

Crotalinae

Metaprogramming for Ethereum Smart Contracts
expressed in Scala's Type System



Wonderful Benefits

- Unlock Smart Contracts **adoption** with a **JVM-based** language:
code them *from* Scala or *in* Scala
- Smart Contracts code generator is itself a **strictly typed program**
- Write a Contract using structs and definitions of **Crotalinae DSL**,
and if it compiles, you're **safe**
- **Export** Smart Contract in *Vyper* as a single plaintext and check it
visually if needed
- *(WIP)* Code directly in Scala: **Scala source code** is translated to
Crotalinae DSL using macros



Functional Things

- Heterogenous lists from Shapeless
- Free Monad from Cats
- Natural transformation from DSL to Writer Monad
- DSL constraints are validated on type level



Demonstration

```
val f = `@public` @:  
  sumArgs.funcDef( name = "sumSome", uint256) { args =>  
    for {  
      c <- 'c' ::= `++`(args.ref('a'), args.ref('b'))  
      d <- 'd' ::= `++`(args.ref('b'), c)  
      _ <- d ::= c  
      sum <- `++`(args.ref('a'), d).toReturn  
    } yield sum  
  }  
  
println(f.toVyper)
```



```
@public  
def sumSome(a: uint256, b: uint256) -> uint256:  
  c = a + b  
  d = b + c  
  d = c  
  return a + d
```


Auction Demo



```
object Auction extends App {
  import Expr.Defs._

  val data = ProductType.self(
    ('beneficiary ->> public(address)) ::
    ('auction_start ->> public(timestamp)) ::
    ('auction_end ->> public(timestamp)) ::
    ('highest_bidder ->> `public`(address)) ::
    ('highest_bid ->> `public`(wei_value)) ::
    ('ended ->> public(bool)) :: HNil
  )

  val beneficiary = data.ref('beneficiary)
  val auction_start = data.ref('auction_start)
  val auction_end = data.ref('auction_end)
  val highest_bid = data.ref('highest_bid)
  val highest_bidder = data.ref('highest_bidder)
  val ended = data.ref('ended)

  val initArgs = ProductType(('beneficiary ->> address) :: ('bidding_time ->> timedelta) :: HNil)

  val _beneficiary = initArgs.ref('_beneficiary)
  val _bidding_time = initArgs.ref('_bidding_time)

  val init = `@public` @: initArgs.funcDef(
    name = "__init__",
    Void
  ) { args =>
    for {
      _ <- beneficiary := _beneficiary
      _ <- auction_start := `block.timestamp`
      _ <- auction_end := `+` (auction_start, _bidding_time)
    } yield Void
  }

  val bid = `@public` @: `@payable` @: ProductType.HNil.funcDef(
    name = "bid",
    Void
  ) { args =>
    for {
      _ <- `assert`(`<`(`block.timestamp`, auction_end))
      _ <- `assert`(`>`(`msg.value`, highest_bid))
      _ <- `if`(`not`(`==` (highest_bid, `msg.value`)), {
        () =>
          send(highest_bidder :: highest_bid :: HNil).liftF.map(_ => Void)
      })
      _ <- highest_bidder := `msg.sender`
      _ <- highest_bid := `msg.value`
    } yield Void
  }

  val end_auction = `@public` @: ProductType.HNil.funcDef(
    name = "end_auction",
    Void
  ) { args =>
    for {
      _ <- `assert`(`>=` (block.timestamp, auction_end))
      _ <- `assert`(`not` ended)
      _ <- ended := True
      _ <- send(beneficiary, highest_bid)
    } yield Void
  }
}
```

Source Code

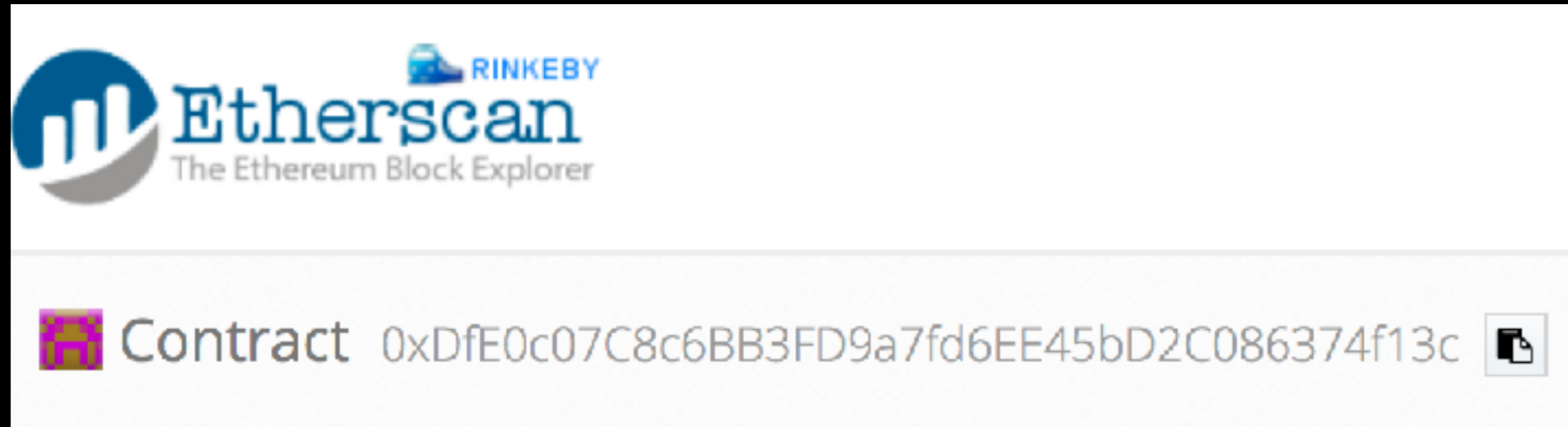
Bytecode ✓

ABI ✓

LLL ✓

```
1 beneficiary: public(address)
2 auction_start: public(timestamp)
3 auction_end: public(timestamp)
4 highest_bidder: public(address)
5 highest_bid: public(wei_value)
6 ended: public(bool)
7
8 @public
9 def __init__(_beneficiary: address, _bidding_time: timedelta):
10     self.beneficiary = _beneficiary
11     self.auction_start = block.timestamp
12     self.auction_end = self.auction_start + _bidding_time
13
14
15 @payable
16 @public
17 def bid():
18     assert block.timestamp < self.auction_end
19     assert msg.value > self.highest_bid
20     if not self.highest_bid == msg.value:
21         send(self.highest_bidder, self.highest_bid)
22
23     self.highest_bidder = msg.sender
24     self.highest_bid = msg.value
25
26
27 @public
28 def end_auction():
29     assert block.timestamp >= self.auction_end
30     assert not self.ended
31     self.ended = True
32     send(self.beneficiary, self.highest_bid)
```

Outside My Computer



- Deployed on Rinkeby Testnet!

For Tech Dive



Visit github.com/fluencelabs/hackethberlin

