VLSM: Validating Labelled State Transition and Message Production Systems

Vlad Zamfir[ξ,*,ψ]

Mihai Calancea[*], Denisa Diaconescu[*,†], Brandon Moore[*],
Karl Palmskog[*,‡], Traian Florin Șerbănuță[*,†], Michael Stay[§]

[*]Runtime Verification, Inc.
[ξ]Ethereum Foundation
[†]Research Center for Logic, Optimization and Security (LOS), University of Bucharest
[‡]KTH Royal Institute of Technology
[§]Pyrofex Corp
[ψ]Lest

Abstract

In this paper we introduce the notion of a validating labelled state transition and message production system (VLSM), a tool for formal analysis and specification of faulty distributed systems. We give theories of VLSM composition, state equivocation, message equivocation, and validators. Then we show that the traces of equivocation-limited compositions of validators are precisely the traces of honest components in the context of limited Byzantine faults in an asynchronous network without a guarantee of eventual message arrival. This result shows that the validation strategies of honest nodes can restrict the effect that Byzantine nodes have on them enough that we can express all of their possible faulty traces in terms of equivocation over valid transitions, circumventing the need for Byzantine fault analysis altogether. Our definitions and results have been formalised and machine-checked in the Coq proof assistant.

Contents

1 Introduction

The theory of Byzantine fault tolerance was conceived by Leslie Lamport [4] for describing the problems that distributed systems face in the context of adversarial faults. Byzantine faulty components in a distributed systems might behave in an arbitrary way in order to disrupt the operations of the system. Byzantine fault tolerance has long been considered to be the most fault tolerance possible, because these faults can exhibit any behaviour whatsoever. Satoshi Nakamoto relied on these concepts in order to describe the fault tolerance of Bitcoin, when using the concept of "honest" and "dishonest" nodes [7]. However the notion that some nodes are honest and some nodes are Byzantine faulty is divorced from the economic realities of Bitcoin. In an economic model we would prefer to assume that every actor is able to modify the code of their component or somehow induce it to commit faulty behaviour, but we would like to imagine that they would do it as a strategic choice. On the other hand, it may be very difficult to provide guarantees in a context where all components are strategically faulty and it may be realistic that only some nodes would be strategic, and therefore mixed models have been proposed [1].

In the Byzantine fault tolerant consensus protocol literature, an "equivocation" refers to inconsistent messages sent by a Byzantine node in order to fool honest, protocol-following nodes into making inconsistent decisions [2, 6, 3]. This paper introduces a more general theory of equivocation, including state equivocations where a component splits off a parallel copy of itself, and message equivocations where nodes receive messages that haven't been sent in the current trace of the system. In consensus protocols, it is common for components to "validate" the received messages in order to be sure that malformed messages are not received. We formalise this idea into a general formal notion of "validators" for a particular system. We are then able to show that the effect that Byzantine nodes can have on honest validators is no different than the effect equivocating validators can have on non-equivocating validators, in the context of a distributed system without synchronization assumptions.

This paper presents these results in terms of a formal composable model of faulty distributed systems called *validating labelled state transition and message production systems*, VLSMs for short. VLSMs were derived in the course of research on the CBC Casper consensus protocols [10], as a tool for specifying both protocols and the properties they should satisfy. In particular, they were developed to give a theory that can be used to express and prove the soundness of equivocation detectors, and to express and prove liveness properties. In the process, we conducted the thorough investigation into full node validators and equivocation that is shared here. The end result is a formal framework for describing faults in distributed systems that is able to account for all the influence of Byzantine nodes on honest validators with only the influence of equivocation faulty validators. Replacing Byzantine nodes with equivocating validators forms the foundation for an alternative to Byzantine fault tolerance analysis. This work opens the way for protocol designers to reason precisely about different types of faults and to budget for them separately, and to thereby create more robust consensus protocols than are possible when budgeting for Byzantine faults.

Coq Formalisation

We formalised and checked our theory of VLSMs using the Coq proof assistant. The formalisation[1] is compatible with Coq 8.13 [9] and uses the Coq-std++ library version 1.6.0[2]. We express VLSM concepts primarily using Coq's type classes [8], and represent finite sets as duplicate-free polymorphic lists. The Coq code consists of two parts:

1. A collection of general utility definitions and results, e.g., on polymorphic lists, that extend the Coq standard library and the Coq-std++ library; this part amounts to around 3 thousand lines of code (kLOC) of specifications and 5 kLOC of proof scripts.
2. The VLSM definitions and results, amounting to around 10 kLOC of specifications and 13 kLOC of proof scripts.

The VLSM part uses two axioms known to be consistent with Coq's logic: functional extensionality and a classical logic property required by Coq's real numbers.

Throughout the paper, we will use the symbol ↗ to reference to the corresponding formalisation of a concept or result in Coq. The paper ends with two appendices in which we discuss some differences between the content of the paper and the Coq formalisation.

2 VLSM: A Simple Theory of Faulty Distributed Systems

In our abstract framework, a validating labelled state transition and message production system will play the role, for example, of a node or a subsystem in a distributed system. The reader may note a familial resemblance between VLSMs and labeled transition systems or I/O automata used in distributed systems [5].

Throughout this paper we will use the symbol $\otimes$ to stand for *no message*. For any set M of messages, we will refer to the set $M \cup \{\otimes\}$ as *the optional messages*. We call a message *proper* if it is not $\otimes$.

Definition 2.1 (↗). A **validating labelled state transition and message production system** (**VLSM**, for short) is a structure of the form

$$\mathcal{V} = (L, S, S_0, M, M_0, \tau, valid),$$

where

- L is a set of **labels** for transitions,
- S is a non-empty set of **states**,
- $S_0 \subseteq S$ is a non-empty set of **initial states**,
- M is set of **messages**,
- $M_0 \subseteq M$ is a set of **initial messages**,

[1] https://github.com/runtimeverification/vlsm/releases/tag/v1.0
[2] https://gitlab.mpi-sws.org/iris/stdpp/

- $\tau : L \times S \times (M \cup \{\otimes\}) \to S \times (M \cup \{\otimes\})$ is a labelled state transition and message production function (**transition function**, for short),
- $valid \subseteq L \times S \times (M \cup \{\otimes\})$ is a predicate describing the **valid inputs** for the transition function.

Note that the transition function is defined as a total function; the validity predicate filters out some of the inputs for the transition function making it partial indirectly. The labels allow a VLSM to have a deterministic behaviour. However, it is possible to have multiple parallel transitions between two states, each with their own label, and to have some labels be invalid.

We will write τ^s to denote the first projection of τ on states and τ^m for the second projection on optional messages, i.e., if $\tau(s, m, l) = (s', m')$, then $\tau^s(l, s, m) = s'$ and $\tau^m(l, s, m) = m'$. We denote a transition $\tau(l, s, m) = (s', m')$ by

$$s \ \xrightarrow[m \,\to\, m']{l}\ s'.$$

Example 2.1. Let us consider a VLSM $\mathcal{C}$ which simply counts how many steps it has taken. This VLSM has a single label c, the set of states is the set of natural numbers, 0 as initial state, no messages and initial messages, no restrictions in the *valid* predicate, and for each natural number n, there is a transition

$$n \ \xrightarrow[\otimes \,\to\, \otimes]{c}\ n + 1.$$

Example 2.2. Let us consider a VLSM $\mathcal{D}$ that counts down from some natural number. $\mathcal{D}$ has a single label d, the set of states to be pairs of integers $\langle p, q\rangle$, initial states to be pairs of the form $\langle p, p\rangle$ with $p \geq 0$, no messages and initial messages, and for any integers p, q there is a transition

$$\langle p, q\rangle \ \xrightarrow[\otimes \,\to\, \otimes]{d}\ \langle p, q - 1\rangle.$$

The *valid* predicate is defined as

$$valid = \{(d, \langle p, q\rangle, \otimes) \mid p \geq q \geq 1\}.$$

Definition 2.2. For a given VLSM $\mathcal{V}$, a transition $\tau(l, s, m) = (s', m')$ is a **valid transition** if $valid(l, s, m)$ holds. We denote a valid transition $\tau(l, s, m) = (s', m')$ by

$$s \xrightarrow[m \,\to\, m']{l} s'.$$

The valid predicate of a VLSM uniquely determine a set of reachable states and messages which we will call *valid states* and *valid messages*, respectively. Note that we consider $\otimes$ to be always valid. Formally, these notions are defined by the following fix point construction.

Definition 2.3 (↗)**.** Let $\mathcal{V} = (L, S, S_0, M, M_0, \tau, valid)$ be a VLSM. The sets $S_{\mathcal{V}}$ of **valid states** and $M_{\mathcal{V}}$ of **valid messages** associated with $\mathcal{V}$ are defined by:

$$S_{\mathcal{V}} = \bigcup_{n=0}^{\infty} S_{\mathcal{V},n} \qquad \text{and} \qquad M_{\mathcal{V}} = \bigcup_{n=0}^{\infty} M_{\mathcal{V},n},$$

where

$$S_{\mathcal{V},0} = S_0, \qquad M_{\mathcal{V},0} = M_0 \cup \{\otimes\},$$

$$S_{\mathcal{V},n+1} = S_{\mathcal{V},n} \cup \bigcup_{\substack{l \in L \\ s \in S_{\mathcal{V},n} \\ m \in M_{\mathcal{V},n} \\ valid(l,s,m)}} \{\tau^s(l, s, m)\}, \qquad M_{\mathcal{V},n+1} = M_{\mathcal{V},n} \cup \bigcup_{\substack{l \in L \\ s \in S_{\mathcal{V},n} \\ m \in M_{\mathcal{V},n} \\ valid(l,s,m)}} \{\tau^m(l, s, m)\}.$$

A **trace** is a succession of valid transitions which starts in an initial state. A **valid trace** is a trace which accepts only valid messages. Note that in a valid trace, all states and messages involved are valid. We use the following notation to indicate a trace t with transition labels $(l_1, l_2, ..., l_n)$ while receiving possible messages $(m_1, m_2, ..., m_n)$, and producing possible messages $(m'_1, m'_2, ..., m'_n)$:

$$s_0 \xrightarrow[m_1 \to m'_1]{l_1} s_1 \xrightarrow[m_2 \to m'_2]{l_2} \dots \xrightarrow[m_n \to m'_n]{l_n} s_n.$$

Terminating traces are those traces that end in states out of which there is no valid transition. **Complete traces** are those traces which are either terminating or infinite.

In general, it is undecidable whether a state is a valid state or whether a message is a valid message. For example, we can take the set of states to be the configurations of a Turing machine, and have valid transitions be the updates of the Turing machine's state. Since the halting problem is undecidable, we cannot know for an arbitrary Turing machine whether it will reach that state. We can also say that the VLSM emits a valid message if and only if the Turing machine halts, which makes the question of whether a message is a valid one undecidable in general. However, we often make extra assumptions about the shape of states and messages that enable us to decide on this problem.

3 Composition of VLSMs

There is a natural way to compose VLSMs with the same set of messages. Composition of VLSMs will allow us, for example, to model distributed networks.

Definition 3.1 (). Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs over the same set of messages M. The **free VLSM composition** of this family is the VLSM

$$\sum_{i=1}^{n} \mathcal{V}_i = (L, S, S_0, M, M_0, \tau, valid),$$

where

- $L = \bigcup_{i=1}^n \{i\} \times L_i$ is the disjoint union of labels,
- $S = \prod_{i=1}^n S_i$ is the product of states,
- $S_0 = \prod_{i=1}^n S_{i,0}$ is the product of all initial states,
- M is the same set of messages as for each VLSM in the family,
- $M_0 = \bigcup_{i=1}^n M_{i,0}$ is the union of all initial messages,
- $\tau : L \times S \times (M \cup \{\maltese\}) \to S \times (M \cup \{\maltese\})$ is defined component-wise, guided by labels,

$$\tau(\langle j, l_j \rangle, \langle s_1, ... s_n \rangle, m) = \\ \left(\langle s_1, ..., s_{j-1}, \tau_j^s(l_j, s_j, m), s_{j+1}, ... s_n \rangle, \tau_j^m(l_j, s_j, m)\right),$$

- $valid \subseteq L \times S \times (M \cup \{\maltese\})$ is defined component-wise, guided by labels,

$$valid(\langle j, l_j \rangle, \langle s_1, ..., s_n \rangle, m) = valid_j(l_j, s_j, m).$$

The fix point construction from Definition 2.3 treats messages output by any component as allowable inputs, so free VLSM composition allows messages produced from one VLSM to be received by another VLSM. Note, however, that a VLSM may receive a message that was not sent earlier in a trace. This choice makes it easier to define the set of valid messages, which would otherwise depend on the current context. We address receipt of messages from an alternate trace in Section 5.

Definition 3.2 (↗). Let $\sum_{i=1}^{n} \mathcal{V}_i = (L, S, S_0, M, M_0, \tau, valid)$ be the free VLSM composition of an indexed set of VLSMs $\{\mathcal{V}_i\}_{i=1}^{n}$. A **composition constraint** φ is

$$\varphi \subseteq L \times S \times (M \cup \{\maltese\}).$$

The **constrained VLSM composition under** φ of $\{\mathcal{V}_i\}_{i=1}^{n}$ is the VLSM which has the same components as the free composition, except for the *valid* predicate which is further constrained by φ:

$$\begin{aligned}\Big(\sum_{i=1}^{n} \mathcal{V}_i\Big)\Big|_{\varphi} &= (L, S, S_0, M, M_0, \tau, valid_{\varphi}),\\ valid_{\varphi} &= valid \cap \varphi.\end{aligned}$$

If φ is empty, then the two notions of compositions coincide. Note that the constrained VLSM composition potentially has a smaller set of valid states and valid messages than the free VLSM composition.

Example 3.1. Let us consider the VLSM $\mathcal{C}$ from Example 2.1. The free VLSM composition of $\mathcal{C}$ with itself has the set $\{\langle 1, c\rangle, \langle 2, c\rangle\}$ of labels, the set of states to be pairs of natural numbers and the initial state $\langle 0, 0\rangle$, no messages and initial messages, no restrictions on the *valid* predicate, and the transition function

$$\tau(\langle j, c\rangle, \langle n_1, n_2\rangle, \maltese) = \begin{cases} (\langle n_1 + 1, n_2\rangle, \maltese), & \text{if } j = 1 \\ (\langle n_1, n_2 + 1\rangle, \maltese), & \text{if } j = 2 \end{cases}.$$

A composition constraint which ensures that counters stay within 5 of each other can be defined by

$$\varphi_{five} = \{(\langle j, c\rangle, \langle n_1, n_2\rangle, \maltese) \mid |n_1 - n_2| \leq 5\}.$$

Example 3.2. The free VLSM composition of the VLSMs $\mathcal{C}$ and $\mathcal{D}$ from Examples 2.1 and 2.2 has the set $\{\langle 1, c\rangle, \langle 2, d\rangle\}$ of labels, states of the form $\langle n, (p, q)\rangle$ where n is a natural number and p, q are integers, initial states of the form $\langle 0, (p, p)\rangle$ with $p \geq 0$, no messages and initial messages, the transition function

$$\tau(\langle j, l\rangle, \langle n, (p, q)\rangle, \maltese) = \begin{cases} (\langle n + 1, (p, q)\rangle, \maltese), & \text{if } j = 1 \\ (\langle n, (p, q - 1)\rangle, \maltese), & \text{if } j = 2 \end{cases},$$

and the *valid* predicate

$$valid = \{(\langle 1, c\rangle, \langle n, (p, q)\rangle, \maltese)\} \cup \{(\langle 2, d\rangle, \langle n, (p, q)\rangle, \maltese) \mid p \geq q \geq 1\}.$$

A composition constraint which ensures that $\mathcal{C}$ and $\mathcal{D}$ alternate steps and that $\mathcal{C}$ takes the first step can be defined by

$$\varphi_{alt} = \ \{\langle j, l\rangle, \langle n, (p, q)\rangle, \maltese) \mid q + n = p, \text{ if } j = 1, \text{ and } q + n = p + 1, \text{ if } j = 2\}.$$

For example, the following is a terminating trace in $(\mathcal{C} + \mathcal{D})\big|_{\varphi_{alt}}$:

$$\begin{aligned}&\langle 0, (2, 2)\rangle \xrightarrow[\maltese \to \maltese]{\langle 1, c\rangle} \langle 1, (2, 2)\rangle \xrightarrow[\maltese \to \maltese]{\langle 2, d\rangle} \langle 1, (2, 1)\rangle \xrightarrow[\maltese \to \maltese]{\langle 1, c\rangle}\\ &\qquad\qquad \langle 2, (2, 1)\rangle \xrightarrow[\maltese \to \maltese]{\langle 2, d\rangle} \langle 2, (2, 0)\rangle \xrightarrow[\maltese \to \maltese]{\langle 1, c\rangle} \langle 3, (2, 0)\rangle.\end{aligned}$$

4 VLSM Projections

Whereas composition allows us to put VLSMs together, projections allow us to take them apart.

Definition 4.1 (↗). Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs and $\mathcal{V} = \left(\sum_{i=1}^n \mathcal{V}_i\right)\Big|_{\varphi} = (L, S, S_0, M, M_0, \tau, valid)$ their constrained VLSM composition under φ. For any $j \in \{1, \ldots, n\}$, the **jth projection** of $\mathcal{V}$ is the VLSM

$$Proj_j(\Big(\sum_{i=1}^n \mathcal{V}_i\Big)\Big|_{\varphi}) = (L_{\pi_j}, S_{\pi_j}, S_{\pi_j,0}, M, M_{\pi_j,0}, \tau_{\pi_j}, valid_{\pi_j}),$$

where

- $L_{\pi_j} = L_j$ is the set of labels from $\mathcal{V}_j$,
- $S_{\pi_j} = S_j$ is the set of states from $\mathcal{V}_j$,
- $S_{\pi_j,0} = S_{j,0}$ is the set of initial states from $\mathcal{V}_j$,
- M is the same set of messages as for each VLSM in the family,
- $M_{\pi_j,0} = M_{\mathcal{V}}$ is the set of all valid messages of the composition $\mathcal{V}$,
- $\tau_{\pi_j} : L_{\pi_j} \times S_{\pi_j} \times (M \cup \{\boxtimes\}) \to S_{\pi_j} \times (M \cup \{\boxtimes\})$ is defined as τ_j of $\mathcal{V}_j$,
- $valid_{\pi_j} \subseteq L_{\pi_j} \times S_{\pi_j} \times (M \cup \{\boxtimes\})$ is defined as

$$\begin{aligned} valid_{\pi_j}(l, s, m) \quad \text{iff} \quad & \text{there are valid } s' = \langle s'_1, \ldots, s'_{j-1}, s, s'_{j+1}, \ldots, s'_n \rangle, s'' \in S \\ & \text{and valid } m' \in M \text{ such that } s' \xrightarrow[m \,\to\, m']{\langle j,l \rangle} s''. \end{aligned}$$

The jth projection for a free VLSM composition can be obtained as a particular case of the above definition when φ is empty. The *valid* predicate in Definition 4.1 says that the valid inputs for a transition are those that can be lifted into valid transitions in the composition. Note that for an indexed set of VLSMs $\{\mathcal{V}_i\}_{i=1}^n$, the jth projection and $\mathcal{V}_j$ do not usually coincide. The jth projection has as initial messages all valid messages of the composition. Moreover, even though the jth projection has the same states as $\mathcal{V}_j$, potentially it has a different set of valid states than $\mathcal{V}_j$, in particular due to the interactions with other components and a possible composition constraint.

Projection of traces. We recall that a trace is a succession of valid transitions. For a composed VLSM, a *subtrace* is a subsequence of valid transitions from a trace.

Let $\mathcal{V} = \left(\sum_{i=1}^n \mathcal{V}_i\right)\Big|_{\varphi}$ be a constrained VLSM composition. Given a trace tr in $\mathcal{V}$, **the projection on component j of the trace** tr is the subtrace of tr which has all the transitions from the component j and no transitions from any of the other components. It is relatively easy to show that the projection on component j of a valid trace from $\mathcal{V}$ is valid in the jth projection $Proj_j(\mathcal{V})$ ↗.

Projection friendliness. Note that the converse implication doesn't always hold. The constraint φ used in the constrained VLSM composition $\mathcal{V} = \left(\sum_{i=1}^n \mathcal{V}_i\right)\Big|_{\varphi}$ is called **projection friendly for the jth component** if the valid traces of $Proj_j(\mathcal{V})$ are precisely the projections on component j of the valid traces of $\mathcal{V}$ ↗.

Example 4.1. The constrained composition from Example 3.1 is projection friendly as every trace in the projection, including infinite traces, occurs as a full subtrace of a trace of the constrained composition.

Example 4.2. The constrained composition from Example 3.2 is not projection friendly. The VLSM $\mathcal{C}$ can increment so long as the VLSM $\mathcal{D}$ is decrementing, and there is no upper bound on what $\mathcal{D}$'s initial state can be, so $\mathcal{C}$'s final state can be arbitrarily large. Because $\mathcal{C}$'s final state is unbounded, the projection of $(\mathcal{C} + \mathcal{D})\big|_{\varphi_{alt}}$ into $\mathcal{C}$ must have an infinite trace, but $(\mathcal{C} + \mathcal{D})\big|_{\varphi_{alt}}$ itself has no infinite traces. Therefore the infinite trace in the projection is not a full subtrace of a trace in $(\mathcal{C} + \mathcal{D})\big|_{\varphi_{alt}}$.

Projections to a subset of components. Definition 4.1 can be generalised for projecting to a subset J of components. Let us denote this VLSM by

$$Proj_J(\Big(\sum_{i=1}^{n}\mathcal{V}_i\Big)\Big|_{\varphi}).$$

If $J = \{j\}$ for some $j \in \{1,\dots,n\}$, then $Proj_J(\Big(\sum_{i=1}^{n}\mathcal{V}_i\Big)\Big|_{\varphi}) = Proj_j(\Big(\sum_{i=1}^{n}\mathcal{V}_i\Big)\Big|_{\varphi})$. The notions and properties of the projection on a component of a trace and projection friendliness can be generalised for projections to a subset J of components.

5 A Theory of Equivocation

The term "Byzantine behaviour" encompasses all possible behaviours an adversary might attempt in order to disrupt a protocol; we explore the general case of Byzantine behaviour in Section 6. An important subset of Byzantine behaviour is *equivocation*. In the consensus literature, equivocation refers to claiming different beliefs about the state of the protocol to different parts of the network [2, 6, 3]. For example, if a network is trying to come to consensus about the value of a bit, an equivocating node may claim to think the bit is 0 to one part of the network and 1 to another part. In CBC Casper, an equivocating node may issue two blocks, neither of which is in the justification of the other [10].

We present two models of equivocation in the VLSM framework: the *state model* and the *message model*. We further prove that these models are equivalent in an appropriate sense. We consider two scenarios: (1) a fixed subset of components can equivocate, and (2) the set of equivocating components is weight limited.

5.1 Equivocation assumptions

Equivocation occurs on the receipt of a message which has not been previously sent in the current trace; the sender of the message is then said to be equivocating. In this subsection we establish some basic assumptions needed for treating equivocation in the VLSM framework.

The hasBeenSent capability assumption allows us to determine whether a message has been sent in traces leading to a state. A VLSM

$$\mathcal{V} = (L, S, S_0, M, M_0, \tau, valid)$$

has the hasBeenSent capability if there is a predicate *sent* over states and messages, i.e., $sent \subseteq S \times M$, with the following properties:

1. no messages have been sent in the initial states, i.e., $(s_0, m) \notin sent$ for any $s_0 \in S_0$ and $m \in M$,
2. for any state s and message m, $sent(s, m)$ holds if for any valid transition

$$s' \xrightarrow[m_1 \;\to\; m_2]{l} s$$

   leading to s, either $sent(s', m)$ or $m_2 = m$.

Note that, under this assumption, a message is observable in a state s iff it is observable on any trace leading to s.

The channel authentication assumption allows us to determine for each message where was it sent from. An indexed set of VLSMs over the same set of messages $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ has the channel authentication assumption if there is a function $sender : M \to \{1,\dots,n\}$ which maps to each message its unique designated sender.

No equivocation constraint for a family $\{\mathcal{V}_i\}_{i=1}^n$ of VLSMs is a composition constraint for their composition which ensures that components may only receive messages that have been sent in a current trace of the composition. Formally,

$$\varphi_{no\_equiv}(\langle i, l\rangle, \langle s_1, ..., s_n\rangle, m) = sent(s_{sender(m)}, m).$$

Message dependencies assumption assures that messages have a way of expressing their (direct) dependencies; practically, most protocols use cryptographic hashes to refer to other messages. Formally, we will assume that each message depends on a (computable) finite set of messages satisfying that:

Necessity The sender must receive all the (direct) dependencies of a message before producing it.

Sufficiency The sender of a message can produce that message by itself (in isolation) if it is pre-loaded with its dependencies as initial messages

The message dependencies assumption implies that messages form a partially ordered set, where $a < b$ if message b depends on message a (a happens before b). Note that the dependencies of a message need not be transitively closed w.r.t. the happens-before relation.

Note that the message dependencies and channel authentication assumptions together imply **unforgeability**, i.e., the fact that a node cannot by itself create messages depending on messages sent by other nodes that it hasn't received yet.

The full node assumption is a way of limiting the amount of new equivocation when receiving a message. A VLSM $\mathcal{V}$ satisfying the message-dependencies assumption also satisfies the full node assumption if for $\mathcal{V}$ to receive a message, it must be the case that it have received all of its dependencies first.

Under this assumption, the only new equivocation which can be introduced when receiving a message is that of the sender of the message, since the equivocation introduced by its dependencies has already be accounted for.

5.2 State and message-equivocation models for a fixed subset of equivocators

Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs over the same set of messages. We assume that each $\mathcal{V}_i$ satisfies the hasBeenSent capability assumption and the family $\{\mathcal{V}_i\}_{i=1}^n$ satisfies the channel authentication assumption.

Let us fix a subset $I \subseteq \{1, ..., n\}$. Any component $\mathcal{V}_i$ with $i \in I$ can equivocate. We describe bellow the two models of equivocation for this scenario and investigate when they are equivalent.

5.2.1 State-equivocation model

In the state-equivocation model we allow an equivocating component to perform **state-equivocations** by forking itself or spawning new machines. We begin by associating to any VLSM its equivocating VLSM.

Definition 5.1 (). The **equivocator VLSM**

$$\mathcal{V}^e = (L^e, S^e, S_0^e, M, M_0, \tau^e, valid^e)$$

associated to a VLSM $\mathcal{V} = (L, S, S_0, M, M_0, \tau, valid)$ is defined as follows:

- the set of labels is $L^e = \mathbb{N}^* \times (L \cup \{\text{duplicate}\} \cup (\{\text{new\_machine}\} \times S_0))$,

- states are lists of states from S, i.e., $S^e = [S]$[3],

- $S_0^e = \{[s] \mid s \in S_0\}$,

- the same set of messages and initial messages as for $\mathcal{V}$,

- the transition function:

$$\tau^e(\langle i, l\rangle, \sigma, m) = ([\sigma[1], ..., \sigma[i-1], \tau^s(l, \sigma[i], m), \sigma[i+1], ...], \tau^m(l, \sigma[i], m)),$$
$$\tau^e(\langle i, \text{duplicate}\rangle, \sigma, m) = ([\sigma[1], ..., \sigma[i-1], \sigma[i], \sigma[i], \sigma[i+1], ...], ⌧),$$
$$\tau^e(\langle i, (\text{new\_machine}, s_0)\rangle, \sigma, m) = ([\sigma[1], ..., \sigma[i], s_0, \sigma[i+1], ...], ⌧),$$

- the valid predicate:

$$\mathit{valid}^e(\langle i, l\rangle, \sigma, m) = i \le \mathit{len}(\sigma) \wedge \mathit{valid}(l, \sigma[i], m),$$
$$\mathit{valid}^e(\langle i, \text{duplicate}\rangle, \sigma, ⌧) = i \le \mathit{len}(\sigma),$$
$$\mathit{valid}^e(\langle i, (\text{new\_machine}, s_0)\rangle, \sigma, ⌧) = i \le \mathit{len}(\sigma).$$

For an equivocator VLSM, the first component of the labels indicate what copy of the machine will be used for a transition. Using the notion of equivocator VLSM, we can define the state-equivocation model as a composition in which we impose the constraint that components may only receive messages that have been sent in a current trace on the composition.

We can show that if $\mathcal{V}$ satisfies the hasBeenSent capability, then $\mathcal{V}^e$ also satisfies the hasBeenSent capability through the following sent^e predicate:

$$\mathit{sent}^e(\sigma, m) ::= \exists k \in \{1, ..., \mathit{len}(\sigma)\} \mathit{sent}(\sigma[k], m).$$

To simplify presentation we will drop the ^{e} superscript from sent^e.

Definition 5.2. The **state-equivocation model of** $\{\mathcal{V}_i\}_{i=1}^n$ **for the fixed set of equivocators I** is the constrained VLSM composition in which we replace each equivocating component with its equivocator VLSM under the composition constraint that components may only receive messages that have been sent in the current trace. Formally, the state-equivocation model is the VLSM

$$\mathcal{V}_S^I = \Big(\sum_{i=1}^n \mathcal{V}_i'\Big)\Big|_{\varphi_{s\_equiv}} = (L, S, S_0, M, M_0, \tau, \mathit{valid}_{\varphi_{s\_equiv}})$$

where, for any $1 \le i \le n$, $\mathcal{V}_i' = \mathcal{V}_i$ if $i \notin I$ and $\mathcal{V}_i' = \mathcal{V}_i^e$ if $i \in I$, and

$$\varphi_{s\_equiv}(\iota, \langle \sigma_1, ..., \sigma_n\rangle, m) = \mathit{sent}(\sigma_{\mathit{sender}(m)}, m).$$

At any point, each equivocating component can perform a state-equivocation either by making a copy of one of the states or introducing a new initial state. Each copy of an equivocating component can evolve independently, but can only receive messages that appear in the current trace of this new machine.

Note that for the empty set of equivocators, the state-equivocation model coincides with the VLSM composition under the constraint composition which ensures that components may only receive messages that have been sent in a current trace of the composition. Hence, for the empty set of equivocators, the two models coincide, and are equal to the composition using the no message-equivocation constraint. On the other hand, if we take the set of equivocators to be the whole set of indices, we obtain a composition in which each machine can state-equivocate freely, while message-equivocation is still not allowed.

[3]For any set A, we denote by $[A]$ the set of all finite lists over A. For any list l over A, we denote by $\mathit{len}(l)$ the length of l.

Example 5.1. The following is a trace in the state-equivocation model of VLSMs $\mathcal{C}$ and $\mathcal{D}$ from Examples 2.1 and 2.2 in which $\mathcal{D}$ is an equivocator:

$$
\begin{aligned}
\langle 0, [(2,2)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 1,c\rangle} \langle 1, [(2,2)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 2,(1,d)\rangle} \langle 1, [(2,1)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 1,c\rangle} \\
\langle 2, [(2,1)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 2,(1,\text{duplicate})\rangle} \langle 2, [(2,1),(2,1)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 2,(2,d)\rangle} \\
\langle 2, [(2,1),(2,0)]\rangle \xrightarrow[\otimes \to \otimes]{\langle 1,c\rangle} \langle 1, [(2,1),(2,0)]\rangle.
\end{aligned}
$$

5.2.2 Message-equivocation model

Recall from Section 3 that in the VLSM framework, messages are always available for receipt (though it might not be valid to receive them). A **message-equivocation** is the receipt of a message that has not yet been sent during that trace.

Definition 5.3. The **message-equivocation model of** $\{\mathcal{V}_i\}_{i=1}^n$ **for the fixed set of equivocators I** is the VLSM composition under the constraint that the only message-equivocations allowed are those between the equivocating components. Formally, the message-equivocation model is the VLSM

$$
\mathcal{V}_M^I = \Big(\sum_{i=1}^{n} \mathcal{V}_i\Big)\Big|_{\varphi_{m\_equiv}} = (L, S, S_0, M, M_0, \tau, valid_{\varphi_{m\_equiv}}),
$$

where

$$
\varphi_{m\_equiv}(\langle i, l\rangle, \langle s_1, \ldots, s_n\rangle, m) \;=\; sent(s_{sender(m)}, m) \vee (i \in I \wedge sender(m) \in I).
$$

Note that for the empty set of equivocators, the message-equivocation model coincides with the VLSM composition under the no-message-equivocation constraint. On the other hand, if we take the set of equivocators to be the whole set of indices, then the composition constraint always holds, so $\mathcal{V}_M^{\{1,\ldots,n\}} = \sum_{i=1}^n \mathcal{V}_i$, the free composition.

Let $_{\neq}I$ denote the set $\{1, \ldots, n\} \setminus I$ representing the non-equivocating nodes.

Lemma 5.1 (). *φ_{m_equiv} is projection friendly for $_{\neq}I$.*

We call the valid traces of $Proj_{_{\neq}I}(\mathcal{V}_M^I)$ *traces exposed to I fixed equivocation.*

5.2.3 Equivalence between state and message-equivocation models

Let us investigate when the state and message-equivocation models coincide. We begin by analysing the following example.

Example 5.2. Let us consider the state and message-equivocation models for just one VLSM $\mathcal{V}$ which is also an equivocator. The machines $\mathcal{V}_S^I$ and $\mathcal{V}_M^I$ are not equivalent. However, consider only the first state in the list of states maintained by $\mathcal{V}_S^I$. Even though a copy of $\mathcal{V}$ in the list cannot receive a message unless at least one of the copies has sent the message earlier in the trace, $\mathcal{V}_S^I$ has the ability to copy the initial state, run that copy ahead to the point where a message has been produced, then go back to the first copy in the list and have that one receive the message. Therefore, given a trace of $\mathcal{V}_S^I$, the subtrace consisting of all transitions of the form $\tau_I(\langle 1, l\rangle, s, m)$ is a trace of $\mathcal{V}_M^I$, and all traces of $\mathcal{V}_M^I$ arise in this way. It is in this sense that we consider message-equivocation equivalent to state-equivocation.

For any indexed set of VLSMs, we might hope that the traces are the same in the two models for equivocation if we restrict our attention to the first element of each list maintained by a component in I, as happened in Example 5.2. This, however, is not true. While components can receive messages from *any* trace of a message equivocator, state equivocators rely on interacting with other components. If those others do not equivocate, it restricts the state equivocators' behaviour in the future. For example, suppose that we have three components $\mathcal{V}_0, \mathcal{V}_1$, and $\mathcal{V}_2$, where $\mathcal{V}_1$ is an equivocator. $\mathcal{V}_0$ can send either the message a or the message b, but not both. $\mathcal{V}_1$ sends c in response to a or d in response to b. In the message-equivocation model, $\mathcal{V}_2$ can receive both c and d, but in the state-equivocation model, it cannot.

Let us assume that all VLSMs $\{\mathcal{V}\}_{i=1}^{n}$ involved satisfy the full node assumption.

Since a state-equivocator is a collection of copies of the original VLSM, it is clear that each of the copies would satisfy the full-node assumption.

Assuming each component from an indexed family of VLSMs has the full node assumption, then their state-equivocation model also has the full node assumption, as it doesn't introduce new messages.

We can extend this to VLSMs with a composition constraint φ by checking if there is some valid state among the states in the product of the equivocator states. However, under some simple composition constraints, state-equivocation is not equivalent to message-equivocation.

Example 5.3. Consider two VLSMs $\mathcal{V}_0$ and $\mathcal{V}_1$, $\mathcal{V}_0$ being an equivocator. The component $\mathcal{V}_0$ has states s_0, s_1, s_0 being initial, while $\mathcal{V}_1$ has states q_0, q_1, q_0 being initial. There is a single message m and three labels, l_0, l_1 and l_2. The initial state in the composition is (s_0, q_0). Let us consider a composition constraint which allows the composite VLSM to transition under l_0 to (s_1, q_0) or under l_1 to (s_0, q_1), while the state (s_1, q_1) transitions under l_2 to itself and sends the message m. In the message-equivocation model, the state (s_1, q_1) is unreachable and m is not valid. But in the state-equivocation model, the system can evolve as follows and emit m:

$$\langle [s_0], q_0\rangle \xrightarrow[\otimes \,\to\, \otimes]{\langle 1,(1,l_0)\rangle} \langle [s_1], q_0\rangle \xrightarrow[\otimes \,\to\, \otimes]{\langle 1,(1,(\text{new\_machine},s_0))\rangle} \langle [s_0, s_1], q_0\rangle$$
$$\xrightarrow[\otimes \,\to\, \otimes]{\langle 2,l_1\rangle} \langle [s_0, s_1], q_1\rangle \xrightarrow[\otimes \,\to\, m]{\langle 1,(2,l_2)\rangle} \langle [s_0, s_1], q_1\rangle.$$

Therefore, for message and state-equivocation to be equivalent, we cannot support all composition constraints. We can support free compositions (as special case of fixed-equivocation), and we will also examine the case of weight limited equivocation in the next sub-section.

Theorem 5.1. *For any fixed-set of equivocators,*

1. *The projection of a valid trace for the state-model keeping only transitions belonging to the first copy of each equivocator is a valid trace for the message-model.*
2. *Under full node assumptions, each valid trace for the message-model can be "lifted" to a valid trace for the state-model such that its projection keeping only transitions belonging to the first copy of each equivocator is the original trace.*

Proof. To complete the argument presented above, it suffices to show that any message received from a message-equivocation model can be sent by a state-equivocation model and vice versa.

Suppose a component receives a message from a message equivocator. The component must have received all the dependencies of that message by the full node assumption. A state equivocator can start a new machine, then receive all the dependencies in the appropriate order to trigger sending the message. Now consider the converse. The state of a message equivocator is irrelevant to what messages other components are able to receive from it; all that matters is whether there exists some trace on which the message can be sent. We will proceed inductively, showing that a message equivocator can emulate any of the substates in a state equivocator by running through the same subtrace. Since it can emulate any of them individually, the set of valid messages is the closure of the states accessible via the messages it emits. For the base case, suppose the state equivocator's substates are all initial; it can receive any message with no dependencies that is valid for that initial substate and some label. If it emits a message in that transition, then there is a trace by which the message equivocator could send the same message. For the inductive case, suppose that the message equivocator is in the same state as one of the substates of the state equivocator and that the substate transitions under some message along some label. Because the message equivocator is in the same state, it can transition in the same way and produce the same message. □

5.3 State and message-equivocation models for a weight limited subset of equivocators

Most consensus protocols require some bound on the number of equivocating parties. Some generalise away from the number of parties to the *weight* of the parties; this reduces to the count when the weights are all 1.

Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs over the same set of messages, equipped with a function *weight* from the set of components to real numbers. As in the previous subsection, we assume that each $\mathcal{V}_i$ satisfies the hasBeenSent capability assumption and the family $\{\mathcal{V}_i\}_{i=1}^n$ satisfies the channel authentication assumption.

We fix an **equivocator threshold** t. We describe bellow the two models of equivocation for this scenario and investigate again when they are equivalent.

5.3.1 State-equivocation model

Definition 5.4 (↗)**.** The **t-limited state-equivocation model of** $\{\mathcal{V}_i\}_{i=1}^n$ is the constrained VLSM composition in which we replace each component with its equivocator VLSM under the composition constraint that components may only receive messages that have been sent in the current trace and the total weight of the equivocators do not exceed t. Formally, the t-limited state-equivocation model is the VLSM

$$\mathcal{V}_S^{<t} = \Big(\sum_{i=1}^n \mathcal{V}_i^e\Big)\Big|_{\varphi_{s\_equiv}^{<t}} = (L, S, S_0, M, M_0, \tau, valid_{\varphi_{s\_equiv}^{<t}})$$

where

$$\varphi_{s\_equiv}^{<t}(\iota, \langle\sigma_1, ..., \sigma_n\rangle, m) \;=\; sent(\sigma_{sender(m)}, m) \;\wedge \sum_{\substack{k=1 \\ 1<len(\sigma_k)}}^{n} weight(k) < t.$$

In this scenario we allow all VLSMs to state-equivocate, but place a limit on the total weight of equivocators allowed. Note that an equivocator which is not allowed to state-equivocate is essentially no different than the corresponding regular node.

Before we continue, let us state two connections between fixed-set and weight limited state-equivocation models. If the added weight of a set of indices I is limited by the threshold t, then, since equivocator traces have no message-equivocation, so they constitute proof for the validness of all their messages, it follows that any valid trace of $\mathcal{V}_S^I$ is also a valid trace of $\mathcal{V}_S^{<t}$, by simply replacing regular nodes with equivocators which are not allowed to equivocate ↗. Conversely, given tr a valid trace for $\mathcal{V}_S^{<t}$ whose last state is $\sigma = \langle\sigma_1, ..., \sigma_n\rangle$, then, by a similar argument as above (replacing equivocators not allowed to equivocate with regular nodes), tr is a valid trace of $\mathcal{V}_S^I$ where I is the set of proper equivocators of tr which can be computed as $I = \{i \mid 1 < len(\sigma_i)\}$ ↗.

5.3.2 Message-equivocation model

Based on the fixed-set equivocation model, we can define the collection of traces with weight limited message-equivocation by simply taking the union of all valid traces for $\mathcal{V}_M^I$ for all subsets I whose weight is limited by

t. We call them *traces under t-limited equivocation*. It is relatively easy to see that these traces correspond to subtraces of the weight limited state-equivocation model, by the following argument:

1. Given tr_M a valid trace for $\mathcal{V}_M^I$ for a subset I whose weight is limited by t, by Theorem 5.1 there is a trace tr_S valid for $\mathcal{V}_M^I$ corresponding to tr. But, from the remark above, tr_S is also valid for $\mathcal{V}_S^{<t}$.
2. Conversely, given a trace tr_S valid for $\mathcal{V}_S^{<t}$, let $\sigma = \langle \sigma_1, ..., \sigma_n \rangle$ be its final state, and let $I = \{i \mid 1 < len(\sigma_i)\}$ be the equivocating indices for tr_S. We have that the weight of I is limited by the threshold, and, by one of the above remarks, tr_S is also valid for $\mathcal{V}_S^I$. By Theorem 5.1, there is a trace tr_M valid for $\mathcal{V}_M^I$ corresponding to tr_S. Since I is of limited weight, tr_M is with limited equivocation.

In the remainder of this section we will define a VLSM whose valid traces are precisely the traces with limited equivocation described above.

First, note that expressing weight limited equivocation as a constraint for the composition of regular nodes is problematic as such a constraint must detect the amount of equivocation encountered by only looking at the states of the individual components. To understand why this is a non-trivial task, consider the following example.

Example 5.4. Given an initial state $\langle \sigma_1, \sigma_2 \rangle$ and two transitions, one on component 1, receiving nothing, transitioning to σ_1' and producing m_1 and one on component 2, receiving m_1, transitioning to σ_2' and producing nothing. After both transitions have occurred, the new state would be $\langle \sigma_1', \sigma_2' \rangle$. However, if the transitions occurred in the order described there should be no equivocation, while if they occurred in the reverse order, component 1 should be considered as equivocating. Hence in some traces the weight of σ should be 0, while in others it should be the weight of component 1.

In what follows, we will try to present a simple model for weight limited message-equivocation based on annotating states with the set of equivocators observed in the current trace so far, which makes the task of detecting the equivocators for a state trivial.

Given a family $\{\mathcal{V}_i\}_{i=1}^n$ of VLSMs, from its free composition $\sum_{i=1}^n V_i$, we can derive a VLSM $\mathcal{V}_M^{<t}$ which annotates the states of the free composition with sets of equivocators, updates those sets during transitions by adding the sender of the input if the input is equivocating, and further constrains the validity predicate to only accept inputs that lead to states whose set of equivocators are of limited weight. Formally,

Definition 5.5 ( ). Let $\sum_{i=1}^n V_i = (L, S, M, S_0, M_0, \tau, valid)$ be free composition $\{\mathcal{V}_i\}_{i=1}^n$. The **t-limited message-equivocation model of** $\{\mathcal{V}_i\}_{i=1}^n$ is the VLSM

$$\mathcal{V}_M^{<t} = (L_{<t}, S_{<t}, M_{<t}, S_{0,<t}, M_{0,<t}, \tau_{<t}, valid_{<t})$$

with the following components:

- $L_{<t} = L$ is the same set of labels as for the free composition,
- $S_{<t} = \{\langle s, Eqv \rangle \mid s \in S \text{ and } Eqv \subseteq \{1, ..., n\}\}$ consists of pairs of states of the free composition and sets of indices,
- $M_{<t} = M$ is the same set of messages as for the free composition,
- $S_{0,<t} = \{\langle s, \varnothing \rangle \mid s \in S_0\}$ pairs the initial states of the free composition with empty sets of indices,
- $M_{0,<t} = M_0$ is the same set of initial messages as for the free composition,

- $\tau_{<t} : L_{<t} \times S_{<t} \times (M_{<t} \cup \{\boxtimes\}) \to S_{<t} \times (M_{<t} \cup \{\boxtimes\})$ is defined as

$$\tau_{<t}(\iota, \langle s, Eqv\rangle, m) = (\langle s', Eqv'\rangle, m'), \quad \text{where } s' = \tau^s(\iota, s, m),$$
$$m' = \tau^m(\iota, s, m), \text{ and } Eqv' = \begin{cases} Eqv \cup \{sender(m)\}, & \text{if } \neg sent(s, m) \\ Eqv, & \text{otherwise} \end{cases},$$

- $valid_{<t} \subseteq L_{<t} \times S_{<t} \times (M_{<t} \cup \{\boxtimes\})$ is defined as

$$valid_{<t}(\iota, \langle s, Eqv\rangle, m) = valid(\iota, s, m) \wedge weight(Eqv') < t,$$

where $\tau_{<t}(\iota, \langle s, Eqv\rangle, m) = (\langle s', Eqv'\rangle, m')$.

5.3.3 Equivalence between state and message-equivocation models

Note that, as in the case of the fixed-set message-equivocation model, the full-node condition is essential in ensuring the adequacy of the model because whenever a message receipt is an equivocation, the only (possibly new) equivocator must be its sender, as all its dependencies were already received (and equivocations accounted for).

Theorem 5.2. *Under full-node assumptions for all components,*

1. *The projection of a valid trace for the state-model keeping only transitions belonging to the first copy of each equivocator is a valid trace for the message-model.*
2. *Each valid trace for the message-model can be "lifted" to a valid trace for the state-model such that its projection keeping only transitions belonging to the first copy of each equivocator is the original trace.*

Proof Sketch. By the correspondence between traces with limited message-equivocation and the valid traces of the limited state-equivocation model from the beginning of the previous subsection, it suffices to show that the traces obtained from the valid traces of $\mathcal{V}_M^{<t}$ by forgetting the equivocators information in each state are precisely the traces with limited equivocation.

Given a trace with t-limited message-equivocation, i.e., a trace valid for the fixed-set message-equivocation model for some set I of limited weight t, annotating each of its states with the equivocators observed so far in the trace, each such set would be a subset of I, thus its weight would be limited by t, so the thus-obtained trace would be valid for $\mathcal{V}_M^{<t}$.

Conversely, given a valid trace tr_E of $\mathcal{V}_M^{<t}$ and the trace tr obtained from tr_E by forgetting the equivocators information in each state, we can observe the following:

- the set of equivocators is monotonously increasing along trace tr_E and limited by the threshold t,
- for each prefix pre_E of tr_E, the equivocators associated to the last state of pre_E are precisely the ones equivocating in that trace (guaranteed by the to full-node condition).

By induction on the trace we can then show that this trace is valid for $\mathcal{V}_M^I$, where I is the set of equivocators of the last state of tr_E. □

6 A Theory of Validation and Byzantine Faults

Traditionally, consensus literature has defined a Byzantine participant in a consensus protocol to be one with arbitrary behaviour [4]. Sometimes Byzantine nodes have a measure of control over the network, with the ability to delay, duplicate, or drop messages. In the VLSM framework, messages can be received at any time, and they may be received multiple times or not at all. Therefore, we can model a Byzantine node as a VLSM that can send or receive any message at any time.

6.1 Validators

Composition constraints are global constraints, but in real networks, components can only enforce constraints on their own behaviour. Given a global composition constraint φ, a VLSM is a *validator for* φ if it enforces φ locally, in the sense that if a transition would cause its view of the network to violate the global constraint, then it is invalid. If $\mathcal{V}$ is the constrained VLSM composition under φ of a family of VLSMs, we will sometimes refer to a validator for φ as a *validator for* $\mathcal{V}$. The free composition of validators does not necessarily globally satisfy φ, but it is the best one can do in a free composition.

Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs and let

$$\mathcal{V} = \Big(\sum_{i=1}^{n} \mathcal{V}_i\Big)\Big|_{\varphi} = (L, S, S_0, M, M_0, \tau, valid)$$

be their constrained VLSM composition under φ.

Definition 6.1. The component $\mathcal{V}_i$ is a **validator** for $\mathcal{V}$ if for any $l \in L_i$, $s_i \in S_i$, and $m \in M \cup \{\boxtimes\}$, if $valid_i(l, s_i, m)$ holds, then there exists a valid state s in S whose ith component is s_i, m is valid in $\mathcal{V}$, and $valid(\langle i, l\rangle, s, m)$.

Lemma 6.1 (↗). *The component $\mathcal{V}_i$ is a validator for $\mathcal{V}$ if $valid_i(l, s_i, m)$ in $\mathcal{V}_i$ implies that $valid_{\pi_i}(l, s_i, m)$ holds and s_i and m are valid in $Proj_i(\mathcal{V})$.*

Proof. Suppose that if $valid(l, s_i, m)$ holds in $\mathcal{V}_i$, then $valid_{\pi_i}(l, s_i, m)$ holds and s_i and m are valid in $Proj_i(\mathcal{V})$. Then, from the definition of projection, there exists a valid state s in $\mathcal{V}$ such that the ith component of s is s_i and $valid(\langle i, l\rangle, s, m)$. □

We can give an alternative definition for the fact that a component is a validator in terms of transitions.

Definition 6.2 (↗). The component $\mathcal{V}_i$ is a **validator** for $\mathcal{V}$ if any valid transition in $\mathcal{V}_i$ can be lifted to a valid transition in $\mathcal{V}$ which involves only valid states and messages. Formally, $\mathcal{V}_i$ is a validator if for any valid transition in $\mathcal{V}_i$

$$s_i \xrightarrow[m \to m']{l} s'_i,$$

there exist a valid transition in $\mathcal{V}$

$$s \xrightarrow[m \to m']{\langle i,l\rangle} s'$$

such that the ith components of s and s' are s_i and s'_i, respectively, and s, s' and m, m' are valid in $\mathcal{V}$.

To see that the definitions are equivalent, suppose that if $valid_i(l, s_i, m)$ holds, there exists a valid state s in S whose ith component is s_i, m is valid in $\mathcal{V}$, and $valid(\langle i, l\rangle, s, m)$. The conclusion follows from the remark that the target state and message of a valid transition from a valid state on receipt of a valid message are again valid ones. The converse direction easily follows.

6.2 Byzantine nodes and validators

We begin by defining the notion of Byzantine nodes in the VLSM framework.

Definition 6.3 (↗). A **Byzantine VLSM** is a VLSM

$$\mathcal{B} = (L, S, S_0, M, M_0, \tau, valid)$$

such that

- the labels are pairs of messages, $L = M$,
- there is only one state which is also an initial one, $S = S_0 = \{s\}$,
- the set of messages and initial messages coincide, $M_0 = M$,
- the transition function ignores the input message and produces the label message, $\tau(m, s, m') = (s, m)$, for any $m, m' \in M$,
- all inputs are valid, $valid = L \times S \times M$.

If a family of VLSMs satisfies the channel authentication assumption, then the Byzantine nodes of the family can send or receive any message at any time, but it is always attributed correctly ↗.

Theorem 6.1 (↗). *Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs and $\mathcal{V} = \left(\sum_{i=1}^n \mathcal{V}_i\right)\Big|_{\varphi} = (L, S, S_0, M, M_0, \tau, valid)$ their constrained VLSM composition under φ such that each $\mathcal{V}_i$ is a validator for $\mathcal{V}$, for any $i \in \{1, ..., n\}$. Replacing any subset of the components of $\mathcal{V}$ with Byzantine nodes has no effect on the valid traces of $Proj_i(\mathcal{V})$ for those i not in the subset.*

Proof. We form the free composition $\mathcal{V}_F = \sum_{i=1}^n \mathcal{V}_i$. Because the composition is free, for each choice of $i \in \{1, ..., n\}$, we can partition $\mathcal{V}_F$ as $\mathcal{V}_i + \mathcal{V}_{\neq i}$, where $\mathcal{V}_{\neq i}$ is the free composition of the remaining components.

Now suppose that we replace any subset of components of $\mathcal{V}_{\neq i}$ with Byzantine nodes to get the VLSM $\mathcal{V}'_{\neq i}$. It suffices to show that any valid trace of $Proj_i(\mathcal{V}_i + \mathcal{V}'_{\neq i})$ is a valid trace of $Proj_i(\mathcal{V})$.

Suppose there is a valid trace tr in $Proj_i(\mathcal{V}_i + \mathcal{V}'_{\neq i})$ that is not a valid trace in $Proj_i(\mathcal{V})$. It must be the case that there is a transition t in tr which is valid in $Proj_i(\mathcal{V}_i + \mathcal{V}'_{\neq i})$ and not valid in $Proj_i(\mathcal{V})$. This means that $\mathcal{V}_i$ receives some message sent by a Byzantine node that is not valid for $\mathcal{V}$ to receive it. However, it must be valid for $\mathcal{V}_i$ to receive it. Since $\mathcal{V}_i$ is a validator for $\mathcal{V}$, any valid transition of $\mathcal{V}_i$ lifts to a valid transition of $\mathcal{V}$ involving only valid states and messages. We get a contradiction since t is not a valid transition of $\mathcal{V}$. Therefore $Proj_i(\mathcal{V}_i + \mathcal{V}'_{\neq i})$ has no such trace. ☐

6.3 Byzantine vs. equivocating behaviour

In this subsection we will argue that validators in a VLSM composition do not distinguish between byzantine nodes and equivocating nodes as defined in Section 5. Therefore, when analysing the security of a protocol it suffices to consider equivocating nodes.

Let $\{\mathcal{V}_i = (L_i, S_i, S_{i,0}, M, M_{i,0}, \tau_i, valid_i)\}_{i=1}^n$ be an indexed set of VLSMs satisfying the channel authentication assumption.

6.3.1 Fixed set of Byzantine nodes

Let us fix $I \subseteq \{1, ..., n\}$. For any $i \in I$, each component $\mathcal{V}_i$ can be replaced with a Byzantine VLSM $\mathcal{V}_i^B$ defined as in Definition 6.3 with the additional requirement that it must also satisfy the channel authentication assumption (that can be enforced by the *valid* predicate). Assuming that the components $\mathcal{V}_i$ with $i \notin I$ are protocol-following, they will only use messages seen in a current trace. Formally, let us define the constrained VLSM

$$\mathcal{V}_B^I = \left(\sum_{i=1}^n \mathcal{V}'_i\right)\Big|_{\varphi_{Byz}} = (L, S, S_0, M, M_0, \tau, valid_{\varphi_{Byz}})$$

where, for any $1 \leq i \leq n$, $\mathcal{V}'_i = \mathcal{V}_i$ if $i \notin I$ and $\mathcal{V}'_i = \mathcal{V}^B_i$ if $i \in I$, and

$$\varphi_{Byz}(\langle i, l\rangle, \langle s_1, \dots, s_n\rangle, m) = sent(s_{sender(m)}, m) \vee i \in I.$$

Let $_{\neq}I$ denote the set $\{1, \dots, n\} \setminus I$ representing the non-Byzantine nodes.

Lemma 6.2 (). *φ_{Byz} is projection friendly for $_{\neq}I$.*

We call the valid traces of $Proj_{_{\neq}I}(\mathcal{V}^I_B)$ *traces exposed to I fixed Byzantine behaviour.*

Theorem 6.2 (). *Under the full node assumption, if the components from $_{\neq}I$ are validators for the message-equivocation model for the fixed set of equivocators I, $\mathcal{V}^I_M$, then the traces exposed to I fixed Byzantine behaviour coincide with the traces exposed to I fixed equivocation.*

Proof Sketch. Since a Byzantine node has less restrictions on its behaviour, it follows easily that traces exposed to I fixed equivocation are included into those exposed to I fixed Byzantine behaviour. For the other inclusion, it suffices to show that any valid trace in $Proj_{_{\neq}I}(\mathcal{V}^I_B)$ is a valid trace in $Proj_{_{\neq}I}(\mathcal{V}^I_M)$.

As in the proof of Theorem 6.1, suppose there is a valid trace tr in $Proj_{_{\neq}I}(\mathcal{V}^I_B)$ that is not a valid trace in $Proj_{_{\neq}I}(\mathcal{V}^I_M)$. It must be the case that there is a transition t in tr where one of the components $j \in {_{\neq}I}$ receives some message sent by a Byzantine node that is not valid for $\mathcal{V}^I_M$ and it must be valid for $\mathcal{V}_j$ to receive it. But since $\mathcal{V}_j$ is a validator for $\mathcal{V}^I_M$, any valid transition lifts to a valid transition involving only valid states and messages. Since by assumption t is not a valid transition of $\mathcal{V}^I_M$, we get a contradiction. Therefore $Proj_{_{\neq}I}(\mathcal{V}^I_B)$ has no such trace. □

6.3.2 Weight limited subset of Byzantine nodes

We fix a threshold t. A *trace under t-limited Byzantine behaviour* is a valid trace of the composition $\mathcal{V}^I_B$ for some I whose weight is limited by t.

As in Definition 6.1, we say that a component $\mathcal{V}_i$ is a *validator for the t-limited message-equivocation model*, $\mathcal{V}^{<t}_M$, if for any $l \in L_i$, $s_i \in S_i$, and $m \in M \cup \{\boxtimes\}$, if $valid_i(l, s_i, m)$ holds, then there exists a valid state $\langle s, Eqv\rangle$ of $\mathcal{V}^{<t}_M$ such that the ith component of s is s_i, m is valid in $\mathcal{V}^{<t}_M$, and $valid_{<t}(\langle i, l\rangle, \langle s, Eqv\rangle, m)$.

Theorem 6.3 (). *Under the full node assumption, if all components are validators for the t-limited message-equivocation model, $\mathcal{V}^{<t}_M$, then the possible behaviours of the non-faulty components are the same under t-limited Byzantine behaviour as under t-limited equivocation.*

Proof Sketch. Let tr be a valid trace under t-limited equivocation in $\mathcal{V}^{<t}_M$ and tr' be the trace obtained from tr by forgetting the second component of the states (i.e, the set of equivocators). Let $\mathcal{V}^{Eqv}_M$ be the message-equivocation model for the fixed set of equivocators Eqv, where Eqv is the set of equivocators of tr (obtained as the second projection of the final state of tr). Note that tr' is a valid trace in $\mathcal{V}^{Eqv}_M$ and the weight of Eqv is limited by t. By the same argument in the proof of Theorem 6.2 (which doesn't depend on the assumption about validators for the message-equivocation model for the fixed set of equivocators), tr' is a valid trace in $\mathcal{V}^{Eqv}_B$. Therefore tr' is a trace under t-limited Byzantine behaviour.

Conversely, let tr be a valid trace under t-limited Byzantine behaviour. Therefore there must be an I whose weight is limited by t and tr is a valid trace in $\mathcal{V}^I_B$. Note that tr has no equivocation from the components in $_{\neq}I$.

Note that we cannot apply Theorem 6.2 for this implication since validators for the t-limited message-equivocation model are not necessarily validators for the message-equivocation model for the fixed set of equivocators I.

First, let us show that the projection on $_{\neq}I$ of tr, $tr_{|_{\neq}I}$, is a valid trace in $Proj_{_{\neq}I}(\mathcal{V}_M^{<t})$. As in the proof of Theorem 6.2, suppose that $tr_{|_{\neq}I}$ is not valid in $Proj_{_{\neq}I}(\mathcal{V}_M^{<t})$. It must be the case that there is a transition t in tr where one of the components $j \in {}_{\neq}I$ receives some message sent by a Byzantine node that is not valid for $\mathcal{V}_M^{<t}$ and it must be valid for $\mathcal{V}_j$ to receive it. But since $\mathcal{V}_j$ is a validator for $\mathcal{V}_M^{<t}$, any valid transition lifts to a valid transition involving only valid states and messages. Since by assumption t is not a valid transition of $\mathcal{V}_M^{<t}$, we get a contradiction. Therefore $tr_{|_{\neq}I}$ must be valid in $Proj_{_{\neq}I}(\mathcal{V}_M^{<t})$.

Finally, let us "lift" $tr_{|_{\neq}I}$ to a trace tr' in the free composition of $\{\mathcal{V}_i\}_{i=1}^n$ by completing the missing state components with initial states. Then we obtain a trace tr'' by tagging each of those new states with the equivocators observed in the trace tr' so far (as in the Definition 5.5). Since tr has no equivocation from the components in $_{\neq}I$, it follows that $tr_{|_{\neq}I}$ has no internal equivocation. Therefore tr'' is valid in $\mathcal{V}_M^{<t}$ as all the equivocating components of tr'' are included in I, thus liomited by t. We conclude that $tr_{|_{\neq}I}$ is the projection of a valid trace in $\mathcal{V}_M^{<t}$. □

Note on the full-node assumption. For both Theorem 6.2 and Theorem 6.3 we assumed that the non-byzantine nodes satisfy the full-node assumption. This assumption is needed because we have used a very simple model for byzantine nodes, assuming only channel authenticity. We believe that the full-node assumption could be dropped if the byzantine nodes would additionally satisfy the message-dependencies assumption (and thus, the unforgeability assumption), and the equivocation models are updated accordingly.

7 Concluding Remarks

The goal of this work is to provide foundations for a theory of typed fault tolerance that can replace Byzantine fault tolerance analysis in settings where it is practical to have validators. We have shown that equivocation faults are exactly as expressive as Byzantine faults, when it comes to their influence on equivocation-limited validators. This result means that in an asynchronous network without guarantee of message arrival, that Byzantine behaviour is exclusively equivocation, as far as equivocation-limited validators are concerned. These traces account for the effects of all possible hostile environments that validators might find themselves in, and therefore form a complete basis for defining and limiting other types of faults.

We showed that Byzantine and equivocating behaviour have the same effect on the validators of an equivocation limited VLSM. Limited equivocation does not guarantee that messages are delivered at all. Our full node assumption insists that messages are received after their dependencies, but no other assumption on the timing or order of the arrival of messages was made during the course of this investigation. This leaves it to later work to define and account for synchronization faults. The equivalence between Byzantine faults and equivocation is therefore in an asynchronous network with no guarantees of message delivery. This means that all other faulty behaviour that Byzantine nodes are capable of inducing can be described and defined in terms of the VLSM traces of these validators. Notably, non-equivocating validators are also capable of exhibiting other kinds of faults, while in contrast non-Byzantine nodes are assumed to not be faulty in other ways.

This invites us to explore distributed systems design in different adversarial settings. For example, we can have distinct and separate limits on equivocation faults and synchronization faults. Extending our fault analysis to include synchronization faults in various configurations is left for later work. Future work will also show examples of equivocation limited validators, including consensus protocols that are safe and non-trivial. Consensus protocols that are live and high performance in limited synchronization faults are also left for later work.

References

[1] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. Bar fault tolerance for cooperative services. In *ACM symposium on Operating systems principles*, pages

45–58, 2005.

[2] Allen Clement, Flavio Junqueira, Aniket Kate, and Rodrigo Rodrigues. On the (limited) power of non-equivocation. In *Symposium on Principles of Distributed Computing*, pages 301–308, 2012.

[3] Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. On the price of equivocation in Byzantine agreement. In *Symposium on Principles of Distributed Computing*, pages 309–318, 2012.

[4] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[5] Nancy A. Lynch and Mark. R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, 1989.

[6] Mads F. Madsen and SØren Debois. On the subject of non-equivocation: defining non-equivocation in synchronous agreement systems. In *Symposium on Principles of Distributed Computing*, pages 159–168, 2020.

[7] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2014.

[8] Matthieu Sozeau and Nicolas Oury. First-class type classes. In *International Conference on Theorem Proving in Higher Order Logics*, pages 278–293, 2008.

[9] The Coq Development Team. The Coq Proof Assistant, 2021. https://doi.org/10.5281/zenodo.4501022.

[10] Vlad Zamfir, Nate Rush, Aditya Asgaonkar, and Georgios Piliouras. Introducing the “minimal CBC Casper” family of consensus protocols. https://github.com/cbc-casper/cbc-casper-paper, 2019.