Task Oriented Pearl:

Distributed Blockchain Applications

M. Lubbers[1,2] J.M. Jansen[1]

[1]Military Technical Sciences
Netherlands Defense Academy

[2]Institute for Computing and Information Sciences
Radboud University Nijmegen

5th January, 2018

Hashing functions

Hash function

$:: \text{HashFun} :== (\text{String} \to \text{String})$

Properties

- Deterministic
- Uniform
- Fixed size output
- Non-invertible

Examples

- $\mathrm{MD}\{2,4,5,6\}$
- $\mathrm{SHA}\{1,224,256,2,385,512,3\}$
- $\mathrm{BLAKE}\{\text{-}256,\text{-}512,2\mathrm{S},2\mathrm{B}\}$

Hashing functions

Hash function

$:: \text{HashFun} :== (\text{String} \to \text{String})$

Properties

- Deterministic
- Uniform
- Fixed size output
- Non-invertible

Examples

- $\mathrm{MD}\{2,4,5,6\}$
- $\mathrm{SHA}\{1,224,256,2,385,512,3\}$
- $\mathrm{BLAKE}\{\text{-}256,\text{-}512,2\mathrm{S},2\mathrm{B}\}$

`e.g. 03897e8ab0b92b39898dc58be3e03e15af4ff710`

Block

What is a block

$$:: \text{Block} = \{\text{data} :: \text{String}, \text{nonce} :: \text{Int}\}$$
$$:: \text{Predicate} :== (\text{String} \to \text{Bool})$$

Block

What is a block

$$:: \text{Block} = \{\text{data} :: \text{String}, \text{nonce} :: \text{Int}\}$$
$$:: \text{Predicate} :== (\text{String} \to \text{Bool})$$

Mining of blocks

- Hash the data appended with the nonce
- Hash predicate
- Leading zeros
- Bitcoin has 18 leading zeros
- Finding the nonce leading to a valid hash

Blockchain

What is a blockchain

```
:: Block = { data      :: String
           , nonce     :: Int
           , prevHash  :: String
           }
:: BlockChain :== [Block]
```

Blockchain

What is a blockchain

```
:: Block = { data      :: String
           , nonce     :: Int
           , prevHash  :: String
           }
:: BlockChain :== [Block]
```

Mining of the blockchain

- Hash the data appended with the nonce and the previous hash
- Block is dependant on the previous block
- Valid only if all hashes match adhere the predicate

Mining a block in FP

Mining a block in FP

```
mine :: HashFun Predicate Block [Int] → [Int]
mine hash pred b nonces
  = filter (λn→pred (hash {b & nonce=n})) nonces
```

Mining the blockchain

```
mineChain :: HashFun Predicate Int String [String] → BlockChain
mineChain hash pred seed prev [] = []
mineChain hash pred seed prev [s:ss]
  # b           = {nonce=0, prev=prev, data=s}
  # b & nonce = hd $ mine hash pred b $ genRandInt seed
  = [b:mineChain hash pred seed (hash b) ss]
```

What was iTasks again?

Task Oriented Programming (TOP)

iTasks

- Tasks are basic blocks
- Combine with combinators
- Generated multi-user web interface

What was iTasks again?

Task Oriented Programming (TOP)

iTasks

- Tasks are basic blocks
- Combine with combinators
- Generated multi-user web interface

Task

- Statefull function
- Observable value

What was iTasks again?

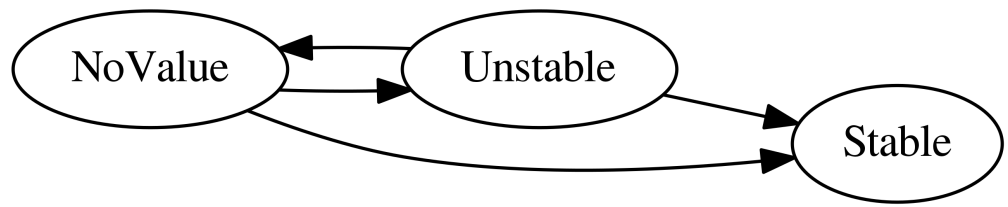
Task Oriented Programming (TOP)

iTasks

- Tasks are basic blocks
- Combine with combinators
- Generated multi-user web interface

Task

- Statefull function
- Observable value



Shared Data Sources

SDS

- JSON file on disk
- iTasks resources
- Hardware access
- Lenses and combinators
- Lean notifications via publish/subscribe

How to store the blockchain

```
:: SDS p r w = ...
:: Shared a :== SDS () a a
:: RWShared
sharedStore :: String a → Shared a
mapRead     :: (r → r`) (SDS p r w) → SDS p r` w
toReadOnly  ::          (SDS p r w) → SDS p r ()
```

How to store the blockchain

```
:: SDS p r w = ...
:: Shared a :== SDS () a a
:: RWShared
sharedStore :: String a → Shared a
mapRead     :: (r → r`) (SDS p r w) → SDS p r` w
toReadOnly  ::          (SDS p r w) → SDS p r ()

blockchain :: Shared BlockChain
blockchain = sharedStore "Blockchain" []

newblock :: ReadOnlyShared Block
newblock = toReadOnly (mapRead read blockchain)
where
  read x = {nonce=0, prev=last [""":map hash x], data=""}
```

Main Task

$$
\begin{aligned}
(\dashv\!|\,) &:: (\text{Task a})\ (\text{Task b}) \to \text{Task a} \\
(\,|\!\vdash) &:: (\text{Task a})\ (\text{Task b}) \to \text{Task b} \\
(\dashv\!\vdash) &:: (\text{Task a})\ (\text{Task a}) \to \text{Task a}
\end{aligned}
$$

Main Task

```
(-||)  :: (Task a) (Task b) → Task a
(||-)  :: (Task a) (Task b) → Task b
(-||-) :: (Task a) (Task a) → Task a

Start w = startEngine bc w

bc :: Task BlockChain
bc = viewSharedInformation () [chainv] blockchain
   -|| whileUnchanged newblock (forever o addBlock)
```

Adding a block

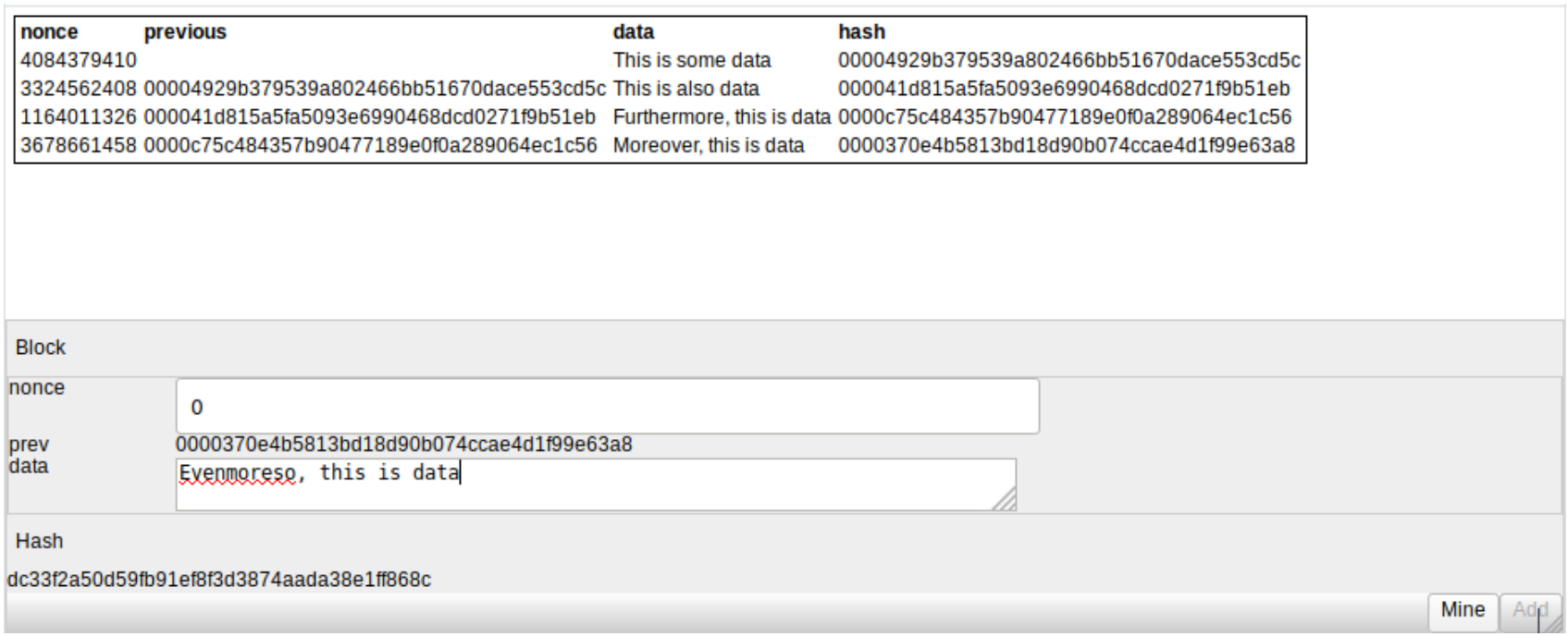
```
addBlock :: Block → Task ()
addBlock b = updateInformation "Block" [blockv] b
    >&^ viewSharedInformation "Hash" [] o mapRead (fmap hash)
    >>* [ OnAction (Action "Mine") $ hasValue mineBlock
        , OnAction (Action "Add") $ ifValue (pred o hash) addToChain
        ]
where
  addToChain b = upd (λc→c ++ [b]) blockchain @! ()

  mineBlock bl = get randomInt
    >>= compute "Mining" o hd o mine bl o genRandInt
    >>=λn→addBlock {bl & nonce=n}

  mine :: Block [Int] → [Int]
```

How does it look

nonce	previous	data	hash
4084379410		This is some data	00004929b379539a802466bb51670dace553cd5c
3324562408	00004929b379539a802466bb51670dace553cd5c	This is also data	000041d815a5fa5093e6990468dcd0271f9b51eb
1164011326	000041d815a5fa5093e6990468dcd0271f9b51eb	Furthermore, this is data	0000c75c484357b90477189e0f0a289064ec1c56
3678661458	0000c75c484357b90477189e0f0a289064ec1c56	Moreover, this is data	0000370e4b5813bd18d90b074ccae4d1f99e63a8



Properties

- Multiuser
- No useless mining
- Example less then 100 LOC
- One source

Conclusion

- Mining on server

Conclusion

- Mining on server, solve with editlets

Conclusion

- Mining on server, solve with editlets
- One blockchain copy

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things:

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things: Distribution,

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things: Distribution, Interfaces,

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things: Distribution, Interfaces, Validation,

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things: Distribution, Interfaces, Validation, Notifications

Conclusion

- Mining on server, solve with editlets
- One blockchain copy, solve with distributed iTasks (also solves previous)
- Difficult things: Distribution, Interfaces, Validation, Notifications
- Free in TOP