BitML^x

Cross-chain Smart Contracts for Bitcoin-style Cryptocurrencies

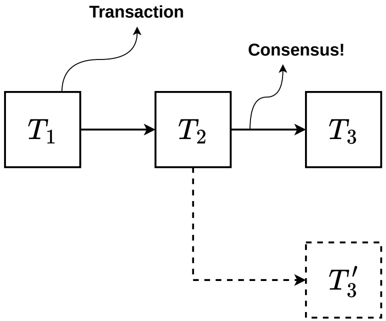
Federico Badaloni[*] Chrysoula Oikonomou[**]
Sebastian Holler[*] Clara Schneidewind[*] Pedro Moreno-Sanchez[**]

[*]Max Planck Institute for Security and Privacy

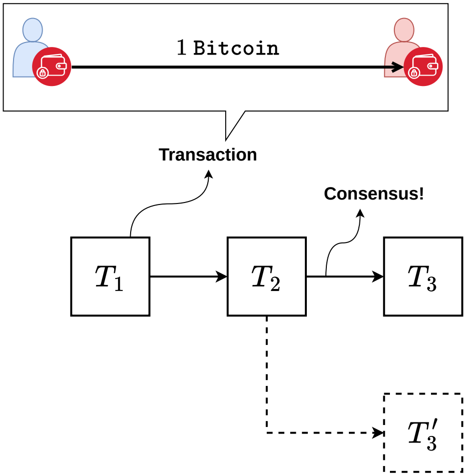
[**]IMDEA Software Institute

FCS 2023
July 9, 2023

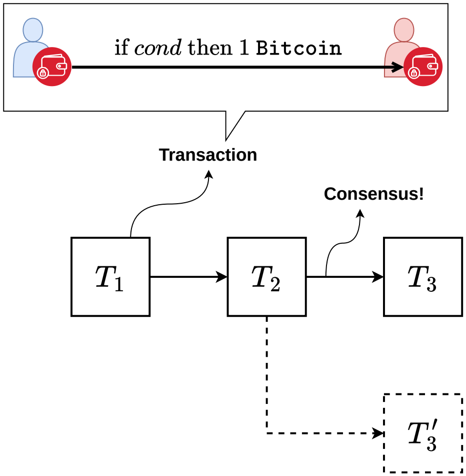
Blockchains



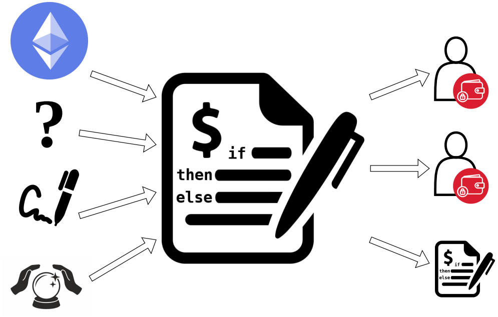
Blockchains



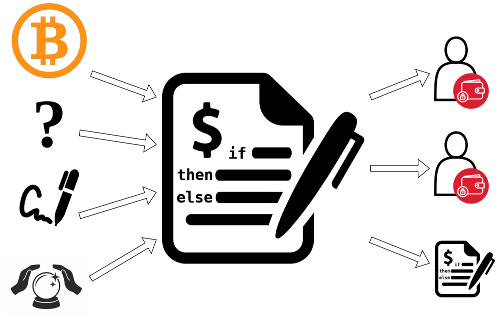
Blockchains



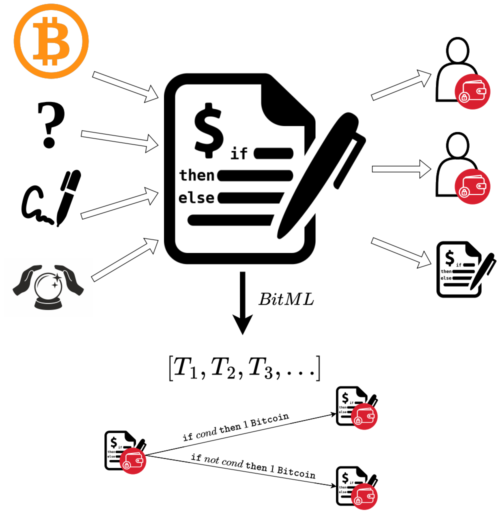
Smart Contracts



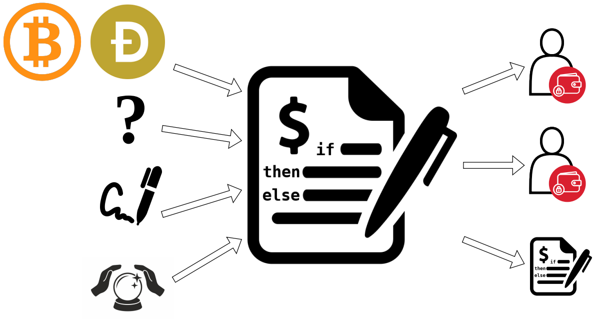
BitML: From Transactions To Smart Contracts



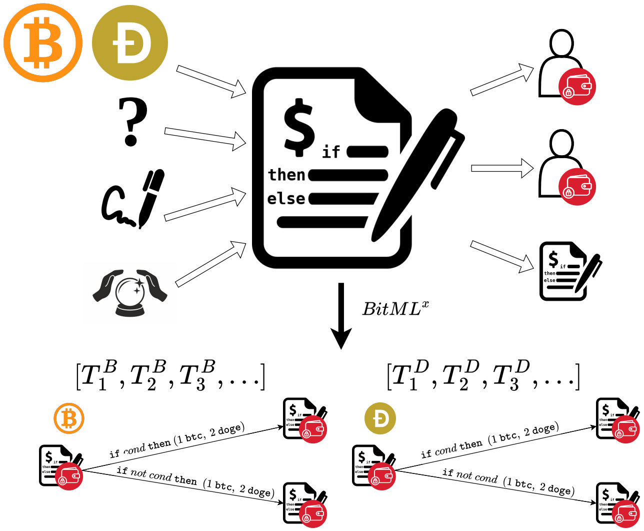
BitML: From Transactions To Smart Contracts



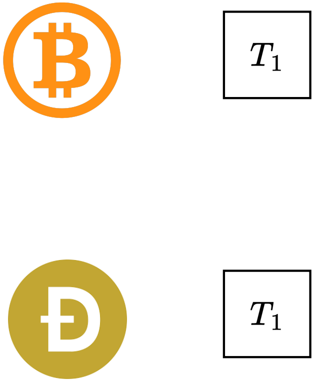
BitML^x: Cross-chain Smart Contracts



BitML^x: Cross-chain Smart Contracts


if
then
else
BitML^x
$[T_1^B, T_2^B, T_3^B, \ldots]$
$[T_1^D, T_2^D, T_3^D, \ldots]$
if cond then (1 btc, 2 doge)
if not cond then (1 btc, 2 doge)
if cond then (1 btc, 2 doge)
if not cond (1 btc, 2 doge)

Cross-chain & Consensus

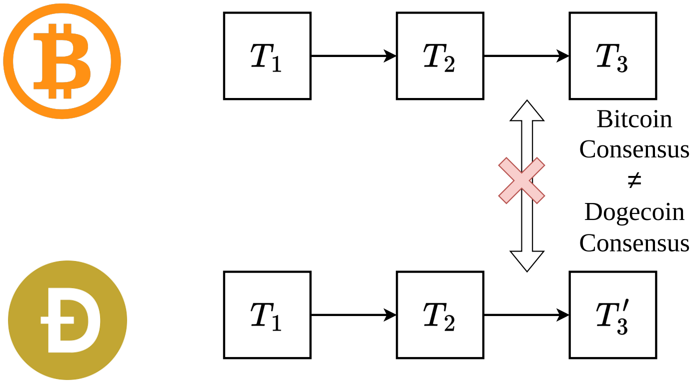


Cross-chain & Consensus

$$\begin{tikzcd} \boxed{T_1} \arrow[r] & \boxed{T_2} \end{tikzcd}$$

$$\begin{tikzcd} \boxed{T_1} \arrow[r] & \boxed{T_2} \end{tikzcd}$$

Cross-chain & Consensus



Presentation Overview

1. BitML & Synchronicity
2. BitML^x
3. Compilation
4. Correctness

BitML syntax

Preconditions:

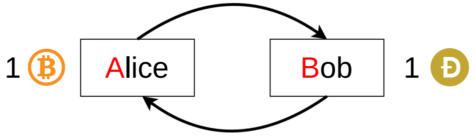
$G ::= A :! v$₿	user deposits
$\mid A :$ `secret` s	secret commitment

Contracts:

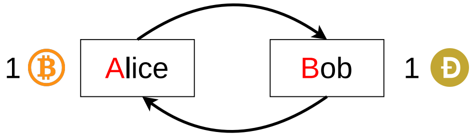
$C ::= D_1 + D_2 + \cdots + D_k$	choose any sub-contract
$D ::= A : \ D$	A signs before executing D
$\mid$ `after` $t : D$	wait until t, execute D
$\mid$ `reveal` s `then` C	reveal s before executing C
$\mid$ `split` $\vec{v}$₿ $\to \vec{C}$	split into many contracts
$\mid$ `withdraw` A	A gets the money

(Don't panic: example in next slide.)

Example: Alice Want To Swap Coins

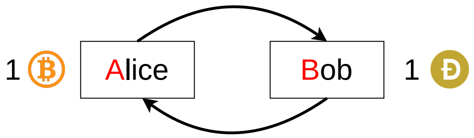


Example: Alice Want To Swap Coins



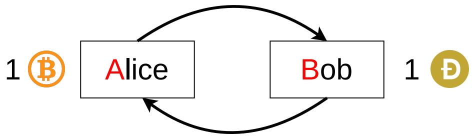
$\{A :! 1฿ \mid B : \texttt{secret } s\}\mathit{Swap}^{฿}$

Example: Alice Want To Swap Coins



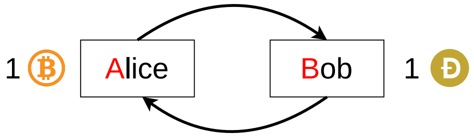
$$\underbrace{\{A :! 1\text{₿} \mid B : \texttt{secret } s\}}_{\text{preconditions}} \; \underbrace{\mathit{Swap}^{\text{₿}}}_{\text{contract}}$$

Example: Alice Want To Swap Coins



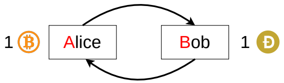
$$\overbrace{\{\underbrace{A :! 1₿}_{\text{user deposits}} \mid \underbrace{\mid B : \texttt{secret}\ s}_{\text{secret commitments}}\}}^{\text{preconditions}} \overbrace{Swap^{₿}}^{\text{contract}}$$

Example: Alice Want To Swap Coins



$$\{ \overbrace{\underbrace{A :! 1\text{฿}}_{\text{user deposits}} \quad \mid \quad \underbrace{\mid B : \texttt{secret}\ s}_{\text{secret commitments}}}^{\text{preconditions}} \} \overbrace{Swap^{\text{฿}}}^{\text{contract}} \quad \{B :! 1\text{Ð} \mid B : \texttt{secret}\ s\} Swap^{\text{Ð}}$$

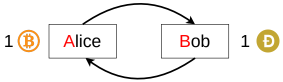
Example: Alice Want To Swap Coins



$$\{A :! 1 ฿ \mid B : \texttt{secret } s\} Swap^{฿}$$

$$\{B :! 1 Ɖ \mid B : \texttt{secret } s\} Swap^{Ɖ}$$

Example: Alice Want To Swap Coins

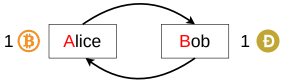


$$\{A :! 1\text{Ƀ} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{Ƀ}}$$

$$\{B :! 1\text{Đ} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{Đ}}$$

$$\mathit{Swap}^{\text{Ƀ}} = \mathit{Exchange}^{\text{Ƀ}} + \mathit{Refund}^{\text{Ƀ}}$$

Example: Alice Want To Swap Coins



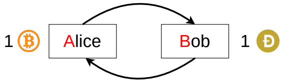
$$\{A :! 1\text{฿} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{฿}}$$

$$\{B :! 1\text{Ɖ} \mid B : \texttt{secret}\ s\}\mathit{Swap}^{\text{Ɖ}}$$

$$\mathit{Swap}^{\text{฿}} = \mathit{Exchange}^{\text{฿}} + \mathit{Refund}^{\text{฿}}$$

$$\mathit{Exchange}^{\text{฿}} = \texttt{reveal}\ s\ .\ \texttt{withdraw}\ B$$

Example: Alice Want To Swap Coins



$$\{A :! 1฿ \mid B : \texttt{secret } s\} Swap^{฿}$$

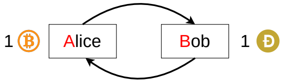
$$\{B :! 1Ɖ \mid B : \texttt{secret } s\} Swap^{Ɖ}$$

$$Swap^{฿} = Exchange^{฿} + Refund^{฿}$$

$$Exchange^{฿} = \texttt{reveal } s \,.\, \texttt{withdraw } B$$

$$Refund^{฿} = \texttt{after } t_0 : \texttt{withdraw } A$$

Example: Alice Want To Swap Coins



$$\{A :! 1\text{฿} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{฿}}$$

$$\{B :! 1\text{Ð} \mid B : \texttt{secret } s\}\mathit{Swap}^{\text{Ð}}$$

$$\mathit{Swap}^{\text{฿}} = \mathit{Exchange}^{\text{฿}} + \mathit{Refund}^{\text{฿}}$$

$$\mathit{Exchange}^{\text{฿}} = \texttt{reveal } s \,.\, \texttt{withdraw } B$$

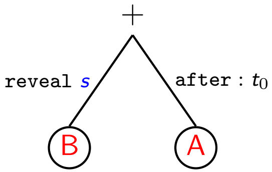
$$\mathit{Refund}^{\text{฿}} = \texttt{after } t_0 : \texttt{withdraw } A$$

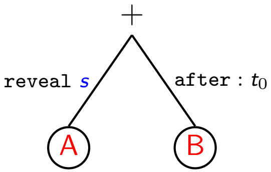
$$\begin{aligned}\mathit{Swap}^{\text{Ð}} = {} & \texttt{reveal } s \,.\, \texttt{withdraw } A \\ & + \texttt{after } t_0 : \texttt{withdraw } B\end{aligned}$$

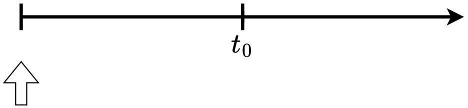
Successful Swap

Alice

Bob





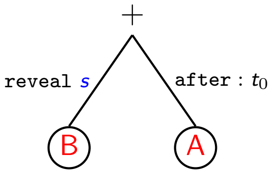


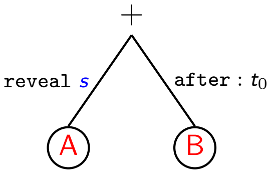
Successful Swap

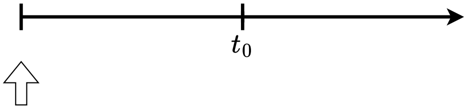
Alice

Bob

- Wanna swap coins?







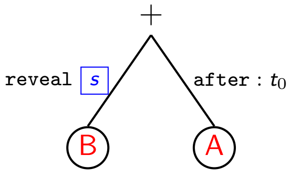
Successful Swap

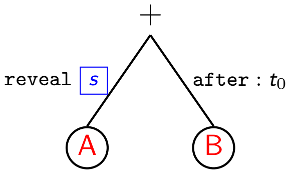
Alice

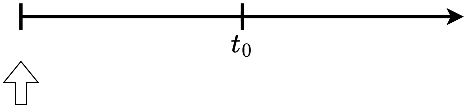
- Wanna swap coins?

Bob

- Sure! Here is $\boxed{s}$.







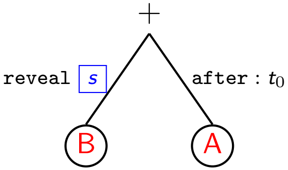
Successful Swap

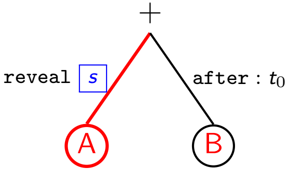
Alice

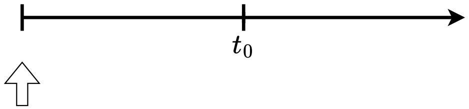
- Wanna swap coins?
- Thanks! I'm taking ₿😄

Bob

- Sure! Here is $\boxed{s}$.







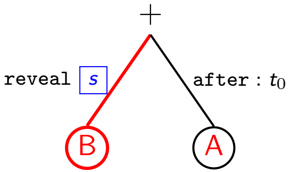
Successful Swap

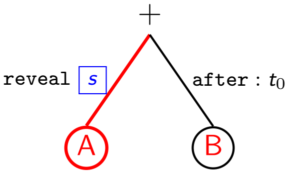
Alice

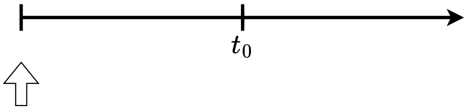
- Wanna swap coins?
- Thanks! I'm taking Đ😄

Bob

- Sure! Here is $\boxed{s}$.
- And I'm taking ฿😄



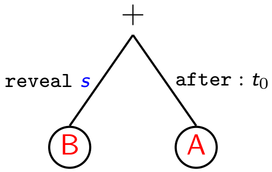


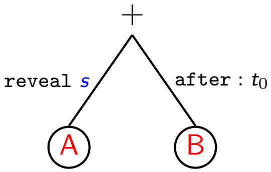


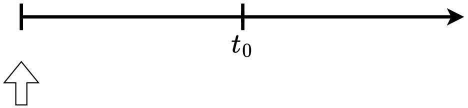
Successful Refund

Alice

Bob





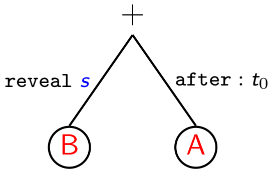


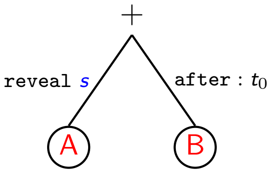
Successful Refund

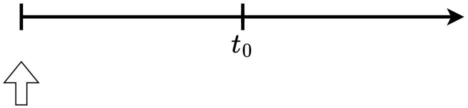
Alice

- Wanna swap coins?

Bob







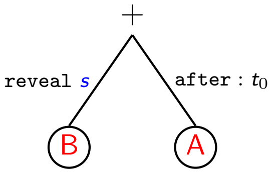
Successful Refund

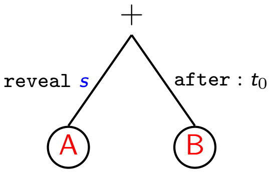
Alice

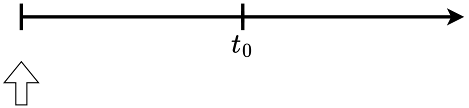
- Wanna swap coins?

Bob

- Not really.







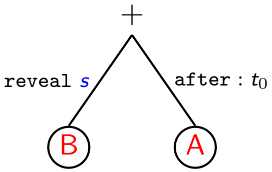
Successful Refund

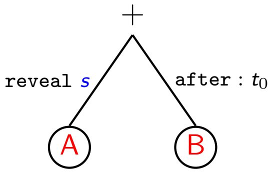
Alice

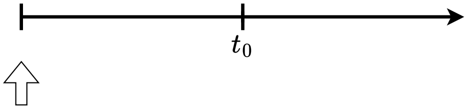
- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.







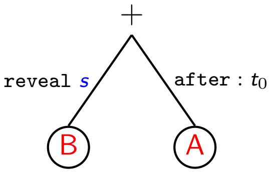
Successful Refund

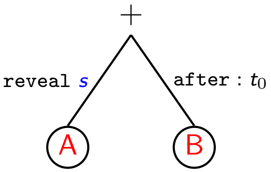
Alice

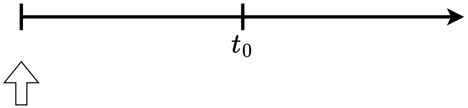
- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.
- Yeah, sorry. 😅







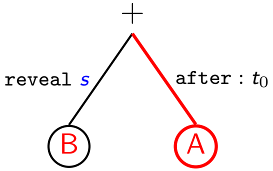
Successful Refund

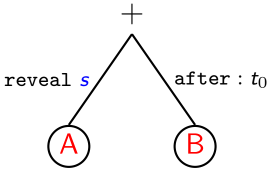
Alice

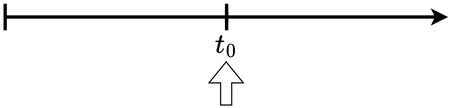
- Wanna swap coins?
- Oh. That's ok. 🥲
- I'll take back my ₿.

Bob

- Not really.
- Yeah, sorry. 😅



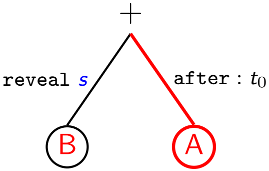




Successful Refund

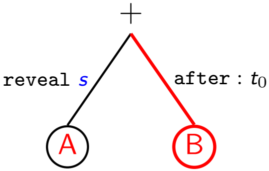
Alice

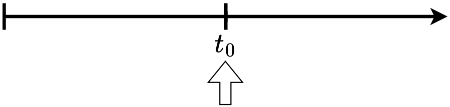
- Wanna swap coins?
- Oh. That's ok. 🥲
- I'll take back my ₿.



Bob

- Not really.
- Yeah, sorry. 😅
- And I'll take back my Ḋ.

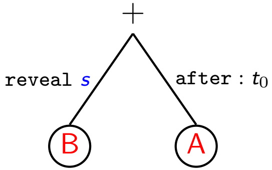


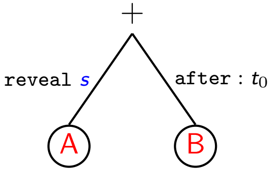


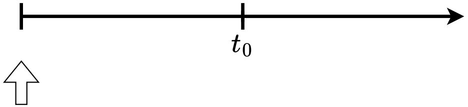
But...

Alice

Bob





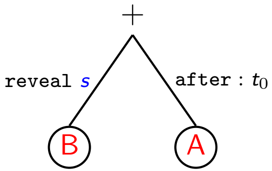


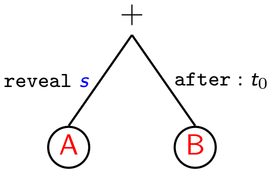
But...

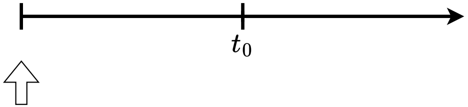
Alice

- Wanna swap coins?

Bob







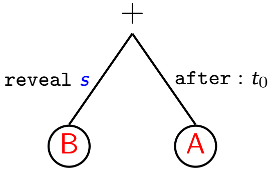
But...

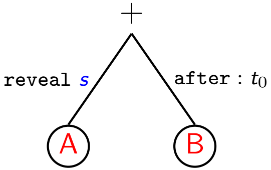
Alice

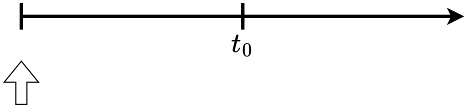
- Wanna swap coins?

Bob

- Not really.







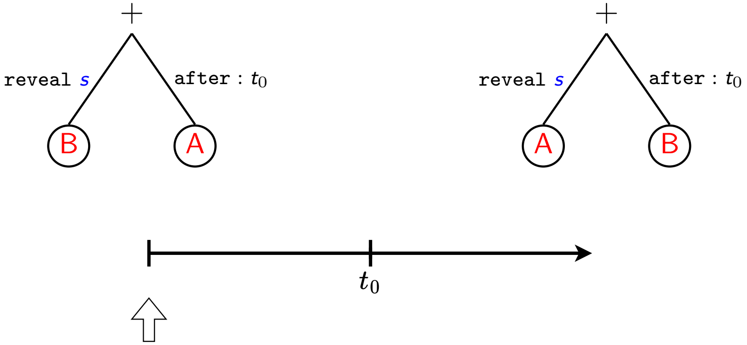
But...

Alice

- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.



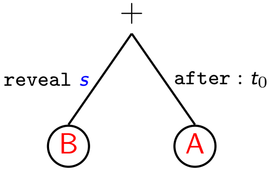
But...

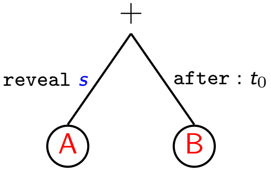
Alice

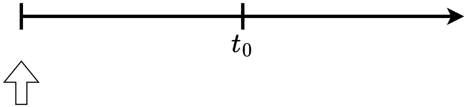
- Wanna swap coins?
- Oh. That's ok. 🥲

Bob

- Not really.
- Yes... totally ok. 😶







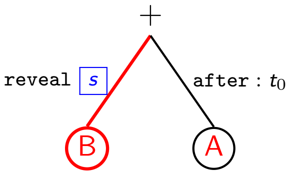
But...

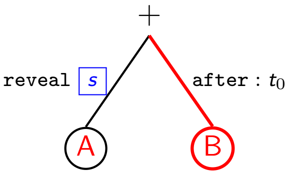
Alice

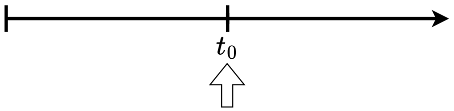
- Wanna swap coins?
- Oh. That's ok. 🥲
- Bob, WTF? 😱

Bob

- Not really.
- Yes... totally ok. 😶







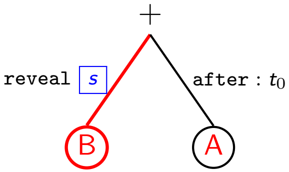
But...

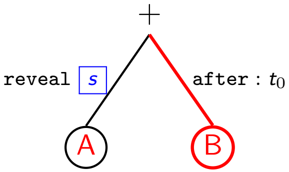
Alice

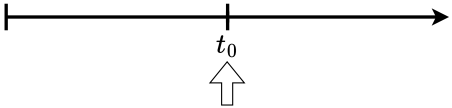
- Wanna swap coins?
- Oh. That's ok. 🥲
- Bob, WTF? 😱

Bob

- Not really.
- Yes... totally ok. 😶
- See ya, looser. 😈







Introducing: BitML^x

Alice should have read the bibliography on sound cryptographic protocol designs.

Introducing: BitML^x

Alice should have ~~read the bibliography on sound cryptographic protocol designs.~~ switched to BitML^x!

$\mathrm{B} = [v_1 \mathbb{B}_1, \ldots, v_k \mathbb{B}_k]$	balance
$G ::= A \;:!\; \mathrm{B}$	user deposit (in all chains)
$\mid A : \texttt{secret}\ s$	secret commitment

$C ::= D \mathrel{+\!\!>} C$	choose D or skip to C
$\mid \texttt{withdraw}\ \vec{\mathrm{B}} \to \vec{A}$	last choice is always `withdraw`
$D ::= A :\ \ D$	
$\mid \texttt{reveal}\ s\ \texttt{then}\ C$	reveal secrets before executing C
$\mid \texttt{split}\ \vec{\mathrm{B}} \to \vec{C}$	split into many contract
$\texttt{withdraw}\ \vec{\mathrm{B}} \to \vec{A}$	distribute the balance among users

Swap In BitML^x

$$\{A\ :!\,(1\text{Ƀ}, 0\text{Đ}) \mid B\ :!\,(0\text{Ƀ}, 1\text{Đ})\}\mathit{Swap}^x$$

Swap In BitML^x

$$\{A \; :! \, (1 Ƀ, 0 Đ) \mid B \; :! \, (0 Ƀ, 1 Đ)\} \mathit{Swap}^x$$
$$\mathit{Swap}^x \; = \mathit{Exchange}^x \; \mathrel{+\!\!>} \; \mathit{Refund}^x$$

Swap In BitML^x

$$\{A :! (1\text{Ƀ}, 0\text{Đ}) \mid B :! (0\text{Ƀ}, 1\text{Đ})\} \mathit{Swap}^x$$
$$\mathit{Swap}^x = \mathit{Exchange}^x \mathrel{+\!\!>} \mathit{Refund}^x$$

$$\begin{aligned}
\mathit{Exchange}^x = \;& \texttt{withdraw}(\\
& \quad (0\text{Ƀ}, 1\text{Đ}) \rightarrow A, \\
& \quad (1\text{Ƀ}, 0\text{Đ}) \rightarrow B \\
&)
\end{aligned}$$

Swap In BitML^x

$$\{A :! (1\text{Ƀ}, 0\text{Đ}) \mid B :! (0\text{Ƀ}, 1\text{Đ})\}\mathit{Swap}^x$$
$$\mathit{Swap}^x = \mathit{Exchange}^x \mathrel{+\!\!>} \mathit{Refund}^x$$

$$\mathit{Exchange}^x = \texttt{withdraw}((0\text{Ƀ}, 1\text{Đ}) \to A, (1\text{Ƀ}, 0\text{Đ}) \to B)$$

$$\mathit{Refund}^x = \texttt{withdraw}((1\text{Ƀ}, 0\text{Đ}) \to A, (0\text{Ƀ}, 1\text{Đ}) \to B)$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{???} \begin{cases} \vec{T}^x_{\text{₿}} \\ \vec{T}^x_{\text{Ð}} \end{cases}$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} \mathit{Swap}^x_{\text{Ƀ}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{Ƀ}} \\ \mathit{Swap}^x_{\text{Đ}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{Đ}} \end{cases}$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} \mathit{Swap}^x_{\text{₿}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{₿}} \\ \mathit{Swap}^x_{\text{Ð}} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{\text{Ð}} \end{cases}$$

$$\mathit{Swap}^x_{\text{₿}} \xLeftrightarrow[\text{collaterals}]{\text{step secrets}} \mathit{Swap}^x_{\text{Ð}}$$

BitML^x Compilation Idea

$$\mathit{Swap}^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} \mathit{Swap}^x_{₿} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{₿} \\ \mathit{Swap}^x_{Đ} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{Đ} \end{cases}$$

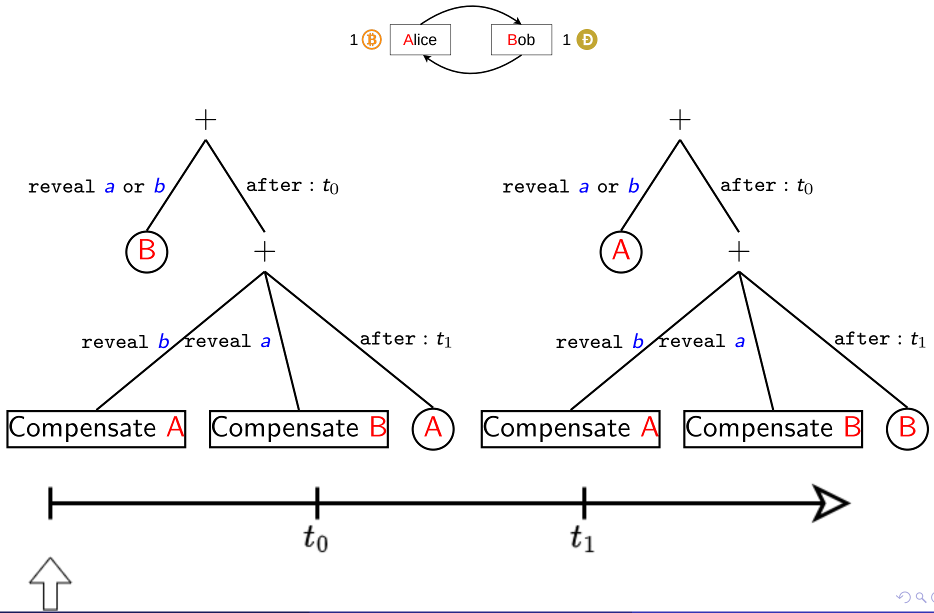
$$\mathit{Swap}^x_{₿} \xLeftrightarrow[\text{collaterals}]{\overbrace{\text{step secrets}}^{\text{prove of execution intent}}} \mathit{Swap}^x_{Đ}$$

BitML^x Compilation Idea

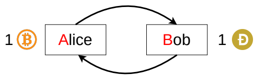
$$Swap^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} Swap^x_{Ƀ} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{Ƀ} \\ Swap^x_{Ð} \xrightarrow{\text{BitML compiler}} \vec{T}^x_{Ð} \end{cases}$$

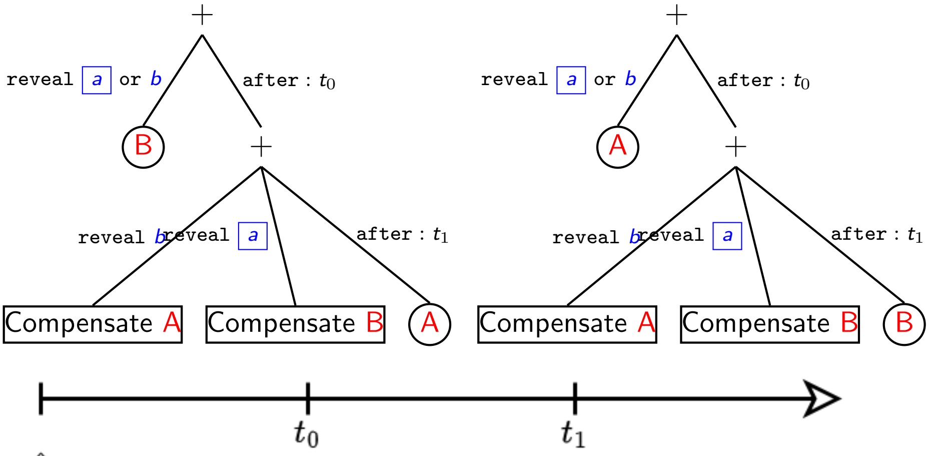
$$Swap^x_{Ƀ} \overset{\overbrace{\text{step secrets}}^{\text{to prove execution intent}}}{\underset{\underbrace{\text{collaterals}}_{\text{to compensate asynchronous behaviours}}}{\Longleftrightarrow}} Swap^x_{Ð}$$

Compilation In Action

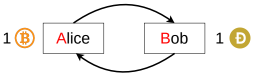


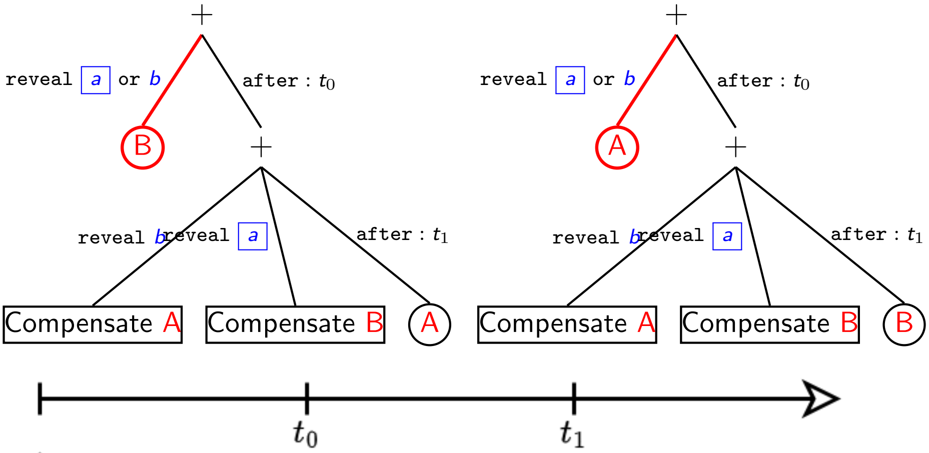
Compilation In Action



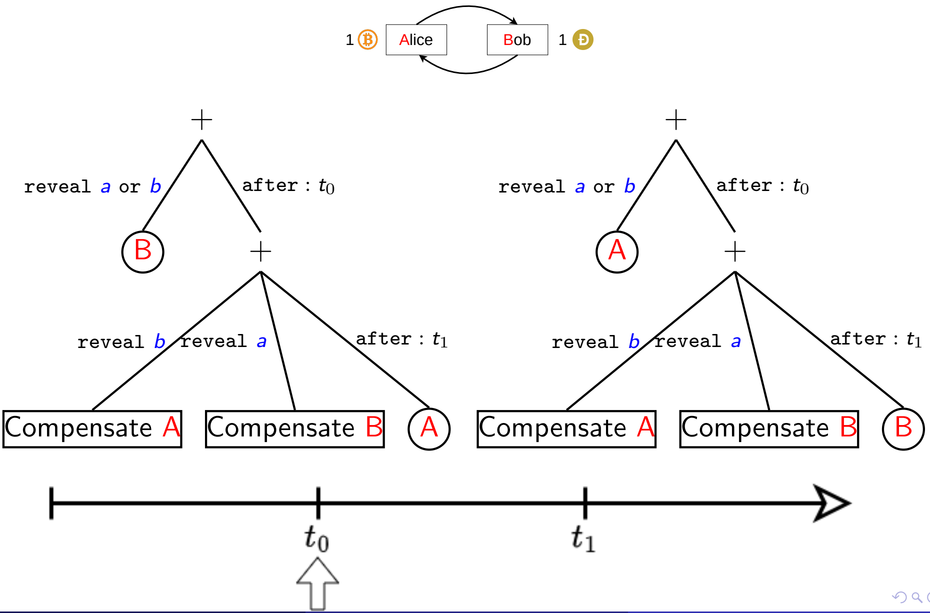


Compilation In Action

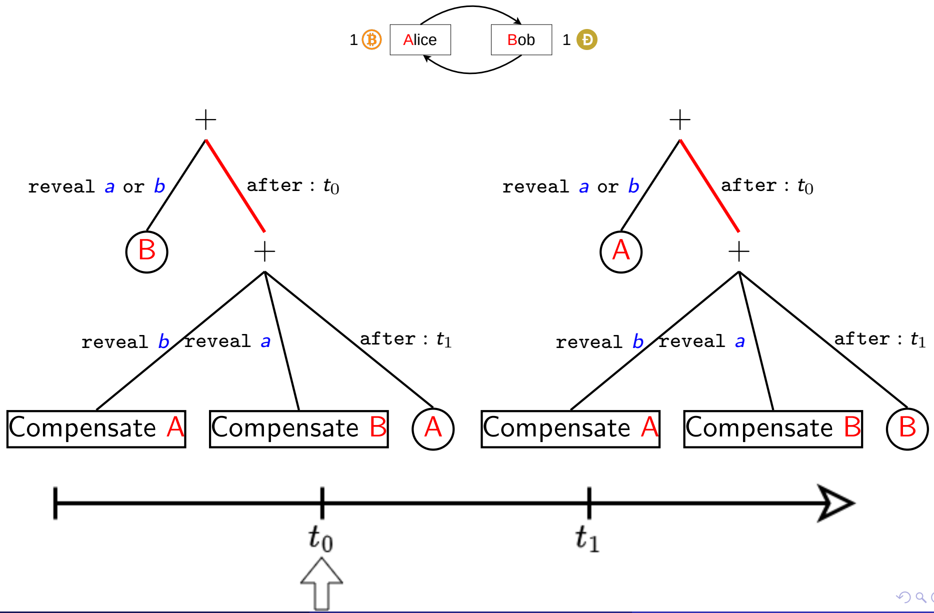




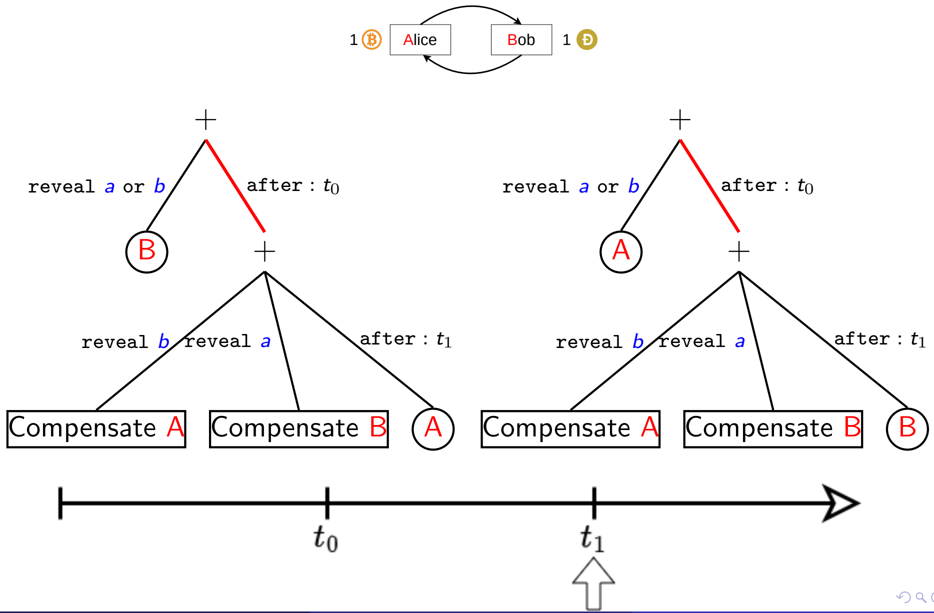
Compilation In Action



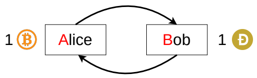
Compilation In Action

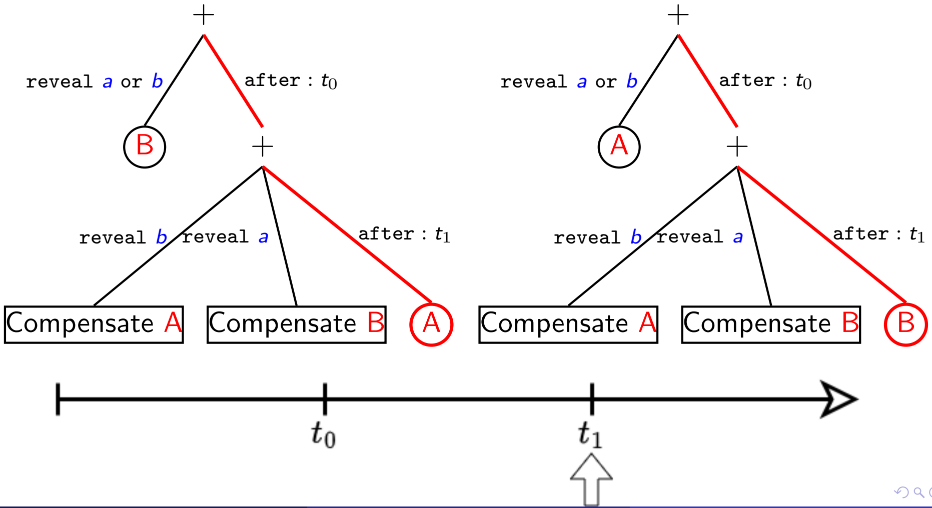


Compilation In Action

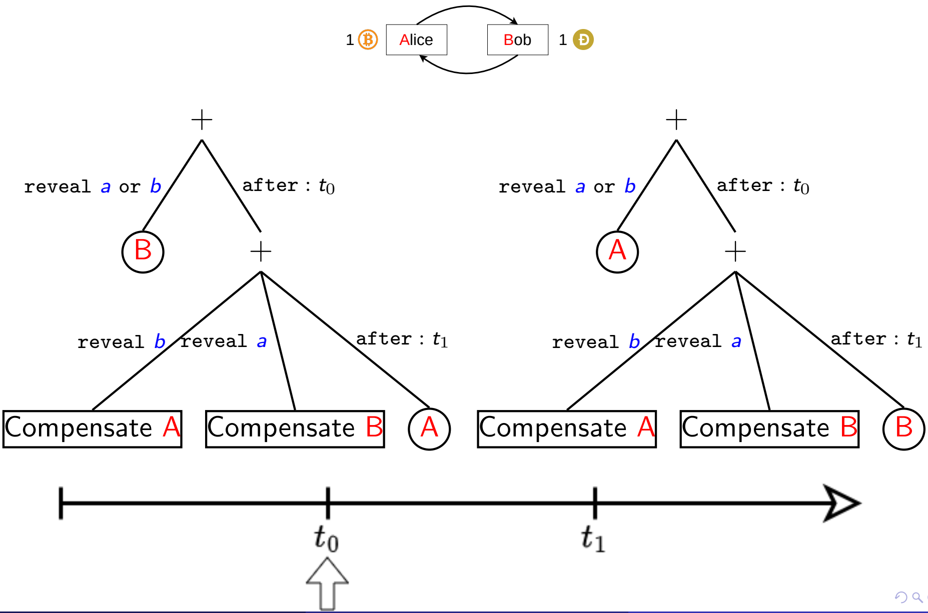


Compilation In Action

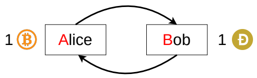


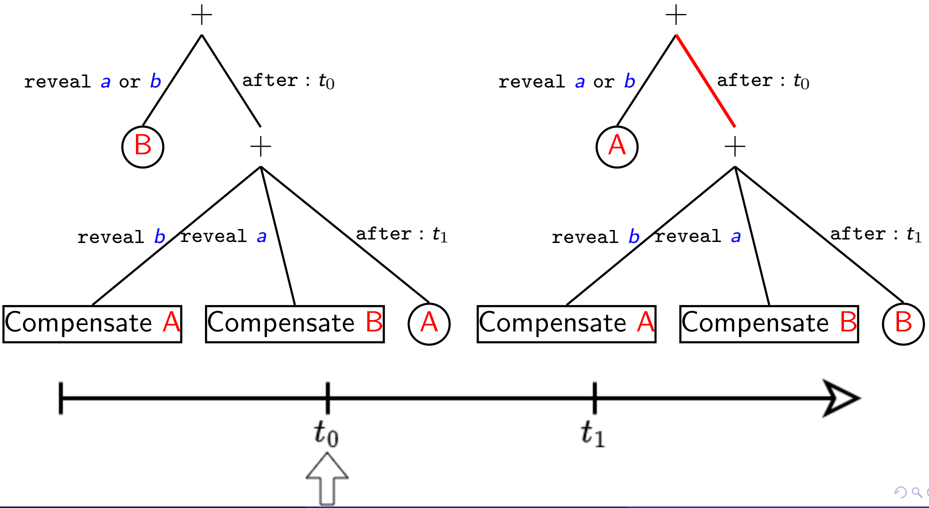


Compilation In Action

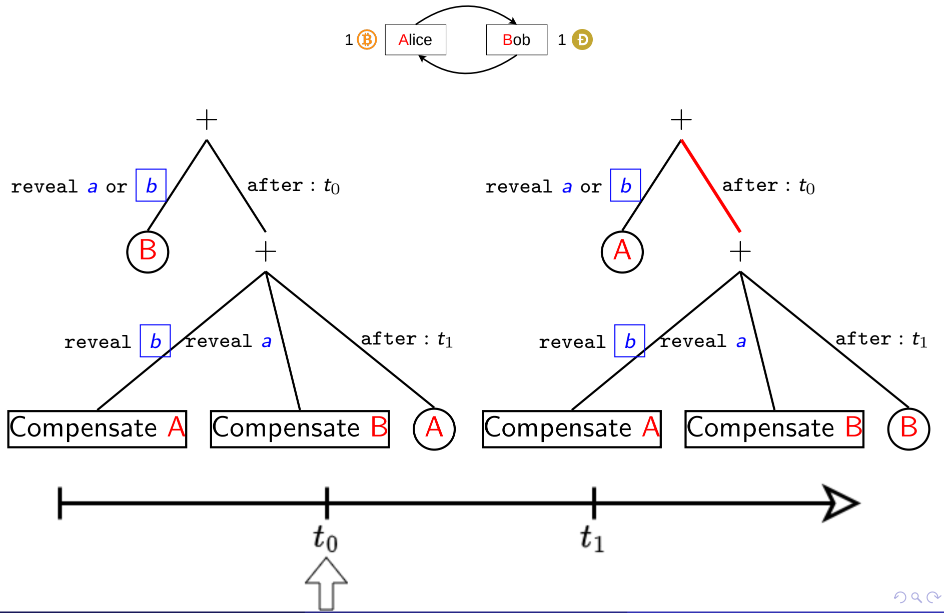


Compilation In Action

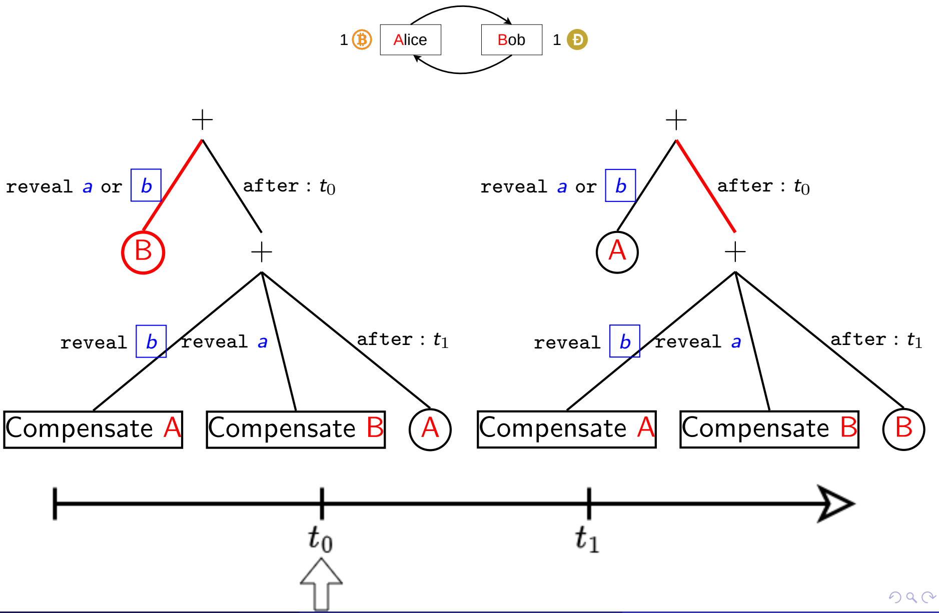




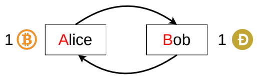
Compilation In Action

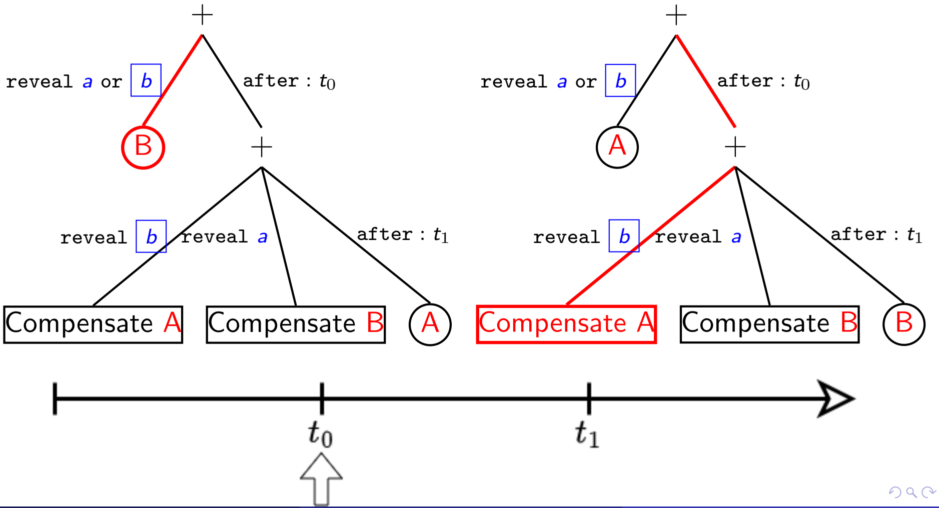


Compilation In Action



Compilation In Action





User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} Swap^x_{₿} \\ Swap^x_{Ð} \end{cases}$$

User Strategies

Suppose Alice follows a strategy S^x_A s.t.

$$S^x_A(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x\text{ compiler}} \begin{cases} Swap^x_{\text{Ƀ}} \\ Swap^x_{\text{Ð}} \end{cases}$$

$$S^x_A \xrightarrow{\text{and strategy compiler!}} \begin{cases} S^{\text{Ƀ}}_A \\ S^{\text{Ð}}_A \end{cases}$$

User Strategies

Suppose Alice follows a strategy S_A^x s.t.

$$S_A^x(Swap^x) = Refund^x$$

$$Swap^x \xrightarrow{\text{BitML}^x \text{ compiler}} \begin{cases} Swap^x_{₿} \\ Swap^x_{Đ} \end{cases}$$

$$S_A^x \xrightarrow{\text{and strategy compiler!}} \begin{cases} S_A^{₿} \\ S_A^{Đ} \end{cases}$$

$$S_A^{₿}(Swap^x_{₿}) = \texttt{if revealed } b$$
$$\texttt{then } Compensate\ A$$
$$\texttt{else wait until } Refund^x_{₿}$$

Correctness

Theorem (Compiler correctness, informal)

Each strategy of an honest user A on a BitML^x contract C translates into a strategy on the concurrently executing compiled BitML contracts $C_{Ƀ} \mid C_{Đ}$ that allows A to extract at least as many assets from $C_{Ƀ} \mid C_{Đ}$ as from C with the original strategy.

Thanks!

- BitML^x allows you to model cross-blockchain smart contracts.
- It's compiled to concurrently executing BitML contracts.
- Security by mechanisms of step secrets and collaterals.
- Work in Progress:
  - Done: BitML^{x}compiler in Haskell.
  - Currently: proving BitML^x correctness.

Download slides and PoC compiler:

Bonus 1: Adversarial Scheduling

BitML

$$C_1 + \cdots + C_k$$

- Users can always execute a valid option.
- Guaranteed to meet deadlines.
- In case of many valid options, adversary decides.

BitML^x

$$C_1^x \mathrel{+\!\!\!>} \ldots \mathrel{+\!\!\!>} C_k^x$$

- Only one valid option at a time.
- Round-based execution.
- Users can act before a subcontract is skipped.

Bonus 2: Collaterals

Every user locks, on each blockchain $\mathbb{B}$, an extra collateral deposit of value:

$$c_{\mathbb{B}} \ = \ b_{\mathbb{B}} \ \times \ (n-2)$$

where $b_{\mathbb{B}}$ is the contract balance in that blockchain, and n is the number of participants.