Computationally sound Bitcoin tokens

Massimo Bartoletti (bart@unica.it)

Stefano Lande

University of Cagliari

Roberto Zunino

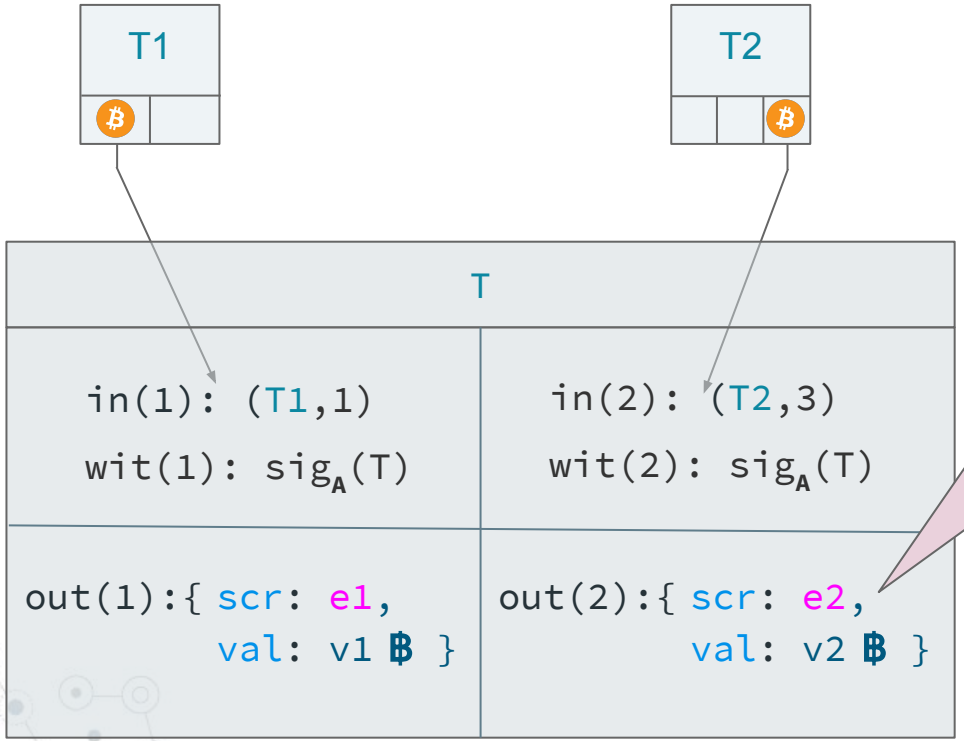
University of Trento

Overview

- Smart contracts on Bitcoin? Tokens??
  - Tokens: only off-chain (no DEX!)

- Bitcoin extension: covenants (BIP 119)
  - Non-Fungible Tokens (NFT)
  - Turing-completeness (⇒ proof in this paper)
  - Fungible Tokens?? (Turing-complete ⇏ efficient)
- New Bitcoin extension: neighbourhood covenants
  - Fungible Tokens, secure & efficient

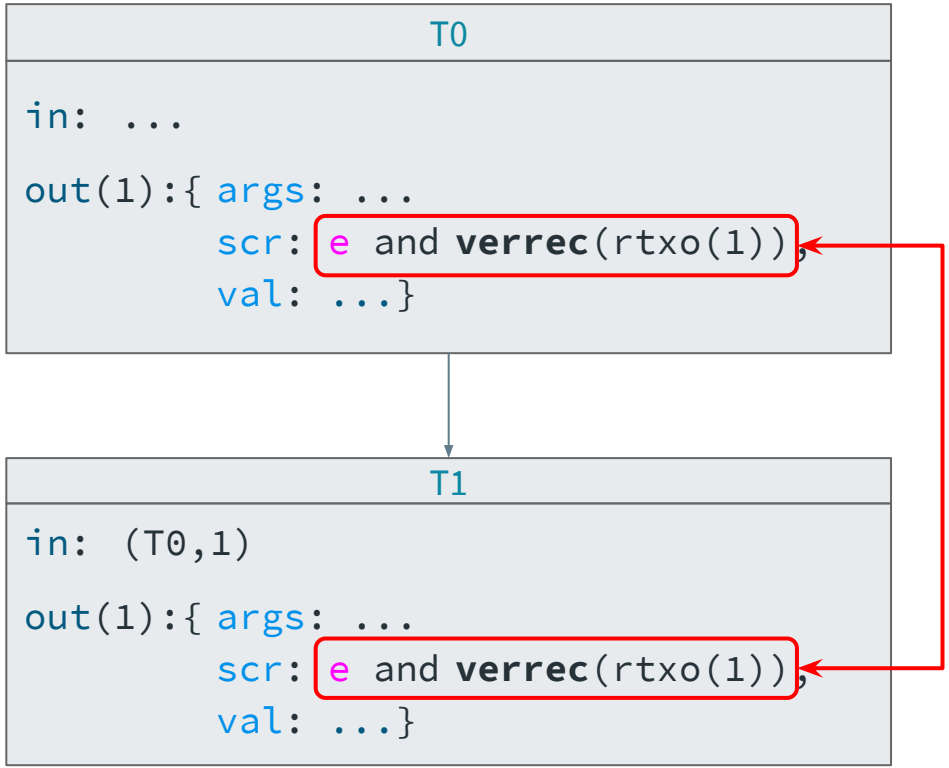
Bitcoin transactions & scripts



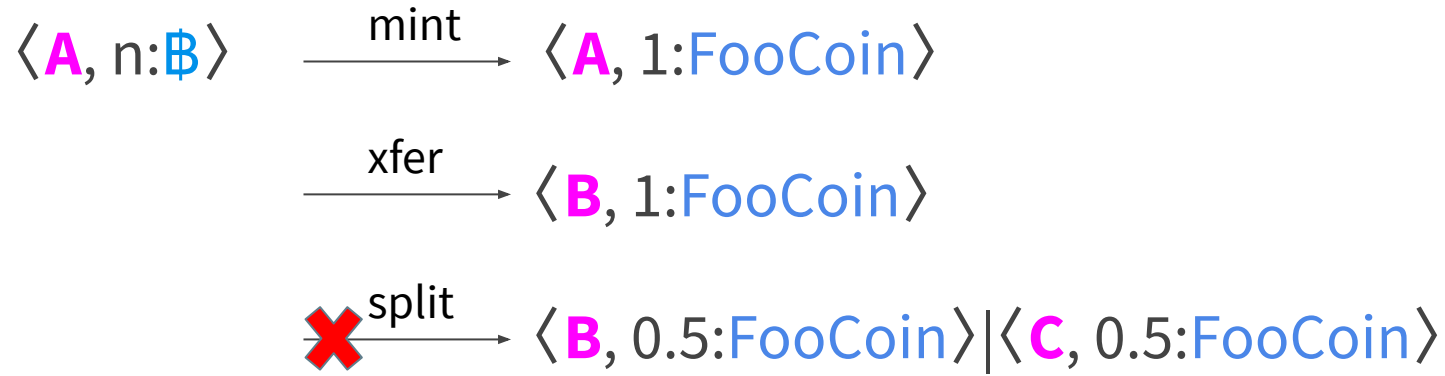

```
k
e.n
e + e
e - e
e < e
e = e
rtx.wit

if e then e else e
versig(k,e)
H(e)
...
```

NO TOKENS!

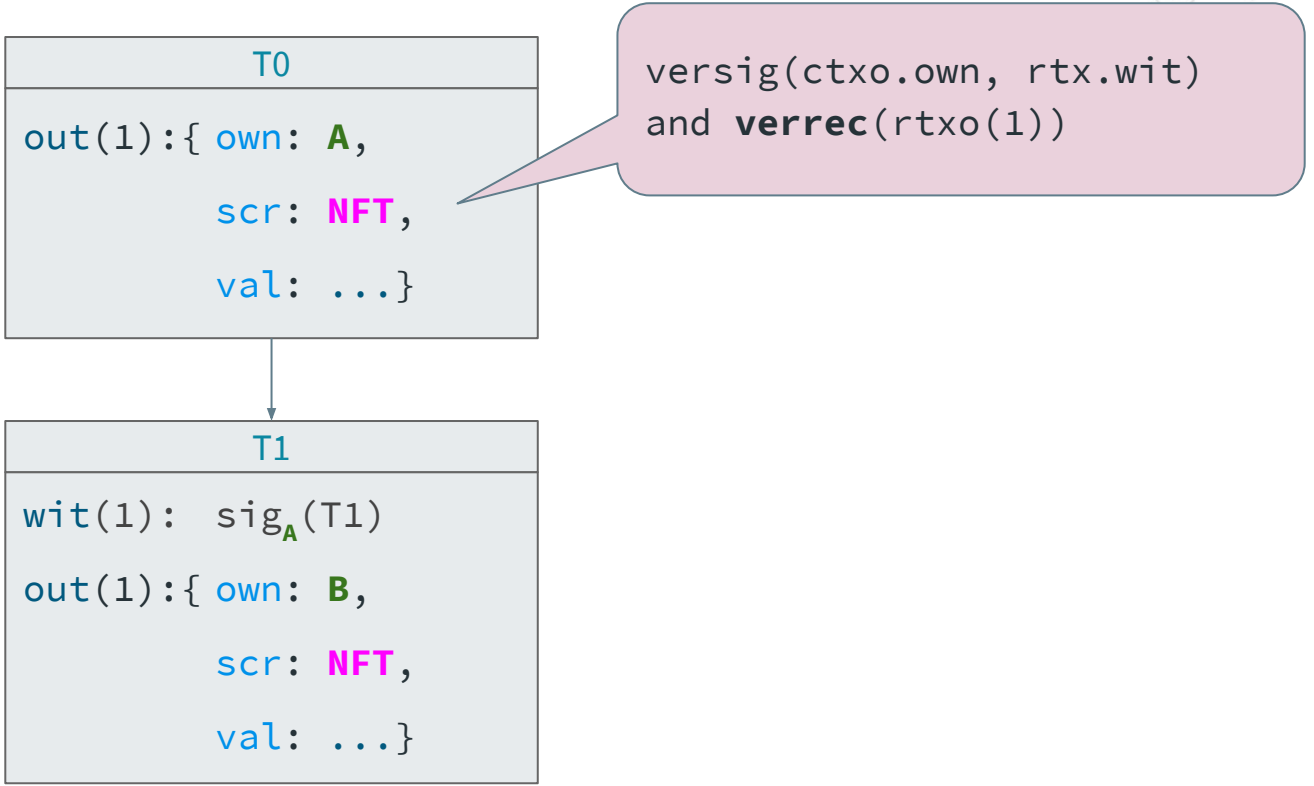
Bitcoin covenants



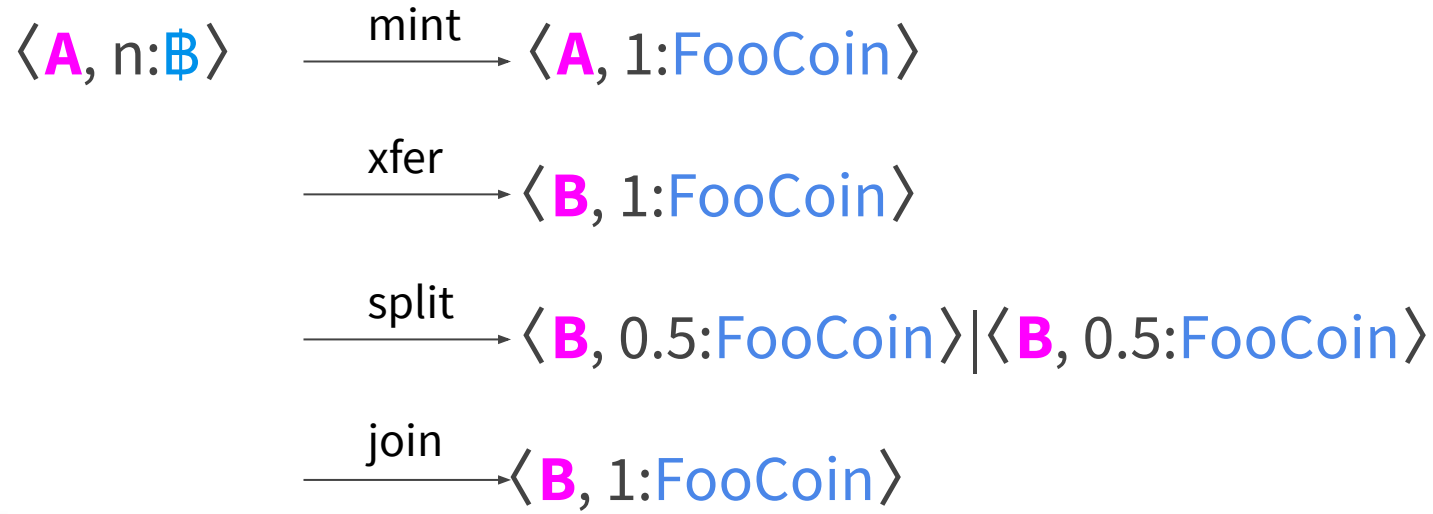
NFTs, symbolically



A non-fungible token



Fungible tokens, symbolically

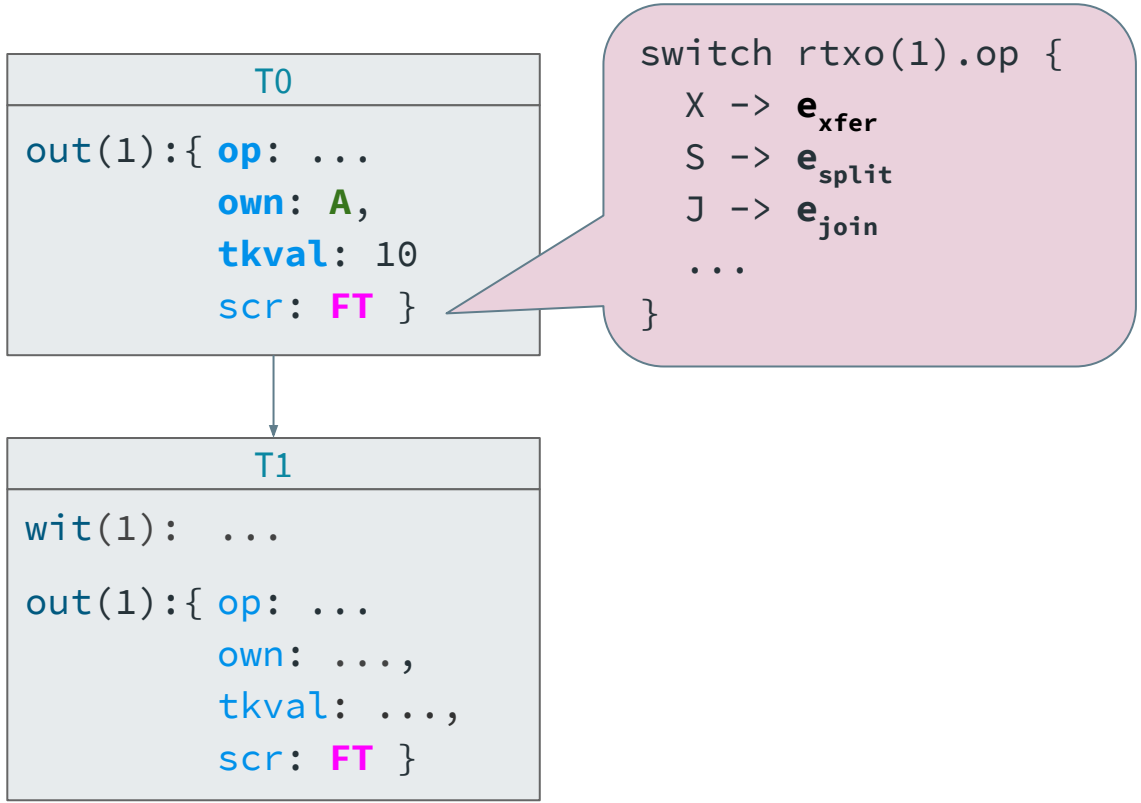


Fungible tokens, symbolically

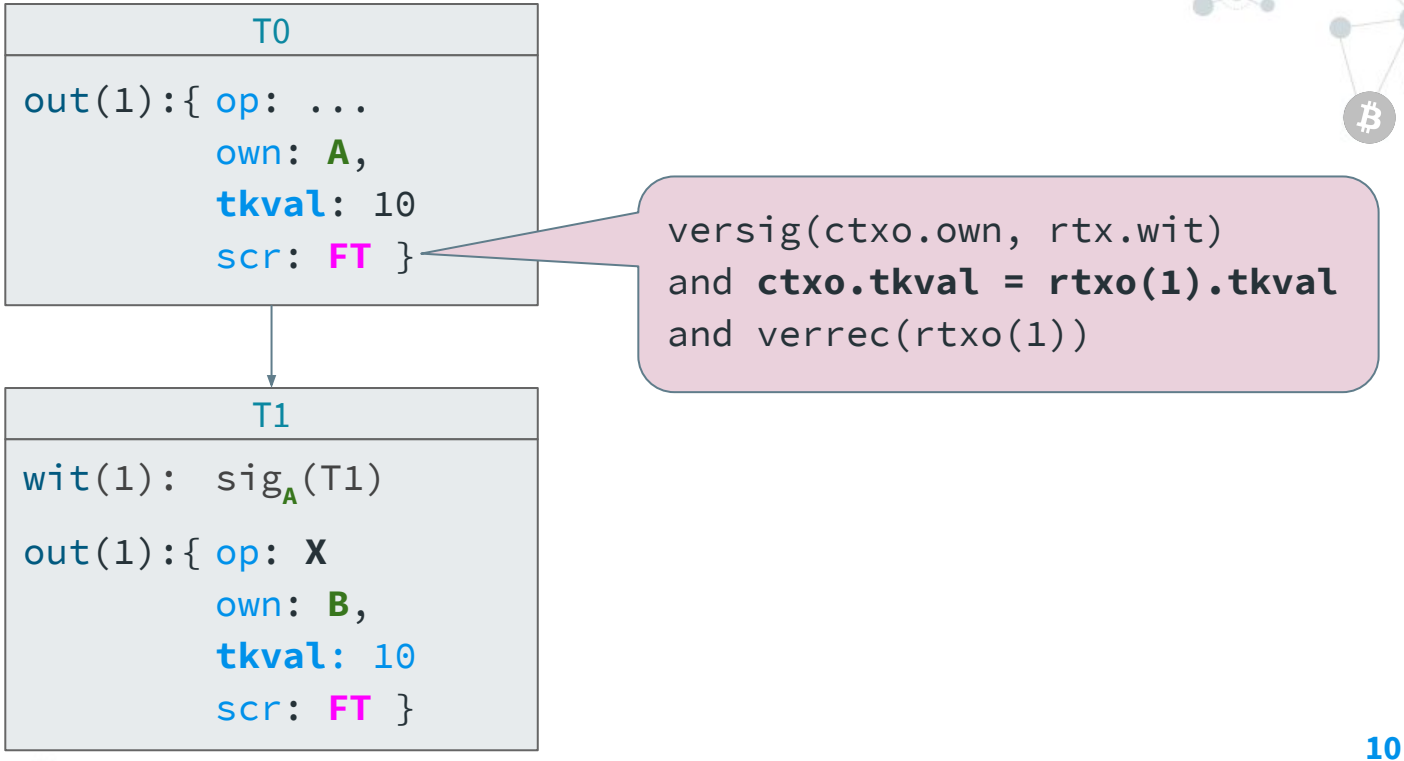
⟨**A**, 1:฿⟩|⟨**B**, 10:FooCoin⟩

⟨**A**, 10:FooCoin⟩|⟨**B**, 1:฿⟩

Fungible tokens



Transfer


T0
out(1):{ op: ...
own: A,
tkval: 10
scr: FT }
versig(ctxo.own, rtx.wit)
and ctxo.tkval = rtxo(1).tkval
and verrec(rtxo(1))
T1
wit(1): sig_A(T1)
out(1):{ op: X
own: B,
tkval: 10
scr: FT }

Split

T0

```
out(1):{ op: ...
         own: A,
         tkval: 10
         scr: FT }
```

```
versig(ctxo.own, rtx.wit)
and ctxo.tkval =
    rtxo(1).tkval + rtxo(2).tkval
and outlen(rtx) = 2
and verrec(rtxo(1))
and verrec(rtxo(2))
```

T1

```
out(1):{ op: S, own: A, tkval: 1, scr: FT }
out(2):{ op: …, own: B, tkval: 9, scr: FT }
```

Join

T0

```
out(1):{ op: ...
         own: B,
         tkval: 8
         scr: FT }
```

T1

```
out(1):{ op: ...
         own: B,
         tkval: 2
         scr: FT }
```

```
versig(ctxo.own, rtx.wit)
and inlen(rtx) = 1
and outlen(rtx) = 1
and verrec(rtxo(1))
and rtxo(1).tkval =
     ctxo.tkval +
     T0.out(1).tkval
and verrec(T0.out(1))
```

needs ref to **sibling**

T2

```
out(1):{ op: J, own: A, tkval: 10, scr: FT }
```

Neighbourhood covenants: checking siblings

T0

```
out(1):{ op: ...
         own: B,
         tkval: 8
         scr: FT }
```

T1

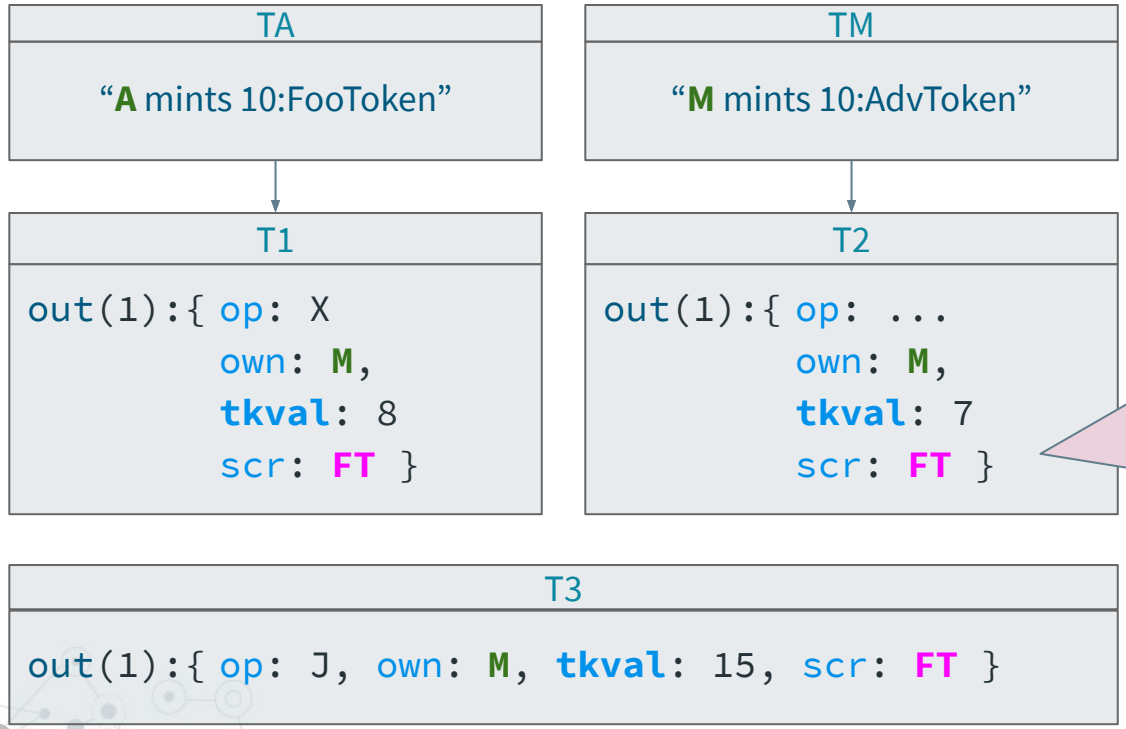
```
out(1):{ op: ...
         own: B,
         tkval: 2
         scr: FT }
```

T2

```
out(1):{ op: J, own: A, tkval: 10, scr: FT }
```

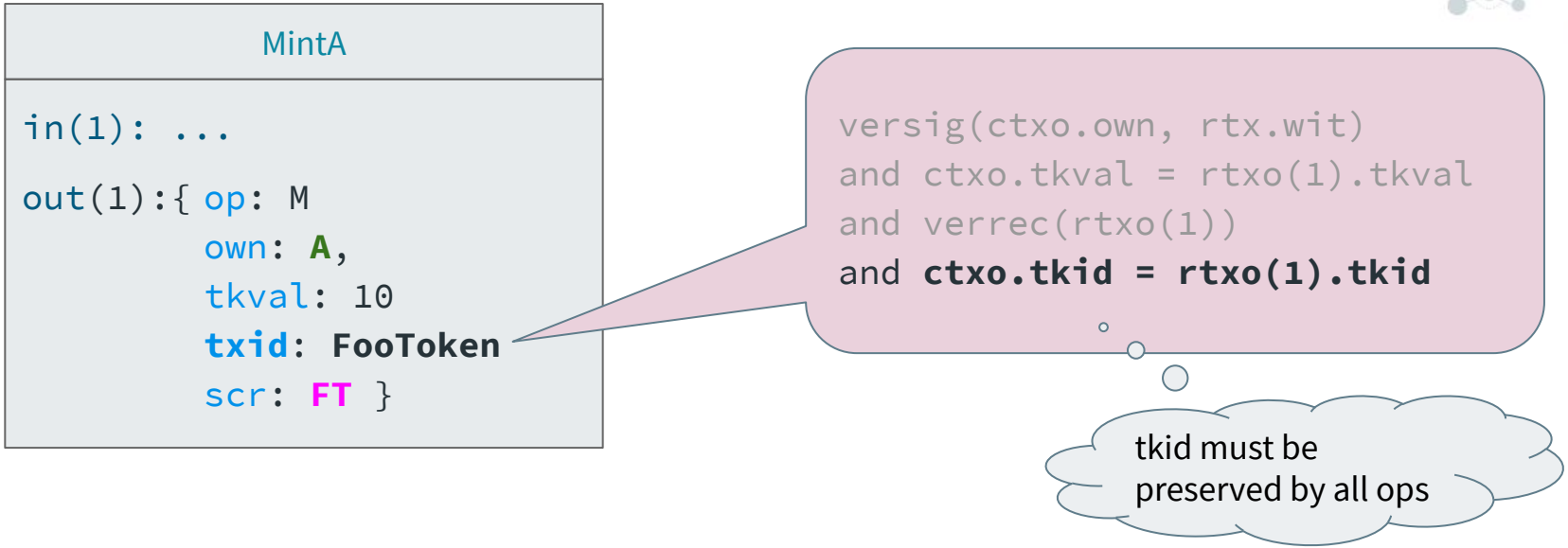
```
versig(ctxo.own, rtx.wit)
and inlen(rtx) = 1
and outlen(rtx) = 1
and verrec(rtxo(1))
and rtxo(1).tkval =
     stxo(1).tkval +
     stxo(2).tkval
and verrec(stxo(1))
and verrec(stxo(2))
```

Attack: joining different tokens

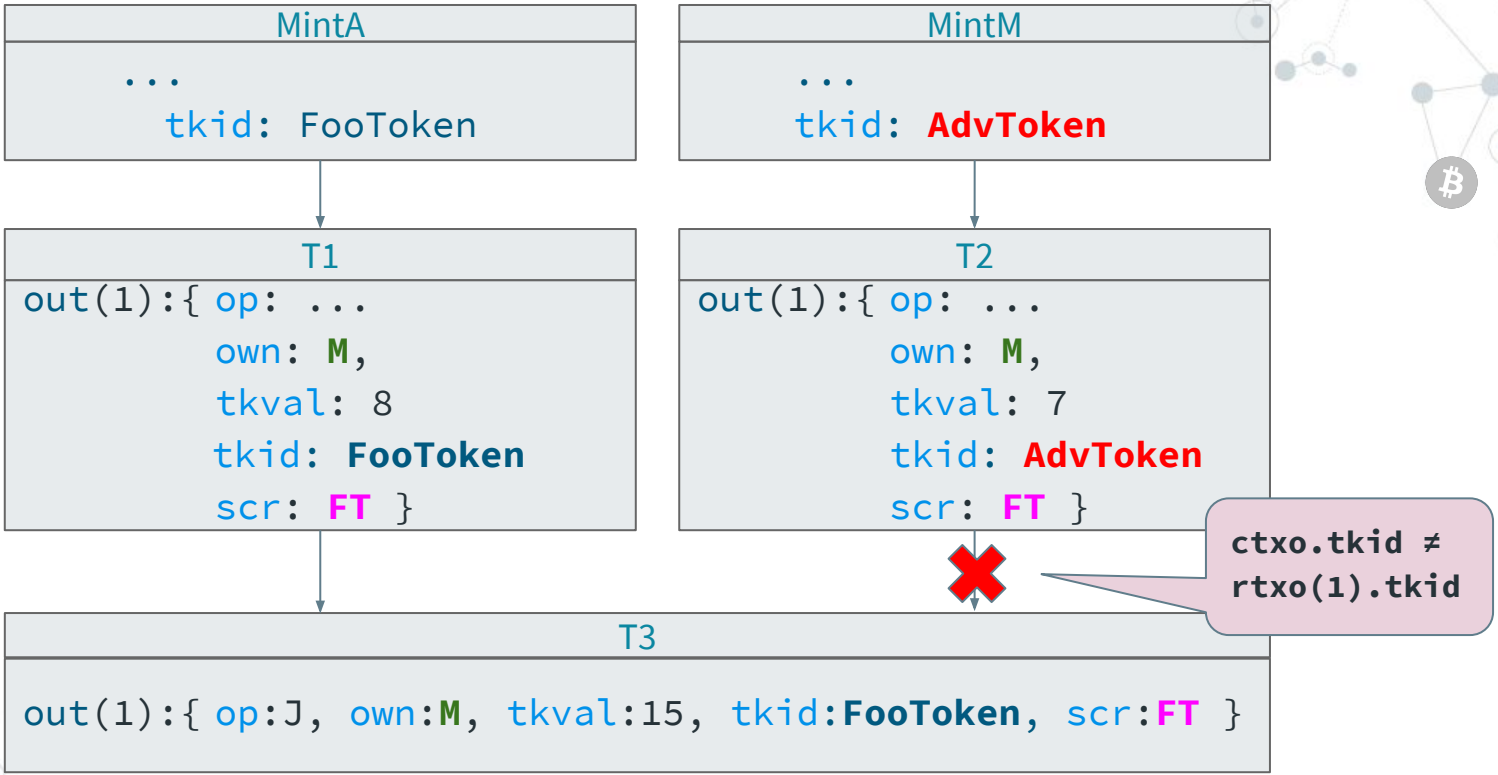



```
versig(ctxo.own, rtx.wit)
and inlen(rtx) = 1
and outlen(rtx) = 1
and verrec(rtxo(1))
and rtxo(1).tkval =
     stxo(1).tkval +
     stxo(2).tkval
and verrec(stxo(1))
and verrec(stxo(2))
```

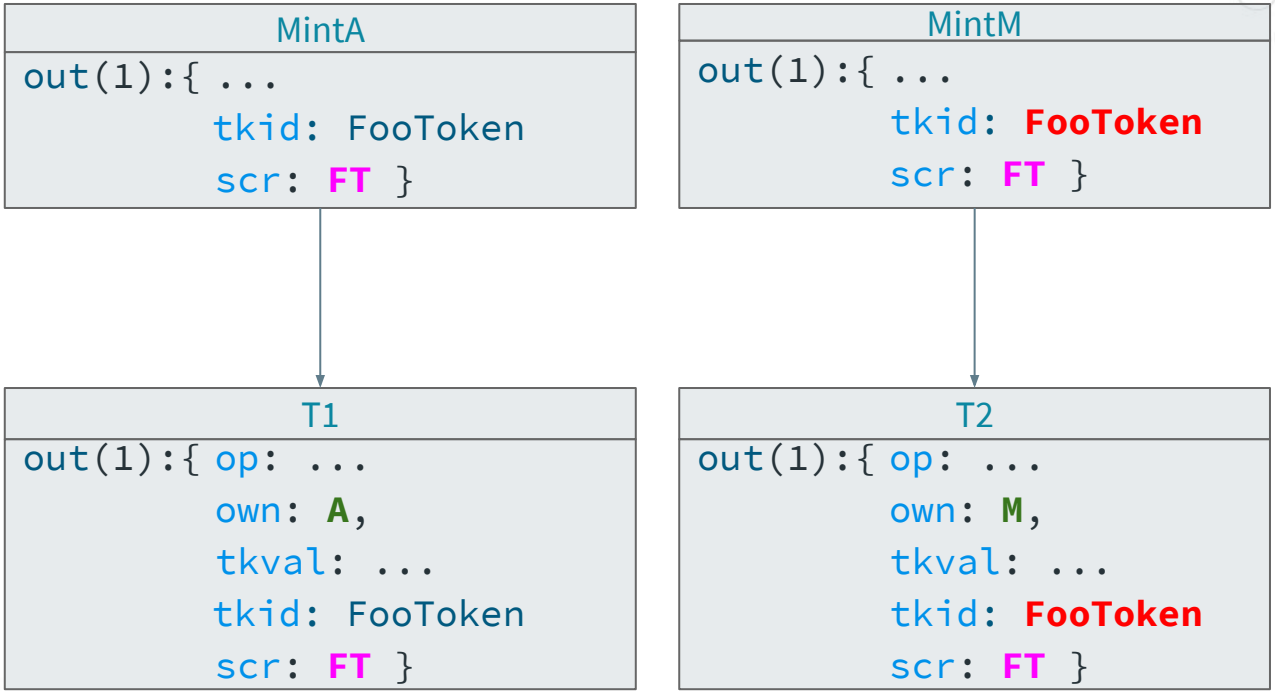
Preserving transaction identifiers



Thwarting the attack



Attack: forging tokens



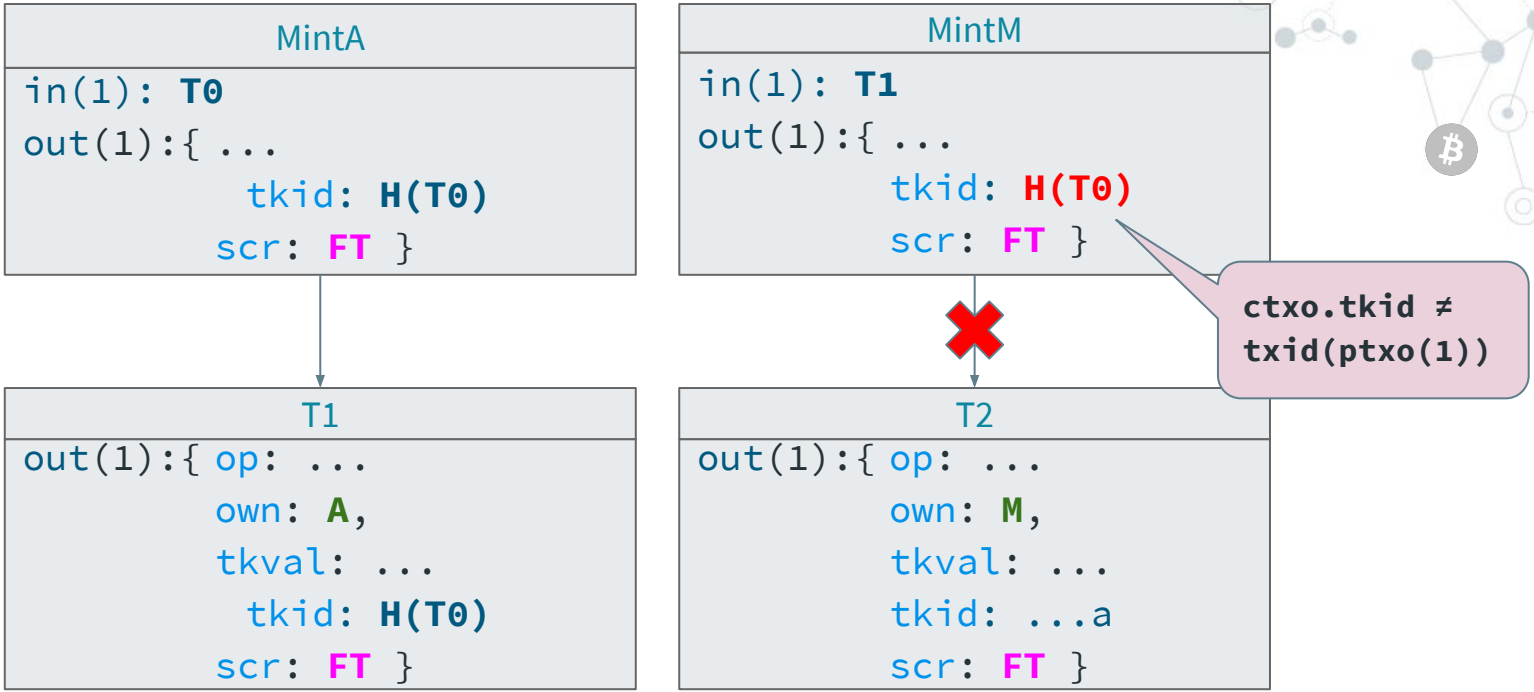
Linking tx identifiers to the token generator

MintA

```
in(1): T0

out(1):{ op: M
         own: A,
         tkval: 10
         txid: H(T0)
         scr: FT }
```

```
versig(ctxo.own, rtx.wit)
and ctxo.tkval = rtxo(1).tkval
and verrec(rtxo(1))
and ctxo.tkid = rtxo(1).tkid
and ctxo.tkid = txid(ptxo(1))
```

Thwarting the attack



Computational soundness

Theorem: any execution in the computational model (Bitcoin)

has a corresponding execution in the symbolic one*

Therefore, the Bitcoin encoding of fungible tokens has **no attacks**

(otherwise, these attacks would have been observable in the symbolic model)

* (with overwhelming probability)

Conclusions

- Covenants increase the expressive power of Bitcoin contracts
- “Forward” covenants not enough for efficient fungible tokens:
  - split & join operations hide security threats!!
  - **neighbourhood covenants** can also inspect siblings & parent
  - minimal impact on the implementation & efficiency of Bitcoin
  - computational soundness of fungible tokens

Thank you! Questions?