A modern language for smart contracts

Carnegie Mellon University

Nick Roberts – Undergraduate Researcher
Jan Hoffmann – Faculty Advisor

Goals and contribution

The blockchain has recently emerged as a technology for distributed consensus; its applications include cryptocurrencies and smart contracts. Our goals are to:

- make **high-level language features** available to blockchain programmers.
- provide **high-performance implementations** of these features.

Our main technical contribution is **retargeting the OCaml compiler to output EVM bytecode.**

Blockchain lingo

Blockchain	Distributed, cryptographically-secured, consensus-based platform used for storing lists of records.
Smart contract	Programs stored on the blockchain that express state transitions and respond to calls from external actors.
Ethereum	Popular blockchain-based smart contract platform.
EVM	(Ethereum Virtual Machine.) Stack-based execution model of Ethereum bytecode.
Ether	Cryptocurrency exchanged in Ethereum transactions.
Gas cost	Cost of executing an EVM smart contract, specified at the bytecode level and paid in Ether.
Solidity	Popular object-oriented language targeting the EVM.

Comparative evaluation of lottery contract

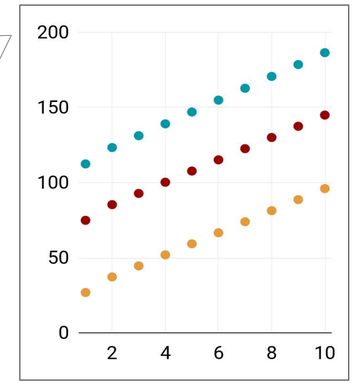
```
(* lottery.ml *)

  (* Module with bindings stored in persistent memory *)
  module S = Setup_lottery
  module P = Primitives

  (* Register a bet for some value of sent Ether *)
  let bet () : unit =
    if !S.num_users >= S.max_users || P.callvalue = 0
      then P.revert ()
      else begin
        S.total += P.callvalue;
        S.bets.(num) ← P.callvalue;
        S.users.(num) ← P.caller;
        S.num_users += 1;
      end

  (* Choose a random bettor to whom to send the contract's
   * total Ether, weighting by the amount bet. *)
  let end_lottery () : unit =
    if !S.num_users = 0 then () else
      let win = P.blockhash (P.number - 1) % !S.total in
      let rec loop (i : int256) (sum : int256) =
        let sum' = sum + S.bets.(i) in
        if sum' > win then P.selfdestruct S.users.(i)
        else loop (i + 1) sum'
      in loop 0 0
```

Gas cost (micro-Ether) **vs. number of participants** for the **lottery contract**, a **lottery contract with frequent function calls**, and an equivalent **Solidity contract**



Runner	Participant
$0.63	$0.06
$0.90	$0.06
$0.20	$0.06

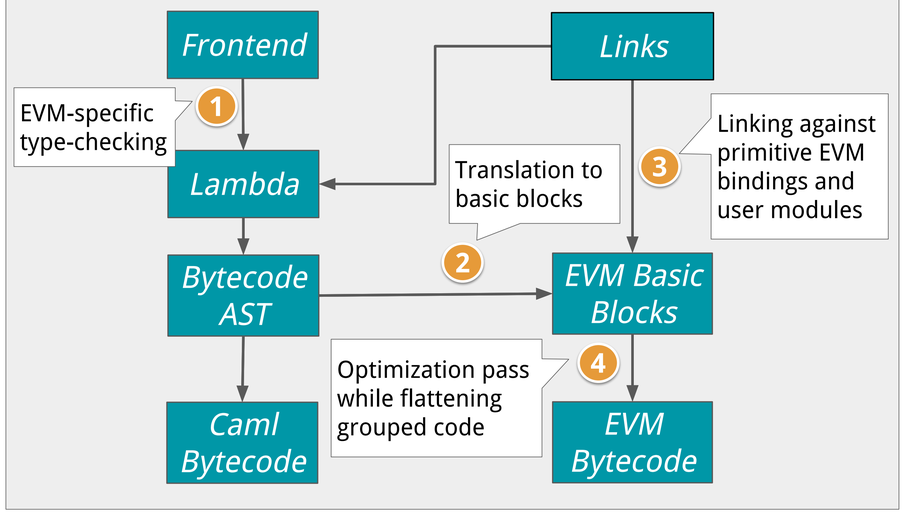
USD overhead (as of May 5, 2018) for the account that runs the lottery and for each participant in the lottery

Cost of application: Swapping of environments makes function calls expensive

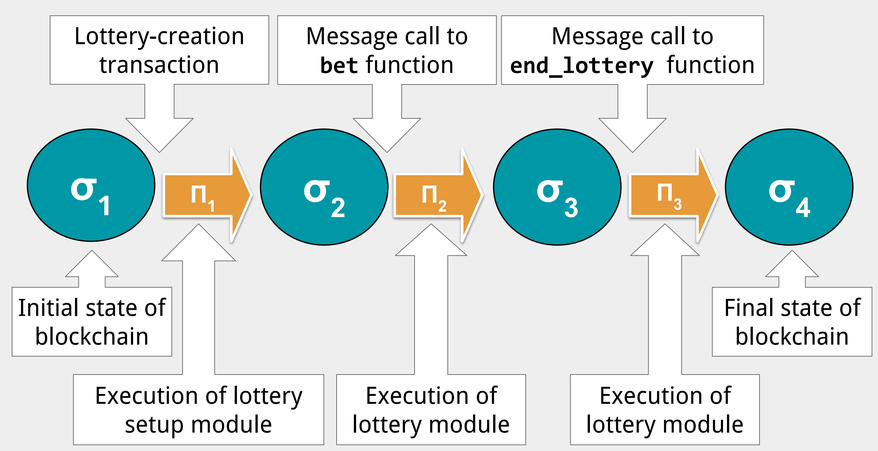
```
let f x =
  let g y = x * y
  in g 3
```

```
g: LOAD ARG 0      2 instrs
   LOAD ENV 0      5 instrs
   MUL             1 instr
   RETURN         31 instrs
f: PUSH 3          1 instr
   CLOSURE g      21 instrs
   APPLY          10 instrs
   RETURN         21 instrs
```

Description of compiler



Execution of code on blockchain



Conclusions

- High-level features are expensive on the blockchain, but this is owed to code size, not execution cost.
- Low-level translations are convenient where a direct correspondence exists, but you forfeit more complex transformations on the structure of code.

Future work

- Optimize code size to reduce contract-creation cost.
- Adapt **Resource Aware ML** to derive polynomial upper bounds on the gas cost of contracts.
- Support a greater subset of the OCaml language, such as the object system and the standard basis.