Formal specification of the Cardano ledger, mechanized in Agda

Andre Knispel, Orestis Melkonian, James Chapman, Alasdair Hill, Joosep Jääger, William DeMeo, Ulf Norell

21 March 2024, FM meeting @ IOG

Intro

‘ Some quotes are worth more than others. ’

—someone

Motivation

- Explore another point in the design space
- Provide a constructive perspective on nominal techniques
- Do this without changing the system itself — as an Agda library
- Make it ergonomic for the user to use the library as a tool for dealing with names (e.g. working on some syntax with binding)
- Mechanise existing (but also new?) meta-theoretical results

Agda Preliminaries

Separation of concerns

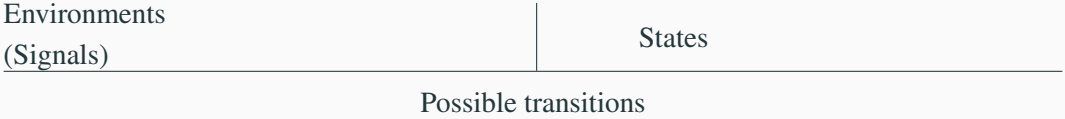
- Networking: deals with sending messages across the internet.
- Consensus: establishes a common order of valid blocks.
- Ledger: decides whether a sequence of blocks is valid.

State transitions

$$\Gamma \vdash s \xrightarrow[X]{b} s'$$

```
_⊢_-⟨_⟩_ : Env → State → Signal → State → Type
```

Triptychs



Reflexive-transitive closure

```
data _⊢_→⟨_⟩*_ : Env → State → List Signal → State → Type where

  base :
    ──────────────────────
    Γ ⊢ s →⟨ [] ⟩* s

  step :
    • Γ ⊢ s  →⟨ b  ⟩ s'
    • Γ ⊢ s' →⟨ bs ⟩* s''
    ──────────────────────
    Γ ⊢ s  →⟨ b :: bs ⟩* s''
```

Set theory

```
_⊆_ _≡ᵉ_ : {A : Type} → ℙ A → ℙ A → Type
X ⊆  Y = ∀ {x} → x ∈ X → x ∈ Y
X ≡ᵉ Y = X ⊆ Y × Y ⊆ X
```

```
_→_ : Type → Type → Type
A → B = ∃ λ (ℛ : ℙ (A × B)) →
  ∀ {a b b'} → (a , b) ∈ ℛ → (a , b') ∈ ℛ → b ≡ b
```

Formalization: basic entities

- crypto
- addresses
- tunable parameters

Formalization: the hierarchy of transitions

- CHAIN
-

Formalization: CHAIN block-by-block transition

-
-
-

Formalization: LEDGER transaction-by-transaction transition

-
-
-

Formalization: UTXO rule (the “core”)

-
-
-

Formalization: the transaction type

-
-
-

Compiling to Haskell

- computational proofs
- manually implementing coercions
- UTXOW example

Future Work

- Prove more interesting meta-theoretical properties
- Finalize conformance testing integration
  - Develop missing infrastructure
  - Randomly test difficult-to-prove properties by translating their Agda statements to Quickcheck properties
- Optimizations → refinements
- Smoother with less boilerplate
  - Transition from builtin GHC backend to agda2hs

Questions?

https://intersectmbo.github.io/formal-ledger-specifications/