

# PROGRAM LOGICS FOR LEDGERS

---

Orestis Melkonian, Wouter Swierstra, James Chapman

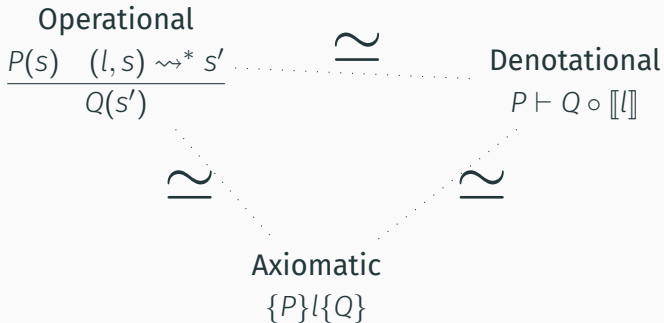
14 June 2023, TYPES @ Valencia

- Local & modular reasoning for UTxO blockchain ledgers
- Entertain the following analogy with concurrency/PL:

Blockchain		Concurrency Theory
ledgers	↔	computer memory
accounts	↔	memory locations
account balances	↔	data values
smart contracts	↔	programs accessing memory

# APPROACH

Investigate multiple semantics in different systems of increasing complexity



# SIMPLE MODEL

```
module ... (Part : Type) { _ : DecEq Part } where
```

```
S = Map< Part → ℤ >
```

```
record Tx : Type where  
  constructor _→⟨_⟩_  
  field sender    : Part  
        value     : ℤ  
        receiver  : Part
```

```
unquoteDecl DecEq-Tx = DERIVE DecEq [ quote Tx , DecEq-Tx ]
```

```
L = List Tx
```

# SIMPLE MODEL: DENOTATIONAL SEMANTICS

Domain =  $S \rightarrow S$

```
record Denotable (A : Type) : Type where
```

```
  field [ _ ] : A → Domain
```

```
instance
```

```
  [T] : Denotable Tx
```

```
  [T] . [ _ ] (A → ⟨ v ⟩ B) s = s [ A ~ _ - v ] [ B ~ _ + v ]
```

```
  [L] : Denotable L
```

```
  [L] . [ _ ] [ ] = id
```

```
  [L] . [ _ ] (t :: l) = [ l ] ∘ [ t ]
```

```
comp : ∀ x → [ l ++ l' ] x ≡ ([ l' ] ∘ [ l ]) x
```

```
comp { [ ] } _ = refl
```

```
comp { t :: l } x = comp { l } ([ t ] x)
```

# SIMPLE MODEL: OPERATIONAL SEMANTICS

data  $\_ \rightarrow \_ : L \times S \rightarrow S \rightarrow \text{Type}$  where

base :

$$\frac{}{\epsilon, s \rightarrow s}$$

step : let  $t = A \rightarrow \langle v \rangle B$  in

$$l, [t] s \rightarrow s'$$
$$\frac{}{t :: l, s \rightarrow s'}$$

denote~~oper~~ :

$$[l] s \equiv s'$$
$$\frac{}{l, s \rightarrow s'}$$

oper-comp :

- $l, s \rightarrow s'$
- $l', s' \rightarrow s''$

$$\frac{}{l ++ l', s \rightarrow s''}$$

# SIMPLE MODEL: AXIOMATIC SEMANTICS (HOARE LOGIC)

Assertion =  $\text{Pred}_0 S$

$\langle \_ \rangle \_ \langle \_ \rangle : \text{Assertion} \rightarrow L \rightarrow \text{Assertion} \rightarrow \text{Type}$

$\langle P \rangle l \langle Q \rangle = P \vdash Q \circ \llbracket l \rrbracket$

hoare-base :

---

$\langle P \rangle \llbracket \_ \rrbracket \langle P \rangle$

hoare-base = id

hoare-step :

$\langle P \rangle l \langle Q \rangle$

---

$\langle P \circ \llbracket t \rrbracket \rangle t :: l \langle Q \rangle$

hoare-step  $P l Q \{ \_ \} = P l Q$

# SIMPLE MODEL: AXIOMATIC SEMANTICS (HOARE LOGIC)

consequence :

- $P' \vdash P$
- $Q \vdash Q'$
- $\langle P \rangle l \langle Q \rangle$

---

$$\langle P' \rangle l \langle Q' \rangle$$

consequence  $\vdash P \ Q \vdash P l Q$

$= Q \vdash \circ P l Q \circ \vdash P$

hoare-step' :

- $\langle P \rangle l \langle Q \rangle$
- $\langle Q \rangle l' \langle R \rangle$

---

$$\langle P \rangle l ++ l' \langle R \rangle$$

hoare-step'  $\{P\}\{l\}\{Q\}\{l'\}\{R\} P l Q Q l R =$

begin  $P$   $\vdash \langle P l Q \rangle$

$Q \circ [ l ]$   $\vdash \langle Q l R \rangle$

$R \circ ([ l' ] \circ [ l ]) \doteq \langle \text{cong } R \circ \text{comp } \{l\} \{l'\} \rangle$

$R \circ [ l ++ l' ]$  ■ where open  $\vdash$ -Reasoning



## SIMPLE MODEL: SEPARATION LOGIC

$\text{emp} : \text{Assertion}$

$\text{emp } m = \forall k \rightarrow m \text{ k} \equiv \epsilon$

$\_ *_ \_ : \text{Op}_2 \text{ Assertion}$

$(P * Q) \text{ s} = \exists \lambda \text{ s}_1 \rightarrow \exists \lambda \text{ s}_2 \rightarrow \langle \text{s}_1 \diamond \text{s}_2 \rangle \equiv \text{s} \times P \text{ s}_1 \times Q \text{ s}_2$

$* \leftrightarrow : P * Q \vdash Q * P$

$* \leftrightarrow (\text{s}_1, \text{s}_2, \equiv \text{s}, P \text{ s}_1, Q \text{ s}_2) = \text{s}_2, \text{s}_1, \diamond \equiv\text{-comm} \{x = \text{s}_1\} \{\text{s}_2\} \equiv \text{s}, Q \text{ s}_2, P \text{ s}_1$

$* \leadsto : P * Q * R \vdash (P * Q) * R$

$* \leadsto \{x = \text{s}\} (\text{s}_1, \text{s}_2 \text{ s}_3, \equiv \text{s}, P \text{ s}_1, (\text{s}_2, \text{s}_3, \equiv \text{s}_2 \text{ s}_3, Q \text{ s}_2, R \text{ s}_3)) =$   
 $(\text{s}_1 \diamond \text{s}_2), \text{s}_3, \diamond \approx\text{-assoc}^r \{m_1 = \text{s}_1\} \equiv \text{s} \equiv \text{s}_2 \text{ s}_3, (\text{s}_1, \text{s}_2, \approx\text{-refl}, P \text{ s}_1, Q \text{ s}_2), R \text{ s}_3$

$\leftarrow * : (P * Q) * R \vdash P * Q * R$

$\leftarrow * \{x = \text{s}\} (\text{s}_1 \text{ s}_2, \text{s}_3, \equiv \text{s}, (\text{s}_1, \text{s}_2, \equiv \text{s}_1 \text{ s}_2, P \text{ s}_1, Q \text{ s}_2), R \text{ s}_3) =$   
 $\text{s}_1, \text{s}_2 \diamond \text{s}_3, \diamond \approx\text{-assoc}^l \{m_1 = \text{s}_1\} \{\text{s}_2\} \equiv \text{s} \equiv \text{s}_1 \text{ s}_2, P \text{ s}_1, (\text{s}_2, \text{s}_3, \approx\text{-refl}, Q \text{ s}_2, R \text{ s}_3)$

## SIMPLE MODEL: FRAME RULE

$\diamond - [] :$

$$\langle s_1 \diamond s_2 \rangle \equiv s$$

---

$$\langle [l] s_1 \diamond s_2 \rangle \equiv [l] s$$

[FRAME] :

$$\langle P \rangle l \langle Q \rangle$$

---

$$\langle P * R \rangle l \langle Q * R \rangle$$

$$[\text{FRAME}] \{l = l\} P l Q (s_1, s_2, \equiv s, Ps_1, Rs_2) =$$

$$[l] s_1, s_2, \diamond - [] \{l = l\} \equiv s, P l Q Ps_1, Rs_2$$

## SIMPLE MODEL: CONCURRENT SEPARATION LOGIC

$\diamond$ -interleave :

- $l_1 \parallel l_2 \equiv l$
- $\langle s_1 \diamond s_2 \rangle \equiv s$

---

$$\langle [l_1] s_1 \diamond [l_2] s_2 \rangle \equiv [l] s$$

[PAR] :

- $l_1 \parallel l_2 \equiv l$
- $\langle P_1 \rangle l_1 \langle Q_1 \rangle$
- $\langle P_2 \rangle l_2 \langle Q_2 \rangle$

---

$$\langle P_1 * P_2 \rangle l \langle Q_1 * Q_2 \rangle$$

$$[PAR] \{l_1\} \{l_2\} \{l\} \equiv l \text{ } P l_1 Q \text{ } P l_2 Q \{s\} (s_1, s_2, \equiv s, P s_1, P s_2) = \\ [l_1] s_1, [l_2] s_2, \diamond\text{-interleave} \equiv l \equiv s, P l_1 Q P s_1, P l_2 Q P s_2$$

# SIMPLE MODEL: EXAMPLE DERIVATION (MONOLITHIC)

A B C D : Part

$t_1 = A \rightarrow \langle 1Z \rangle B$ ;  $t_2 = D \rightarrow \langle 1Z \rangle C$ ;  $t_3 = B \rightarrow \langle 1Z \rangle A$ ;  $t_4 = C \rightarrow \langle 1Z \rangle D$

$t_{1-4} = L \ni [t_1, t_2, t_3, t_4]$

$\_ : \langle A \mapsto 1Z * B \mapsto 0Z * C \mapsto 0Z * D \mapsto 1Z \rangle t_{1-4} \langle A \mapsto 1Z * B \mapsto 0Z * C \mapsto 0Z * D \mapsto 1Z \rangle$

$\_ = \text{begin } A \mapsto 1Z * B \mapsto 0Z \quad * C \mapsto 0Z * D \mapsto 1Z \sim \langle \rightsquigarrow \rangle$

$(A \mapsto 1Z * B \mapsto 0Z) * C \mapsto 0Z * D \mapsto 1Z \sim \langle t_1 :- [\text{FRAME}] (C \mapsto 0Z * D \mapsto 1Z) (A \rightsquigarrow B) \rangle$

$(A \mapsto 0Z * B \mapsto 1Z) * C \mapsto 0Z * D \mapsto 1Z \sim \langle \rightsquigarrow \rangle$

$(C \mapsto 0Z * D \mapsto 1Z) * A \mapsto 0Z * B \mapsto 1Z \sim \langle t_2 :- [\text{FRAME}] (A \mapsto 0Z * B \mapsto 1Z) (C \leftarrow D) \rangle$

$(C \mapsto 1Z * D \mapsto 0Z) * A \mapsto 0Z * B \mapsto 1Z \sim \langle \rightsquigarrow \rangle$

$(A \mapsto 0Z * B \mapsto 1Z) * C \mapsto 1Z * D \mapsto 0Z \sim \langle t_3 :- [\text{FRAME}] (C \mapsto 1Z * D \mapsto 0Z) (A \leftarrow B) \rangle$

$(A \mapsto 1Z * B \mapsto 0Z) * C \mapsto 1Z * D \mapsto 0Z \sim \langle \rightsquigarrow \rangle$

$(C \mapsto 1Z * D \mapsto 0Z) * A \mapsto 1Z * B \mapsto 0Z \sim \langle t_4 :- [\text{FRAME}] (A \mapsto 1Z * B \mapsto 0Z) (C \rightsquigarrow D) \rangle$

$(C \mapsto 0Z * D \mapsto 1Z) * A \mapsto 1Z * B \mapsto 0Z \sim \langle \rightsquigarrow \rangle$

$(A \mapsto 1Z * B \mapsto 0Z) * C \mapsto 0Z * D \mapsto 1Z \sim \langle \rightsquigarrow \rangle$

$A \mapsto 1Z * B \mapsto 0Z \quad * C \mapsto 0Z * D \mapsto 1Z \blacksquare$

## SIMPLE MODEL: EXAMPLE DERIVATION (MODULAR)

```
_ : < A ↦ 1Z * B ↦ 0Z * C ↦ 0Z * D ↦ 1Z > t1-4 < A ↦ 1Z * B ↦ 0Z * C ↦ 0Z * D ↦ 1Z >  
_ = begin A ↦ 1Z * B ↦ 0Z    * C ↦ 0Z * D ↦ 1Z ~⟨ *~ >  
      (A ↦ 1Z * B ↦ 0Z) * C ↦ 0Z * D ↦ 1Z ~⟨ t1-4 :- [PAR] auto H1 H2 > ++  
      (A ↦ 1Z * B ↦ 0Z) * C ↦ 0Z * D ↦ 1Z ~⟨ *~ >  
      A ↦ 1Z * B ↦ 0Z    * C ↦ 0Z * D ↦ 1Z ■
```

where

$H_1 : \mathbb{R} \langle A \mapsto 1Z * B \mapsto 0Z \rangle t_1 :: t_3 :: [] \langle A \mapsto 1Z * B \mapsto 0Z \rangle$

$H_1 = A \mapsto 1Z * B \mapsto 0Z \sim \langle t_1 :- A \rightsquigarrow B \rangle$

$A \mapsto 0Z * B \mapsto 1Z \sim \langle t_3 :- A \leftarrow B \rangle$

$A \mapsto 1Z * B \mapsto 0Z \blacksquare$

$H_2 : \mathbb{R} \langle C \mapsto 0Z * D \mapsto 1Z \rangle t_2 :: t_4 :: [] \langle C \mapsto 0Z * D \mapsto 1Z \rangle$

$H_2 = C \mapsto 0Z * D \mapsto 1Z \sim \langle t_2 :- C \leftarrow D \rangle$

$C \mapsto 1Z * D \mapsto 0Z \sim \langle t_4 :- C \rightsquigarrow D \rangle$

$C \mapsto 0Z * D \mapsto 1Z \blacksquare$

## ADDING PARTIALITY

$S = \text{Map} \langle \text{Part} \mapsto \mathbb{N} \rangle$

$\text{Domain} = S \rightarrow \text{Maybe } S$

$\llbracket T \rrbracket : \text{Denotable Tx}$

$\llbracket T \rrbracket . \llbracket - \rrbracket t s = \text{M.when } (\text{isValidTx } t s) (\llbracket t \rrbracket \circ s)$

$\llbracket L \rrbracket : \text{Denotable L}$

$\llbracket L \rrbracket . \llbracket - \rrbracket \llbracket \rrbracket s = \text{just } s$

$\llbracket L \rrbracket . \llbracket - \rrbracket (t :: l) = \llbracket t \rrbracket \Rightarrow \llbracket l \rrbracket$

$\text{comp} : \forall x \rightarrow \llbracket l ++ l' \rrbracket x \equiv (\llbracket l \rrbracket \Rightarrow \llbracket l' \rrbracket) x$

# ADDING PARTIALITY: OPERATIONAL SEMANTICS

data  $\_ \rightarrow \_ : L \times S \rightarrow S \rightarrow \text{Type}$  where

base :

---

$\epsilon, s \rightarrow s$

step :

- $\text{IsValidTx } t \ s$

- $l, \llbracket t \rrbracket_0 s \rightarrow s'$

---

$t :: l, s \rightarrow s'$

denote~~oper~~ :

$\llbracket l \rrbracket s \equiv \text{just } s'$

---

---

$l, s \rightarrow s'$

## ADDING PARTIALITY: LIFTING PREDICATES FOR HOARE LOGIC

$\text{weak}\uparrow \text{strong}\uparrow : \text{Pred}_0 S \rightarrow \text{Pred}_0 (\text{Maybe } S)$

$\text{weak}\uparrow = \text{M.All.All}$

$\text{strong}\uparrow = \text{M.Any.Any}$

$\_ \uparrow \circ \_ : \text{Pred}_0 S \rightarrow (S \rightarrow \text{Maybe } S) \rightarrow \text{Pred}_0 S$

$P \uparrow \circ f = \text{strong}\uparrow P \circ f$

$\langle \_ \rangle \_ \langle \_ \rangle : \text{Assertion} \rightarrow L \rightarrow \text{Assertion} \rightarrow \text{Type}$

$\langle P \rangle l \langle Q \rangle = P \vdash Q \uparrow \circ [ l ]$



## ADDING PARTIALITY: FRAME RULE

$\diamond - [\ ] : \forall s_1' \rightarrow$

- $[\ ] s_1 \equiv \text{just } s_1'$
- $\langle s_1 \diamond s_2 \rangle \equiv s$

---

$(\langle s_1' \diamond s_2 \rangle \equiv \_ \uparrow \circ [\ ]) s$

$[\text{FRAME}] : \forall R \rightarrow$

$\langle P \rangle \text{ l } \langle Q \rangle$

---

$\langle P * R \rangle \text{ l } \langle Q * R \rangle$

## ADDING PARTIALITY: PARALLEL RULE

$\diamond$ -interleave :

- $(l_1 \parallel l_2 \equiv l)$
- $\langle s_1 \diamond s_2 \rangle \equiv s$
- $\llbracket l_1 \rrbracket s_1 \equiv \text{just } s_1'$
- $\llbracket l_2 \rrbracket s_2 \equiv \text{just } s_2'$

---

$$\begin{aligned} \exists \lambda s' \rightarrow & (\llbracket l \rrbracket s \equiv \text{just } s') \\ & \times (\langle s_1' \diamond s_2' \rangle \equiv s') \end{aligned}$$

[PAR] :

- $l_1 \parallel l_2 \equiv l$
- $\langle P_1 \rangle l_1 \langle Q_1 \rangle$
- $\langle P_2 \rangle l_2 \langle Q_2 \rangle$

---

$$\langle P_1 * P_2 \rangle l \langle Q_1 * Q_2 \rangle$$

## ADDING PARTIALITY: EXAMPLE DERIVATION (MONOLITHIC)

A B C D : Part

$t_1 = A \rightarrow \langle 1 \rangle B$ ;  $t_2 = D \rightarrow \langle 1 \rangle C$ ;  $t_3 = B \rightarrow \langle 1 \rangle A$ ;  $t_4 = C \rightarrow \langle 1 \rangle D$

$t_{1-4} = L \ni [t_1, t_2, t_3, t_4]$

$\_ : \langle A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \rangle t_{1-4} \langle A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \rangle$

$\_ = \text{begin } A \mapsto 1 * B \mapsto 0 \quad * C \mapsto 0 * D \mapsto 1 \sim \langle \langle * \rangle \rangle$

$(A \mapsto 1 * B \mapsto 0) * C \mapsto 0 * D \mapsto 1 \sim \langle t_1 :- [\text{FRAME}] (C \mapsto 0 * D \mapsto 1) (A \leadsto B) \rangle$

$(A \mapsto 0 * B \mapsto 1) * C \mapsto 0 * D \mapsto 1 \sim \langle \langle * \rangle \rangle$

$(C \mapsto 0 * D \mapsto 1) * A \mapsto 0 * B \mapsto 1 \sim \langle t_2 :- [\text{FRAME}] (A \mapsto 0 * B \mapsto 1) (C \leadsto D) \rangle$

$(C \mapsto 1 * D \mapsto 0) * A \mapsto 0 * B \mapsto 1 \sim \langle \langle * \rangle \rangle$

$(A \mapsto 0 * B \mapsto 1) * C \mapsto 1 * D \mapsto 0 \sim \langle t_3 :- [\text{FRAME}] (C \mapsto 1 * D \mapsto 0) (A \leadsto B) \rangle$

$(A \mapsto 1 * B \mapsto 0) * C \mapsto 1 * D \mapsto 0 \sim \langle \langle * \rangle \rangle$

$(C \mapsto 1 * D \mapsto 0) * A \mapsto 1 * B \mapsto 0 \sim \langle t_4 :- [\text{FRAME}] (A \mapsto 1 * B \mapsto 0) (C \leadsto D) \rangle$

$(C \mapsto 0 * D \mapsto 1) * A \mapsto 1 * B \mapsto 0 \sim \langle \langle * \rangle \rangle$

$(A \mapsto 1 * B \mapsto 0) * C \mapsto 0 * D \mapsto 1 \sim \langle \langle * \rangle \rangle$

$A \mapsto 1 * B \mapsto 0 \quad * C \mapsto 0 * D \mapsto 1 \blacksquare$

## ADDING PARTIALITY: EXAMPLE DERIVATION (MODULAR)

```
_ : < A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 > t1-4 < A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 >  
_ = begin A ↦ 1 * B ↦ 0      * C ↦ 0 * D ↦ 1 ~⟨ * ↦ >  
      (A ↦ 1 * B ↦ 0) * C ↦ 0 * D ↦ 1 ~⟨ t1-4 :- [PAR] auto H1 H2 > ++  
      (A ↦ 1 * B ↦ 0) * C ↦ 0 * D ↦ 1 ~⟨ ↦ * >  
      A ↦ 1 * B ↦ 0      * C ↦ 0 * D ↦ 1 ■
```

where

```
H1 : < A ↦ 1 * B ↦ 0 > t1 :: t3 :: [] < A ↦ 1 * B ↦ 0 >  
H1 = begin A ↦ 1 * B ↦ 0 ~⟨ t1 :- A ↦ B >  
      A ↦ 0 * B ↦ 1 ~⟨ t3 :- A ↦ B >  
      A ↦ 1 * B ↦ 0 ■  
H2 : < C ↦ 0 * D ↦ 1 > t2 :: t4 :: [] < C ↦ 0 * D ↦ 1 >  
H2 = begin C ↦ 0 * D ↦ 1 ~⟨ t2 :- C ↦ D >  
      C ↦ 1 * D ↦ 0 ~⟨ t4 :- C ↦ D >  
      C ↦ 0 * D ↦ 1 ■
```

## UTxO: BAREBONES SETUP

$S = \text{Map}\langle \text{TxOutputRef} \mapsto \text{TxOutput} \rangle$

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

    validOutputRefs :
       $\forall [ \text{ref} \in \text{outputRefs tx} ] (\text{ref} \in^d \text{utxos})$ 

    preservesValues :
       $\text{tx} . \text{forge} + \sum \text{resolvedInputs} (\text{value} \circ \text{proj}_2) \equiv \sum (\text{tx} . \text{outputs}) \text{value}$ 

    allInputsValidate :
       $\forall [ i \in \text{tx} . \text{inputs} ] T (i . \text{validator txInfo } (i . \text{redeemer}))$ 

    validateValidHashes :
       $\forall [ (i, o) \in \text{resolvedInputs} ] (o . \text{address} \equiv i . \text{validator } \#)$ 
```

## UTxO: DENOTATIONAL SEMANTICS

instance

$\llbracket T \rrbracket : \text{Denotable Tx}$

$\llbracket T \rrbracket . \llbracket - \rrbracket \text{ tx } s = \text{M.when } (\text{isValidTx tx } s) (s - \text{outputRefs tx } \cup \text{utxoTx tx})$

$\llbracket L \rrbracket : \text{Denotable L}$

$\llbracket L \rrbracket . \llbracket - \rrbracket [] \quad s = \text{just } s$

$\llbracket L \rrbracket . \llbracket - \rrbracket (t :: l) = \llbracket t \rrbracket \Rightarrow \llbracket l \rrbracket$

$\text{comp} : \forall x \rightarrow \llbracket l ++ l' \rrbracket x \equiv (\llbracket l \rrbracket \Rightarrow \llbracket l' \rrbracket) x$

$\text{comp } \{[]\} \quad \_ = \text{refl}$

$\text{comp } \{t :: l\} x \text{ with } \llbracket t \rrbracket x$

$\dots \mid \text{nothing} = \text{refl}$

$\dots \mid \text{just } s = \text{comp } \{l\} s$

# UTxO: SEPARATION VIA DISJOINTNESS

$\_ *_\_ : \text{Op}_2$  Assertion

$$(P *_ Q) s = \exists \lambda s_1 \rightarrow \exists \lambda s_2 \rightarrow \langle s_1 \uplus s_2 \rangle \equiv s \times P s_1 \times Q s_2$$

$\uplus - [\ ] : \forall s_1' \rightarrow$

- $[ \textcolor{teal}{l} ] s_1 \equiv \textcolor{teal}{just} s_1'$
- $\langle s_1 \uplus s_2 \rangle \equiv s$

---

$$(\langle s_1' \uplus s_2 \rangle \equiv\_ \uparrow \circ [ \textcolor{teal}{l} ] ) s$$

$\textcolor{blue}{[FRAME]} : \forall R \rightarrow$

- $\textcolor{teal}{l} \# R$
- $\langle P \rangle \textcolor{teal}{l} \langle Q \rangle$

---

$$\langle P *_ R \rangle \textcolor{teal}{l} \langle Q *_ R \rangle$$

$\textcolor{blue}{[PAR]} :$

- $\textcolor{teal}{l}_1 \# P_2$
- $\textcolor{teal}{l}_2 \# P_1$
- $\textcolor{teal}{l}_1 \parallel \textcolor{teal}{l}_2 \equiv \textcolor{teal}{l}$
- $\langle P_1 \rangle \textcolor{teal}{l}_1 \langle Q_1 \rangle$
- $\langle P_2 \rangle \textcolor{teal}{l}_2 \langle Q_2 \rangle$

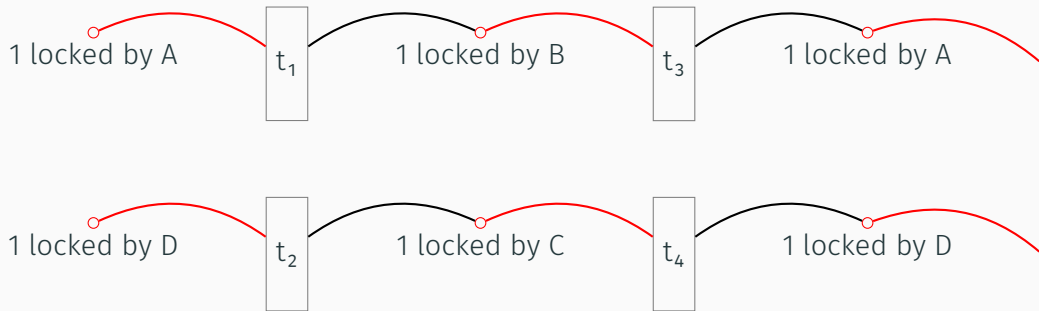
---

$$\langle P_1 *_ P_2 \rangle \textcolor{teal}{l} \langle Q_1 *_ Q_2 \rangle$$

# UTxO: EXAMPLE TRANSACTION GRAPH

A B C D : Address

$t_{1-4} = L \ni [ t_1, t_2, t_3, t_4 ]$





## UTxO: EXAMPLE DERIVATION (MONOLITHIC)

```
_ : < t00 ↦ 1 at A * t01 ↦ 1 at D > t1-4 < t30 ↦ 1 at A * t40 ↦ 1 at D >  
_ = begin t00 ↦ 1 at A * t01 ↦ 1 at D ~ < t1 :- [FRAME] (t01 ↦ 1 at D) t1 # ... >  
      t10 ↦ 1 at B * t01 ↦ 1 at D ~ << *↔ >  
      t01 ↦ 1 at D * t10 ↦ 1 at B ~ < t2 :- [FRAME] (t10 ↦ 1 at B) t2 # ... >  
      t20 ↦ 1 at C * t10 ↦ 1 at B ~ << *↔ >  
      t10 ↦ 1 at B * t20 ↦ 1 at C ~ < t3 :- [FRAME] (t20 ↦ 1 at C) t3 # ... >  
      t30 ↦ 1 at A * t20 ↦ 1 at C ~ << *↔ >  
      t20 ↦ 1 at C * t30 ↦ 1 at A ~ < t4 :- [FRAME] (t30 ↦ 1 at A) t4 # ... >  
      t40 ↦ 1 at D * t30 ↦ 1 at A ~ << *↔ >  
      t30 ↦ 1 at A * t40 ↦ 1 at D ■
```

```
where postulate t1 # : [ t1 ] # (t01 ↦ 1 at D)  
               t2 # : [ t2 ] # (t10 ↦ 1 at B)  
               t3 # : [ t3 ] # (t20 ↦ 1 at C)  
               t4 # : [ t4 ] # (t30 ↦ 1 at A)
```

## UTxO: EXAMPLE DERIVATION (MODULAR)

```
_ : < t00 ↦ 1 at A * t01 ↦ 1 at D > t1-4 < t30 ↦ 1 at A * t40 ↦ 1 at D >  
_ = begin t00 ↦ 1 at A * t01 ↦ 1 at D ~< t1-4 :- [PAR] ... auto H1 H2 > ++  
      t30 ↦ 1 at A * t40 ↦ 1 at D ■
```

where

```
H1 : < t00 ↦ 1 at A > t1 :: t3 :: [] < t30 ↦ 1 at A >
```

```
H1 = begin t00 ↦ 1 at A ~< t1 :- ... >
```

```
      t10 ↦ 1 at B ~< t3 :- ... >
```

```
      t30 ↦ 1 at A ■
```

```
H2 : < t01 ↦ 1 at D > t2 :: t4 :: [] < t40 ↦ 1 at D >
```

```
H2 = begin t01 ↦ 1 at D ~< t2 :- ... >
```

```
      t20 ↦ 1 at C ~< t4 :- ... >
```

```
      t40 ↦ 1 at D ■
```

## ABSTRACT UTxO: SETUP

$S = \text{Bag}\langle \text{TxOutput} \rangle$

```
record IsValidTx (tx : Tx) (utxos : S) : Type where
  field
    validOutputRefs :
      stxoTx tx  $\subseteq^s$  utxos

    preservesValues :
      tx .forge +  $\sum$  (tx .inputs) (value  $\circ$  outputRef)  $\equiv$   $\sum$  (tx .outputs) value

    allInputsValidate :
       $\forall [i \in \text{tx} . \text{inputs}] T (i . \text{validator } \text{txInfo } (i . \text{redeemer}))$ 

    validateValidHashes :
       $\forall [i \in \text{tx} . \text{inputs}] (i . \text{outputRef} . \text{address} \equiv i . \text{validator } \#)$ 
```

# ABSTRACT UTxO: DENOTATIONAL SEMANTICS

instance

$\llbracket T \rrbracket : \text{Denotable Tx}$

$\llbracket T \rrbracket . \llbracket - \rrbracket tx\ s = \text{M.when } (\text{isValidTx } tx\ s) (s - \text{stxoTx } tx \cup \text{utxoTx } tx)$

$\llbracket L \rrbracket : \text{Denotable L}$

$\llbracket L \rrbracket . \llbracket - \rrbracket [] \quad s = \text{just } s$

$\llbracket L \rrbracket . \llbracket - \rrbracket (t :: l) = \llbracket t \rrbracket \Rightarrow \llbracket l \rrbracket$

# ABSTRACT UTxO: MONOIDAL SEPARATION ONCE AGAIN

$\_ *_\_ : \text{Op}_2 \text{ Assertion}$

$$(P * Q) \ s = \exists \lambda \ s_1 \rightarrow \exists \lambda \ s_2 \rightarrow \langle s_1 \diamond s_2 \rangle \equiv s \times P \ s_1 \times Q \ s_2$$

$* \leftrightarrow : P * Q \vdash Q * P$

$$* \leftrightarrow \{x = s\} (s_1, s_2, \equiv s, P s_1, Q s_2) = s_2, s_1, \diamond \equiv \text{-comm} \{s = s\} \{s_1\} \{s_2\} \equiv s, Q s_2, P s_1$$

$* \leadsto : P * Q * R \vdash (P * Q) * R$

$$\begin{aligned} * \leadsto \{x = s\} (s_1, s_{23}, \equiv s, P s_1, (s_2, s_3, \equiv s_{23}, Q s_2, R s_3)) = \\ \text{let } \equiv s_{12} = \diamond \approx \text{-assoc}^r \{s_1 = s_1\} \{s_{23}\} \{s\} \{s_2\} \{s_3\} \equiv s \equiv s_{23} \text{ in} \\ (s_1 \diamond s_2), s_3, \equiv s_{12}, (s_1, s_2, \approx \text{-refl} \{x = s_1 \cup s_2\}, P s_1, Q s_2), R s_3 \end{aligned}$$

$\leftarrow * : (P * Q) * R \vdash P * Q * R$

$$\begin{aligned} \leftarrow * \{x = s\} (s_{12}, s_3, \equiv s, (s_1, s_2, \equiv s_{12}, P s_1, Q s_2), R s_3) = \\ \text{let } \equiv s_{23} = \diamond \approx \text{-assoc}^l \{s_{12} = s_{12}\} \{s_3\} \{s\} \{s_1\} \{s_2\} \equiv s \equiv s_{12} \text{ in} \\ s_1, s_2 \diamond s_3, \equiv s_{23}, P s_1, (s_2, s_3, \approx \text{-refl} \{x = s_2 \cup s_3\}, Q s_2, R s_3) \end{aligned}$$

# ABSTRACT UTXO: SEPARATION LOGIC RULES

[FRAME] :  $\forall R \rightarrow$

$\langle P \rangle \mathbin{\text{!}} \langle Q \rangle$

---

$\langle P * R \rangle \mathbin{\text{!}} \langle Q * R \rangle$

[PAR] :

- $\mathcal{l}_1 \parallel \mathcal{l}_2 \equiv \mathcal{l}$

- $\langle P_1 \rangle \mathbin{\text{!}}_1 \langle Q_1 \rangle$

- $\langle P_2 \rangle \mathbin{\text{!}}_2 \langle Q_2 \rangle$

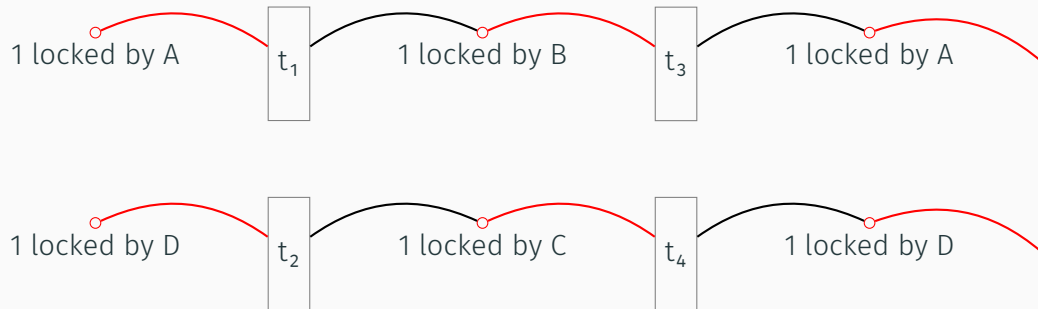
---

$\langle P_1 * P_2 \rangle \mathbin{\text{!}} \langle Q_1 * Q_2 \rangle$

# ABSTRACT UTXO: EXAMPLE TRANSACTION GRAPH

A B C D : Address

$t_{1-4} = L \ni [ t_1, t_2, t_3, t_4 ]$



## ABSTRACT UTXO: EXAMPLE DERIVATION (MONOLITHIC)

```
_ : < A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 > t1-4 < A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 >  
_ = begin A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 ~⟨⟨ * ↦ ⟩⟩  
    (A ↦ 1 * B ↦ 0) * C ↦ 0 * D ↦ 1 ~⟨ t1 :- [FRAME] (C ↦ 0 * D ↦ 1) (A ↦ B) >  
    (A ↦ 0 * B ↦ 1) * C ↦ 0 * D ↦ 1 ~⟨⟨ * ↦ ⟩⟩  
    (C ↦ 0 * D ↦ 1) * A ↦ 0 * B ↦ 1 ~⟨ t2 :- [FRAME] (A ↦ 0 * B ↦ 1) (C ↦ D) >  
    (C ↦ 1 * D ↦ 0) * A ↦ 0 * B ↦ 1 ~⟨⟨ * ↦ ⟩⟩  
    (A ↦ 0 * B ↦ 1) * C ↦ 1 * D ↦ 0 ~⟨ t3 :- [FRAME] (C ↦ 1 * D ↦ 0) (A ↦ B) >  
    (A ↦ 1 * B ↦ 0) * C ↦ 1 * D ↦ 0 ~⟨⟨ * ↦ ⟩⟩  
    (C ↦ 1 * D ↦ 0) * A ↦ 1 * B ↦ 0 ~⟨ t4 :- [FRAME] (A ↦ 1 * B ↦ 0) (C ↦ D) >  
    (C ↦ 0 * D ↦ 1) * A ↦ 1 * B ↦ 0 ~⟨⟨ * ↦ ⟩⟩  
    (A ↦ 1 * B ↦ 0) * C ↦ 0 * D ↦ 1 ~⟨⟨ ↦ * ⟩⟩  
A ↦ 1 * B ↦ 0 * C ↦ 0 * D ↦ 1 ■
```



## ABSTRACT UTxO: EXAMPLE DERIVATION (MODULAR)

$\_ : \langle A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \rangle t_{1-4} \langle A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \rangle$   
 $\_ = \text{begin } A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \quad \sim \langle * \sim \rangle$   
     $(A \mapsto 1 * B \mapsto 0) * C \mapsto 0 * D \mapsto 1 \sim \langle t_{1-4} :- [\text{PAR}] \text{ auto } H_1 H_2 \rangle ++$   
     $(A \mapsto 1 * B \mapsto 0) * C \mapsto 0 * D \mapsto 1 \sim \langle \leftarrow * \rangle$   
     $A \mapsto 1 * B \mapsto 0 * C \mapsto 0 * D \mapsto 1 \quad \blacksquare$

where

$H_1 : \mathbb{R} \langle A \mapsto 1 * B \mapsto 0 \rangle t_1 :: t_3 :: [] \langle A \mapsto 1 * B \mapsto 0 \rangle$   
 $H_1 = A \mapsto 1 * B \mapsto 0 \sim \langle t_1 :- A \sim B \rangle$   
     $A \mapsto 0 * B \mapsto 1 \sim \langle t_3 :- A \leftarrow B \rangle$   
     $A \mapsto 1 * B \mapsto 0 \quad \blacksquare$

$H_2 : \mathbb{R} \langle C \mapsto 0 * D \mapsto 1 \rangle t_2 :: t_4 :: [] \langle C \mapsto 0 * D \mapsto 1 \rangle$   
 $H_2 = C \mapsto 0 * D \mapsto 1 \sim \langle t_2 :- C \leftarrow D \rangle$   
     $C \mapsto 1 * D \mapsto 0 \sim \langle t_4 :- C \sim D \rangle$   
     $C \mapsto 0 * D \mapsto 1 \quad \blacksquare$

## SOUND ABSTRACTION: STATES AND VALIDITY

$\text{absS} : \mathbb{C}.S \rightarrow \mathbb{A}.S$

$\text{absVT} : \mathbb{C}.\text{IsValidTx } t \ s \rightarrow \exists \lambda \hat{t} \rightarrow \mathbb{A}.\text{IsValidTx } \hat{t} \ (\text{absS } s)$

$\text{absVL} : \mathbb{C}.\text{ValidLedger } s \ l \rightarrow \exists \lambda \hat{l} \rightarrow \mathbb{A}.\text{ValidLedger } (\text{absS } s) \ \hat{l}$

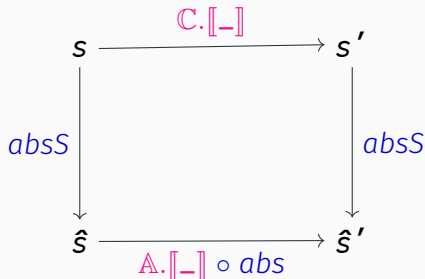
## SOUND ABSTRACTION: DENOTATIONS COINCIDE

$\text{denot-abs-t} : \forall (vt : \mathbb{C}.\text{IsValidTx } t \ s) \rightarrow$

$\mathbb{A}.\llbracket \text{absT } vt \rrbracket (\text{absS } s) \equiv (\text{absS } \langle \$ \rangle \mathbb{C}.\llbracket t \rrbracket s)$

$\text{denot-abs} : \forall (vl : \mathbb{C}.\text{ValidLedger } s \ l) \rightarrow$

$\mathbb{A}.\llbracket \text{absL } vl \rrbracket (\text{absS } s) \equiv (\text{absS } \langle \$ \rangle \mathbb{C}.\llbracket l \rrbracket s)$



# SOUND ABSTRACTION

soundness :

$\forall (vl : \mathbb{C}.ValidLedger\ s\ l) \rightarrow$

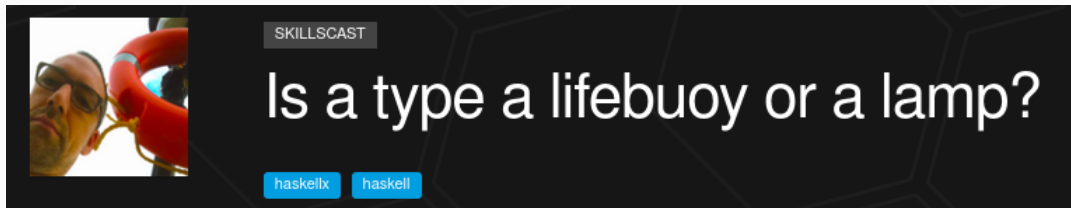
$A \langle P \rangle\ absL\ vl \langle Q \rangle$

---

$\mathbb{C} \langle P \circ absS \rangle\ l \langle Q \circ absS \rangle$

- Deeper compositionality (i.e. monoidally exploit the values in the bag)
  - will require further abstraction of split/merge transactions
- Go beyond the monetary values (states, transaction data)
  - leads to more practical verification of smart contracts
- Generalise to multiple separation views, aka zooming levels
- Generically grow such separation logics, i.e. “Separation Logics à la carte”

Agda as a design guide, rather than merely a verification tool of existing systems.



QUESTIONS?