UTxO: Barebones Setup

```
S = Map⟨ TxOutputRef ↦ TxOutput ⟩
```

```
record IsValidTx (tx : Tx) (utxos : S) : Type where
  field
    noDoubleSpending :
      ·Unique (outputRefs tx)

    validOutputRefs :
      ∀[ ref ∈ outputRefs tx ] (ref ∈ᵈ utxos)
    preservesValues :
      tx .forge + ∑ resolvedInputs (value ∘ proj₂) ≡ ∑ (tx .outputs) value

    allInputsValidate :
      ∀[ i ∈ tx .inputs ] T (i .validator txInfo (i .redeemer))

    validateValidHashes :
      ∀[ (i , o) ∈ resolvedInputs ] (o .address ≡ i .validator ♯)
```

UTxO: Denotational Semantics

```
instance
  ⟦T⟧ : Denotable Tx
  ⟦T⟧ .⟦_⟧ tx s = M.when (isValidTx tx s) (s — outputRefs tx ∪ utxoTx tx)

  ⟦L⟧ : Denotable L
  ⟦L⟧ .⟦_⟧ []       s = just s
  ⟦L⟧ .⟦_⟧ (t :: l) = ⟦ t ⟧ >⇒ ⟦ l ⟧
```

```
comp : ∀ x → ⟦ l ++ l' ⟧ x ≡ (⟦ l ⟧ >⇒ ⟦ l' ⟧) x
comp {[]}     _ = refl
comp {t :: l} x with ⟦ t ⟧ x
... | nothing = refl
... | just s  = comp {l} s
```

UTxO: Separation via Disjointness

```
_*_ : Op₂ Assertion
(P * Q) s = ∃ λ s₁ → ∃ λ s₂ → ⟨ s₁ ⊎ s₂ ⟩≡ s × P s₁ × Q s₂
```

⊎-⟦⟧ : ∀ s_1' →
- ⟦ l ⟧ s_1 ≡ just s_1'
- ⟨ s_1 ⊎ s_2 ⟩≡ s

(⟨ s_1' ⊎ s_2 ⟩≡_ ↑∘ ⟦ l ⟧) s

[FRAME] : ∀ R →
- l ♯ R
- ⟨ P ⟩ l ⟨ Q ⟩

⟨ P * R ⟩ l ⟨ Q * R ⟩

[PAR] :
- l_1 ♯ P_2
- l_2 ♯ P_1
- l_1 ∥ l_2 ≡ l
- ⟨ P_1 ⟩ l_1 ⟨ Q_1 ⟩
- ⟨ P_2 ⟩ l_2 ⟨ Q_2 ⟩

⟨ P_1 * P_2 ⟩ l ⟨ Q_1 * Q_2 ⟩