Formalizing Correct-by-Construction Casper in Coq

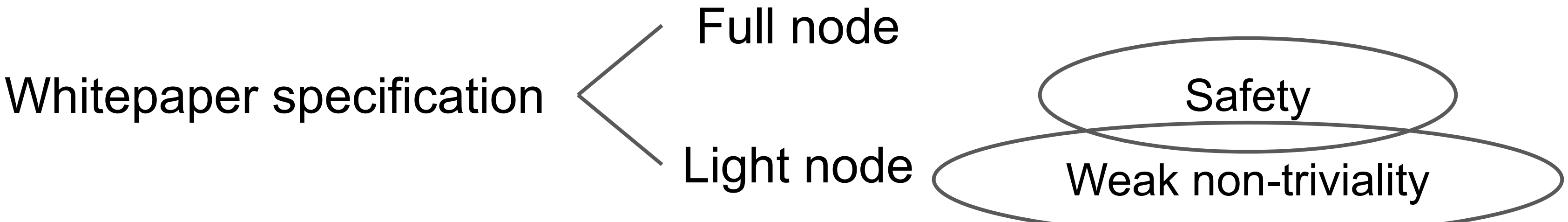
Elaine Li, Traian Șerbănuță, Denisa Diaconescu, Vlad Zamfir, Grigore Rosu

runtime verification

ethereum foundation

Correct-by-Construction (CBC) Casper

- CBC Casper is a *partial* specification of a family of consensus protocols with five parameters: consensus values, estimator, validators, validator weights, fault tolerance threshold
- Each CBC Casper family member shares the same proofs of protocol properties: safety and non-triviality
- https://github.com/cbc-casper/cbc-casper-paper/blob/master/cbc-casper-paper-draft.pdf



Whitepaper specification

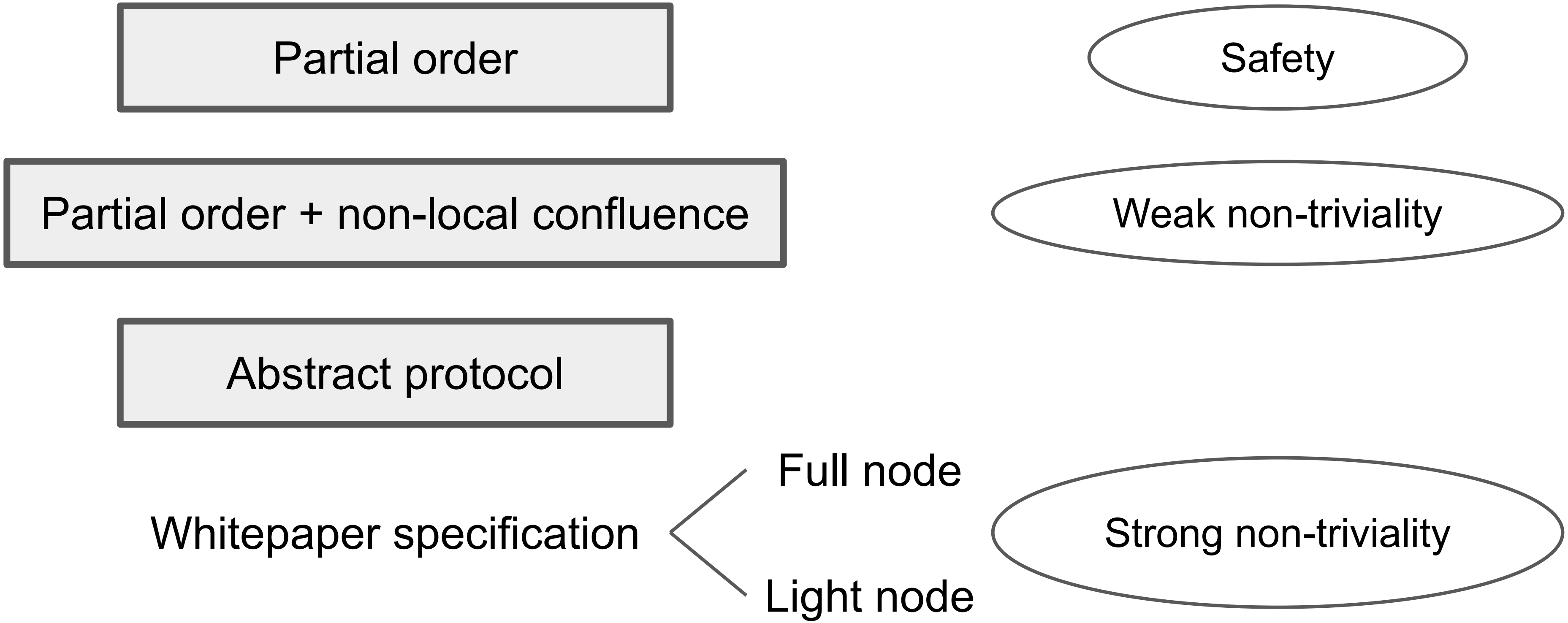
Full node

Light node

Safety

Weak non-triviality

Formalization approach: abstraction hierarchy



Partial order

Safety

Partial order + non-local confluence

Weak non-triviality

Abstract protocol

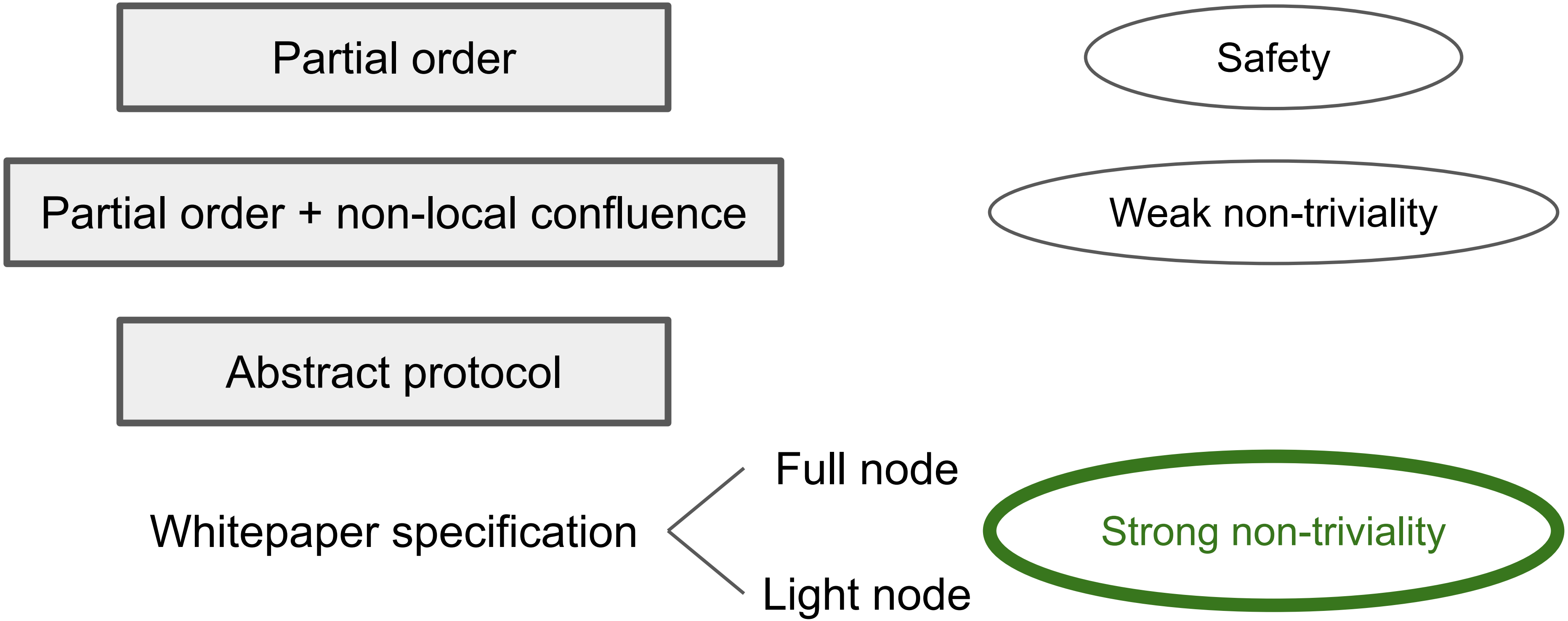
Whitepaper specification

Full node

Light node

Strong non-triviality

Formalization approach: abstraction hierarchy



Partial order

Safety

Partial order + non-local confluence

Weak non-triviality

Abstract protocol

Whitepaper specification

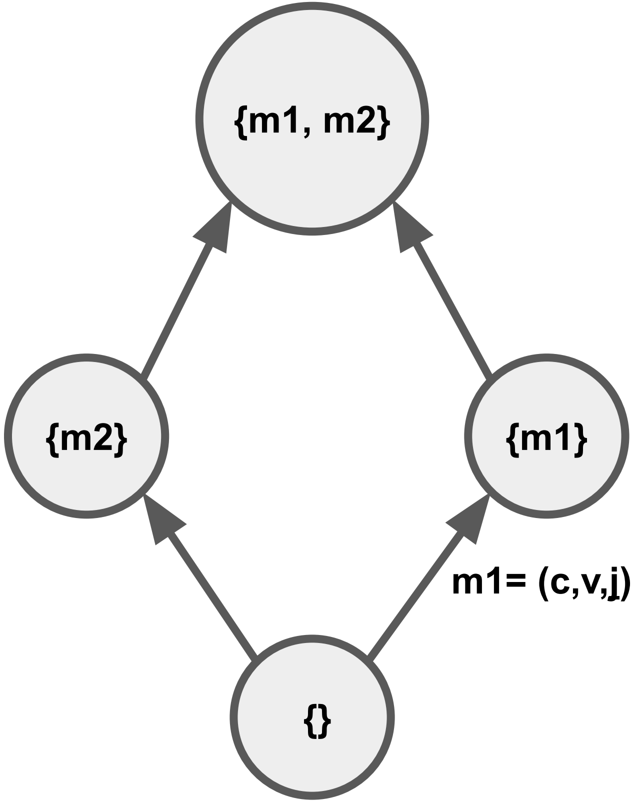
Full node

Light node

Strong non-triviality

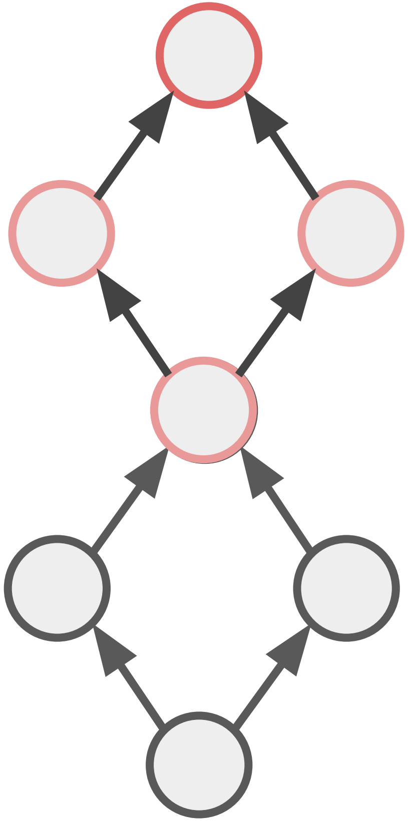
Mutually recursive states and messages

- States are sets of messages
- Messages contain states
- State transition is equivalent to set inclusion
- Future states are equivalent to set union

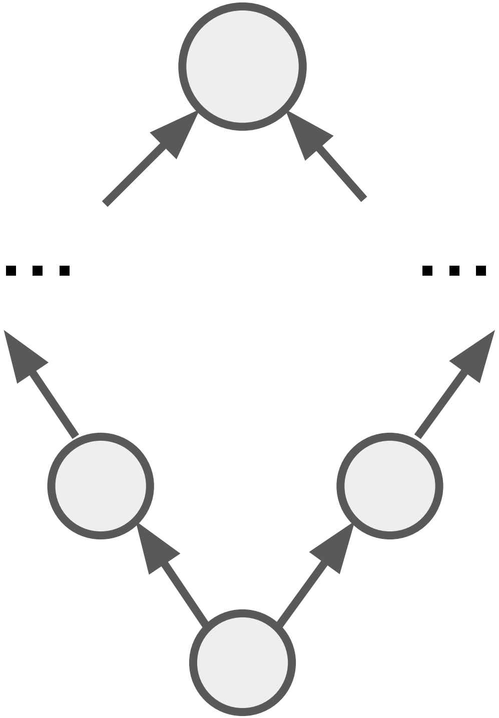


Fault weight increasing

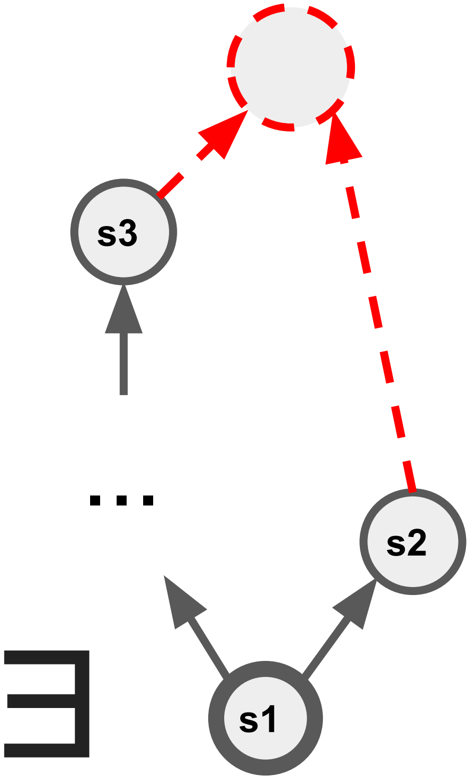
- Validators have weights
- Validators “fault” by sending “equivocating” messages
- Fault weight of a state: sum weight of faulty senders
- Protocol states must stay within the fault tolerance threshold



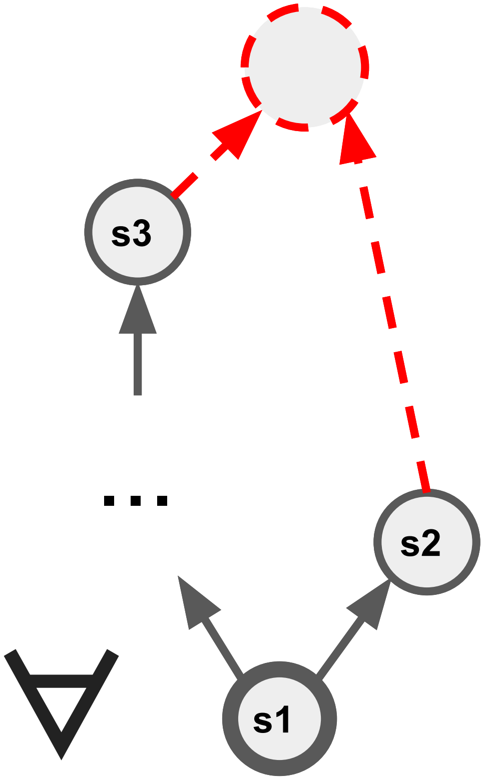
Safety properties, in pictures



Weak non-triviality

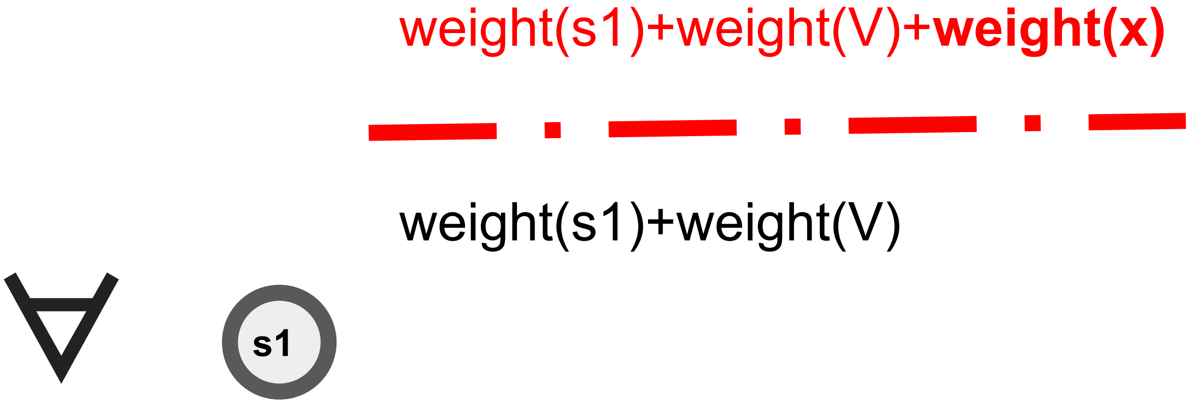


Strong non-triviality



Strong non-triviality

- **w** - current fault weight
- **T** - fault tolerance threshold
- Every state has a set of validators **V** and a *pivotal* validator **x**

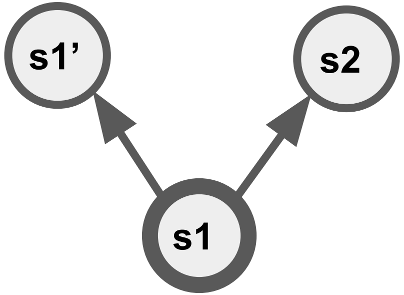


weight(s1)+weight(V)+**weight(x)**

weight(s1)+weight(V)

∀

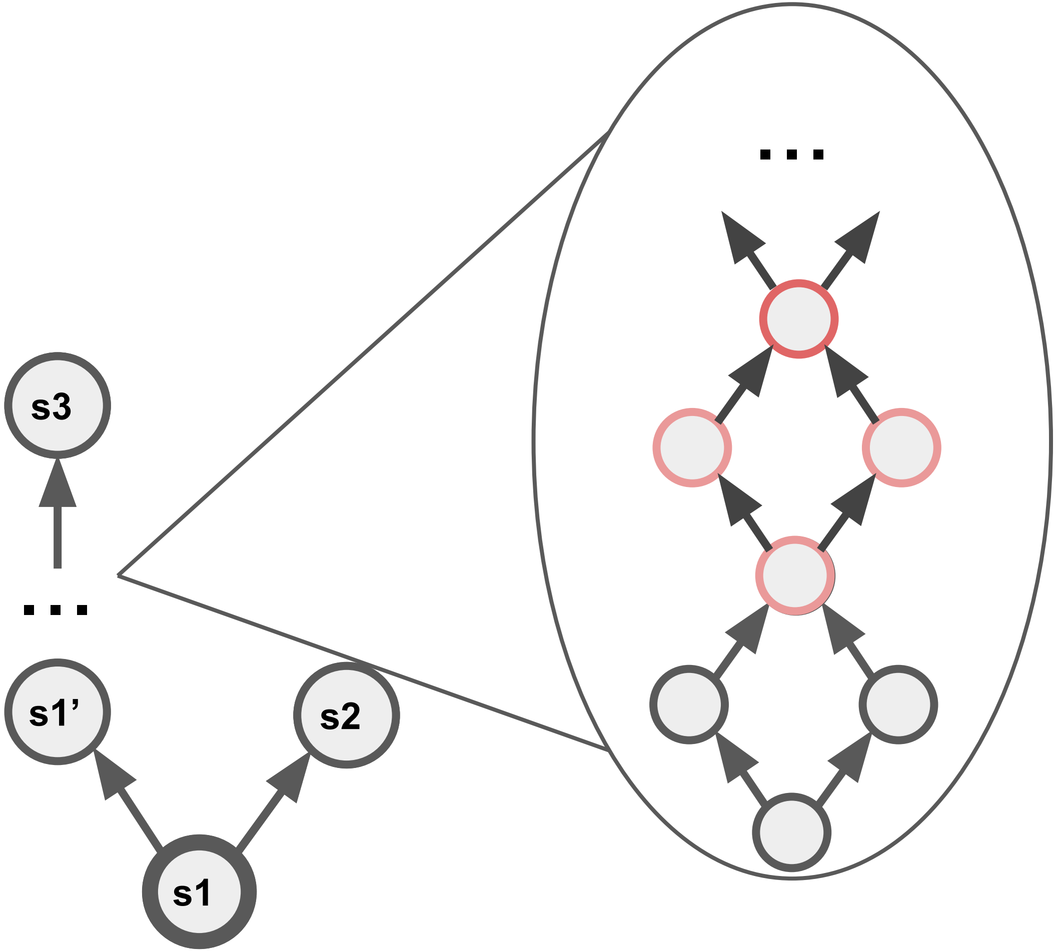
Strong non-triviality



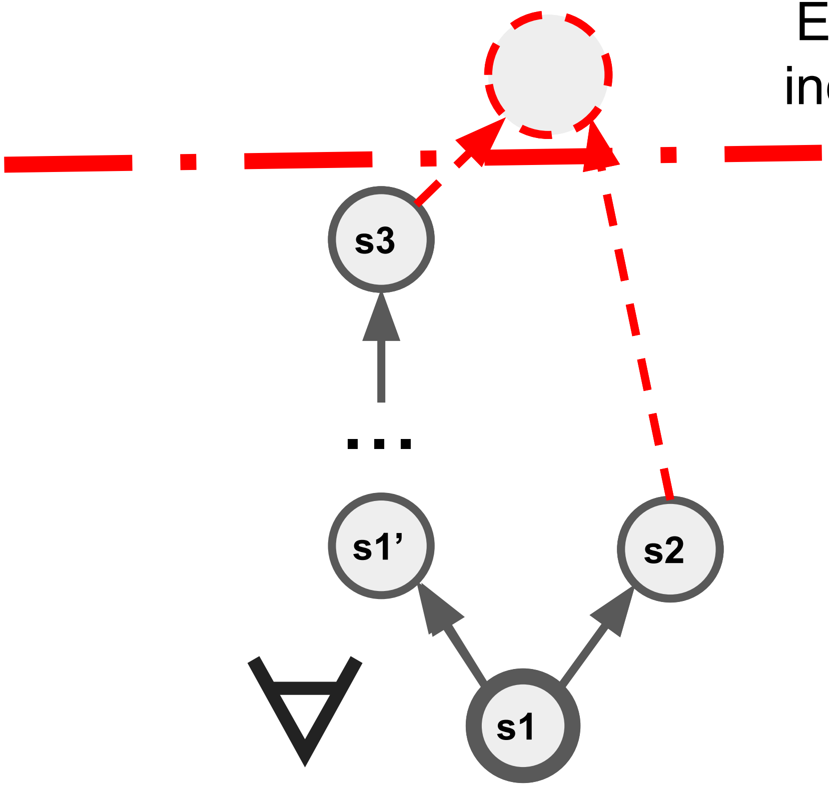
Pair of equivocating messages
sent by pivotal sender **x**

Strong non-triviality

Pairs of equivocating messages
sent by the remaining validators in **V**



Strong non-triviality



Equivocation weight includes weights in **V** and weight of **x**

Formal verification takeaways

- Proof engineering: Coq type classes are a suitable mechanism for abstraction
- Protocol engineering: formal approach fosters better understanding of the protocol

Formal verification takeaways

- Proof engineering: Coq type classes are a suitable mechanism for abstraction
- Protocol engineering: formal approach fosters better understanding of the protocol

Thank you!