.
- Tipos em Agda são uma generalização de tipos de dados algébricos encontrados em Haskell e ML.

# Agda — III

- Definindo os números naturais com axiomas de peano

```
data  $\mathbb{N}$  : Set where  
  zero :  $\mathbb{N}$   
  suc   :  $\mathbb{N} \rightarrow \mathbb{N}$ 
```



# Agda — IV

- Adição em Agda:

$\_ + \_ : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$

$\text{zero} + n = n$

$\text{suc } m + n = \text{suc } (m + n)$

## Agda — V

- Dizemos que um certo tipo é dependente se este depende de um valor.
- Exemplo — Listas indexadas por seu tamanho:

```
data Vec (A : Set) : ℕ → Set where
  [] : Vec A 0
  _::_ : ∀ {n} → A → Vec A n → Vec A (suc n)
```

# Agda — VI

**data** GenesisBlock :  $\mathbb{N} \rightarrow \text{Set}$  **where**

**block** :  $(n : \mathbb{N}) \rightarrow (sb : \text{SimpleBlock}) \rightarrow n \equiv \text{hashBlock } sb \rightarrow \text{GenesisBlock } n$

**data** Block :  $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \text{Set}$  **where**

**block** :  $(n : \mathbb{N}) \rightarrow (m : \mathbb{N}) \rightarrow (sb : \text{SimpleBlock}) \rightarrow m \equiv \text{hashBlock } sb \rightarrow \text{Block } n m$

**data** Blockchain :  $\mathbb{N} \rightarrow \text{Set}$  **where**

**gen** :  $\{n : \mathbb{N}\} \rightarrow \text{GenesisBlock } n \rightarrow \text{Blockchain } n$

**cons** :  $\{n m : \mathbb{N}\} \rightarrow \text{Block } n m \rightarrow \text{Blockchain } n \rightarrow \text{Blockchain } m$

# Objetivos

- Programar uma criptomoeda (parecida com o Bitcoin) em Agda. Uma linguagem com tipos dependentes.



# Justificativa

- Programar um protocolo de criptomoedas livre de BUGs
- Com Agda, além da programação da criptomoeda, temos a sua especificação de como ela deveria se comportar

# Maleabilidade de transacao

- Nesse tipo de bug, era possível alterar o hash da transação depois que ela foi enviada
- Todos os dados para calcular do hash não eram previamente calculados. Ou seja, o minerador conseguiria alterar o hash da transação
- Possível ataque seria um usuário enviar uma transação e não receber a confirmação do envio da transação. Logo em seguida, este usuário enviaria de novo outra transação. Dessa forma, ele gastaria 2 vezes a mesma quantidade
- Esse tipo de BUG poderia ser evitado usando tipos dependentes. Colocando como característica da transação, o fato de seu ID ser único

# DAO Bug

- Bug que aconteceu em um cripto-contrato da rede Ethereum com um prejuízo de mais do que 250 milhões de dólares
- No cripto-contrato, existia uma função recursiva que não terminava. Ou seja, o usuário enviava uma quantidade de ethereum, depois acontecia um loop infinito e só depois era feito a atualização do seu balanço
- Em Agda, esse tipo de bug seria evitado, pois é necessário provar que a função termina. Logo, loops infinitos não são possíveis em Agda

# O que já foi feito

- Programada cripto-moeda em python
- Programada parte da blockchain em Agda
- Pelo paper, já foi feito parte de transações. UTXO (Unspent transaction output)



# Próximos passos

- Juntar a parte da blockchain com transações
- Se necessário, provar alguns teoremas relacionados ao bitcoin

# Teoremas

- Se uma transação tem algum output e ele não foi usada em nenhuma outra transação. Então ela deve estar na lista de outputs transactions não usados
- Se uma transação tem algum output que foi gasto. Ele não pode ser usado novamente.
- Provar que transações e mensagens ids são únicos

# O que não será feito

- Modelo de cripto-moeda em que é possível algum tipo de fork. No bitcoin, é possível que exista algum tipo de fork temporário
- Pool de transações. Sua utilidade é apenas para guardar as transações que ainda não foram adicionadas na blockchain. Isso pode ser feito fora do protocolo principal
- Otimização e protocolos RPC. O objetivo do projeto é definir as propriedades da cripto-moeda, não como ela será implementada e usada

## Livros

