Knowledge in Byzantine Message-Passing Systems I Framework and the Causal Cone

Roman Kuznets[1], Laurent Prosperi[2], Ulrich Schmid[1],
Krisztina Fruzsa[1], and Lucas Gréaux[2]

[1]Embedded Computing Systems, Institute of Computer Engineering, TU Wien
[2]École normale supérieure Paris-Saclay

January 17, 2019
v.1.0.1

Abstract

We[1] present an extension incorporating Byzantine agents into the epistemic runs-and-systems framework for modeling distributed systems introduced by Fagin et al. [FHMV95]. Our framework relies on a careful separation of concerns for various actors involved in the evolution of a message-passing distributed system: the agents' protocols, the underlying computational model, and the adversary controlling Byzantine faulty behavior, with each component adjustable individually. This modularity will allow our framework to be eventually extended to most existing distributed computing models. The main novelty of our framework is its ability to reason about knowledge of Byzantine agents who need not follow their protocols and may or may not be aware of their own faultiness.

We demonstrate the utility of this framework by first reproducing the standard results regarding Lamport's *happened-before* causal relation [Lam78] and then identifying its Byzantine analog representing the necessary communication structure for attaining knowledge in asynchronous systems with Byzantine faulty agents.

[1]Roman Kuznets is supported by the Austrian Science Fund (FWF) project RiSE/SHiNE (S11405), Krisztina Fruzsa is a PhD student in the FWF doctoral program LogiCS (W1255).

Contents

Chapter 1

The Byzantine Message-Passing Framework

1.1 Agents and States

We consider distributed multi-agent systems with agents viewed as abstract processes, which can represent humans, computers, or more basic devices.

Definition 1.1.1 (Agents and nodes)**.** We consider a non-empty finite set $\mathcal{A}$ of **agents**. Without loss of generality, we assume that the agents are numbered: $\mathcal{A} = [\![1; n]\!]$ for some integer $n > 1$.[1] (The case of a distributed system of only one agent is considered degenerate as it does not exhibit typical properties of a distributed system.) **Local timestamps**, or simply **nodes**, are identified by pairs $(i, t) \in \mathcal{A} \times \mathbb{N}$ of an agent i and a **timestamp** t. For a set $X \subseteq \mathcal{A} \times \mathbb{N}$ of local timestamps, we define the set

$$\mathcal{A}(X) := \{i \mid (\exists t \in \mathbb{N})(i, t) \in X\}$$

of involved agents

In our distributed model, non-negative integer timestamps $0, 1, 2, \ldots$ are used exclusively for snapshots of the system's state. All actions are performed during the open intervals in between, called **rounds**: $]0; 1[,]1; 2[,]2; 3[, \ldots$[2] We use the abbreviation $t.5$ to denote the round $]t; t + 1[$. For instance, it is equivalent to discuss one agent initiating the sending of a message at timestamp t and another agent receiving this message by timestamp $t + 1$, on the one hand, or to discuss the message sent and received during the round $t.5$, on the other hand.

The system begins with each agent in one of its *initial states*.

Definition 1.1.2 (Initial states)**.** Each agent $i \in \mathcal{A}$ has a set Σ_i of **local initial states**. A **joint initial state**, or **global initial state**, is a tuple of local initial states from

$$\mathscr{G}(0) := \prod_{i \in \mathcal{A}} \Sigma_i.$$

An agent's state can be modified due to *internal actions* of the agent itself and/or *external events* triggered by the **environment**, represented as a designated agent ϵ, which is *not* considered a member of $\mathcal{A}$. Since we do not assume agents to be of the same type, their sets of initial states, available internal actions, and observable external events may differ. An example of a local internal action is incrementing a local counter. An example of a local external event is receiving input from a motion-detection sensor.

[1] We use the notation $[\![k; m]\!]$ to denote the set of integers from k to m, i.e., $\{i \in \mathbb{N} \mid k \leq i \leq m\}$.

[2] We use the notation $]k; m[$ to denote the set of real numbers strictly between k and m, i.e., $\{x \in \mathbb{R} \mid k < x < m\}$.

Definition 1.1.3 (Local internal actions and events)**.** Int_i denotes the set of all **local internal actions** of agent $i \in \mathcal{A}$. Ext_i denotes the set of all **local external events** it can observe. We use

- $a, a', a'', \ldots, a_1, a_2, \ldots$, for local internal actions,
- $e, e', e'', \ldots, e_1, e_2, \ldots$, etc. for local external events, and
- $o, o', o'', \ldots, o_1, o_2, \ldots, u, u', u'', \ldots, u_1, u_2, \ldots$ as a generic notation for both events and actions.

We consider a message-passing system whereby agents communicate exclusively by messages. Unlike other actions, which may be specific to an agent, messages should be understandable for both the sender and receiver. Hence, we assume one uniform set of messages comprehensible for all agents.

Definition 1.1.4 (Messages)**.** We denote by *Msgs* the (possibly infinite) set of **messages** that agents can send to each other. For any two agents $i, j \in \mathcal{A}$, a message $\mu \in Msgs$ can be

- sent by agent i to agent j (possibly in multiple copies), which constitutes an internal action of i and is recorded[3] in agent i's history as $send(j, \mu_k)$ for the kth copy of the message; we consider $send(j, \mu_0)$ to be the *master copy* and denote it simply by $send(j, \mu)$ in protocols where multiple copies are not necessary;
- received by agent j from agent i, which constitutes an external event for j and is recorded in agent j's history as $recv(i, \mu)$ (note that j is not aware whether multiple copies of this message have been sent by i to j and cannot tell which copy it has received).

Remark 1.1.5 (Channel implementation)**.** We chose not to model communication channels explicitly. The way messages are delivered from one agent to another is governed by the environment ϵ but this process remains a black box with certain postulated guaranteed properties, e.g., we will present tools capable of ensuring that successfully delivered messages arrive in the same round they have been sent. The most important property of our message-passing system is that it is incorruptible. Agents do not have access to it. In particular, Byzantine agents cannot in any way affect message delivery. The most they can do is to send false information, but cannot pretend that this false information comes from a reliable source.

This might be viewed as a restriction on their Byzantine power. Fortunately, the model is flexible enough to also implement agents impersonating other agents. The underlying assumption of our model is that two communicating agents would necessarily recognize each other (this is sometimes referred to as "oral messages"). Thus, in order to mask one's identity, one simply needs to transfer messages through a relay station. Thus, the unfalsifiable identity of the last sender of the message will belong to such a relay station, whereas the identity of the originator of the message would have to be part of the message content and, hence, malleable. In cases where there is no obvious physical manifestation for such a relay station, e.g., for wireless communication, we can still use a relay station to represent the medium that makes the communication possible. Without going into the technical details, we also remark that relay stations can be implemented without the increase in latency in message delivery.

Remark 1.1.6 (Global ids and global view)**.** Allowing multiple copies of the same message serves only one purpose: to enable sending several duplicates of the same message in the same round. Generally, these copies need not be used and, conversely, their use does not have any effect on the sender's behavior. In particular, there is no obligation to use consecutive numbers for copies nor to always use a fresh copy number in following rounds.[4] Thus, despite having a possibility to distinguish all sent messages, agents are under no obligation to do so.

[3]The exception is the case when sending of the message was a Byzantine action. Then the record of sending can be missing or corrupted, in particular, it can look like another action.

[4]However, within one round each new copy requires a fresh number. Otherwise, it will be conflated with the same-numbered copy because messages form a set.

Thus, the same copy k of the same message μ can be sent from i and received by j multiple times with the sent messages $send(j, \mu_k)$ and received messages $recv(i, \mu)$ all looking the same for agents i and j respectively. The environment ϵ, which also plays the role of the delivery system, must, however, be able to distinguish among identical copies of messages with identical contents but sent/received at different times. This is modelled by a **global message identifier** $id \in \mathbb{N}$, or simply **GMI**, which can be compared to a tracking number used by the environment to uniquely identify each message. Agents never observe this GMI.

Further, while agent i only observes messages sent by itself and its own actions and events, the environment should distinguish between a message μ sent to j from i and a message with the same content μ sent to j by another agent i', between action a by i and the same action a by j, between event e observed by i and the same event e observed by j. Thus, the environment represents

- copy k of a message μ sent from i to j and assigned GMI id in the format $gsend(i, j, \mu, id)$ with the information of the copy number k transferred to the GMI id,
- the same copy of the same message when received by j in the format $grecv(j, i, \mu, id)$,
- action a by agent i in the format $A = internal\,(i, a)$,
- event e observed by i in the format $E = external\,(i, e)$.

We will refer to this as the *global* or *environment's view*, as opposed to the *local view* $send(j, \mu_k)$ and $recv(i, \mu)$ of the sending and receiving agents respectively, a of the acting agent, and e of the observing agent. We denote

- globally presented actions by A, A', A_1, etc.,
- globally presented events by E, E', E_1, etc., and
- globally presented either by $O, O', O_1, \ldots, U, U', U_1, \ldots$.

We assume that a'' and A'' or e_{13} and E_{13}, etc. represent the same action/event presented locally and globally respectively.

After we define protocols and the normative, by-the-protocol behavior for agents, we will introduce agents with Byzantine behavior, i.e., behavior defying their protocols.

Remark 1.1.7 (Modelling asynchronous agents)**.** Asynchronous agents do not have access to the global clock of the system. In particular, they should not be able to count the rounds passed from the beginning. This is implemented by letting agents skip one or more rounds completely. For each round, the environment ϵ controls whether an agent is to be awoken to implement its protocol and/or observe some external events or is to skip the round.

This choice for agent i is implemented by three system events: $go(i)$, $sleep\,(i)$, and $hibernate\,(i)$. None of these events are registered by the agents. The $go(i)$ event, unless countermanded by $sleep\,(i)$ or $hibernate\,(i)$ causes the agent to implement its protocol for this round. Even if the protocol prescribes to stand by and do nothing, the agent would still record the passage of time. Independently, the agent records the passage of time whenever it observes any event[5] other than a Byzantine event perceived as the special **no-op** action 🗓.

Events $sleep\,(i)$ and $hibernate\,(i)$ model situations when the agent fails to act due to a malfunction. They differ in whether, despite not acting, the agent is supposed to notice the passing round or not. Note that we are distinguishing between the agent *actively doing nothing* (observing the round passing without any action) and *passively not doing anything* (not being aware of the passing round at all). For either option, we present a possible malfunction resulting in it (in the absence of any actions or events).

[5]It might seem that waking up agents to receive messages interferes with asynchronicity. However, just like $go(i)$, the delivery of messages is controlled by the environment. In particular, the environment can make the agent skip rounds by postponing all message deliveries.

	Correct	Byzantine
Observing the round	$go(i)$	$sleep\,(i)$
Not being aware of the round		$hibernate\,(i)$

Table 1.1: Possibilities regarding agent's actions in a round.

Remark 1.1.8 (Sufficient conditions for message transfer)**.** As can be seen from the preceding remark, it may happen that despite an agent's protocol prescribing it to send a message, this action is thwarted by the environment not waking up the agent. Additionally, a properly functioning agent should not be able to receive a message that was not yet sent (i.e., sent no later than in the same round). This causally motivated restriction on the model will be implemented by a special filter (see Def. 1.2.19) that uses the GMI of the message to distinguish among multiple duplicate messages if need be. This filter can be viewed as part of the environment.

Because of the active role agents play in actions, there are more ways to violate rules while performing them. We discuss Byzantine events and actions separately.

Remark 1.1.9 (Modelling Byzantine events)**.** In the spirit of the distributed paradigm, i.e., in the absence of a global observer, we do not model global events. Even if one event is observed by several agents, we model these observations independently and do not generally postulate the event to be registered by all agents in the same round. In other words, the fact that one agent failed to observe event e or observed it too late, despite another agent having observed it (earlier) does not generally make the first agent's behavior incorrect.[6]

Instead the faults are agent-centric. It is not an event itself that causes the fault, it is the agent's perception of the event (with the exception of system events). Consequently, the only difference we model is between the agent recording an event e that happened to it vs. the agent recording e even though e did not take place. Note that the two versions are incompatible. Consequently, they never happen simultaneously. More precisely, the environment never attempts simultaneously a correct and faulty event that leave the same trace in the agent's local history.

Remark 1.1.10 (Modelling Byzantine actions)**.** Unlike events, which are not controlled by the agent, actions depend on its will. Accordingly, apart from recording its own actions incorrectly, the agent can also perform wrong actions, i.e., a set of actions not envisioned by its protocol. It is also clear that whether the action is correct and whether it is correctly recorded are independent of each other. Accordingly, there are four possibilities:

- a correct action is correctly recorded;
- a correct action is mistaken for another action or possibly not recorded at all;
- an incorrect action is correctly recorded;
- an incorrect action is mistaken for another (possibly correct) action or possibly not recorded.

Clearly, all but the first case represent faulty agents. It is especially important to distinguish which action took place and which action (if any) the agent thinks took place for the action of sending a message. We assume that all messages actually sent, whether correctly or otherwise and independent of whether the agent retains a record of sending it, are treated by the environment in the same way. In particular, the environment always assigns a correct GMI for all messages actually sent meaning that Byzantine agents are not able to deceive the environment as to the identity of their messages or, indeed, as to the Byzantine nature of these messages. However, agent i may, for instance, think that a message was sent to agent j, whereas in fact it was sent to k. The GMI for such a message and all the delivery procedures will correspond to the true state of affairs. In particular, the environment would not be allowed to deliver this message to j

[6] If it is absolutely necessary for correct agents to observe a particular event, it is still possible to model agents guilty of not observing it by creating a special Byzantine event *distracted*.

in a correct manner. Similarly, the agent sending a message without being aware of it is modeled by the send action being mistaken for the special Byzantine **no-op** action 🗓. More generally, any action may remain unregistered by the agent when mistaken for 🗓. Conversely, the agent may believe to have acted despite not doing anything if 🗓 is mistaken for this action. The latter situation still falls under "actively doing nothing," at least from the agent's point of view. Thus, 🗓 is used as a Byzantine action that has no consequences and leaves no record. In particular, if 🗓 is the only action/event of agent i during a round (independently of which actions it erroneously performed in reality), then the local history of the agent remains unchanged, unless $\mathit{sleep}\,(i)$ forces the agent to mark time.

Remark 1.1.11 (The culprit of Byzantaneity)**.** Despite the Byzantine agents acting any way they like, they should be able to reason about their actions the same way as uncorrupted agents. This is especially crucial for agents who are corrupted due to malfunction rather than malfeasance. This need is at the heart of our decision to shift the control of Byzantine behavior from corrupted agents to the environment. Thus, formally, a Byzantine "action" of an agent, including sending a Byzantine message, is not modelled as the agent's action, but is instead an external event imposed on the agent by the environment, an event specifying both which action took place and which action the agent thinks took place. The agent is unable to distinguish between itself performing some action a and the environment imposing some action that looks like a to the agent. In particular, as will be shown later, the only way for the agent to learn that it is corrupted is by observing the discrepancies between its actions and those demanded by its protocol, more precisely, those discrepancies the agent is able to perceive.

Remark 1.1.12 (How environment communicates)**.** Our model does not provide for message passing between an agent and the environment because such messages would be redundant. On the one hand, the environment is considered to be omniscient and already knows everything the agents might want to communicate to it. As for the information flow in the opposite direction, the environment is not viewed as a conscious entity trying to communicate (though, in principle, information can be delivered to an agent from the environment by means of external events).

We now flesh out the formal definitions for the concepts just discussed:

Definition 1.1.13 (Global message identifier function)**.** We fix a function for computing GMIs to be any computable one-to-one total function $id\colon \mathcal{A} \times \mathcal{A} \times \mathit{Msgs} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[7]

Note that the two functions retrieving the arguments t and k from $id(i, j, \mu, k, t)$ respectively would always be computable due to the injectivity and totality of $id(\cdot)$. However, it is beneficial to choose $id(\cdot)$ so as to additionally make these two functions computable efficiently.

Definition 1.1.14 (Internal actions)**.** From the point of view of agent $i \in \mathcal{A}$, its correct **internal actions**, which can be prescribed by its protocol, consist of the *send* actions from Def. 1.1.4 and local internal actions $a \in \mathit{Int}_i$ (Def. 1.1.3):

$$\overline{\mathit{Actions}_i} := \{\mathit{send}(j, \mu_k) \mid j \in \mathcal{A}, \mu \in \mathit{Msgs}, k \in \mathbb{N}\} \sqcup \mathit{Int}_i. \tag{1.1}$$

The same actions from the point of view of the environment look like

$$\overline{\mathit{GActions}_i} := \{\mathit{gsend}(i, j, \mu, \mathit{id}) \mid j \in \mathcal{A}, \mu \in \mathit{Msgs}, \mathit{id} \in \mathbb{N}\} \sqcup \{\mathit{internal}\,(i, a) \mid a \in \mathit{Int}_i\} \tag{1.2}$$

We also define the sets of all (correct) internal actions that at least one of the agents can take:

$$\overline{\mathit{Actions}} := \bigcup_{i \in \mathcal{A}} \overline{\mathit{Actions}_i} \tag{1.3}$$

$$\overline{\mathit{GActions}} := \bigsqcup_{i \in \mathcal{A}} \overline{\mathit{GActions}_i} \tag{1.4}$$

[7] A simple though not necessarily the most efficient possibility is to use $2^i \cdot 3^j \cdot 5^{\lceil \mu \rceil} \cdot 7^k \cdot 11^t$, where $\lceil \mu \rceil$ represents the numerical code of the message μ according to some arbitrary but fixed coding scheme.

Remark 1.1.15. Note that the union in (1.4) is disjoint because each action viewed globally necessarily specifies the acting agent. Even if the same local internal action $a \in Int_i \cap Int_j$ can be performed by several agents, the representations $internal\,(i,a) \neq internal\,(j,a)$ of the action for these agents $i \neq j$ are distinct.

Definition 1.1.16 (External events)**.** From the point of view of agent $i \in \mathcal{A}$, **correct external events** that it can observe consist of the *recv* actions from Def. 1.1.4 and local external events $e \in Ext_i$ (Def. 1.1.3):

$$\overline{Events_i} := \{recv(j,\mu) \mid j \in \mathcal{A}, \mu \in Msgs\} \sqcup Ext_i \tag{1.5}$$

The same events from the point of view of the environment look like

$$\overline{GEvents_i} := \{grecv(i,j,\mu,id) \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external\,(i,e) \mid e \in Ext_i\} \tag{1.6}$$

For each correct external event $E \in \overline{GEvents_i}$ of agent i, there is a matching **Byzantine external event**

$$fake\,(i,E)$$

representing the agent being mistaken about observing the (local version of the) event E. For $A, A' \in \{🗓\} \sqcup \overline{GActions_i}$, each of which is either a correct global action of agent i or **no-op** 🗓, there is a matching **Byzantine external event**

$$fake\,(i, A \mapsto A')$$

representing the situation when the agent performs (the local version of) action A but thinks that it performed (the local version of) action A'. When the agent faithfully records the performed Byzantine action, we abbreviate

$$fake\,(i, A \mapsto A) = fake\,(i,A).$$

Note that $fake\,(i,🗓)$ acts as a malfunction without any action or any trace in the local history. Hence, we abbreviate it as

$$fake\,(i,🗓) = fail\,(i).$$

Correct actions are always recorded faithfully. (It can, however, happen that one or several faulty actions are merged with a correctly performed action in the local history.) We do not impose any *a priory* restrictions on E, A, or A', i.e., a Byzantine agent can mistakenly observe any event, mistakenly perform any action, and mistake any performed action or inaction for any other action or for inaction.

In particular, for received Byzantine messages, the environment creates the message "out of thin air" as if it has been delivered. Such a message will be supplied with a GMI for uniformity's sake but such GMIs is not assumed to carry any information. Indeed, this GMI will play no role whatsoever as it will be stripped from the message upon delivery. It can even violate the uniqueness of GMIs. Similarly, a Byzantine sent message A is created already with a well-formed GMI, unlike the correctly sent messages, which are supplied with a GMI in a separate step after creation. However, the delivery of messages A with GMI does not depend on whether they originate as Byzantine or correct ones. At the same time, if the agent is mistaken about having sent a message A', its GMI is again immaterial as the agent will not see it nor the environment will ever deliver this "message."

We use $fake\,(i,U)$ to denote an arbitrary fake event $fake\,(i,E)$ or action $fake\,(i, A \mapsto A')$ and denote the set of all such Byzantine events of agent i by

$$BEvents_i := \left\{fake\,(i,E) \mid E \in \overline{GEvents_i}\right\} \sqcup \left\{fake\,(i, A \mapsto A') \mid A, A' \in \{🗓\} \sqcup \overline{GActions_i}\right\} \tag{1.7}$$

In addition to correct and Byzantine events, the environment uses system events to determine which agents can act in each round:

- $go(i)$ approves i's actions prescribed by its protocol;
- $sleep\,(i)$ instructs i to forfeit acting in this round but wakes it up anyways, marking it Byzantine;
- $hibernate\,(i)$ instructs i to forfeit acting in this round without waking it, marking it Byzantine.

System events $SysEvents_i := \{go(i), sleep\,(i), hibernate\,(i)\}$ are not (directly[8]) observable by agents and, hence, are not part of $\overline{Events_i}$.[9]

The complete set of events affecting agent i that the environment can trigger is

$$GEvents_i := \overline{GEvents_i} \sqcup BEvents_i \sqcup SysEvents_i \tag{1.8}$$

We also define the sets of external events affecting all agents as

$$\overline{Events} := \bigcup_{i\in\mathcal{A}} \overline{Events_i}, \qquad \overline{GEvents} := \bigsqcup_{i\in\mathcal{A}} \overline{GEvents_i},$$
$$GEvents := \bigsqcup_{i\in\mathcal{A}} GEvents_i, \qquad BEvents := \bigsqcup_{i\in\mathcal{A}} BEvents_i,$$
$$SysEvents := \bigsqcup_{i\in\mathcal{A}} SysEvents_i.$$

Remark 1.1.17. Since action A' in $fake\,(i, A \mapsto A')$ is a complete fiction only existing in i's imagination, we could have written it in the format $fake\,(i, A \mapsto a')$ for the action a' already presented from i's local point of view. The only reason we do not do that is to preserve uniformity and avoid mixing local and global points of view.

Remark 1.1.18 (Modelling Byzantine inaction)**.** As can be seen from Table 1.1, these three events governing actions have the following meanings:

- the event $hibernate\,(i)$ prevents the agent from acting or from waking up other than to observe other events and makes the agent Byzantine;
- the event $sleep\,(i)$ wakes the agent up but prevents it from acting and makes it Byzantine;
- the event $go(i)$ wakes up the agent and prompts it to fulfill its protocol;
- finally, without any of the three events, the agent would not act, would not wake up unless to observe other events and would not become Byzantine unless other events cause it.

Since $hibernate\,(i)$, $sleep\,(i)$, and $go(i)$ are generally pairwise incompatible, the environment never issues more than one event from $SysEvents_i$, much like she never attempts both a correct event $E \in \overline{GEvents_i}$ and its fake version $fake\,(i, E)$.

Note that $\{hibernate\,(i)\}$ is similar in its effect to $\{fail\,(i)\}$ in the absence of $go(i)$: namely, the Byzantine inaction when the agent does not act, does not mark time (unless because of other events), and becomes Byzantine. However, the similarity is not perfect: $fail\,(i)$ is a generic failure due to the lack of some action, whereas $hibernate\,(i)$ signifies the specific failure to act in the round altogether. Accordingly, $fail\,(i)$ is compatible with $go(i)$, whereas $hibernate\,(i)$ is not. Thus, having this separation makes sense for diagnostic purposes. We generally intend to have a full taxonomy of possible failures: failures based on events, based on actions, based on the absence of actions, and based on failure to follow system instruction $go(i)$.

Remark 1.1.19. It is possible to define the correct version $internal\,(i, \text{📅})$ of $fake\,(i, \text{📅})$ but, given that 📅 is not recorded in the local history, this would be a wholly redundant operation.

[8] Events $go(i)$ can, under certain circumstances, be detected by the acting agent based on the fact that it is acting. However, mere acting does not generally imply $go(i)$ by itself since the actions can also be Byzantine.

[9] Another group of events not observed by the agents are $fake\left(i, A \mapsto \text{📅}\right)$.

Remark 1.1.20. There can be two main causes for Byzantine behavior:

- the agent may want to subvert the correct procedures and actively engages in sabotage;
- the agent is malfunctioning, which can result in incorrect sensor data being recorded, actions performed in reaction to this erroneous data, and/or actions in response to correct data but not correctly implemented.

Our formal model is attuned to the latter case, where the malfunctions are imposed by the environment. While the former case remains faithfully represented as the environment is free to implement any malicious intent by the agent, the epistemic approach is primarily relevant in the setting with malfunctioning agents. Indeed, if the agent has a complete freedom to misbehave to achieve a goal different from the stated protocol, then its protocol need not contain any self-diagnostic tools whereas for other agents its Byzantine actions are indistinguishable from random actions. By contrast, a malfunctioning agent that is capable of detecting its own faultiness can make necessary adjustments or shut down.

Remark 1.1.21 (Perfect recall)**.** We consider agents capable of perfect recall. More precisely, they do have perfect recall while acting correctly and their memory remains stable once recorded. However, Byzantine agents may misremember some of their events/actions. In particular, Byzantine agents may not remember any actions they performed. If perfect recall is desired for all agents, then Byzantine actions of the type $\mathit{fake}\,(i, A \mapsto A')$ with $A \neq A'$ should be prohibited.

Perfect recall imposes certain requirements on the memory available to the agent. The choice between perfect recall and history-free agents is akin to the difference between Turing machines and finite-state automata, i.e., the choice between expressivity and efficiency. In this report, we concentrate on the formalism that is expressive. Hence, the state of an agent is defined as its local *history*. In the string of following definitions as well as in the following statements and proofs, we assume that a set $\mathcal{A} = [\![1; n]\!]$ of agents, sets Σ_i of initial states, sets Int_i of local internal actions and sets Ext_i of local external events for each agent $i \in \mathcal{A}$, as well as a set *Msgs* of messages are arbitrary but fixed and we do not repeat this list every time.

Definition 1.1.22 (Agent's history)**.** A **history** h_i **of agent** $i \in \mathcal{A}$, or its **local state**, is a non-empty sequence

$$h_i = [\lambda_m, \dots, \lambda_1, \lambda_0]$$

for some $m \geq 0$ such that $\lambda_0 \in \Sigma_i$ and $\forall j \in [\![1; m]\!]$ we have $\lambda_j \subset \overline{Actions_i} \sqcup \overline{Events_i}$. In this case m is called the **length of history** h_i and denoted $|h_i|$. We say that a set $\lambda \subset \overline{Actions_i} \sqcup \overline{Events_i}$ **is recorded** in the history h_i of agent i and write $\lambda \subset h_i$ iff $\lambda = \lambda_j$ for some $j \in [\![1; m]\!]$. We say that $o \in \overline{Actions_i} \sqcup \overline{Events_i}$ **is recorded** in the history h_i and write $o \in h_i$ iff $o \in \lambda$ for some set $\lambda \subset h_i$.

Definition 1.1.23 (Environment's history)**.** A **history** h **of the system** with n agents, or the **global state**, is a tuple

$$h := (h_\epsilon, h_1, \dots, h_n)$$

where the **history of the environment** is a sequence

$$h_\epsilon = [\Lambda_m, \dots, \Lambda_1]$$

for some $m \geq 0$ such that $\forall j \in [\![1; m]\!]$ we have $\Lambda_j \subseteq \overline{GActions} \sqcup GEvents$ and h_i is a local state of each agent $i \in [\![1; n]\!]$. In this case m is called the **length of history** h and denoted $|h| := |h_\epsilon|$, i.e., the environment has the true global clock. We say that a set $\Lambda \subset \overline{GActions} \sqcup GEvents$ **happens** in the environment's history h_ϵ or in the system history h and write $\Lambda \subset h_\epsilon$ iff $\Lambda = \Lambda_j$ for some $j \in [\![1; m]\!]$. We say that $O \in \overline{GActions} \sqcup GEvents$ **happens** in the environment's history h_ϵ or in the system history h and write $O \in h_\epsilon$ iff $O \in \Lambda$ for some set $\Lambda \subset h_\epsilon$.

Definition 1.1.24 (Sets of local and global states)**.** $\mathscr{L}_i$ is the set of **local states** of agent i, i.e., the set of all histories of agent i. $\mathscr{L}_\epsilon$ is the set of histories of the environment. $\mathscr{L} := \prod_{i \in \mathcal{A}} \mathscr{L}_i$ is the set of **joint local states**. $\mathscr{G}$ is the set of **global states**.

The global state of the system contains both the local states of all agents and the omniscient view of the environment. It provides a complete snapshot of the system at a specific time, including the real picture of events and how these events are perceived by agents.

The following is an exhaustive list of the notation we use to describe main stages in a lifecycle of events and actions. For completeness purposes, this list also mentions protocols that will be formally introduced later. In this list, $i, j \in \mathcal{A}$ are agents, $\mu \in \mathit{Msgs}$ is a message, and $id \in \mathbb{N}$ is a GMI.

Correct internal actions and their Byzantine copies:

- $a \in \mathit{Int}_i$ represents the following:
  - in i's protocol, this prescribes agent i to perform the local internal action a if it is woken up for the round; this command by itself does not affect whether the agent marks the passage of time: that depends on whether the agent is woken up;
  - in i's local history, this means that agent i marked the passage of time and thinks it performed a, but does not necessarily know whether it was really performed and, if so, whether it was performed according to the protocol or in a Byzantine fashion.
- $\mathit{internal}\,(i, a)$ for $a \in \mathit{Int}_i$ represents the following:
  - in the environment's history, this means that i performed a according to i's protocol and marked the passage of time.
- $\mathit{fake}\,(i, \mathit{internal}\,(i, a) \mapsto A')$ for $a \in \mathit{Int}_i$ and $A' \in \overline{\mathit{GActions}_i}$ represents the following:
  - in the environment's protocol, this prescribes agent i to perform a in a Byzantine fashion (i.e., irrespective of both the protocol and whether the agent is woken up) but believe that the local version of A' was performed instead; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, this means that i performed a in a Byzantine fashion but believed that the local version of A' was performed instead and marked the passage of time.
- $\mathit{fake}\,(i, \mathit{internal}\,(i, a) \mapsto \text{📅})$ for $a \in \mathit{Int}_i$ represents the following:
  - in the environment's protocol, this prescribes agent i to perform a in a Byzantine fashion but forget about it; this command does not affect whether the agent marks the passage of time;
  - in the environment's history, this means that i performed a in a Byzantine fashion without recording it in its local history; whether i marked the passage of time depends exclusively on other actions/events of the round.
- $\mathit{fake}\,(i, A' \mapsto \mathit{internal}\,(i, a))$ for $a \in \mathit{Int}_i$ and $A' \in \overline{\mathit{GActions}_i}$ represents the following:
  - in the environment's protocol, this prescribes agent i to perform the local version of A' but believe that a was performed instead while marking the passage of time;
  - in the environment's history, this means that i performed the local version of A' in a Byzantine fashion but believed that a was performed instead and marked the passage of time.
- $\mathit{fake}\,(i, \text{📅} \mapsto \mathit{internal}\,(i, a))$ for $a \in \mathit{Int}_i$ represents the following:
  - in the environment's protocol, this prescribes agent i to mistake inaction for performing a while marking the passage of time;
  - in the environment's history, this means that i did not do anything but believes to have performed a and marked the passage of time.

Sending messages, correctly or in a Byzantine way:

- $send(j, \mu_k)$ represents the following:
  - in i's protocol, this prescribes agent i to send kth copy of a message μ to j if it is woken up for the round; in most cases, only one copy, the master copy is sent, which is denoted μ_0 or simply μ; this command by itself does not affect whether the agent marks the passage of time: that depends on whether the agent is woken up;
  - in i's local history, this means that agent i marked the passage of time and thinks it sent kth copy of a message μ to j, but does not necessarily know whether it was really sent and, if so, whether it was sent according to the protocol or in a Byzantine fashion.
- $gsend(i, j, \mu, id)$ represents the following:
  - in the environment's history, this means that i marked the passage of time, sent a message μ to j according to i's protocol, and the message was assigned the GMI id (which contains information about the copy number).
- $fake\,(i, gsend(i, j, \mu, id) \mapsto A')$ for $A' \in \overline{GActions_i}$ represents the following:
  - in the environment's protocol, this prescribes agent i to send a message μ to j in a Byzantine fashion and the environment to assign the GMI id (which contains information about the copy number) to the message and is computed correctly with respect to the current timestamp;, but i would believe that the local version of A' was performed instead; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, this means that i marked the passage of time, sent a message μ to j in a Byzantine fashion and the message was assigned the GMI id, but i believes that the local version of A' was performed instead; notwithstanding this belief, μ can be correctly received by j.
- $fake\,(i, gsend(i, j, \mu, id) \mapsto \text{📅})$ represents the following:
  - in the environment's protocol, this prescribes agent i to send a message μ to j in a Byzantine fashion and the environment to assign the GMI id to the message and forget about it; the GMI is computed correctly with respect to the current timestamp; this command does not affect whether the agent marks the passage of time;
  - in the environment's history, this means that i sent a message μ to j in a Byzantine fashion and the message was assigned the GMI id, but forgot about it; whether i marked the passage of time depends exclusively on other actions/events of the round; notwithstanding, μ can be correctly received by j.
- $fake\,(i, A \mapsto gsend(i, j, \mu, id))$ for $A \in \overline{GActions_i}$ represents the following:
  - in the environment's protocol, this prescribes agent i to perform the local version of A but mistakenly believe that it is sending a message μ to j (GMI id does not play a role); if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, this means that i marked the passage of time and performed the local version of A but mistakenly believes to have sent copy μ to j; despite i's belief, this event does not enable j to receive μ correctly.
- $fake\,(i, \text{📅} \mapsto gsend(i, j, \mu, id))$ represents the following:
  - in the environment's protocol, this prescribes agent i to mistakenly believe that it is sending a message μ to j; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, this means that i marked time and mistakenly believes to have sent a message μ to j; despite i's belief, this event does not enable j to receive μ correctly.

Byzantine inaction

- $\mathit{fake}\,(i, \boxminus)$ represents the following:
  - in the environment's protocol, this prescribes agent i to not do anything in a Byzantine fashion; this command does not affect whether the agent marks the passage of time;
  - in the environment's history, this means that i became Byzantine if it wasn't already; whether i marked the passage of time depends exclusively on other actions/events of the round.

Correct external events and their Byzantine copies:

- $e \in Ext_i$ represents the following:
  - in i's local history, it means that i marked the passage of time and believes to have observed e happened, but does not necessarily know whether it really happened;
- $\mathit{external}\,(i, e)$ for $e \in Ext_i$ represents the following:
  - in the environment's protocol, it prescribes the environment to impose e on agent i; it is incompatible with $\mathit{fake}\,(i, \mathit{external}\,(i, e))$; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, it means that i marked the passage of time and observed e.
- $\mathit{fake}\,(i, \mathit{external}\,(i, e))$ for $e \in Ext_i$ represents the following:
  - in the environment's protocol, it prescribes agent i to mistakenly believe to have observed e; it is incompatible with $\mathit{external}\,(i, e)$; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, it means that i marked the passage of time and mistakenly believes to have observed e.

Receiving messages, correctly or in a Byzantine fashion:

- $\mathit{recv}(j, \mu)$ represents the following:
  - in i's local history, it means that i marked the passage of time and believes to have received a message μ from agent j, but does not necessarily know whether the receipt of the message really happened.
- $\mathit{grecv}(i, j, \mu, id)$ represents the following:
  - in the environment's protocol, it prescribes to deliver to i a message μ sent earlier with GMI id by j; it is incompatible with $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id'))$ even if the fake id' is different; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, it means that i marked the passage of time and received a message μ sent earlier by j with GMI id.
- $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id))$ represents the following:
  - in the environment's protocol, it prescribes agent i to falsely believe to have received a message μ from j (GMI id does not play a role); it is incompatible with $\mathit{grecv}(i, j, \mu, id')$ even if the correct id' is different; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, it means that i marked the passage of time and falsely believes to have received μ from j.

Environment controlling agents' actions:

- $go(i)$ represents the following:
  - in the environment's protocol, it prescribes agent i to wake up and perform some set of actions prescribed by i's protocol; it is incompatible with $sleep\,(i)$ and $hibernate\,(i)$; if approved, this event causes the agent to mark the passage of time (even if no actions are prescribed by the protocol);
  - in the environment's history, it means that i was woken up, marked the passage of time, and performed some set of actions prescribed by i's protocol.
- $sleep\,(i)$ represents the following:
  - in the environment's protocol, it prescribes agent i to malfunction by marking the passage of time but skipping whatever actions prescribed by i's protocol; it is incompatible with $go(i)$ and $hibernate\,(i)$; if approved, this event causes the agent to mark the passage of time;
  - in the environment's history, it means that i malfunctioned, was woken up and marked the passage of time but was not allowed to perform any actions prescribed by i's protocol.
- $hibernate\,(i)$ represents the following:
  - in the environment's protocol, it prescribes agent i to malfunction by skipping whatever actions prescribed by i's protocol; it is incompatible with $go(i)$ and $sleep\,(i)$; if approved, this event prevents marking the passage of time due to actions; however, the passage of time may be triggered by other events;
  - in the environment's history, it means that i malfunctioned, was not woken up and did not perform any actions prescribed by i's protocol; moreover, the passage of time was not marked due to $go(i)$ or $sleep\,(i)$ but may have been marked due to other events.

1.2 Transition Function

There are multiple consistency restrictions to be imposed on the histories to ensure that information from a local history h_i, incomplete as it might be, does not contradict what is recorded by the environment objectively and omnisciently in h_ϵ. We now start introducing these restrictions, dividing them into several types according to the parts of the framework responsible for upholding them.

Our general ideology is that (correct) agents act to achieve a particular goal, for which the agent's protocol takes the responsibility. The environment plays a triple role. Firstly, it is an impartial physical medium enforcing the consistency of histories and the laws of causality. In particular the environment increments all histories, local and global, in a coherent way and filters out events that are considered "physically" impossible, such as a (non-Byzantine) delivery of a message that was never sent. Secondly, the environment is the source of external unbiased indeterminacy: it simply records all possibilities the future can have in store. "Unbiased" here means that possibilities should not be omitted in the interests of short-term expedience, such as achieving the worst-case scenario. The latter is done using the third part of the environment that performs the non-deterministic choice and is designated the *adversary*. In other words, it should not be possible to guarantee avoiding good (or any other possible) choices but it should be possible to avoid them by chance.

Since agents are assumed not to have the complete overview of the system, agent i's protocol P_i can only rely on i's local view, i.e., its local state at the moment. In particular, it is crucial for implementing asynchronous agents that P_i cannot use timestamp t as a parameter.

Conversely, the protocol P_ϵ of the omniscient environment can use timestamp t, in fact using t is necessary to correctly forge GMIs for sent Byzantine messages. At the same time, P_ϵ should

not depend on the current (global) state to preserve the unbiased representation of the physical laws and to facilitate proofs of properties of our framework.

For instance, a (synchronous-communication) receiver can be modeled by an environment protocol that "listens on all frequencies", i.e., attempts to deliver all messages at all times. However, all messages that have not been sent are filtered out and not delivered. Thus, the behavior of the system does depend on the global state, but the environment's protocol does not.

Definition 1.2.1 (Coherent events)**.** Let $t \in \mathbb{N}$ be a timestamp. A set $S \subset \mathit{GEvents}$ of events is called **t-coherent** if it satisfies the following conditions:

1. for any $\mathit{fake}\,(i, \mathit{gsend}(i, j, \mu, id) \mapsto A) \in S$, the GMI $id = id(i, j, \mu, k, t)$ for some $k \in \mathbb{N}$;
2. for any $i \in \mathcal{A}$ at most one of $\mathit{go}(i)$, $\mathit{sleep}\,(i)$, and $\mathit{hibernate}\,(i)$ is present in S;
3. for any $i \in \mathcal{A}$ and any $e \in \mathit{Ext}_i$ at most one of $\mathit{external}\,(i, e)$ and $\mathit{fake}\,(i, \mathit{external}\,(i, e))$ is present in S;
4. for any $\mathit{grecv}(i, j, \mu, id_1) \in S$, no event of the form $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id_2))$ belongs to S for any $id_2 \in \mathbb{N}$;
5. for any $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id_1)) \in S$, no event of the form $\mathit{grecv}(i, j, \mu, id_2)$ belongs to S for any $id_2 \in \mathbb{N}$;

Remark 1.2.2. It is possible that two copies $\mathit{grecv}(i, j, \mu, id_1)$ and $\mathit{grecv}(i, j, \mu, id_2)$ of the same message, possibly sent at different rounds, arrive simultaneously. While the receiving agent i would only know that the message μ from j is received, without being aware of various copies or their multiplicity, the ability to receive multiple copies is important, for instance, when message delivery has to be reliable.

We also leave a possibility of $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id_1))$ and $\mathit{fake}\,(i, \mathit{grecv}(i, j, \mu, id_2))$ at the same time making agent i falsely think that it received the message μ from j. While one such error makes all further ones redundant, there is no material difference in the system's behavior when two or more of such errors are present. Hence, to avoid unnecessary technical work, we leave this as a possibility. Needless to say, this possibility can always be precluded in specific protocols.

Lemma 1.2.3. *Any subset of a t-coherent set is itself t-coherent.*

Definition 1.2.4 (Protocol)**.**

1. A (non-deterministic) **protocol for agent** $i \in \mathcal{A}$ is any function

$$P_i \colon \mathscr{L}_i \to 2^{2^{\overline{\mathit{Actions}_i}}} \setminus \{\varnothing\} \tag{1.9}$$

   For a local state $h_i \in \mathscr{L}_i$ of agent i, each member $S \in P_i\,(h_i)$ is a subset of $\overline{\mathit{Actions}_i}$ and represents one of non-deterministic choices prescribing a set of actions for i in this local state. Note that $P_i\,(h_i) \neq \varnothing$ means that an agent always has at least one such choice S, which might be to perform no actions if $S = \varnothing$.

2. Given individual agents' protocols $P_1, \ldots, P_n$, their **joint protocol** is a function of global states that returns a tuple of action sets computed according to agent's protocols, one set per agent: for a global state $h = (h_\epsilon, h_1, \ldots, h_n)$,

$$P\,(h) := \big(P_1\,(h_1), \ldots, P_n\,(h_n)\big) \tag{1.10}$$

3. A (non-deterministic) **protocol for the environment** is any function

$$P_\epsilon \colon \mathbb{N} \longrightarrow 2^{2^{\mathit{GEvents}}} \setminus \{\varnothing\} \tag{1.11}$$

such that every $S \in P_\epsilon(t)$ is t-coherent. In other words, for each $t \in \mathbb{N}$, each member $S \in P_\epsilon(t)$ is a t-coherent subset of $GEvents$, i.e.,

$$\begin{aligned} S \quad \subset \quad & \{grecv(i,j,\mu,id) \mid i,j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \quad \sqcup \quad \{go(i) \mid i \in \mathcal{A}\} \quad \sqcup \\ & \{external(i,e) \mid i \in \mathcal{A}, e \in Ext_i\} \quad \sqcup \quad \{sleep(i) \mid i \in \mathcal{A}\} \quad \sqcup \\ & \{fake(i,E) \mid i \in \mathcal{A}, E \in \overline{GEvents_i}\} \quad \sqcup \quad \{hibernate(i) \mid i \in \mathcal{A}\} \quad \sqcup \\ & \left\{fake(i, A \mapsto A') \mid i \in \mathcal{A}, A, A' \in \{\text{🗃}\} \sqcup \overline{GActions_i}\right\}, \end{aligned}$$

and represents one of the non-deterministic possibilities for what can happen in the system at time $t.5$. The two conditions mean that no correct event can be accompanied by its faulty version and any faulty send has a correctly computed GMI. Note that $P_\epsilon(t) \neq \varnothing$ means that the environment always has at least one such choice S, which might be to impose no events if $S = \varnothing$.

Definition 1.2.5. For $\sigma \in \{0,1\}$ and a set X we define

$$X^\sigma := \begin{cases} X & \text{if } \sigma = 1, \\ \varnothing & \text{if } \sigma = 0. \end{cases}$$

Notation 1.2.6. We denote by $\mathscr{C}$ the set of all joint protocols.

Notation 1.2.7. We denote by $\mathscr{C}_\epsilon$ the set of all environment protocols.

Remark 1.2.8. All sets produced by protocols in $\mathscr{C}_\epsilon$ at time t are t-coherent.

Remark 1.2.9. Depending on the intended strength and type of Byzantine agents, we may further restrict the set of functions allowed as protocols of the environment.

Remark 1.2.10 (Time sensitive actions)**.** The dependence of the environment's protocol on time enables modelling of time-sensitive actions. For instance, such a protocol can implement a global prohibition on message delivery during designated quiet time.

Remark 1.2.11 (Life must go on)**.** Both the environment and each of the agents always has at least one (possibly empty) set of actions/events at its disposal (*no-apocalypse clause*). The situation when the agents crash and cannot proceed further can still be represented, e.g., by designating a special crash action.

As we saw, Byzantine send and receive events, as well as correct receive events, are both initiated and performed by the environment, which is why we chose to represent these events as fully formed, i.e., supplied with a GMI, from the very beginning. In fact, correct receive events must contain a GMI to determine whether a message with such a GMI was sent earlier, which is part of the definition of its correctness. The situation with correct send actions is different: they are initiated by an agent, who must remain unaware of a GMI, but the message is propagated to the recipient by the environment. Thus, when an active agent sends a message, it must first be transformed from the local to the global view. This task is performed by the *labeling functions* $label_i$ for each $i \in \mathcal{A}$. Similarly, when the message, correct or fake, is delivered or a fake event occurs for an agent, the event must be transformed into its local format before being recorded in the local history. This is done by the "*reverse*" function $label^{-1}$.

Definition 1.2.12 (Labeling functions)**.** For an agent $i \in \mathcal{A}$, we define a function

$$label_i \colon \overline{Actions_i} \times \mathbb{N} \longrightarrow \overline{GActions_i}$$

converting the local representation of actions to the global format as follows:

$$label_i(a,t) := \begin{cases} gsend(i,j,\mu,id(i,j,\mu,k,t)) & \text{if } a = send(j,\mu_k) \\ internal(i,a) & \text{if } a \in Int_i \end{cases}$$

We collect all these functions into one tuple $label := (label_1, \ldots, label_n)$.

We also define a function converting actions and events from the global format into the local ones. This function is applied after all fake events are already turned into their benign counterparts and 📅's are removed by a separate function. Thus, this function does not deal with fake events or no events.

$$label^{-1} \colon \overline{GActions} \sqcup \overline{GEvents} \longrightarrow \overline{Actions} \sqcup \overline{Events}$$

as follows:

$$label^{-1}(U) := \begin{cases} send(j, \mu_k) & \text{if } U = gsend(i, j, \mu, id(i, j, \mu, k, t)) \\ send(j, \mu_0) & \text{if } U = gsend(i, j, \mu, M) \text{ and } M \neq id(i, j, \mu, k, t) \text{ for any } k, t \in \mathbb{N} \\ recv(j, \mu) & \text{if } U = grecv(i, j, \mu, id) \\ a & \text{if } U = internal(i, a) \\ e & \text{if } U = external(i, e) \end{cases}$$

Function $label^{-1}$ extends to sets in the standard way: $label^{-1}(X) := \{label^{-1}(U) \mid U \in X\}$. For the functions $label_i$, we distribute the timestamp parameter to all elements of the set: $label_i(X, t) := \{label_i(a, t) \mid a \in X\}$.

Remark 1.2.13. The injectivity of the function id used in $label_i$ ensures that each message is unique from the point of view of the environment.

Remark 1.2.14. The second clause in the definition of $label^{-1}(U)$ is mostly cosmetic: we make GMIs id unforgeable, and, hence, this clause will never be used. It is added solely to make the function $label^{-1}(U)$ total, thus, avoiding irrelevant complications stemming from the use of potentially partial functions.

Definition 1.2.15 (Non-deterministic choice for protocols)**.** Given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and protocols $P_\epsilon \in \mathscr{C}_\epsilon$ for the environment and $(P_1, \ldots, P_n) \in \mathscr{C}$ for the agents, we obtain, for agent $i \in \mathcal{A}$ and timestamp $t \in \mathbb{N}$, the sets of global actions and events to be attempted at the global state h at t, i.e., in the round $t.5$, as follows:

1. Events imposed by the environment are a t-coherent set

$$\alpha_\epsilon^t = X_\epsilon \tag{1.12}$$

   for some set $X_\epsilon \in P_\epsilon(t)$ non-deterministically chosen by the adversary.

2. Actions agent $i \in \mathcal{A}$ would perform if woken up are a set

$$\alpha_i^{h,t} = label_i(X_i, t) \tag{1.13}$$

   for some set $X_i \in P_i(h_i)$ non-deterministically chosen by the adversary.

3. All these choices are combined in the **joint attempted action**

$$\alpha^{h,t} := (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t}).$$

Among the events α_ϵ^t we distinguish the following subsets for each agent $i \in \mathcal{A}$:

1. *Regular events* for $i \in \mathcal{A}$

$$\overline{\alpha}_{\epsilon_i}^t := \alpha_\epsilon^t \cap \overline{GEvents}_i = \{grecv(i, j, \mu, id) \in \alpha_\epsilon^t \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external(i, e) \in \alpha_\epsilon^t \mid e \in Ext_i\} \tag{1.14}$$

2. Instructions regarding waking up for $i \in \mathcal{A}$

$$\alpha_{g_i}^t = \alpha_\epsilon^t \cap SysEvents_i \tag{1.15}$$

3. Fake events for agent $i \in \mathcal{A}$, including those mimicking the agent's actions:

$$\alpha^t_{b_i} := \alpha^t_\epsilon \cap \mathit{BEvents}_i = \\ \left\{\mathit{fake}\,(i, A \mapsto A') \in \alpha^t_\epsilon \mid A, A' \in \{\text{🗓}\} \sqcup \overline{\mathit{GActions}}_i\right\} \sqcup \{\mathit{fake}\,(i, E) \in \alpha^t_\epsilon \mid E \in \overline{\mathit{GEvents}}_i\} \tag{1.16}$$

4. Instructions making agent i Byzantine:

$$\alpha^t_{f_i} := \alpha^t_{b_i} \sqcup \left(\alpha^t_\epsilon \cap \{\mathit{sleep}\,(i), \mathit{hibernate}\,(i)\}\right) \tag{1.17}$$

Note that $\mathit{sleep}\,(i)$ and $\mathit{hibernate}\,(i)$ may be present in both $\alpha^t_{g_i}$ and $\alpha^t_{f_i}$ Finally, we define

$$\overline{\alpha}^t_\epsilon := \bigsqcup_{i\in\mathcal{A}} \overline{\alpha}^t_{\epsilon_i} \qquad \alpha^t_g := \bigsqcup_{i\in\mathcal{A}} \alpha^t_{g_i} \qquad \alpha^t_b := \bigsqcup_{i\in\mathcal{A}} \alpha^t_{b_i} \qquad \alpha^t_f := \bigsqcup_{i\in\mathcal{A}} \alpha^t_{f_i}$$

Lemma 1.2.16. *Given a global history $h \in \mathscr{G}$ and protocols P_ϵ for the environment and $P_1, \ldots, P_n$ for the agents, for agent $i \in \mathcal{A}$ and for timestamp $t \in \mathbb{N}$, we have that*

$$\alpha^t_\epsilon \subset \mathit{GEvents} \qquad \text{and} \qquad \alpha^{h,t}_i \subset \overline{\mathit{GActions}}_i.$$

The environment should not create impossible situations. Most of them are prohibited by the definition of environment's protocol. For instance, an event recorded by an agent cannot both happen and not happen. Accordingly, the environment can only try one of these two possibilities but not both at the same time.

There is, however, one common type of "causal" impossibility that is not restricted to environment alone or a particular moment in time: a message *cannot* be delivered without being previously[10] sent. Due to our necessity to make the environment's protocol independent of the global history, which is crucial for many proofs, the environment's protocol cannot check whether the message was actually sent in previous rounds (based on the global history) or in the current round (based in part on the actions chosen by the adversary for the sending agent, the presence of the *go* command for it, and other events imposed on this agent). But correctly receiving an unsent message would break the laws of causality. Since these events cannot be handled by the environment alone, we create a special filter that weeds them out.

To simplify notation we introduce the following abbreviation:

Definition 1.2.17 (Active/passive, aware/unaware)**.** For a set $X \subseteq \mathit{GEvents}$ we define

$$\mathit{active}(i, X) := \begin{cases} t & \text{if } X \cap \mathit{SysEvents}_i = \{\mathit{go}(i)\}, \\ f & \mathit{otherwise}. \end{cases} \tag{1.18}$$

$$\mathit{aware}(i, X) := \begin{cases} t & \text{if } \varnothing \neq X \cap \mathit{SysEvents}_i \in \left\{\{\mathit{go}(i)\}, \{\mathit{sleep}\,(i)\}\right\}, \\ f & \mathit{otherwise}. \end{cases} \tag{1.19}$$

For readability's sake we write $\mathit{active}(i, X)$ instead of $\mathit{active}(i, X) = t$ and $\mathit{passive}(i, X)$ instead of $\mathit{active}(i, X) = f$, as well as $\mathit{aware}(i, X)$ instead of $\mathit{aware}(i, X) = t$ and $\mathit{unaware}(i, X)$ instead of $\mathit{aware}(i, X) = f$.

Corollary 1.2.18. *Given that for any $S \in P_\epsilon\,(t)$, at most one of system actions can be present*

$$\mathit{passive}(i, S) \iff S \cap \mathit{SysEvents}_i \in \left\{\varnothing, \{\mathit{sleep}\,(i)\}, \{\mathit{hibernate}\,(i)\}\right\}, \tag{1.20}$$

$$\mathit{unaware}(i, S) \iff S \cap \mathit{SysEvents}_i \in \left\{\varnothing, \{\mathit{hibernate}\,(i)\}\right\}. \tag{1.21}$$

$$\mathit{active}(i, X) \implies \mathit{aware}(i, X) \tag{1.22}$$

$$\mathit{unaware}(i, S) \implies \mathit{passive}(i, S) \tag{1.23}$$

[10] Here *previously* sent means sent in one of the preceding rounds or in the same round, whether correctly or in a Byzantine fashion. On the other hand, when an agent mistakenly thinks the message was sent, it is not considered sent previously.

Definition 1.2.19 (Event and action filter functions)**.** We define an **event filter function for Byzantine agents**

$$filter_\epsilon^B : \mathscr{G} \times 2^{GEvents} \times 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n} \longrightarrow 2^{GEvents}$$

as follows. Given a global history h, a set X_ϵ of events attempted by the environment (chosen by the adversary) and sets X_i of actions to be performed by the agents (also chosen by the adversary), the function returns the set of all attempted events that are "causally" possible as the set of events to be actually performed by the environment. Formally, for a set $X_\epsilon \subset GEvents$, sets $X_i \subset \overline{GActions}_i$ for each agent $i \in \mathcal{A}$, and a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, we define

$$\begin{aligned} filter_\epsilon^B (h, X_\epsilon, X_1, \ldots, X_n) \quad := \\ X_\epsilon \quad \setminus \quad \Big\{ grecv(j, i, \mu, id) \quad | \quad gsend(i, j, \mu, id) \notin h_\epsilon \quad \wedge \\ (\forall A \in \{\boxminus\} \sqcup \overline{GActions}_i)\, fake\,(i, gsend(i, j, \mu, id) \mapsto A) \notin h_\epsilon \quad \wedge \\ (gsend(i, j, \mu, id) \notin X_i \vee passive(i, X_\epsilon)) \quad \wedge \\ (\forall A \in \{\boxminus\} \sqcup \overline{GActions}_i)\, fake\,(i, gsend(i, j, \mu, id) \mapsto A) \notin X_\epsilon \Big\} \end{aligned} \tag{1.24}$$

In addition, we define **action filter functions for Byzantine agents** $i \in \mathcal{A}$

$$filter_i^B : 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n} \times 2^{GEvents} \longrightarrow 2^{\overline{GActions}_i}$$

as follows. Given a set of actions $X_j \subset \overline{GActions}_j$ prescribed for each agent $j \in \mathcal{A}$ by its protocol (as chosen by the adversary) and a set of events $X_\epsilon \subset GEvents$ that are performed by the environment, we define an all-or-nothing

$$filter_i^B (X_1, \ldots, X_n, X_\epsilon) = \begin{cases} X_i & \text{if } active(i, X_\epsilon) \\ \varnothing & \text{otherwise} \end{cases} \tag{1.25}$$

It is obvious from these definitions that these are indeed filter functions on X_ϵ and X_i respectively:

$$filter_\epsilon^B (h, X_\epsilon, X_1, \ldots, X_n) \subset X_\epsilon \tag{1.26}$$

$$filter_i^B (X_1, \ldots, X_n, X_\epsilon) \subset X_i \tag{1.27}$$

Thus, after protocols provided a range of possible event/action collections $P_\epsilon(t)$ and $P_i(h_i)$ and the adversary chose the collection α_ϵ^t of events to be attempted by the environment and collections $\alpha_i^{h,t}$ of actions to be performed by each agent if it is awoken, the filter functions determine which of these events and actions are to actually happen during the round. For this second stage, the resulting sets are called β-sets by analogy with α-sets.

Remark 1.2.20. While it is clear that an agent observing an event that actually happens cannot be mistaken about it, the situation with actions is more complex because the agent may not be certain about the exact action it performs. We chose to leave the agent with the widest variety of possibilities:

- several faulty actions can be mistaken for one (including the no-op action $\boxminus$ or some action that actually was performed): $fake\,(i, A_1 \mapsto A)$, ..., $fake\,(i, A_m \mapsto A)$ are generally compatible;
- one faulty action can be mistaken for several actions: $fake\,(i, A \mapsto A_1)$, ..., $fake\,(i, A \mapsto A_m)$ are generally compatible (this can also happen when A was actually performed or when A is the no-op action $\boxminus$).

In other words, agents can be confused not only regarding which actions they have performed but also regarding how many have been performed. Further, the agent may think it has done a lot without doing anything or vice versa may think it has done nothing despite frantic activity.

Remark 1.2.21. For the case of general Byzantine agents, we could directly define a local version of the **action filter function** $filter_i^B : 2^{\overline{GActions}_i} \times 2^{GEvents} \longrightarrow 2^{\overline{GActions}_i}$ because the choices of other agents do not affect the filtering for general Byzantine agents. But we will need the definition of other variants of $filter_i$ to be $(n+1)$-ary functions to refine the model to implement, for example, *rendez-vous communication.*

Definition 1.2.22. For a global history $h \in \mathscr{G}$, a timestamp $t \in \mathbb{N}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t})$, and agent $i \in \mathcal{A}$,

1. $\beta_\epsilon^{h,\alpha^{h,t}} := filter_\epsilon^B \left(h, \quad \alpha_\epsilon^t, \quad \alpha_1^{h,t}, \quad \ldots, \quad \alpha_n^{h,t}\right)$
2. $\beta_i^{h,\alpha^{h,t}} := filter_i^B \left(\alpha_1^{h,t}, \ldots, \alpha_n^{h,t}, \beta_\epsilon^{h,\alpha^{h,t}}\right)$
3. $\beta^{h,\alpha^{h,t}} := (\beta_\epsilon^{h,\alpha^{h,t}}, \beta_1^{h,\alpha^{h,t}}, \ldots, \beta_n^{h,\alpha^{h,t}})$

As for α_ϵ^t we also distinguish the following subsets of $\beta_\epsilon^{h,\alpha^{h,t}}$ for each agent $i \in \mathcal{A}$:

1. *Regular events* for agent i:

$$\overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} := \beta_\epsilon^{h,\alpha^{h,t}} \cap \overline{GEvents}_i = \{grecv(i,j,\mu,id) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external\,(i,e) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid e \in Ext_i\} \subset \overline{\alpha}_{\epsilon_i}^t \quad (1.28)$$

2. Instructions regarding waking up agent i:

$$\beta_{g_i}^{h,\alpha^{h,t}} := \beta_\epsilon^{h,\alpha^{h,t}} \cap SysEvents_i \subset \alpha_{g_i}^t \quad (1.29)$$

3. Fake events for agent i, including those mimicking the agent's actions:

$$\beta_{b_i}^{h,\alpha^{h,t}} := \beta_\epsilon^{h,\alpha^{h,t}} \cap BEvents_i = \left\{fake\,(i, A \mapsto A') \in \beta_\epsilon^{h,\alpha^{h,t}} \mid A, A' \in \{\text{📅}\} \sqcup \overline{GActions}_i\right\} \sqcup \{fake\,(i,E) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid E \in \overline{GEvents}_i\} \subset \alpha_{b_i}^t \quad (1.30)$$

4. Instructions making agent i Byzantine:

$$\beta_{f_i}^{h,\alpha^{h,t}} := \beta_{b_i}^{h,\alpha^{h,t}} \sqcup \left(\beta_\epsilon^{h,\alpha^{h,t}} \cap \{sleep\,(i), hibernate\,(i)\}\right) \subset \alpha_{f_i}^t \quad (1.31)$$

Finally, we define

$$\overline{\beta}_\epsilon^{h,\alpha^{h,t}} := \bigsqcup_{i \in A} \overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} \subset \overline{\alpha}_\epsilon^t \qquad \beta_g^{h,\alpha^{h,t}} := \bigsqcup_{i \in A} \beta_{g_i}^{h,\alpha^{h,t}} \subset \alpha_g^t$$

$$\beta_b^{h,\alpha^{h,t}} := \bigsqcup_{i \in A} \beta_{b_i}^{h,\alpha^{h,t}} \subset \alpha_b^t \qquad \beta_f^{h,\alpha^{h,t}} := \bigsqcup_{i \in A} \beta_{f_i}^{h,\alpha^{h,t}} \subset \alpha_f^t$$

$$\beta_{\epsilon_i}^{h,\alpha^{h,t}} := \overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} \sqcup \beta_{g_i}^{h,\alpha^{h,t}} \sqcup \beta_{b_i}^{h,\alpha^{h,t}}$$

Remark 1.2.23. The filtering is split into two steps: first filtering events $\beta_\epsilon^{h,\alpha^{h,t}}$ and then filtering actions based on the results of event filtering $\beta_i^{h,\alpha^{h,t}} = filter_i\left(\alpha_1^{h,t},\dots,\alpha_n^{h,t},\beta_\epsilon^{h,\alpha^{h,t}}\right)$. Such two-step filtering enables us to represent communication scenarios that rely on coordination among agents by making it possible to filter out *go* events that violate the coordination requirements.

Remark 1.2.24. Consider a global history $h \in \mathscr{G}$, a timestamp $t \in \mathbb{N}$, and a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t},\dots,\alpha_n^{h,t})$. Then

$$\beta_\epsilon^{h,\alpha^{h,t}} = \overline{\beta}_\epsilon^{h,\alpha^{h,t}} \sqcup \beta_g^{h,\alpha^{h,t}} \sqcup \beta_b^{h,\alpha^{h,t}}.$$

Proposition 1.2.25. *Consider a global history $h \in \mathscr{G}$, a timestamp $t \in \mathbb{N}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t},\dots,\alpha_n^{h,t})$, and an agent $i \in \mathcal{A}$. Then action filter function $filter_i$ for agent i ensures that*

$$\beta_i^{h,\alpha^{h,t}} \neq \varnothing \quad\Longrightarrow\quad active(i, \beta_\epsilon^{h,\alpha^{h,t}});$$
$$\beta_i^{h,\alpha^{h,t}} \neq \varnothing \quad\Longrightarrow\quad aware(i, \beta_\epsilon^{h,\alpha^{h,t}}).$$

Proof. The first statement follows directly from Def. 1.2.22(2) and equation (1.25). The second statement follows from the first and (1.22). □

Once again, it is easy to see that

Lemma 1.2.26. *Given a global history $h \in \mathscr{G}$, a timestamp $t \in \mathbb{N}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t},\dots,\alpha_n^{h,t})$, and agent $i \in \mathcal{A}$*

$$\beta_\epsilon^{h,\alpha^{h,t}} \subset GEvents \qquad and \qquad \beta_i^{h,\alpha^{h,t}} \subset \overline{GActions}_i$$

It is important to separate the complete knowledge required of the environment to perform the transition from state to state from the limited local view that the agents have. In particular, it is a central assumption of distributed systems in general and of the proposed framework in particular that agents should not be able to tell the difference between an external event they actually observed and a fake external event their sensors mistakenly registered, nor between performing action A' and thinking they have performed A' when A was the actual action performed, as represented by $fake\,(i, A \mapsto A')$. In this respect, the agents can be viewed as malfunctioning drones rather than scheming moles: They always mean well but are sometimes prevented by the environment from behaving correctly. In such cases, they can juxtapose their own intentions with their perception of the resulting actions and events but cannot directly detect the environment's meddling.

Formally, this means that the local histories must be purged of

(1) *fake* modifiers,

(2) GMIs,

(3) controlling commands $go(i)$, $sleep\,(i)$, and $hibernate\,(i)$.

These tasks are performed by the localization function σ: both on the action/event level and on the set level:

Definition 1.2.27 (Localization function)**.** The function

$$\sigma\colon 2^{\overline{GActions} \sqcup GEvents} \longrightarrow 2^{\overline{Actions} \sqcup \overline{Events}}$$

is defined as follows

$$\sigma\big(X\big) := label^{-1}\Big(\big(X \cap (\overline{GActions} \sqcup \overline{GEvents})\big) \cup \{E \mid (\exists i)\, fake\,(i, E) \in X\} \cup \{A' \neq \text{🗓} \mid (\exists i)(\exists A)\, fake\,(i, A \mapsto A') \in X\}\Big) \quad (1.32)$$

If $U \in \overline{GActions} \sqcup GEvents$ and $\sigma\big(\{U\}\big) \neq \varnothing$, i.e., U is not one of $go(i)$, $sleep\,(i)$, $hibernate\,(i)$, nor is a $fake\,(i, A \mapsto \text{📅})$ for some i and A, we also write $\sigma\big(U\big)$ to denote the only element of the set $\sigma\big(\{U\}\big)$, i.e., $\sigma\big(\{U\}\big) = \{\sigma(U)\}$.

The following lemma directly follows from the definitions of P_ϵ (Def. 1.2.4(3) and of t-coherence (Def. 1.2.1).

Lemma 1.2.28. *Given a global history $h \in \mathscr{G}$ and protocols $P_\epsilon \in \mathscr{C}_\epsilon$ for the environment and $P_1, \ldots, P_n$ for the agents, for any agent $i \in \mathcal{A}$ and for timestamp $t \in \mathbb{N}$,*

$$\sigma\big(\alpha^t_{b_i}\big) \cap label^{-1}\left(\overline{\alpha}^t_{\epsilon_i}\right) = \varnothing \tag{1.33}$$

$$|\alpha^t_{g_i}| \leq 1 \tag{1.34}$$

$$fake\,(i, gsend(i, j, \mu, id) \mapsto A) \in \alpha^t_{b_i} \qquad \Longrightarrow \qquad (\exists k \in \mathbb{N})\, id = id(i, j, \mu, k, t) \tag{1.35}$$

The following properties are inherited from the α-sets because filtering does not add new things. The next lemma follows from Lemma 1.2.3.

Lemma 1.2.29. *Given a global history $h \in \mathscr{G}$ and protocols $P_\epsilon \in \mathscr{C}_\epsilon$ for the environment and $P_1, \ldots, P_n$ for the agents, for any agent $i \in \mathcal{A}$ and for timestamp $t \in \mathbb{N}$, the set $\beta^{h,\alpha^{h,t}}_\epsilon$ is t-coherent, in particular,*

$$\sigma\big(\beta^{h,\alpha^{h,t}}_{b_i}\big) \cap label^{-1}\left(\overline{\beta}^{h,\alpha^{h,t}}_{\epsilon_i}\right) = \varnothing \tag{1.36}$$

$$|\beta^{h,\alpha^{h,t}}_{g_i}| \leq 1 \tag{1.37}$$

$$fake\,(i, gsend(i, j, \mu, id) \mapsto A) \in \beta^{h,\alpha^{h,t}}_{b_i} \qquad \Longrightarrow \qquad (\exists k \in \mathbb{N})\, id = id(i, j, \mu, k, t) \tag{1.38}$$

Remark 1.2.30. Since $\left|\beta^{h,\alpha^{h,t}}_{g_i}\right| \leq 1$ in all cases, we write $go(i) \in \beta^{h,\alpha^{h,t}}_{g_i}$ instead of the equivalent statement $\beta^{h,\alpha^{h,t}}_{g_i} = \{go(i)\}$.

The last piece of the puzzle is state update functions that record the events and actions performed in a round into all the histories.

Definition 1.2.31 (State update functions)**.** Given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of performed actions/events $X = (X_\epsilon, X_1, \ldots, X_n) \in 2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}$, we use the following abbreviation $X_{\epsilon_i} = X_\epsilon \cap GEvents_i$ for each $i \in \mathcal{A}$. Agents i's update function

$$update_i \colon \mathscr{L}_i \times 2^{\overline{GActions_i}} \times 2^{GEvents} \to \mathscr{L}_i$$

outputs a new local history from $\mathscr{L}_i$ based on i's actions X_i and environment-controlled events X_ϵ as follows:

$$update_i\,(h_i, X_i, X_\epsilon) := \begin{cases} h_i & \text{if } \sigma(X_{\epsilon_i}) = \varnothing \text{ and } unaware(i, X_\epsilon) \\ \Big[\sigma\big(X_{\epsilon_i} \sqcup X_i\big)\Big] : h_i & \text{otherwise} \end{cases} \tag{1.39}$$

where : represents sequence concatenation. Similarly, the environment's state update function

$$update_\epsilon \colon \mathscr{L}_\epsilon \times \left(2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}\right) \to \mathscr{L}_\epsilon$$

outputs a new state of the environment based on X_ϵ:

$$update_\epsilon\,(h_\epsilon, X) := (X_\epsilon \sqcup X_1 \sqcup \cdots \sqcup X_n) \colon h_\epsilon \tag{1.40}$$

Thus, the global state is modified as follows:

$$update\,(h, X) := \Big(update_\epsilon\,(h_\epsilon, X)\,, update_1\,(h_1, X_1, X_\epsilon)\,, \ldots, update_n\,(h_n, X_n, X_\epsilon)\Big) \tag{1.41}$$

Remark 1.2.32. The first clause in (1.39) corresponds to the situation when the agent is not woken up by actions or events. In particular, $unaware(i, X_\epsilon)$ states that i is denied both actions in the round and even awareness of the round itself. Virtually always function $update_i$ will be applied to $X_\epsilon = \beta_\epsilon^{h,\alpha^{h,t}}$, which is a t-coherent set. Thus, the condition $unaware(i, \beta_\epsilon^{h,\alpha^{h,t}})$ is equivalent to $\beta_\epsilon^{h,\alpha^{h,t}} \cap SysEvents_i \in \{\varnothing, \{hibernate\,(i)\}\}$, in other words,

$$unaware(i, \beta_\epsilon^{h,\alpha^{h,t}}) \qquad \Longleftrightarrow \qquad \beta_\epsilon^{h,\alpha^{h,t}} \cap SysEvents_i \subset \{hibernate\,(i)\}. \tag{1.42}$$

Following [FHMV95], we define transition functions as follows:

Definition 1.2.33 (Transition function)**.** For agents' protocols $P = (P_1, \ldots, P_n)$ and a protocol P_ϵ of the environment, we define a **Byzantine transition function**

$$\tau^B_{P_\epsilon,P} \quad : \quad 2^{GEvents} \times 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n} \quad \to \quad (\mathscr{G} \quad \to \quad \mathscr{G})$$

as a function that outputs a **global state transformer function**

$$\tau^B_{P_\epsilon,P}(Y) \colon \mathscr{G} \to \mathscr{G}$$

from global states to global states given joint attempted actions/events

$$Y \quad \in \quad 2^{GEvents} \times 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n}$$

defined as follows. For a global state $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and such joint attempted actions/events Y we consider two possibilities:

- if $Y = \alpha^{h,|h|} = \left(\alpha_\epsilon^{|h|}, \alpha_1^{h,|h|}, \ldots, \alpha_n^{h,|h|}\right)$ for some $\alpha_\epsilon^{|h|} \in P_\epsilon\,(|h|)$ and some $X_i \in P_i\,(h_i)$ for each $i \in \mathcal{A}$ such that $\alpha_i^{h,|h|} = label_i\,(X_i, |h|)$ then we define

$$\tau^B_{P_\epsilon,P}(Y)(h) := update\left(h, \quad \beta^{h,\alpha^{h,|h|}}\right) \tag{1.43}$$

  where the β-sets are computed from $\alpha^{h,|h|}$ by Def. 1.2.22 and the *update* function is defined in (1.41);

- otherwise, we define $\tau^B_{P_\epsilon,P}(Y)(h) = h$.[11]

Remark 1.2.34. By a slight abuse of notation, we write $h' \in \tau^B_{P_\epsilon,P}(h)$ to mean that there is a protocol-conformant set of joint actions $\alpha^{h,|h|}$ satisfying the first clause of the above definition such that $\tau^B_{P_\epsilon,P}(\alpha^{h,|h|})(h) = h'$.

1.3 Runs and Contexts

As already mentioned, integer timestamps are used exclusively to take snapshots of the local and global states. A sequence of such snapshots as the time progresses is called a *run*. Our goal is to model systems that are, in general, asynchronous, meaning that the agents can neither know the global time nor count the number of rounds since the beginning of the run. Without loss of generality, we consider runs that encompass the whole infinite set $\mathbb{N}$ of timestamps.

Definition 1.3.1 (Run)**.** A **run** is a function that assigns a global state to each integer timestamp.

$$r : \mathbb{N} \longrightarrow \mathscr{G} \tag{1.44}$$

We denote the set of all runs by R. The part of the run that an agent i can see is called i's **local view**. It is a function that assigns i's local state to each integer timestamp.

$$r_i \colon \mathbb{N} \longrightarrow \mathscr{L}_i \tag{1.45}$$

[11] The latter case will never be used and is only provided to make the transition function total.

It is clear that each local view r_i is uniquely determined by the run r:

$$r_i(t) := \pi_{i+1} r(t)$$

where π_j is the jth projection function for tuples/sequences. Similarly, we define the environment's history

$$r_\epsilon \colon \mathbb{N} \longrightarrow \mathscr{L}_\epsilon$$

to be

$$r_\epsilon(t) := \pi_1 r(t)$$

Definition 1.3.2. For a set $X \subset \mathcal{A} \times \mathbb{N}$ of nodes we define the upper **time bound** $T(X)$ to be the largest $T \in \mathbb{N}$ such that $(i, T) \in X$ for some agent $i \in \mathcal{A}$ if such a T exists. $T(\varnothing)$ is defined to be 0. A set X is called **bounded** if it has a time bound or **unbounded** otherwise.

Each agent initially starts off in a correct state and may become Byzantine when its actions stop being dictated by the protocol or its perception of events is compromised. Thus, it is more precise to talk about Byzantine states of agents, i.e., about Byzantine nodes $(i, t) \in \mathcal{A} \times \mathbb{N}$ instead of announcing the agents themselves to be universally Byzantine. Local timestamps (nodes) directly resulting from a violation of the agent's protocol or from a Byzantine event, including a Byzantine system event, are called *Bad*. All nodes of an agent starting from the first *Bad* local timestamp are called *Failed*. Recall that time in global histories h is represented by $|h|$.

Definition 1.3.3. Consider a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ of length $|h|$, so that $h_\epsilon = [\Lambda_{|h|}, \ldots, \Lambda_1]$. We define the sets of *Bad* and *Failed* nodes

$$\begin{aligned} Bad(h) &:= \left\{(i,t) \in \mathcal{A} \times \mathbb{N} \mid \Lambda_t \cap \left(BEvents_i \sqcup \{sleep(i), hibernate(i)\}\right) \neq \varnothing\right\} \\ Failed(h) &:= \{(i,t) \in \mathcal{A} \times \mathbb{N} \mid (\exists t' \leq t)\,(i,t') \in Bad(h)\} \end{aligned}$$

if $|h| > 0$ and $Bad(h) = Failed(h) := \varnothing$ otherwise.

Remark 1.3.4. For any global history $h \in \mathscr{G}$, $Bad(h) \subset Failed(h)$. Indeed, the former represents nodes that experienced a malfunction in the immediately preceding round, whereas the latter is comprised of nodes with some malfunction possibly further in the past.

Definition 1.3.5. For a run $r \in R$, timestamp $t \in \mathbb{N}$, and bounded set $X \subset \mathcal{A} \times \mathbb{N}$ of nodes, we define

$$\begin{aligned} Bad(r,t) &:= Bad(r(t)) & Bad_X(r) &:= X \cap Bad(r, T(X)) \\ Failed(r,t) &:= Failed(r(t)) & Failed_X(r) &:= X \cap Failed(r, T(X)) \end{aligned}$$

For an unbounded set $X \subset \mathcal{A} \times \mathbb{N}$ of nodes, we define

$$Bad_X(r) := X \cap \left(\bigcup_{t=1}^{\infty} Bad(r,t)\right) \qquad Failed_X(r) := X \cap \left(\bigcup_{t=1}^{\infty} Failed(r,t)\right)$$

Remark 1.3.6. The unions in the unbounded case begin from $t = 1$ because for any run r, we have $Bad(r, 0) = Failed(r, 0) = \varnothing$.

Remark 1.3.7. The definition of $Bad_X(r)$ and $Failed_X(r)$ for unbounded sets X is compatible with that for bounded sets, when applied to bounded sets X. The benefit of the latter definition is that it is efficiently computable.

Remark 1.3.8. For agents' protocols P and the environment's protocol P_ϵ, we are mostly interested in runs $r \in R$ that are built according to these protocols by some transition function. For the time being, we use the Byzantine transition function $\tau^B_{P_\epsilon, P}$, i.e., consider runs that begin from a proper initial state and such that for each timestamp $t \in \mathbb{N}$,

$$r(t+1) \in \tau^B_{P_\epsilon, P}(r(t)) \tag{1.46}$$

Sometimes we call such runs $\tau^B_{P_\epsilon, P}$**-transitional**, or simply **transitional**. For such a transitional run r, we denote its initial state by $r(0)$ and the global state after round $(t-1).5$ is $r(t)$.

It immediately follows from (1.43), (1.40), and Def. 1.3.2 that

Proposition 1.3.9. *For any transitional run r,*

$$Bad\,(r,t) \subset Bad\,(r,t+1) \qquad and \qquad Failed\,(r,t) \subset Failed\,(r,t+1).$$

In addition, for $X \subset X'$,

$$Bad_X\,(r) \subset Bad_{X'}\,(r) \qquad and \qquad Failed_X\,(r) \subset Failed_{X'}\,(r),$$

independent of whether both sets are bounded, X is bounded while X' is not, or both sets are unbounded.

Proof. The crucial observation is that, for transitional runs, $r\,(t+1)$ is either equal to $r\,(t)$ or obtained by prepending it. In either case, $r\,(t+1)$ contains $r\,(t)$ without modifications. □

Remark 1.3.10. In the interests of generality and modularity of concepts, we defined the transition function in terms of arbitrary histories. For the case of histories comprising a transitional run, the notation can be simplified. We will now provide a concise digest of one step of transition for transitional runs (see also Fig. 1.1) with the dual purpose: to give a compact summary of the procedure and introduce a simpler notation mostly used in the rest of our work. This will also form a crucial part of the notion of *(weak) consistency with a context and a joint protocol* in Def. 1.3.27 after we introduce all parts comprising a context.

In Def. 1.2.33, we defined the basic Byzantine transition function $\tau^B_{P_\epsilon,P}$ based on protocols P of the agents and P_ϵ of the environment. As already mentioned, we will sometimes need to change the filtering phase of the transition function. Hence, we leave the exact details of the transition as a parameter τ that converts protocols into a transition function.

Definition 1.3.11 (Transition template)**.** A **transition template**

$$\tau : \mathscr{C}_\epsilon \times \mathscr{C} \to \left(2^{GEvents} \times 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n} \quad \to \quad (\mathscr{G} \quad \to \quad \mathscr{G})\right) \tag{1.47}$$

is a two-place function that takes a protocol $P_\epsilon \in \mathscr{C}_\epsilon$ of the environment and a joint agents' protocol $P \in \mathscr{C}$ and outputs a **transition function** $\tau(P_\epsilon, P)$, which we denote by $\tau_{P_\epsilon,P}$

$$\tau_{P_\epsilon,P} \qquad : \qquad 2^{GEvents} \times 2^{\overline{GActions}_1} \times \cdots \times 2^{\overline{GActions}_n} \quad \to \quad (\mathscr{G} \quad \to \quad \mathscr{G})$$

Thus $\tau^B_{P_\epsilon,P}$ is only one possible transition function, the *Byzantine* transition function, obtained from protocols P_ϵ and P: namely, the one resulting from using the Byzantine filter functions $filter^B_i$ and $filter^B_\epsilon$.

Whichever filtering is used, one round of transition consists of the following phases:

One step of $\tau_{P_\epsilon,P}$-transition for runs One transition made according to a transition function $\tau_{P_\epsilon,P}$ consists of five consecutive phases, which are visually represented in Fig. 1.1:

1. *Protocol phase* (protocols are explicit arguments to the transition template τ)
   First, the protocol P_i for each agent i lays out a range $P_i\,(r_i\,(t))$ of possible sets of i's actions in the round based on the local state $r_i\,(t)$ of the agent. Similarly, the protocol P_ϵ of the environment lays out a range $P_\epsilon\,(t)$ of possible t-coherent sets of events in the round based on time t.

2. *Adversary phase* (this phase is stable: it does not change from template to template or from protocol to protocol)
   From these ranges, the adversary non-deterministically picks one set

$$X_i \in P_i\,(r_i\,(t)) \tag{1.48}$$

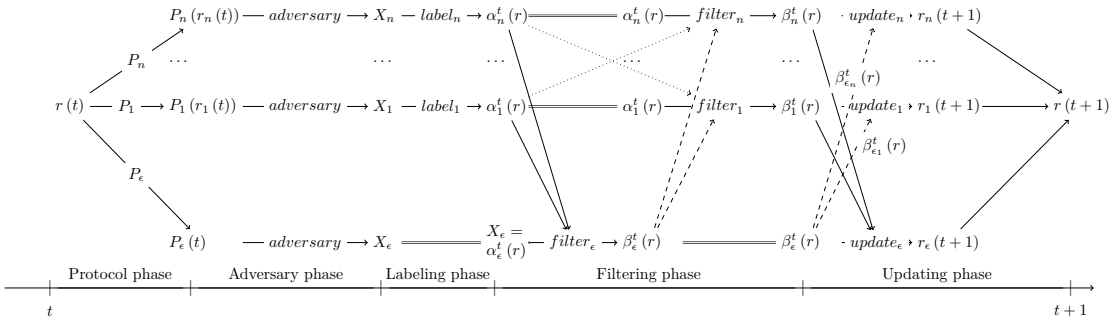

Figure 1.1: The evolution of states in round $t.5$ (from timestamp $t \in \mathbb{N}$ to $t+1$) inside a run r constructed according to the transition function $\tau_{P_\epsilon,P}$. Different communication models require changes to the filtering functions $filter_\epsilon$ and $filter_i$.

of actions for each agent i and a set

$$X_\epsilon \in P_\epsilon(t) \tag{1.49}$$

of events for the environment. These are actions the agents intend to perform and events the environment intends to impose in the round. Note that $X_i \subset \overline{Actions_i}$ and $X_\epsilon \subset GEvents$.

3. *Labeling phase* (this phase is stable: it does not change from template to template or from protocol to protocol)
The environment processes the intended actions X_i of each agent i converting them into the global format, in particular, assigning GMIs to message send requests from agents. We denote the resulting sets
$$\alpha_i^t(r) := label_i(X_i, t). \tag{1.50}$$
The set of environmental events X_ϵ is already in the global format and requires no modifications:
$$\alpha_\epsilon^t(r) := X_\epsilon. \tag{1.51}$$
Note that $\alpha_i^t(r) \subset \overline{GActions_i}$ and $\alpha_\epsilon^t(r) \subset GEvents$.

4. *Filtering phase* (this phase depends on the filtering functions $filter_\epsilon$ and $filter_i$, which are considered to be part of the template)
In this phase, intended actions and events that are deemed "causally impossible" in the underlying communication model are filtered out: though they may be requested by the agents/environment, they are not performed and not recorded in histories. Thus, the exact nature of filtering depends on the intended model, and different filtering functions produce different transition functions. The following requirements are imposed on filtering functions that can be used in any template:
$$passive(i, X_\epsilon) \implies filter_i(X_1, \ldots, X_n, X_\epsilon) = \varnothing \tag{1.52}$$
$$filter_i(X_1, \ldots, X_n, X_\epsilon) \subset X_i \tag{1.53}$$
$$filter_\epsilon(h, X_\epsilon, X_1, \ldots, X_n) \subset X_\epsilon \tag{1.54}$$
In other words, no actions by agent i are allowed by the environment unless $go(i)$ is issued and filtering is a non-increasing function with respect to the relevant argument. Note that some environment's events may also be "causally impossible", such as, e.g., receiving a message that was never sent. The filtering phase is further divided into two subphases:

(a) first impossible environment events are filtered out by the function $filter_\epsilon$ based on the intended environment's events X_ϵ and intended actions $\alpha_i^t(r)$ of all agents, resulting in the set $\beta_\epsilon^t(r)$ of performed environment's events:

$$\beta_\epsilon^t(r) := filter_\epsilon\left(r(t), \quad \alpha_\epsilon^t(r), \quad \alpha_1^t(r), \quad \ldots, \quad \alpha_n^t(r)\right), \tag{1.55}$$

(b) then for each agent i, the filtering function $filter_i$ performs the same task on the agents' actions, but taking into account the already filtered events $\beta_\epsilon^t(r)$ and intended actions $\alpha_j^t(r)$ of all agents $j \in \mathcal{A}$. The resulting sets of actions actually performed by agents are denoted $\beta_i^t(r)$:

$$\beta_i^t(r) := filter_i\left(\alpha_1^t(r), \quad \ldots, \quad \alpha_n^t(r), \quad \beta_\epsilon^t(r)\right) \tag{1.56}$$

Note that $\beta_i^t(r) \subset \alpha_i^t(r) \subset \overline{GActions}_i$ and $\beta_\epsilon^t(r) \subset \alpha_\epsilon^t(r) \subset GEvents$ by (1.53) and (1.54) and that the latter is always a t-coherent set.

5. *Updating phase* (this phase is stable: it does not change from template to template or from protocol to protocol)
The events $\beta_\epsilon^t(r)$ and actions $\beta_i^t(r)$ actually happening in the round are faithfully recorded into the global history and are translated into the simplified local form for being recorded into the local histories of each agent by the update functions. The crucial point of this translation is stripping out the GMIs and any information that would allow an agent to easily distinguish a correct event from a faulty one. Once again, the local history of each agent i is only affected by the actions $\beta_i^t(r)$ it performs and environment's events $\beta_{\epsilon_i}^t(r)$ it observes, whereas the global history is modified based on the complete information about all events and actions performed in the round.

$$r_i(t+1) := update_i\left(r_i(t), \quad \beta_i^t(r), \quad \beta_\epsilon^t(r)\right) \tag{1.57}$$

$$\beta^t(r) := \left(\beta_\epsilon^t(r), \quad \beta_1^t(r), \quad \ldots, \quad \beta_n^t(r)\right), \tag{1.58}$$

$$r_\epsilon(t+1) := update_\epsilon\left(r_\epsilon(t), \quad \beta^t(r)\right) \tag{1.59}$$

We will routinely use (1.48)–(1.59) to prove properties of transitional runs. However, it should be noted that in this respect β sets have a different status from α sets and X sets. Indeed, by (1.59), all β sets can be easily retrieved from a given transitional run. We do not even have to assume the transitionality of a run to define the β sets, though this definition would make sense mostly for transitional runs. In compliance with (1.40), for an arbitrary global history h, we define

$$\beta_i(h) := \pi_1 h_\epsilon \cap \overline{GActions}_i \tag{1.60}$$

$$\beta_\epsilon(h) := \pi_1 h_\epsilon \cap GEvents \tag{1.61}$$

For the case of runs

$$\beta_i^t(r) = \beta_i(r(t+1)) = \pi_1 r_\epsilon(t+1) \cap \overline{GActions}_i \tag{1.62}$$

$$\beta_\epsilon^t(r) = \beta_\epsilon(r(t+1)) = \pi_1 r_\epsilon(t+1) \cap GEvents \tag{1.63}$$

The latter set we further partition (we only show the notation for the case of runs, the case of histories is processed analogously):

- correct external events: $\overline{\beta}_\epsilon^t(r)$ observed by all agents and $\overline{\beta}_{\epsilon_i}^t(r)$ observed specifically by agent i;
- system events (*go*, *sleep*, and *hibernate*): $\beta_g^t(r)$ imposed on all agents and $\beta_{g_i}^t(r)$ imposed specifically on agent i;
- Byzantine events: $\beta_b^t(r)$ observed by all agents and $\beta_{b_i}^t(r)$ observed specifically by agent i;

- all external events that make the agent Byzantine until the end of the run, including faultily skipped rounds $sleep\,(i)$ and $hibernate\,(i)$: $\beta_f^t\,(r)$ imposed on all agents and $\beta_{f_i}^t\,(r)$ imposed specifically on agent i.

On the other hand, parts of the X and α sets are filtered out and have no effect on the transitions in the run. Hence, given a transitional run, it is not generally possible to retrieve the exact X and α sets used in each transition. All that is required is that there exist a collection of sets $X_1, \ldots, X_n, X_\epsilon$ satisfying (1.48)–(1.49) that eventually generate $\beta_1^t\,(r), \ldots, \beta_n^t\,(r), \beta_\epsilon^t\,(r)$ from (1.62)–(1.63) according to (1.50)–(1.56). Despite this subtlety, we still use function-like notation for α sets to keep the notation uniform with β sets. In particular, we use this uniformity to identify the same parts of the α sets by the same subscripts, e.g., $\overline{\alpha}_{\epsilon_i}^t\,(r)$ represents the set of correct internal events the environment is intending to impose on agent i, which will be filtered and become $\overline{\beta}_{\epsilon_i}^t\,(r)$, the set of correct external events actually imposed by the environment on agent i during round $t.5$ of the run r.

If we write $\beta^t\,(r)\,, \beta_\epsilon^t\,(r), \beta_1^t\,(r), \ldots, \beta_n^t\,(r)$, etc., it means that we assume the run r to be transitional and use $\alpha_\epsilon^t\,(r), \alpha_1^t\,(r), \ldots, \alpha_n^t\,(r)$, etc. for one possible choice of sets that could lead to such sets β according to a transition function $\tau_{P_\epsilon,P}$.

Transitional runs have several useful properties:

Remark 1.3.12 (Global total recall)**.** Not only $\beta^t\,(r)$ but also all $\beta^{t'}\,(r)$ for $t' \leq t$ can be extracted from $r(t+1)$, or even from $r_\epsilon(t+1)$ in a $\tau_{P_\epsilon,P}$-transitional run r: for instance,

$$\beta_\epsilon^{t'}\,(r) = \pi_1 r_\epsilon\,(t'+1) \cap \mathit{GEvents} = \pi_{t-t'+1} r_\epsilon\,(t+1) \cap \mathit{GEvents}$$

Lemma 1.3.13 (Objective global time)**.** *For any $\tau_{P_\epsilon,P}$-transitional run r and any timestamp t,*

$$|r\,(t)\,| = t.$$

The following properties rely on the restrictions imposed on the environment's protocol as well as on the filtering functions implementing general Byzantine agents. In order to make this and further results more general we define pointwise order on filtering functions and formulate many of the statements for any filter up to the general Byzantine one.

Definition 1.3.14. We say that a filter $filter_\epsilon^1$ is **stricter** than a filter $filter_\epsilon^2$ or that $filter_\epsilon^2$ is more **liberal** than $filter_\epsilon^1$ and we write $filter_\epsilon^1 \subset filter_\epsilon^2$ if the inclusion holds pointwise

$$filter_\epsilon^1\,(h, X_\epsilon, X_1, \ldots, X_n) \subset filter_\epsilon^2\,(h, X_\epsilon, X_1, \ldots, X_n)$$

for any global history $h \in \mathscr{G}$, any $X_\epsilon \subset \mathit{GEvents}$, and arbitrary $X_i \subset \overline{\mathit{GActions}}_i$ for each $i \in \mathcal{A}$.

Lemma 1.3.15 (GMIs are correct)**.** *For any $\tau_{P_\epsilon,P}$-transitional run r for some $P_\epsilon \in \mathscr{C}_\epsilon$, for $i, j \in \mathcal{A}$, $\mu \in \mathit{Msgs}$, $t \in \mathbb{N}$, $A \in \overline{\mathit{GActions}}_i \sqcup \{\text{📅}\}$, and $id \in \mathbb{N}$*

$$\mathit{gsend}(i, j, \mu, id) \in \beta_i^t\,(r) \quad \Longrightarrow \quad id = id(i, j, \mu, k, t) \text{ for some } k \in \mathbb{N} \tag{1.64}$$

$$\mathit{fake}\,(i, \mathit{gsend}(i, j, \mu, id) \mapsto A) \in \beta_{b_i}^t\,(r) \quad \Longrightarrow \quad id = id(i, j, \mu, k, t) \text{ for some } k \in \mathbb{N} \tag{1.65}$$

In addition, for any transition template τ with a $filter_\epsilon \subset filter_\epsilon^B$ and for any $\tau_{P_\epsilon,P}$-transitional run r

$$\mathit{grecv}(i, j, \mu, id) \in \overline{\beta}_{\epsilon_i}^t\,(r) \quad \Longrightarrow \quad id = id(j, i, \mu, k, t') \text{ for some } k \in \mathbb{N} \text{ and } t' \leq t \tag{1.66}$$

Proof. By (1.56), (1.53), (1.50), and Def. 1.2.12, the id for correct *gsend* instructions is supplied by the function $label_i$, which guarantees the first statement.

Similarly, for the second statement, by (1.55), (1.51), Def. 1.2.4(3), the id's for Byzantine *gsend* instructions are created by the environment in a manner that guarantees the second statement.

Finally, for the third statement, by the same (1.55) and (1.24), if a *grecv* command was not filtered out by $filter_\epsilon$, it was not filtered by the more liberal $filter_\epsilon^B$, hence, the matching *gsend* command or its Byzantine version must have occurred at the latest by the same round. In other words, taking into account (1.55)–(1.56), there are four possibilities:

1. $gsend(j,i,\mu,id) \in r_\epsilon(t) \quad \Longrightarrow \quad gsend(j,i,\mu,id) \in \beta_j^{t'}(r)$ for some $t' < t$
2. $gsend(j,i,\mu,id) \in \beta_j^t(r)$
3. $fake(j, gsend(j,i,\mu,id) \mapsto A) \in r_\epsilon(t)$ for some $A \in \{\text{📅}\} \sqcup \overline{GActions}_j \implies$
   $fake(j, gsend(j,i,\mu,id) \mapsto A) \in \beta_{b_j}^{t'}(r)$ for some $t' < t$ and $A \in \{\text{📅}\} \sqcup \overline{GActions}_j$
4. $fake(j, gsend(j,i,\mu,id) \mapsto A) \in \beta_{b_j}^t(r)$ for some $A \in \{\text{📅}\} \sqcup \overline{GActions}_j$

In other words,

$$gsend(j,i,\mu,id) \in \beta_j^{t'}(r) \text{ or } fake(j, gsend(j,i,\mu,id) \mapsto A) \in \beta_{b_j}^{t'}(r) \text{ for an } A \in \{\text{📅}\} \sqcup \overline{GActions}_j$$

for some $t' \le t$. It remains to use the already proved (1.64) or (1.65) respectively. □

Corollary 1.3.16. *For any $\tau_{P_\epsilon,P}$-transitional run r with a $filter_\epsilon \subset filter_\epsilon^B$, for $i,j \in \mathcal{A}$, $\mu \in Msgs$, $t \in \mathbb{N}$, and $id \in \mathbb{N}$*

$$grecv(i,j,\mu,id) \in \overline{\beta}_{\epsilon_i}^t(r) \quad \Longrightarrow \quad (\exists t' \le t)\Big(gsend(j,i,\mu,id) \in \beta_j^{t'}(r) \quad or$$
$$(\exists A \in \{\text{📅}\} \sqcup \overline{GActions}_j)\, fake(j, gsend(j,i,\mu,id) \mapsto A) \in \beta_{b_j}^{t'}(r)\Big) \tag{1.67}$$

Corollary 1.3.17 (GMIs are unique)**.** *For any transitional run r, for $i,j,k,l \in \mathcal{A}$, $\mu,\mu' \in Msgs$, $t,t' \in \mathbb{N}$, $id \in \mathbb{N}$, $A \in \{\text{📅}\} \sqcup \overline{GActions}_i$, and $A' \in \{\text{📅}\} \sqcup \overline{GActions}_k$:*

$$\begin{cases} gsend(i,j,\mu,id) \in \beta_i^t(r) \\ gsend(k,l,\mu',id) \in \beta_k^{t'}(r) \end{cases} \quad \Longrightarrow \quad \begin{cases} k = i \\ l = j \\ t' = t \\ \mu' = \mu \end{cases} \tag{1.68}$$

$$\begin{cases} fake(i, gsend(i,j,\mu,id) \mapsto A) \in \beta_{b_i}^t(r) \\ fake(k, gsend(k,l,\mu',id) \mapsto A') \in \beta_{b_k}^{t'}(r) \end{cases} \quad \Longrightarrow \quad \begin{cases} k = i \\ l = j \\ t' = t \\ \mu' = \mu \end{cases} \tag{1.69}$$

$$\begin{cases} gsend(i,j,\mu,id) \in \beta_i^t(r) \\ fake(k, gsend(k,l,\mu',id) \mapsto A') \in \beta_{b_k}^{t'}(r) \end{cases} \quad \Longrightarrow \quad \begin{cases} k = i \\ l = j \\ t' = t \\ \mu' = \mu \end{cases} \tag{1.70}$$

In other words, the GMI id completely determines the sender, the recipient, the sent message and the time of sending for both correct and Byzantine messages processed by the environment.

Proof. The statements follow from Lemma 1.3.15 (from (1.64) for the first statement, from (1.65) for the second one, and from both (1.64) and (1.65) for the third one) and the injectivity of $id(\cdot)$ from Def. 1.1.13. □

Corollary 1.3.18 (Send–receive causality)**.** *For any $\tau_{P_\epsilon,P}$-transitional run r with a $filter_\epsilon \subset$*

$filter_\epsilon^B$, for $i,j,k,l \in \mathcal{A}$, $\mu,\mu' \in Msgs$, $t,t' \in \mathbb{N}$, $A \in \{\text{🗓}\} \sqcup \overline{GActions_i}$, and $id \in \mathbb{N}$:

$$\begin{cases} gsend(i,j,\mu,id) \in \beta_i^t(r) \\ grecv(k,l,\mu',id) \in \overline{\beta}_{\epsilon_k}^{t'}(r) \end{cases} \implies \begin{cases} k=j \\ l=i \\ t' \geq t \\ \mu'=\mu \end{cases} \tag{1.71}$$

$$\begin{cases} fake\,(i, gsend(i,j,\mu,id) \mapsto A) \in \beta_{b_i}^t(r) \\ grecv(k,l,\mu',id) \in \overline{\beta}_{\epsilon_k}^{t'}(r) \end{cases} \implies \begin{cases} k=j \\ l=i \\ t' \geq t \\ \mu'=\mu \end{cases} \tag{1.72}$$

In other words, whether a message is sent correctly or faultily, the receipt of the message cannot happen before it was sent and the senders/recipients/content at the time of receipt must match those at the time of sending.

Proof. The statements follow from Lemma 1.3.15 and the injectivity of $id(\cdot)$ from Def. 1.1.13. □

Remark 1.3.19. While GMIs for sent messages are unforgeable for all transition templates, the correctness of GMIs for correctly received messages relies on the filtering performed by the general Byzantine environment filter and any stricter filters. It could be argued that Byzantine behavior can be strengthened and/or reliability of the communication channel can be weakened to enable Byzantine agents to forge GMIs, but this is outside the scope of this report, especially given that the man in the middle attack can be represented without forged GMIs (see Remark 1.1.5 for details).

In order to discuss what it means to behave the same way at a node $(i,t) \in \mathcal{A} \times \mathbb{N}$ in two distinct runs r and r' we define the notion of *agreement*:

Definition 1.3.20 (Agreement on a node)**.** For two runs r and r' from R, we say that r and r' **agree on** a node $(i,t) \in \mathcal{A} \times \mathbb{N}$ iff

1. $r_i(t) = r'_i(t)$
2. $\beta_{\epsilon_i}^t(r) = \beta_{\epsilon_i}^t(r')$
3. $\beta_i^t(r) = \beta_i^t(r')$

We extend this notion to sets of nodes $X \subset \mathcal{A} \times \mathbb{N}$: runs r and r' agree on X iff

$$\big(\forall (i,t) \in X\big) \;\; r \text{ and } r' \text{ agree on } (i,t)$$

Remark 1.3.21. In the above definition, Requirement 1 states that the local states of i at t are identical in both runs. Requirement 3 expresses that actions of i chosen by the adversary based on the protocol for the upcoming round $t.5$ are identical in both runs. Requirement 2 ensures three properties: correct external events imposed on i in round $t.5$ are identical, there is no difference as to whether i is awoken for round $t.5$ or not, faulty behavior of i in round $t.5$ is identical.

Lemma 1.3.22. *For two transitional runs r and r' and a node $(i,t) \in \mathcal{A} \times \mathbb{N}$*

$$r \text{ and } r' \text{ agree on } (i,t) \qquad \text{implies} \qquad r_i(t+1) = r'_i(t+1)$$

Proof. By (1.57) and (1.39). □

Remark 1.3.23. For two transitional runs r and r' and a node $(i,t) \in \mathcal{A} \times \mathbb{N}$,

$$r \text{ and } r' \text{ agreeing on } (i,t) \quad \text{is strictly stronger than} \quad \begin{cases} r_i(t) = r'_i(t) \\ r_i(t+1) = r'_i(t+1) \end{cases}$$

The most important case when local states remain in sync between timestamps t and $t+1$ despite different things happening during round $t.5$ is when a correct action/event in run r is replaced with its Byzantine version in run r'.

While it is preferrable to directly build desired properties of runs into the transition functions, in a manner of speech, to hardwire them, there are characteristics that cannot be implemented on a round-by-round basis. The most familiar of them is the *liveness condition* that requires that certain things happen *eventually* in a run. If no bound on the delay is given, this requirement cannot be translated into local terms because this is the property of the whole infinite run. Therefore, to enforce such properties we have to restrict the set of runs being considered.

Definition 1.3.24 (Admissibility condition)**.** An **admissibility condition** Ψ is any subset of the set R of all runs.

Now we have all the ingredients to define sets of runs for particular communication models. A **context** is essentially an extended environment where a **joint protocol** is executed.

Definition 1.3.25 (Context)**.** A **context**

$$\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi) \tag{1.73}$$

consists of an environment protocol $P_\epsilon \in \mathscr{C}_\epsilon$, a set of global initial states $\mathscr{G}(0)$, a transition template τ, and an admissibility condition Ψ.

Definition 1.3.26 (Agent-context)**.** Given a context γ and joint protocol P, we can combine them in an **agent-context** $\chi = (\gamma, P)$.

Definition 1.3.27 (Consistency)**.** For a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi)$ and a joint protocol P, we define the set of runs **weakly consistent** with P in γ (or weakly consistent with $\chi = (\gamma, P)$), denoted $R^{w\chi} = R^{w(\gamma,P)}$, to be the set of $\tau_{P_\epsilon,P}$-transitional runs that start at some global initial state from $\mathscr{G}(0)$:

$$R^{w(\gamma,P)} \quad := \quad \{r \in R \quad | \quad r(0) \in \mathscr{G}(0) \quad \text{and} \quad (\forall t \in \mathbb{N})\, r(t+1) \in \tau_{P_\epsilon,P}(r(t)) \quad \} \tag{1.74}$$

A run r is called **strongly consistent**, or simply **consistent**, with P in γ (or with χ) if it is weakly consistent with P in γ and, additionally, satisfies the admissibility condition: $r \in \Psi$. We denote the system of all runs consistent with P in γ by

$$R^{(\gamma,P)} := R^{w(\gamma,P)} \cap \Psi. \tag{1.75}$$

We say that an agent-context $\chi = (\gamma, P)$ is **non-excluding** if any prefix of a run weakly consistent with P in γ can be extended to a run strongly consistent with P in γ.

Definition 1.3.28 (Non-excluding agent-context)**.** For an agent-context χ, χ is non-excluding iff

$$R^\chi \neq \varnothing \quad \text{and} \quad (\forall r \in R^{w\chi})(\forall t \in \mathbb{N})(\exists r' \in R^\chi)(\forall t' \leq t)\, r'(t') = r(t')$$

The full formalism will be introduced in Sect. 1.4.

A local state of a run is called *coherent* if the agent could have arrived at the same local state without exhibiting any Byzantine behavior.

Definition 1.3.29 (Coherence)**.** For an agent-context χ, a run $r \in R^\chi$, and a node $(i,t) \in \mathcal{A} \times \mathbb{N}$, we say the local state $r_i(t)$ is χ-**coherent with respect to** i, or simply **coherent with respect to** i, iff

$$(\exists r' \in R^\chi)(\exists t' \in \mathbb{N})\Big(r'_i(t') = r_i(t) \text{ and } i \notin \mathcal{A}(Failed(r', t'))\Big).$$

Definition 1.3.30 (Failure free)**.** For an agent-context χ, a run $r \in R^\chi$, and a node $(i,t) \in \mathcal{A} \times \mathbb{N}$, we say the local state $r_i(t)$ is χ-**failure free with respect to** i, or simply **failure free with respect to** i, iff

$$(\exists r' \in R^\chi)(\exists t' \in \mathbb{N})\Big(r'_i(t') = r_i(t) \text{ and } \mathcal{A}(Failed(r', t')) = \varnothing\Big).$$

1.4 Syntax and Semantics

We define a formal language and its semantics in order to express knowledge of an agent in distributed systems. We will be using the standard adaptation of Kripke models to the run-based environment. Kripke models are based on abstract worlds or states supplied with the indistinguishability relations for the agents. For a collection of runs, it is quite natural to consider states to be various global states achievable during these runs and define the indistinguishability relation for an agent based on its knowledge of the local state: global states are indistinguishable for agent $i \in \mathcal{A}$ if and only if i's local state in these states is the same (i.e., the states are exactly the same from the point of view of agent i).

For multiple reasons, we consider the general set up in the form of agent-context to be common knowledge among agents. In other words, for an agent-context χ, the only possibilities agents consider are global states from various runs from R^χ. For instance, a synchronous agent who determined that it had skipped a round should conclude that it is compromised rather than imagining itself in an asynchronous context. By the same token, the same local state should give rise to different epistemic states depending on the type of distributed system. This is the reason why the Consensus problem with Byzantine failures *can* be solved in the synchronous context but *not* in the asynchronous one.

Definition 1.4.1 (Atomic propositions)**.** We consider an infinitely countable set Π of **atomic propositions**.

Definition 1.4.2 (Interpretation function)**.** An **interpretation function** $\pi\colon \mathscr{G} \to \{\bot, \top\}^\Pi$ assigns, for a given global state $h \in \mathscr{G}$, a propositional valuation function $\pi(h)\colon \Pi \to \{\bot, \top\}$.

Hence, for a global state $h \in \mathscr{G}$ the truth value $\pi(h)(p)$ of an atomic proposition $p \in \Pi$ is either $\bot$ (*false*) or $\top$ (*true*).

Definition 1.4.3 (Interpreted system)**.** A set $R' \subset R$ of runs and an interpretation function π yield an **interpreted system** $I = (R', \pi)$. For an agent-context $\chi = (\gamma, P)$, an interpreted system (R', π) is called **weakly χ-based** if $R' = R^{w(\chi)}$ and **χ-based** if $R' = R^\chi$.

Definition 1.4.4 (Indistinguishability relation)**.** For agent $i \in \mathcal{A} = [\![1; n]\!]$, the **indistinguishability relation** $\sim_i \subset \mathscr{G}^2$ is formally defined as follows:

$$\sim_i := \{\ (h, h') \mid \pi_{i+1} h = \pi_{i+1} h'\} \tag{1.76}$$

In other words, agent i cannot distinguish between global histories $h = (h_\epsilon, h_1, \ldots, h_n)$ and $h' = (h'_\epsilon, h'_1, \ldots, h'_n)$ iff $h_i = h'_i$, i.e., i sees exactly the same local history at h and h'.

Remark 1.4.5. Generally, for a particular R', the interpretation functions π and the indistinguishability relation $\sim_i$ are also defined for global states only appearing in the runs from $R \setminus R'$. This creates no problems but makes the formalism simpler.

We define a **language** $\mathfrak{L}$ to deal with the expression of knowledge in a *system*. For this we extend the propositional logic with the following operators:

1. three modal operators:
   - K_i for each agent $i \in \mathcal{A}$. For a global state $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, the formula $K_i\varphi$ can be read as "agent i knows φ (based on its local state h_i)": this means that, in every global state indistinguishable from h for i, the proposition φ holds;
   - E_G for each group $G \subset \mathcal{A}$ of agents. It means that "everyone in the group G of agents knows φ (based on their respective local states)." E_G is naturally defined to be the conjunction of all operators K_i over $i \in G$. We generally assume $G \neq \varnothing$ unless stated otherwise;

- C_G for each group $G \subset \mathcal{A}$ of agents. It means that "φ is common knowledge among the agents of G," i.e., everyone in G knows that everyone in G knows ... that everyone in G knows φ. We generally assume $G \neq \varnothing$ unless stated otherwise;

2. one temporal operator:

   - $\Box$ is the "always in the future" operator. It expresses statements like "the sender will never forget that he has sent *Hello*."[12]

Definition 1.4.6. For an agent $i \in \mathcal{A}$, a group of agents $\varnothing \neq G \subset \mathcal{A}$ and an atomic proposition $p \in \Pi$, the **language** $\mathfrak{L}$ is generated by the following BNF specification

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid C_G\varphi \mid \Box\varphi$$

We define the remaining Boolean connectives such as $\vee$, $\rightarrow$, and $\leftrightarrow$ in the standard way.[13] Mutual knowledge E_G and iterated mutual knowledge E_G^m are defined by

$$E_G\varphi := \bigwedge_{i\in G} K_i\varphi \qquad E_G^0\varphi := \varphi \qquad E_G^{n+1}\varphi := E_G E_G^n\varphi$$

In addition, each modal operator $\heartsuit$ has its dual $\neg\heartsuit\neg$. The duals of epistemic operators are denoted by putting $\hat{}$ over the operator, e.g., $\hat{K}_i\varphi = \neg K_i\neg\varphi$. The dual of $\Box$ is traditionally denoted by $\Diamond$.[14] Note that $E_G^1\varphi = E_G\varphi$ is syntactically the same formula.

To simplify the definition of truth, we define the following binary relations on the set of global states, using the standard notion of relation composition for binary relations $\star$ and $*$:

$$\star \circ * := \big\{(x,z) \mid (\exists y)\big(x \star y \wedge y * z\big)\big\}$$

Definition 1.4.7. Other binary relations on $\mathscr{G}$ are defined as follows:

$$\sim_G := \bigcup_{i\in G} \sim_i, \qquad \sim_G^0 := \{(h,h) \mid h \in \mathscr{G}\},$$

$$\sim_G^m := \underbrace{\sim_G \circ \cdots \circ \sim_G}_{m} \text{ (for } m > 1), \qquad \sim_G^C := \bigcup_{m=1}^{\infty} \sim_G^m$$

With the language $\mathfrak{L}$ we can express statements about the knowledge of an agent (or of a group of agents) or about the temporal properties of a formula. The semantics with respect to interpreted systems is as follows:

Definition 1.4.8. For an interpreted system $I = (R', \pi)$ with the set $R' \subset R$ of runs and the interpretation function π, for an agent $i \in \mathcal{A}$, a group of agents $\varnothing \neq G \subset \mathcal{A}$, a run $r \in R'$, and a timestamp $t \in \mathbb{N}$:

$$\begin{array}{lll}
(I,r,t) \models p & \text{iff} & \pi(r(t))(p) = \top \\
(I,r,t) \models \neg\varphi & \text{iff} & (I,r,t) \not\models \varphi \\
(I,r,t) \models \varphi \wedge \varphi' & \text{iff} & (I,r,t) \models \varphi \text{ and } (I,r,t) \models \varphi' \\
(I,r,t) \models K_i\varphi & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})\Big(r'(t') \sim_i r(t) \Rightarrow (I,r',t') \models \varphi\Big) \\
(I,r,t) \models C_G\varphi & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})\Big(r'(t') \sim_G^C r(t) \Rightarrow (I,r',t') \models \varphi\Big) \\
(I,r,t) \models \Box\varphi & \text{iff} & (\forall t' \geq t)\,(I,r,t') \models \varphi \\
(I,r,t) \models Y\varphi & \text{iff} & (t > 0) \text{ and } (I,r,t-1) \models \varphi
\end{array}$$

[12] In temporal logic, this operator is usually denoted G.

[13] We also use the common ranking of binding strength: $\neg$ is the strongest, then $\vee$ and $\wedge$ which bind equally strong, then $\rightarrow$, and the weakest is $\leftrightarrow$.

[14] In temporal logic, it is usually denoted F.

Note that the "yesterday" modality satisfies $(I, r, 0) \not\models Y\varphi$ for any φ.

It immediately follows from the definition, based on the meaning of the secondary connectives, that

$$
\begin{array}{lll}
(I,r,t) \models \varphi \vee \varphi' & \text{iff} & (I,r,t) \models \varphi \ \text{ or } (I,r,t) \models \varphi' \\
(I,r,t) \models \varphi \to \varphi' & \text{iff} & (I,r,t) \not\models \varphi \ \text{ or } (I,r,t) \models \varphi' \\
(I,r,t) \models E_G\varphi & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})\Big(r'(t') \sim_G r(t) \ \Rightarrow\ (I,r',t') \models \varphi\Big) \\
 & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})(\forall i \in G)\Big(r'(t') \sim_i r(t) \ \Rightarrow\ (I,r',t') \models \varphi\Big) \\
(I,r,t) \models E^0_G\varphi & \text{iff} & (I,r,t) \models \varphi \\
(I,r,t) \models E^m_G\varphi & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})\Big(r'(t') \sim^m_G r(t) \ \Rightarrow\ (I,r',t') \models \varphi\Big) \\
 & \text{iff} & (\forall r^0,\ldots,r^m \in R')(\forall t_0,\ldots,t_m \in \mathbb{N})\Big(r^0 = r \wedge t_0 = t \wedge \\
 & & \qquad (\forall k < m)(\exists i_k \in G) r^k(t_k) \sim_{i_k} r^{k+1}(t_{k+1}) \quad \Rightarrow \quad (I,r^m,t_m) \models \varphi\Big) \\
(I,r,t) \models C_G\varphi & \text{iff} & (\forall m \in \mathbb{N} \setminus \{0\})(I,r,t) \models E^m_G\varphi \\
 & \text{iff} & (\forall m \in \mathbb{N} \setminus \{0\})(\forall r^0,\ldots,r^m \in R')(\forall t_0,\ldots,t_m \in \mathbb{N})\Big(r^0 = r \wedge t_0 = t \wedge \\
 & & \qquad (\forall k < m)(\exists i_k \in G) r^k(t_k) \sim_{i_k} r^{k+1}(t_{k+1}) \quad \Rightarrow \quad (I,r^m,t_m) \models \varphi\Big) \\
(I,r,t) \models \Diamond\varphi & \text{iff} & (\exists t' \geq t)\,(I,r,t') \models \varphi
\end{array}
$$

It is also easy to see that truth is defined with respect to global histories rather than points in a run, i.e., for any formula φ, we have

$$r(t) = r'(t) \qquad \Rightarrow \qquad (\forall \varphi \in \mathfrak{L})\Big((I,r,t) \models \varphi \quad \Leftrightarrow \quad (I,r',t') \models \varphi\Big)$$

even though $r(t+1)$ may differ from $r'(t+1)$. (Note that $r(t) \neq r'(t')$ for any $t \neq t'$ because the length of the environment's history is a function of time.)

1.5 Atomic Propositions

We have defined the language $\mathfrak{L}$ as the syntax and the associated semantics to tell the truth value of a formula for a given interpreted system I, run $r \in R$ and timestamp $t' \in \mathbb{N}$. We now designate some of the atomic propositions from Π as special and consider their truth values to be fully determined by $r(t')$ rather than arbitrary. In other words, we will restrict interpretations π so as to adhere to the following intended meanings for a given $r(t')$ with $t \leq t'$ and $i \in \mathcal{A}$. We also introduce useful abbreviations for negations of some atomic propositions.

- $correct_{(i,t)}$ states that by timestamp $t \in \mathbb{N}$, i.e., in rounds $0.5, 1.5, \ldots, (t-1).5$, agent $i \in \mathcal{A}$ did not violate its protocol through improper action or improper inaction, i.e., did not exhibit any Byzantine actions or events and was not marked with $sleep\,(i)$ or $hibernate\,(i)$ from timestamp 0 to timestamp t.
- $correct_i$ states that by the time of evaluation ($t' \in \mathbb{N}$, i.e., in rounds $0.5, 1.5, \ldots, (t'-1).5$) agent $i \in \mathcal{A}$ did not violate its protocol through improper action or improper inaction.
- $faulty_{(i,t)} := \neg correct_{(i,t)}$ states that by timestamp $t \in \mathbb{N}$ agent $i \in \mathcal{A}$ violated its protocol through improper action or improper inaction.
- $faulty_i := \neg correct_i$ states that by the time of evaluation agent $i \in \mathcal{A}$ violated its protocol through improper action or improper inaction.

- $fake_{(i,t)}(o)$ states that agent $i \in \mathcal{A}$ thinks that $o \in \overline{Actions} \sqcup \overline{Events}$ occurred in round $(t-1).5$ but thinks so for a wrong reason, i.e., $o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. Note that in the Byzantine setting, if o is an action, it is still possible that agent i did perform o in round $(t-1).5$ but mistook it for another action o'', while at the same time mistaking some third action o' for o, e.g., when $fake(i, O' \mapsto O), fake(i, O \mapsto O'') \in \beta_{b_i}^{t-1}(r)$. It is also possible that action o was performed according to the protocol, e.g., when $fake(i, O' \mapsto O) \in \beta_{b_i}^{t-1}(r)$ and $O \in \beta_i^{t-1}(r)$.
- $\overline{occurred}_{(i,t)}(o)$ states that agent i thinks that $o \in \overline{Actions} \sqcup \overline{Events}$ occurred in round $(t-1).5$ for a right reason, i.e., $o \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. Note that in the Byzantine setting, if o is an action, it is possible that agent i has both a right and a wrong reason to believe it performed o in round $(t-1).5$, e.g., when both $O \in \beta_i^{t-1}(r)$ and $fake(i, O' \mapsto O) \in \beta_{b_i}^{t-1}(r)$. In this case, agent i unwittingly piggybacks O' onto the action O. For instance, one might accidentally throw away a postcard with an unwanted catalog, or a ticket-vending machine might accidentally print two tickets instead of one.
- $\overline{occurred}_i(o)$ states that by the time t' of evaluation, agent i correctly registered $o \in \overline{Actions} \sqcup \overline{Events}$ occurring in some previous round, i.e., $(\exists t < t')\, o \in label^{-1}\left(\beta_i^t(r) \sqcup \overline{\beta}_{\epsilon_i}^t(r)\right)$.
- $occurred_i(o)$ states that by the time of evaluation agent i believes that $o \in \overline{Actions} \sqcup \overline{Events}$ occurred.

We now give formal definitions and discuss properties of these atomic propositions.

Definition 1.5.1. A (weakly) χ-based interpreted system $I = (R', \pi)$ and its interpretation π are called **proper** if, for any run $r \in R'$, any agent $i \in \mathcal{A}$, arbitrary two timestamps $t \leq t'$, and any $o \in \overline{Actions} \sqcup \overline{Events}$, the interpretation π satisfies the following properties:

$$\pi(r(t'))\left(correct_{(i,t)}\right) = \top \quad \text{iff} \quad (i,t) \notin Failed(r,t') \tag{1.77}$$

$$\pi(r(t'))(correct_i) = \top \quad \text{iff} \quad (i,t') \notin Failed(r,t') \tag{1.78}$$

$$\pi(r(t'))\left(fake_{(i,t)}(o)\right) = \top \quad \text{iff} \quad t \geq 1 \text{ and } o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right) \tag{1.79}$$

$$\pi(r(t'))\left(\overline{occurred}_{(i,t)}(o)\right) = \top \quad \text{iff} \quad t \geq 1 \text{ and } o \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right) \tag{1.80}$$

$$\pi(r(t'))\left(\overline{occurred}_i(o)\right) = \top \quad \text{iff} \quad (\exists t < t')\, o \in label^{-1}\left(\beta_i^t(r) \sqcup \overline{\beta}_{\epsilon_i}^t(r)\right) \tag{1.81}$$

$$\pi(r(t'))(occurred_i(o)) = \top \quad \text{iff} \quad o \in r_i(t') \tag{1.82}$$

Proposition 1.5.2. *The following truth values coincide in all weakly (strongly) χ-interpreted systems:*[15]

$$\begin{aligned}
\pi(r(t'))(correct_i) &= \pi(r(t'))\left(correct_{(i,t')}\right)\\
(I,r,t') \models faulty_i &\Leftrightarrow (I,r,t') \models faulty_{(i,t')}\\
(I,r,t') \models faulty_{(i,t)} &\Leftrightarrow (i,t) \in Failed(r,t')\\
(I,r,t') \models faulty_i &\Leftrightarrow (i,t') \in Failed(r,t')\\
\pi(r(t'))\left(\overline{occurred}_i(o)\right) &= \bigvee_{t=1}^{t'} \pi(r(t'))\left(\overline{occurred}_{(i,t)}(o)\right)\\
\pi(r(t'))(occurred_i(o)) &= \bigvee_{t=1}^{t'} \left(\pi(r(t'))\left(\overline{occurred}_{(i,t)}(o)\right) \vee \pi(r(t'))\left(fake_{(i,t)}(o)\right)\right)
\end{aligned}$$

Remark 1.5.3. Although these atomic propositions are objective properties, which are typically imperceptible for agents, some of them are formulated for locally representated actions/events o because they represent objective properties of agents' subjective views.

[15] $\top \vee \top = \top \vee \bot = \bot \vee \top = \top$ and $\bot \vee \bot = \bot$.

Note that no conditions are postulated for such atomic propositions if $t > t'$. This is due to the fact that π is defined on finite global histories rather than on infinite runs. The global history $r(t')$ at timestamp t' contains no information about later timestamps $t > t'$. Indeed, there generally exist multiple $\tau_{P_\epsilon,P}$-transitional runs extending the global history $r(t')$, due to the non-deterministic capabilities of the adversary. Since the run r cannot be singled out based on $r(t')$ only, only the features of r already present in $r(t')$ can be relied upon.

Note also that, using (1.60)–(1.63), we could have easily given the same definitions in terms of global histories h rather than considering them as prefixes $r(t')$ of a transitional run r. The latter is simply what we are interested in.

Remark 1.5.4. Agents can only record their own actions and events: if an agent believes something happened, it could happen to this agent in principle.

$$\begin{aligned}
\pi(r(t'))\left(fake_{(i,t)}(o)\right) &= \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i \\
\pi(r(t'))\left(\overline{occurred}_{(i,t)}(o)\right) &= \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i \\
\pi(r(t'))\left(\overline{occurred}_i(o)\right) &= \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i \\
\pi(r(t'))\left(occurred_i(o)\right) &= \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i
\end{aligned}$$

The omniscient environment does not forget. Note that this is independent of whether agents have perfect recall because $\beta_i(h)$ is defined in (1.60) based on the environment's history.

Lemma 1.5.5. *Consider an agent-context χ, a proper (weakly) χ-based interpreted system $I = (R', \pi)$, some $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R'$, a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, a timestamp $t' \geq t$, and:*

$$\begin{aligned}
(I,r,t') \models \quad & correct_\theta \quad \leftrightarrow \quad \Box correct_\theta \\
(I,r,t') \models \quad & faulty_\theta \quad \leftrightarrow \quad \Box faulty_\theta \\
(I,r,t') \models \quad & fake_\theta(o) \quad \leftrightarrow \quad \Box fake_\theta(o) \\
(I,r,t') \models \quad & \overline{occurred}_\theta(o) \quad \leftrightarrow \quad \Box\overline{occurred}_\theta(o) \\
(I,r,t') \models \quad & faulty_i \quad \leftrightarrow \quad \Box faulty_i \\
(I,r,t') \models \quad & \overline{occurred}_i(o) \quad \leftrightarrow \quad \Box\overline{occurred}_i(o) \\
(I,r,t') \models \quad & occurred_i(o) \quad \leftrightarrow \quad \Box occurred_i(o)
\end{aligned}$$

Proof. The direction from right to left is trivial in all cases because $t' \geq t'$, hence being true at t' is part of being true in all futures of t'.

From left to right, for the first four statements, the truth is based on a particular event/action, correct or Byzantine, occurring in the global run at round $(t-1).5$, at a specific past of t', whereas for the remaining three equivalences something must have happened at an unspecified past of t'. Since all futures of t' lie to the future of this event/action and the round enumeration remains stable, the requisite event remains in the global run. □

Remark 1.5.6. Note that, unlike the atomic propositions from Lemma 1.5.5, atoms $correct_i$ are based on certain kinds of events/actions *not* having occurred yet. Thus, they may not be preserved temporally.

Further, the first four equivalences from Lemma 1.5.5 are not universal validities as they rely on $t' \geq t$. Indeed, for $t > t'$ the truth value of these atoms is not restricted, in particular, it does not depend on the run and can change arbitrarily with time.

Agents cannot both observe an event and be mistaken about observing it. More formally,

Lemma 1.5.7. *Consider a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi)$, a proper weakly (strongly) χ-based interpreted system $I = (R', \pi)$, an agent-context $\chi = (\gamma, P)$, an event $e \in \overline{Events}$, a run $r \in R'$, a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, and a timestamp $t' \geq t$:*

- $(I, r, t') \models \overline{occurred}_\theta(e) \to \neg fake_\theta(e)$
- $(I, r, t') \models fake_\theta(e) \to \neg\overline{occurred}_\theta(e)$

Proof. To prove the first implication, assume $(I, r, t') \models \overline{occurred}_\theta(e)$. By the definition of properness, this means that $e \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. Since e is an event, the only option is $e \in label^{-1}\left(\overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$, i.e., there must exist $E \in \overline{\beta}_{\epsilon_i}^{t-1}(r) \subset \beta_\epsilon^{t-1}(r)$ such that $e = label^{-1}(E)$. Our goal is to show that $e \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. The situation splits into two cases:

Case I: $e \in Ext_i$ is an external event. Then $E = external(i, e)$. By the $(t-1)$-coherence of $\beta_\epsilon^{t-1}(r) \supset \beta_{b_i}^{t-1}(r)$ we have $fake(i, external(i, e)) \notin \beta_{b_i}^{t-1}(r)$. Hence, $e \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$.
Case II: $e = recv(j, \mu)$ is a message delivery. Then $E = grecv(i, j, \mu, id)$ for some $id \in \mathbb{N}$ (this id cannot be entirely arbitrary but this is irrelevant for the proof). By the $(t-1)$-coherence of $\beta_\epsilon^{t-1}(r) \supset \beta_{b_i}^{t-1}(r)$ we have $fake(i, grecv(i, j, \mu, id')) \notin \beta_{b_i}^{t-1}(r)$ for any $id' \in \mathbb{N}$. Hence, $e \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$.

We have demonstrated that $(I, r, t') \models \overline{occurred}_\theta(e) \to \neg fake_\theta(e)$. Now the second implication $(I, r, t') \models fake_\theta(e) \to \neg\overline{occurred}_\theta(e)$ follows by contraposition. □

Remark 1.5.8. Needless to say, the absence of a correct (Byzantine) occurrence does not mean that there was a Byzantine (correct) one.

Remark 1.5.9. The same statement does not apply to actions a. For instance, the correct internal action $internal(i, a)$ is generally compatible with a Byzantine action $fake(i, A' \mapsto internal(i, a))$ the agent mistakes for a.

Using these atomic propositions with fixed evaluations, we can define derived concepts with similarly fixed meanings. For instance, the **absolute occurrence** represents information about local actions and events accessible only for the environment.

Definition 1.5.10 (Absolute occurrence)**.** Consider any integer $k \geq 1$ and any $o \in \overline{Actions} \sqcup \overline{Events}$,

$$\overline{occurred}^{(k)}(o) := \bigvee_{\substack{S \subset \mathcal{A} \\ |S| = k}} \bigwedge_{i \in S} \overline{occurred}_i(o) \tag{1.83}$$

$$\overline{occurred}(o) := \overline{occurred}^{(1)}(o) \tag{1.84}$$

We now define several notions related to **relative occurrence** $occurred_{(i,t)}(o)$ that represents agents' information about the same events:

Definition 1.5.11 (Relative occurrence)**.** Consider any agent $i \in \mathcal{A}$, any timestamp $t \in \mathbb{N}$, and any integer $k \geq 1$. For any $o \in \overline{Actions} \sqcup \overline{Events}$,

$$occurred_{(i,t)}(o) := \overline{occurred}_{(i,t)}(o) \vee fake_{(i,t)}(o) \tag{1.85}$$

$$occurred^{(k)}(o) := \bigvee_{\substack{S \subset \mathcal{A} \\ |S| = k}} \bigwedge_{i \in S} occurred_i(o) \tag{1.86}$$

$$occurred(o) := occurred^{(1)}(o) \tag{1.87}$$

Informally speaking,

- $\overline{occurred}(o)$ says that some non-Byzantine version of o happened for at least one agent;
- $\overline{occurred}^{(k)}(o)$ says that some non-Byzantine versions of o happened for at least k distinct agents.

- $occurred_{(i,t)}(o)$ says that some version of event/action o was entered into agent i's history at local timestamp (i,t), i.e., during round $(t-1).5$;
- $occurred(o)$ says that at least one agent believes some version of o happened;
- $occurred^{(k)}(o)$ says that at least k distinct agents believe some versions of o happened.

In all the three last cases agents are not aware of whether o was correct or Byzantine, and whether it really happened or they imagine it did; in the case of k agents, a mixture of correct and Byzantine entries also satisfies the conditions.

Remark 1.5.12. Note that $occurred^{(k)}(o)$ (resp $\overline{occurred}^{(k)}(o)$) requires the existence of *some* k distinct agents. In particular, in order to fulfill $K_i occurred^{(k)}(o)$ (resp. $K_i\overline{occurred}^{(k)}(o)$), it is not necessary that the same k agents observe o in all global states i considers possible. It is sufficient that in each such possible state, there be a group of k agents who have observed o.

The following is a direct corollary of Lemma 1.5.5.

Remark 1.5.13. Consider an agent-context χ, a proper weakly (strongly) χ-based interpreted system $I=(R',\pi)$, some $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R'$, a node $\theta=(i,t)\in \mathcal{A}\times\mathbb{N}$, and a timestamp $t' \geq t$,

$$\begin{aligned}
(I,r,t') \models \quad & \overline{occurred}(o) \quad \leftrightarrow \quad \Box\overline{occurred}(o)\\
(I,r,t') \models \quad & \overline{occurred}^{(k)}(o) \quad \leftrightarrow \quad \Box\overline{occurred}^{(k)}(o)\\
(I,r,t') \models \quad & occurred_{(i,t)}(o) \quad \leftrightarrow \quad \Box occurred_{(i,t)}(o)\\
(I,r,t') \models \quad & occurred(o) \quad \leftrightarrow \quad \Box occurred(o)\\
(I,r,t') \models \quad & occurred^{(k)}(o) \quad \leftrightarrow \quad \Box occurred^{(k)}(o)
\end{aligned}$$

Lemma 1.5.14. *Consider a context $\gamma=(P_\epsilon,\mathscr{G}(0),\tau,\Psi)$, an agent-context $\chi=(\gamma,P)$, a proper weakly (strongly) χ-based interpreted system $I=(R',\pi)$, some $o\in\overline{Actions}\sqcup\overline{Events}$, a run $r\in R'$, a node $(i,t)\in\mathcal{A}\times\mathbb{N}$, and a timestamp $t'\geq t$.*

$$(I,r,t') \models occurred_{(i,t)}(o) \qquad \textit{iff} \qquad t\geq 1 \textit{ and } (\exists\lambda)\big(r_i(t)=\lambda : r_i(t-1) \textit{ and } o\in\lambda\big)$$

Proof. First we prove the direction from left to right. Assume $(I,r,t')\models occurred_{(i,t)}(o)$, i.e.,

$$(I,r,t') \models \overline{occurred}_{(i,t)}(o) \vee fake_{(i,t)}(o)$$

It is clear that $t\geq 1$.

Case I: $(I,r,t')\models\overline{occurred}_{(i,t)}(o)$. Then $o\in label^{-1}\left(\beta_i^{t-1}(r)\sqcup\overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. Thus, $o=label^{-1}(O)$ for some $O\in\beta_i^{t-1}(r)\sqcup\overline{\beta}_{\epsilon_i}^{t-1}(r)\subset\left(\beta_i^{t-1}(r)\sqcup\beta_{\epsilon_i}^{t-1}(r)\right)\cap\left(\overline{GActions}\sqcup\overline{GEvents}\right)$. Additionally, by Prop. 1.2.25, if O is an action from $\beta_i^{t-1}(r)$, then by Prop. 1.2.25,

$$aware(i,\beta_\epsilon^{t-1}(r)). \tag{1.88}$$

So by (1.57), definition (1.39) of $update_i$, and definition (1.32) of σ we conclude that $r_i(t)=\lambda : r_i(t-1)$ and $o=label^{-1}(O)\in\lambda$.

Case II: $(I,r,t')\models fake_{(i,t)}(o)$. Then $o\in\sigma\left(\beta_{b_i}^{t-1}(r)\right)$. It remains to note that $\sigma\left(\beta_{b_i}^{t-1}(r)\right)\neq\varnothing$ implies that

$$\sigma\left(\beta_{\epsilon_i}^{t-1}(r)\right)\neq\varnothing \tag{1.89}$$

and $r_i(t)=\lambda : r_i(t-1)$. Once again, the definition (1.39) of $update_i$ implies $o\in\lambda$.

We proved that in either case $r_i(t)=\lambda\colon r_i(t-1)$ and $o\in\lambda$.

Now we demonstrate the opposite direction from right to left. Assume $r_i(t)=\lambda\colon r_i(t-1)$ and $o\in\lambda$. By definition (1.39) of $update_i$ it means that either (1.88) or (1.89) holds.

Case I: $o \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. We have $(I,r,t') \models \overline{occurred}_{(i,t)}(o)$, and, hence, $(I,r,t') \models occurred_{(i,t)}(o)$.

Case II: $o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. We have $(I,r,t') \models fake_{(i,t)}(o)$, and, hence, $(I,r,t') \models occurred_{(i,t)}(o)$

We proved that in either case $(I,r,t') \models occurred_{(i,t)}(o)$. □

Lemma 1.5.15. *Consider a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi)$, an agent-context $\chi = (\gamma, P)$, a proper weakly (strongly) χ-based interpreted system $I = (R', \pi)$, some $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R'$, an agent $i \in \mathcal{A}$, and a timestamp $t \in \mathbb{N}$.*

$$\begin{aligned}(I,r,t) &\models \quad occurred_i(o) \quad \leftrightarrow \quad K_i\, occurred_i(o)\\ (I,r,t) &\models \quad \neg occurred_i(o) \quad \leftrightarrow \quad K_i \neg occurred_i(o)\end{aligned}$$

Proof. The directions from right to left are trivial because the indistinguishability relation $\sim_i$ is reflexive. We prove the direction from left to right for the case of $(I,r,t) \models occurred_i(o)$ as the other statement is completely analogous. For any $(r',t') \in R' \times \mathbb{N}$ such that $r(t) \sim_i r'(t')$, i.e., $r_i(t) = r'_i(t')$, we have

$$(I,r,t) \models occurred_i(o) \iff o \in r_i(t) \iff o \in r'_i(t') \iff (I,r',t') \models occurred_i(o)$$

□

Formulas $occurred_i(o)$ and $\overline{occurred}_i(o)$ represent events occurring in the system. As shown in Lemma 1.5.15, the former event is detectable by agent i and, hence, can be used by its protocol, whereas the latter may not be detectable by any agents but is fully determined by the global state, i.e., "detectable" by the environment. Following [FHMV99], we define conditions under which formulas can be treated as events:

Definition 1.5.16. A formula φ is called an **i-internal event (within an agent-context χ)** iff

$$r_i(t) = r'_i(t') \quad \Longrightarrow \quad \big((I,r,t) \models \varphi \iff (I,r',t') \models \varphi\big)$$

for all χ-based interpreted systems $I = (R', \pi)$, arbitrary runs $r, r' \in R'$, and arbitrary timestamps $t, t' \in \mathbb{N}$.

A formula φ is called a **state event (within an agent-context χ)** iff

$$r(t) = r'(t) \quad \Longrightarrow \quad \big((I,r,t) \models \varphi \iff (I,r',t) \models \varphi\big)$$

for all χ-based interpreted systems $I = (R', \pi)$, arbitrary runs $r, r' \in R'$, and any timestamp $t \in \mathbb{N}$.

Lemma 1.5.17. *For any i-internal event within an agent context χ and any χ-based interpreted system $I = (R', \pi)$, any run $r \in R'$, and any timestamp $t \in \mathbb{N}$.,*

$$\begin{aligned}(I,r,t) &\models \varphi \leftrightarrow K_i\varphi,\\ (I,r,t) &\models \neg\varphi \leftrightarrow K_i\neg\varphi,\end{aligned}$$

1.6 Past and Causality Relationship

For a run r the related *causal graph* represents all the causality dependencies of nodes in r. We say that there is a causality dependence between two nodes (i,t) and (j,t') iff the state of j at time t' in r can, in principle, depend on the state of i at time t. For instance when during round 0.5 agent i sends a message to j, which it receives during the same round 0.5, then all the future states of j after timestamp 1 contain the receipt of the message at node $(j,1)$ in the history and, hence, are directly affected by it and through it can be influenced by the node $(i,0)$.

Definition 1.6.1 (Causal graph)**.** For a run $r \in R$ and a timestamp $t \in \mathbb{N}$, we define the **causal graph of** r **at** t as

$$G(r,t) := \big(V(t),\ E(r,t)\big)$$

such that

$$V(t) := \mathcal{A} \times [\![0;t]\!] \tag{1.90}$$

$$E_{loc}(t) := \big\{\big((i,t'),(i,t'+1)\big) \mid (i,t') \in \mathcal{A} \times [\![0;t-1]\!]\big\} \tag{1.91}$$

$$E_{msg}(r,t) := \Big\{\big((i,t'),(j,t'')\big) \mid t'' \le t \quad \text{and} \quad (\exists \mu \in Msgs)(\exists id \in \mathbb{N})\Big(grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t''-1}(r) \quad \text{and} \quad \big(gsend(i,j,\mu,id) \in \beta_i^{t'}(r) \quad \text{or} \quad (\exists A \in \{\text{📅}\} \sqcup \overline{GActions_i})\, fake(i, gsend(i,j,\mu,id) \mapsto A) \in \beta_{b_i}^{t'}(r)\big)\Big)\Big\} \tag{1.92}$$

$$E(r,t) := E_{loc}(t) \cup E_{msg}(r,t) \tag{1.93}$$

(strictly speaking, these two sets need not be disjoint because agents are not prohibited to send messages to themselves).

Remark 1.6.2. If $\big((i,t'),(j,t'')\big) \in E(r,t)$, then $t' < t'' \le t$.

Remark 1.6.3. The causal graph can be naturally defined from a Byzantine extension of *Lamport's Happened Before relation* and vice versa.

Definition 1.6.4 (Path)**.** For a run $r \in R$ and two nodes $\beta = (i,t_1)$ and $\theta = (j,t_2)$ from $\mathcal{A} \times \mathbb{N}$, we define the following notations for paths in the causal graph $G(r,t)$ of r at $t \ge \max(t_1,t_2)$.

We denote by $\to^{r,t}$ an **edge of the causal graph** $G(r,t)$:

$$\beta \to^{r,t} \theta \quad \text{iff} \quad (\beta,\theta) \in E(r,t), \tag{1.94}$$

and by $\rightsquigarrow_\xi^{r,t}$ a **finite path** ξ **in the causal graph** $G(r,t)$:

$$\beta \rightsquigarrow_\xi^{r,t} \theta \qquad \text{iff} \qquad \xi = \beta : \eta_1 : \cdots : \eta_k : \theta \text{ and } \beta \to^{r,t} \eta_1 \to^{r,t} \ldots \to^{r,t} \eta_k = \theta, \tag{1.95}$$

where $k \ge 0$ is the **length** of the path. In particular, every node β is connected to itself via the trivial path β (path of lengh 0)

We sometimes apply notions defined for sets of nodes to a path ξ to mean the application to the set of nodes of ξ. For instance,

$$\mathcal{A}((i_1,t_1) : \ldots : (i_k,t_k)) \qquad := \qquad \{i_1,\ldots,i_k\} = \mathcal{A}(\{(i_1,t_1),\ldots,(i_k,t_k)\}).$$

Other notions, when applied to a path, require minor modifications. For instance, intuitively, agents being correct along a path means that all the actions/events forming the path, notably all sends and receives, are necessarily correct. However, all nodes of the path being correct is a weaker requirement because an agent correct at a node on the path may turn Byzantine in exactly the next round when it sends the path-forming message, which may, in this case, be a Byzantine send. To account for sends as well are receives being necessarily correct along a **correct path** we present the following definition:

$$Failed_{(i_1,t_1) : \ldots : (i_{k-1},t_{k-1}) : (i_k,t_k)}(r(t)) \qquad := \qquad Failed_{\{(i_1,t_1+1),\ldots,(i_{k-1},t_{k-1}+1),(i_k,t_k)\}}(r(t)).$$

In particular, for a trivial path of length 0 connecting β to itself, $Failed_\beta(r(t)) = Failed_{\{\beta\}}(r(t))$. In other words, for the path to be correct, we require agents to be correct up to one round after each node on the path (except the last node need only be correct itself as no more sends are issued along the path).

Remark 1.6.5. We could have imposed a weaker condition that the path-forming sends be correct, leaving open the possibility of agents becoming Byzantine in the same round as sending these path-forming messages. However, it makes little sense to trust an agent's correct send if its other actions in the same round are provably faulty.

The **set of all paths of** $G(r,t)$ is denoted by $\Xi(r,t)$. We write $\beta \rightsquigarrow^{r,t} \theta$ if there exists a path $\xi \in \Xi(r,t)$ such that $\beta \rightsquigarrow_{\xi}^{r,t} \theta$. Thus, in particular, $\beta \rightsquigarrow^{r,t} \beta$ for any node $\beta \in G(r,t)$ because it is reachable from itself via the trivial path β of length 0. We say that an edge $e \in (\mathcal{A} \times \mathbb{N})^2$ is part of a path $\xi = \eta_0 : \eta_1 : \cdots : \eta_{k+1}$ and write $e \in \xi$ iff $e = (\eta_i, \eta_{i+1})$ for some $0 \leq i \leq k$. When talking about graphs on subsets of $\mathcal{A} \times \mathbb{N}$ independent of any particular run, we use $\rightsquigarrow_{\xi}$ to describe a path ξ, $\rightsquigarrow$ to postulate the existence of a path, and $\rightarrow$ to describe an edge.

Remark 1.6.6. If $(i,t') \rightsquigarrow^{r,t} (j,t'')$, then $t' \leq t'' \leq t$.

The causal cone of a node $\theta \in \mathcal{A} \times \mathbb{N}$ in a run $r \in R$ is the set of nodes that are part of the past of θ, i.e, nodes that can impact the state of θ.

Definition 1.6.7 (Past)**.** For a run $r \in R$, a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, we define the causal cone of θ in r as follows:

$$V_{(i,t)}(r) := \{\beta \in V(t) \mid \beta \rightsquigarrow^{r,t} (i,t)\} \tag{1.96}$$

Hence, we define the related edges as follows:

$$E_{(i,t)}(r) := \left\{ e \in E(r,t) \,\middle|\, \big(\exists \beta \in V(t)\big)\big(\exists \xi \in \Xi(r,t)\big)\left(\beta \rightsquigarrow_{\xi}^{r,t} (i,t) \text{ and } e \in \xi\right) \right\}. \tag{1.97}$$

Finally, the **causal cone**, or past, of θ in r is denoted by $Past_{\theta}(r)$ and formally defined as

$$Past_{\theta}(r) := \big(V_{\theta}(r), E_{\theta}(r)\big) \tag{1.98}$$

Lemma 1.6.8 (Correctness)**.** *The causal cone $Past_{(i,t)}(r)$ is the restriction of the causal graph $G(r,t)$ to the set $V_{(i,t)}(r)$ of all vertices that have a path to (i,t) (including (i,t) itself).*

Proof. We will sometimes use the abbreviation $\theta = (i,t)$.

$V_{(i,t)}(r)$ is the set of those vertices of the causal graph $G(r,t)$ that have a path to (i,t) by definition, cf. (1.96). In particular, $(i,t) \in V_{(i,t)}(r)$.

We show that

$$(\eta, \eta') \in E_{(i,t)}(r) \qquad \Longleftrightarrow \qquad (\eta, \eta') \in E(r,t) \text{ and } \eta, \eta' \in V_{\theta}(r).$$

By definition, cf. (1.97), each edge $e = (\eta, \eta') \in E_{(i,t)}(r)$ belongs to $E(r,t)$ and lies on some path of the form

$$\gamma \rightsquigarrow^{r,t} \eta \rightarrow^{r,t} \eta' \rightsquigarrow_{\xi}^{r,t} (i,t) \in \Xi(r,t).$$

Clearly, the paths ξ and

$$\eta \rightarrow^{r,t} \eta' \rightsquigarrow_{\xi}^{r,t} (i,t) \in \Xi(r,t) \tag{1.99}$$

connect the ends η' and η of edge e to (i,t). Thus, $\eta, \eta' \in V_{(i,t)}(r)$. This completes the proof from left to right. In particular, it shows that each edge of $Past_{(i,t)}(r)$ connects two of its vertices, making it a graph.

For the direction from right to left. If $\eta' \in V_{\theta}(r)$, then there must exist some path ξ from η' to (i,t). Prepending it with the edge $(\eta, \eta') \in E(r,t)$ yields a path (1.99) from η to (i,t). It remains to note that (η, η') is an edge on this latter path. □

The intuition tells us that if a node β is in the past of a node θ in some run r, then any past of β is also a past of θ. The next lemma states this property.

Lemma 1.6.9. *For a run $r \in R$ and a node $\theta \in \mathcal{A} \times \mathbb{N}$*

$$\beta \in V_{\theta}(r) \qquad \Longrightarrow \qquad Past_{\beta}(r) \text{ is a subgraph of } Past_{\theta}(r),$$

i.e., $V_{\beta}(r) \subset V_{\theta}(r)$ and $E_{\beta}(r) \subset E_{\theta}(r)$.

Proof. Let $\theta = (i,t)$ and $\beta = (j,t')$. First, let us prove that $V_\beta(r) \subset V_\theta(r)$. For any node $(k,t'') \in V_\beta(r)$ there must exist a path $(k,t'') \rightsquigarrow_{\xi_1}^{r,t'} (j,t')$ that consists of edges from $E(r,t') \subset E(r,t)$. Similarly, there exists a path $(j,t') \rightsquigarrow_{\xi_2}^{r,t} (i,t)$ that consists of edges from $E(r,t)$. In particular, it means that $t'' \leq t' \leq t$. It remains to note that the concatenation of these two paths

$$(k,t'') \rightsquigarrow_{\xi_1} (j,t') \rightsquigarrow_{\xi_2} (i,t)$$

leads from (k,t'') to (i,t) and consists of edges from $E(r,t)$. Thus, it is a path from $\Xi(r,t)$ that witnesses $(k,t'') \in V_{(i,t)}(r)$.

By a similar reasoning, $E_\beta(r) \subset E_\theta(r)$. □

Lemma 1.6.10. *Consider a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi)$ with a transition template τ such that $filter_\epsilon \subset filter_\epsilon^B$, an agent-context $\chi = (\gamma, P)$, two runs $r, r' \in R^{w(\chi)}$ and a node $(i,t) \in \mathcal{A} \times \mathbb{N}$ such that r and r' agree on $V_{(i,t)}(r) \setminus \{(i,t)\}$. Then*

$$Past_{(i,t)}(r) = Past_{(i,t)}(r').$$

Proof. Let us denote $\theta = (i,t)$. By Lemma 1.6.8, the graphs $Past_{(i,t)}(r)$ and $Past_{(i,t)}(r')$ are restrictions of the graphs $G(r,t)$ and $G(r',t)$ respectively onto the sets $V_{(i,t)}(r)$ and $V_{(i,t)}(r')$ respectively. Note that $G(r,t)$ and $G(r',t)$ have the same set $V(t)$ of vertices and may differ only in edges.

We show by induction on the length of ξ that

$$\beta \rightsquigarrow_\xi^{r,t} (i,t) \qquad \Longleftrightarrow \qquad \beta \rightsquigarrow_\xi^{r',t} (i,t). \tag{1.100}$$

Induction base. If $\xi = (i,t)$ is a path of length zero and one of the two statements holds, then $\beta = (i,t)$. Accordingly, both statements $(i,t) \rightsquigarrow_{(i,t)}^{r,t} (i,t)$ and $(i,t) \rightsquigarrow_{(i,t)}^{r',t} (i,t)$ in (1.100) hold.

Induction step. Suppose (1.100) holds for any path ξ of length k. Consider a path κ of length $k+1$. Then κ has the form

$$\nu_1 \to \nu_2 \rightsquigarrow_\xi (i,t)$$

for some path ξ of length k. Assume that κ is a path in one of $G(r,t)$ or $G(r',t)$. Then so is ξ and, by the induction hypothesis, ξ is also a path in the other of the two. Thus, all we need to show is that

$$\nu_1 \to^{r,t} \nu_2 \qquad \Longleftrightarrow \qquad \nu_1 \to^{r',t} \nu_2 \tag{1.101}$$

provided $\nu_2 \rightsquigarrow_\xi^{r,t} (i,t)$ and $\nu_2 \rightsquigarrow_\xi^{r',t} (i,t)$ (these two statements are equivalent by the induction hypothesis). If the edge $(\nu_1, \nu_2) \in E_{loc}(t)$ is a simple time edge, which are the same in $G(r,t)$ and $G(r',t)$, then (1.101) is trivial. Thus, assume first that $(\nu_1, \nu_2) \in E_{msg}(r,t)$ is a message edge in the first graph. Since $\nu_1 \in V_{(i,t)}(r)$ by virtue of κ, runs r and r' agree on ν_1 by assumption. Let $\nu_1 = (l', t')$ and $\nu_2 = (l'', t'')$. Note that $t'' > 0$ due to an incoming edge from ν_1 and $t'' \leq t$. Then, $\beta_{\epsilon_{l'}}^{t'}(r) = \beta_{\epsilon_{l'}}^{t'}(r')$ and $\beta_{l'}^{t'}(r) = \beta_{l'}^{t'}(r')$. Therefore,

$$\begin{aligned} &gsend(l', l'', \mu, id) \in \ \beta_{l'}^{t'}(r) \text{ or} \\ &\qquad (\exists A \in \{\text{📅}\} \sqcup \overline{GActions}_{l'})\, fake(l', gsend(l', l'', \mu, id) \mapsto A) \in \beta_{b_{l'}}^{t'}(r) \subset \beta_{\epsilon_{l'}}^{t'}(r) \end{aligned} \tag{1.102}$$

implies

$$\begin{aligned} &gsend(l', l'', \mu, id) \in \ \beta_{l'}^{t'}(r') \text{ or} \\ &\qquad (\exists A \in \{\text{📅}\} \sqcup \overline{GActions}_{l'})\, fake(l', gsend(l', l'', \mu, id) \mapsto A) \in \beta_{b_{l'}}^{t'}(r') \subset \beta_{\epsilon_{l'}}^{t'}(r'). \end{aligned} \tag{1.103}$$

Similarly, $(l'', t''-1) \in V_{(i,t)}(r)$ by virtue of the path

$$(l'', t''-1) \to^{r,t} (l'', t'') \rightsquigarrow_\xi^{r,t} (i,t), \tag{1.104}$$

and $(l'', t''-1) \neq (i,t)$. Hence, r and r' also agree on $(l'', t''-1)$, meaning, in particular, $\beta_{\epsilon_{l''}}^{t''-1}(r) = \beta_{\epsilon_{l''}}^{t''-1}(r')$. Thus,

$$grecv(l'', l', \mu, id) \in \overline{\beta}_{\epsilon_{l''}}^{t''-1}(r) \subset \beta_{\epsilon_{l''}}^{t''-1}(r) \tag{1.105}$$

implies

$$grecv(l'', l', \mu, id) \in \overline{\beta}_{\epsilon_{l''}}^{t''-1}(r') \subset \beta_{\epsilon_{l''}}^{t''-1}(r'). \tag{1.106}$$

We conclude that $(\nu_1, \nu_2) \in E_{msg}(r', t)$, which completes the proof of the left-to-right direction of (1.101).

Let us now assume that $(\nu_1, \nu_2) \in E_{msg}(r', t)$ is a message edge in the second graph, i.e., assume that (1.103) and (1.106). Since ξ is a path in the second graph, it is also a path in the first graph by the induction hypothesis, making (1.104) a witness that $(l'', t''-1) \in V_{(i,t)}(r)$ and that r agrees with r' on $(l'', t''-1)$ because $(l'', t''-1) \neq (i,t)$. Consequently, (1.106) implies (1.105), which, in turn, by (1.67), implies

$$\begin{aligned} &gsend(l', l'', \mu, id) \in \beta_{l'}^{t'''}(r) \quad \text{or} \\ &\qquad (\exists A \in \{\text{📅}\} \sqcup \overline{GActions}_{l'})\, fake(l', gsend(l', l'', \mu, id) \mapsto A) \in \beta_{b_{l'}}^{t'''}(r) \end{aligned} \tag{1.107}$$

for some $t''' \leq t''-1$. From (1.105) and (1.107), it follows that

$$(l', t''') \rightarrow^{r,t} (l'', t'') \rightsquigarrow_{\xi}^{r,t} (i,t)$$

witnesses $(l', t''') \in V_{(i,t)}(r)$. Therefore, r agrees with r' on (l', t''') because $(l', t''') \neq (i,t)$ and

$$\begin{aligned} &gsend(l', l'', \mu, id) \in \beta_{l'}^{t'''}(r') \quad \text{or} \\ &\qquad (\exists A \in \{\text{📅}\} \sqcup \overline{GActions}_{l'})\, fake(l', gsend(l', l'', \mu, id) \mapsto A) \in \beta_{b_{l'}}^{t'''}(r'). \end{aligned} \tag{1.108}$$

Thus, (1.103) in combination with (1.108) yields $t' = t'''$ by Corollary 1.3.17. Thus, $\nu_1 = (l', t') \in V_{(i,t)}(r)$, and (ν_1, ν_2) is an edge of $G(r,t)$, completing the proof of (1.101) from right to left.

It easily follows that

- $V_{(i,t)}(r) = V_{(i,t)}(r')$, i.e., that $\beta \rightsquigarrow^{r,t} (i,t)$ iff $\beta \rightsquigarrow^{r',t} (i,t)$;
- $\nu_1 \rightarrow^{r,t} \nu_2$ iff $\nu_1 \rightarrow^{r',t} \nu_2$ for any $\nu_1, \nu_2 \in V_{(i,t)}(r)$.

Hence, the two causal cones are, indeed identical. □

Definition 1.6.11 (Source nodes)**.** For an agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau, \Psi), P\big)$, a χ-based interpreted system $I = (R^\chi, \pi)$, a run $r \in R^\chi$, a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$ such that $t > 0$ and a local event or action $o \in \overline{Actions} \sqcup \overline{Events}$, we define the **non-Byzantine source nodes of o within the causal cone of θ in the run** r to be all nodes in the causal cone of θ where o occurred correctly:

$$\overline{\Theta_\theta^o}(r) := \{\eta \in V_\theta(r) \,|\, (I, r, t) \models \overline{occurred}_\eta(o)\}. \tag{1.109}$$

Similarly, we define the **source nodes of o within the causal cone of θ in the run** r as all nodes in the causal cone where o occurred, whether correctly or in a Byzantine way:

$$\Theta_\theta^o(r) := \{\eta \in V_\theta(r) \,|\, (I, r, t) \models occurred_\theta(o)\}. \tag{1.110}$$

Definition 1.6.12. For a set of nodes $Sources \subset \mathcal{A} \times \mathbb{N}$

$$\Xi_\theta^{Sources}(r) := \{\xi \in \Xi(r,t) \mid (\exists \eta \in Sources)\, \eta \rightsquigarrow_\xi \theta\} \tag{1.111}$$

$$\overline{\Xi_\theta^{Sources}}(r) := \{\xi \in \Xi(r,t) \mid (\exists \eta \in Sources)\, \eta \rightsquigarrow_\xi \theta \wedge Failed_\xi(r(t)) = \varnothing\} \tag{1.112}$$

We denote information flows from correct sources of o to θ as

$$\Xi_\theta^o(r) := \Xi_\theta^{\overline{\Theta_\theta^o}(r)}(r), \tag{1.113}$$

$$\overline{\Xi_\theta^o}(r) := \overline{\Xi_\theta^{\overline{\Theta_\theta^o}(r)}}(r). \tag{1.114}$$

Definition 1.6.13 (Resiliency)**.** Let $\Xi \subset \Xi(r,t)$ be a set of paths for some run r and timestamp t, let $k \in \mathbb{N}$ be a number, and let $D, R \subset \mathcal{A}$ be sets of agents:

1. D **disconnects** Ξ iff $(\forall \xi \in \Xi)\ \mathcal{A}(\xi) \cap D \neq \varnothing$.
2. Ξ is k**-agent-resilient around** R, or simply k**-resilient around** R, iff no set $B \subset \mathcal{A} \setminus R$ of size $|B| \leq k$ can disconnect Ξ. Formally stated,

$$(\forall B \subset \mathcal{A} \setminus R)\left(|B| \leq k \implies \neg(\forall \xi \in \Xi)\ \mathcal{A}(\xi) \cap B \neq \varnothing\right) \iff (\forall B \subset \mathcal{A} \setminus R)\left(|B| \leq k \implies (\exists \xi \in \Xi)\ \mathcal{A}(\xi) \cap B = \varnothing\right) \quad (1.115)$$

Remark 1.6.14 (Inapplicability of Menger's theorem)**.** Note carefully that Menger's theorem cannot be applied in our setting, as the past graph consists of process-time nodes that are not independent of each other: Both (i,t) and (i,t') are affected if i is removed. Taking out a set of k *agents* may hence disconnect more than k agent-disjoint paths.

1.7 Fully Byzantine Asynchronous Agents

In the previous sections, we defined the general framework using transition templates and introduced the Byzantine transition template. It covers a wide range of settings, from crash failures to fully Byzantine agents and represents asynchronous agents with no additional requirements on communication.

However, no task can be guaranteed if agents are never allowed to act. Thus, it is standard to impose the **Fair Schedule (FS)** admissibility condition, which ensures that each correct agent will eventually be given a possibility to follow its protocol.

Definition 1.7.1 (Fair schedule)**.**

$$FS = \left\{ r \in R \mid \left(\forall (i,t) \in \mathcal{A} \times \mathbb{N}\right)(\exists t' \geq t) \quad \beta_{g_i}^{t'}(r) \neq \varnothing \right\}.$$

Remark 1.7.2. The condition $\beta_{g_i}^{t'}(r) \neq \varnothing$ is equivalent to demanding that eventually $go(i)$, $sleep(i)$, or $hibernate(i)$ be present in $\beta_{\epsilon_i}^{t'}(r)$. In other words, the FS admissibility condition demands that the environment either provide CPU time or *wrongfully* deny CPU time for every processor infinitely many times. This means that correct processes will be treated fairly, i.e., would always be given an opportunity to act, whereas faulty processes can stop their by-the-protocol actions from some point onward. Note, however, that for this to happen, the environment still has to deal with this agent acting infinitely often, via $sleep(i)$ and/or $hibernate(i)$ commands. In other words, an agent malfunctioning only due to wrong actions/events would still be regularly fulfilling its protocol.[16] Avoiding the protocol altogether constitutes a separate type of malfunction.

Thus, allowing all agents to go rogue may also result in the complete crash failure of the whole system, which would preclude any guarantees of fulfilling the goal(s) of the joint protocol. It is, therefore, common to restrict the maximal number f of agents that can become Byzantine.

In preventing the environment from failing too many agents within a round, several options are *a priori* possible. The most general manifestation of this problem is when $f - k$ agents are already faulty and the environment's protocol attempts to fail $k + l + 1$ more agents for some $k, l \geq 0$. In the situation when k is positive, i.e., some but not all agents can still be failed, the choices are

1. to fail as many agents as possible, i.e., to fail some k of the proposed $k + l + 1$ agents but filter out Byzantine events for the other $l + 1$ agents;

[16] This might appear to be a restriction artificially making agents perform actions. However, an agent that can stay in standby indefinitely would simply have the empty set of actions as an option in its protocol.

2. to fail nobody, i.e., to filter all Byzantine events while keeping all correct events intact;

3. to filter all events, Byzantine or correct alike.

We find that the first option is too arbitrary as it requires to randomly choose which $l+1$ agents from $k+l+1$ are to stay correct. The third option, on the contrary, is too invasive. Given the general postulate that, unlike agents, the environment is not acting according to any plan, it is more natural to consider each event attempted by the environment in isolation, much like the inability to receive an unsent message does not preclude the environment from implementing other events.

Thus, we choose the second option and implement it by adding an additional function $filter_\epsilon^{\leq f}$ for the environment.

Definition 1.7.3 (Filtering for at most f Byzantine agents)**.** For a set $X_\epsilon \subset GEvents$ and agent $i \in \mathcal{A}$, we abbreviate

$$X_{\epsilon_i}^B := X_\epsilon \cap \left(BEvents_i \sqcup \{sleep\,(i), hibernate\,(i)\}\right) \tag{1.116}$$

and define

$$filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) := \begin{cases} X_\epsilon & \text{if } \left|\mathcal{A}(Failed\,(h)) \cup \left\{i \mid X_{\epsilon_i}^B \neq \varnothing\right\}\right| \leq f \\ X_\epsilon \setminus \bigcup\limits_{i\in\mathcal{A}} X_{\epsilon_i}^B & \text{otherwise,} \end{cases} \tag{1.117}$$

which removes all Byzantine commands from X_ϵ whenever they would have led to creating more than f faulty agents.

Remark 1.7.4. It might seem that the filters $filter_\epsilon^{\leq f}$ and $filter_\epsilon^B$ are completely independent, i.e., they can be applied in any order. After all, by (1.117) one of them only removes Byzantine events, while by (1.24) the other only removes *grecv* events, which are correct. Unfortunately, this is not entirely accurate. It is possible that a *grecv* command is not filtered by $filter_\epsilon^B$ based on a Byzantine send from the same round. If $filter_\epsilon^{\leq f}$ is applied after that and happens to filter this fake send out, the receipt of the message becomes causally problematic. Thus, the only correct order of applying these two filters is

$$filter_\epsilon^B\left(h, filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n)\,,\, X_1,\, \ldots,\, X_n\right).$$

Definition 1.7.5. Given filters $filter_\epsilon^{Z_1}$ and $filter_\epsilon^{Z_2}$ we write

$$filter_\epsilon^{Z_2\circ Z_1}\,(h, X_\epsilon,\, X_1,\, \ldots,\, X_n) := filter_\epsilon^{Z_2}\left(h, filter_\epsilon^{Z_1}\,(h, X_\epsilon,\, X_1,\, \ldots,\, X_n)\,,\, X_1,\, \ldots,\, X_n\right). \tag{1.118}$$

We also simplify this notation for the combination of the Byzantine and at most f faults filters

$$filter_\epsilon^{B_f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) := filter_\epsilon^{B\circ\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n)\,. \tag{1.119}$$

Remark 1.7.6. Unlike FS, the upper bound on Byzantine agents cannot be formulated as an admissibility condition if agent-contexts are to be non-excluding. Indeed, a run without such an admissibility condition imposed may incur $> f$ Byzantine agents already in a finite prefix, in which case it would be impossible to extend such a prefix to a run satisfying the upper bound.

This is a typical example of separation between properties determined by a finite prefix of a run (safety properties) on the one hand and properties of the run as a whole (liveness properties) on the other hand. The non-exclusion requirement precludes the former from being imposed via admissibility conditions, which are non-constructive and, hence, should only be used as the last resort.

Definition 1.7.7 (f-Byzantine transition template)**.** For a bound $f \geq 0$, the **f-Byzantine transition template** τ^{B_f} is obtained by replacing $filter_\epsilon^B$ in the definition of τ^B with $filter_\epsilon^{B_f}$.

Definition 1.7.8 (f-Byzantine agent-context)**.** For a bound $f \geq 0$, we call an agent-context

$$\chi = \Big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, FS), P\Big) \tag{1.120}$$

f-Byzantine when it is based on the transition template τ^{B_f} and has the admissibility condition FS.

Remark 1.7.9. It is easy to see that for $f \geq |\mathcal{A}|$, we have $filter_\epsilon^{B_f} = filter_\epsilon^B$. Indeed, if all agents can become Byzantine simultaneously, the need to restrict their number never materializes. Hence, we generally assume that $f \leq |\mathcal{A}|$. The case of $f = |\mathcal{A}|$ is useful for uniform statements about both $filter_\epsilon^{B_f}$ and $filter_\epsilon^B$.

Lemma 1.7.10. *If the set*

$$\{t \mid (\exists X_\epsilon \in P_\epsilon(t))\, go(i) \in X_\epsilon\} \tag{1.121}$$

is infinite for each $i \in \mathcal{A}$, then the f-Byzantine agent-context (1.120) *is non-excluding.*

Proof. In order to extend a given finite prefix of a weakly χ-consistent run to a consistent one, it is sufficient to make $\beta_{g_i}^t(r)$ non-empty infinitely many times. Since the filter function $filter_\epsilon^{B_f}$ never removes $go(i)$, it is sufficient that $\alpha_{g_i}^t(r)$ contain it infinitely many times, which can be achieved in the rest of the run by the adversary if $(\exists X_\epsilon \in P_\epsilon(t))\, go(i) \in X_\epsilon$ holds for infinitely many t's. □

Remark 1.7.11. Note that, despite its name, a non-excluding f-Byzantine agent-context χ provides neither a guarantee that Byzantine events *will* happen in a particular run nor, indeed, that they *can* happen in a transitional run at all. Thus, our model can represent both correct runs and infallible systems. Indeed, Byzantine events will not occur in a run if the adversary part of the environment never chooses them. But it chooses them out of the possibilities afforded by the environment's protocol P_ϵ. If P_ϵ contains no Byzantine events, the adversary is powerless to effect them.

Lemma 1.7.12. $filter_\epsilon^{B_f} \subset filter_\epsilon^B$.

Proof. If $E \in GEvents$ is any event other than a correct receive, then

$$E \in filter_\epsilon^{B_f}(h, X_\epsilon, X_1, \ldots, X_n) \quad \Rightarrow \quad E \in X_\epsilon \quad \Rightarrow \quad E \in filter_\epsilon^B(h, X_\epsilon, X_1, \ldots, X_n).$$

by (1.54) and the fact that $filter_\epsilon^B$ only removes correct receives. If

$$E = grecv(i,j,\mu,id) \in filter_\epsilon^{B_f}(h, X_\epsilon, X_1, \ldots, X_n) = filter_\epsilon^B\big(h, filter_\epsilon^{\leq f}(h, X_\epsilon, X_1, \ldots, X_n), X_1, \ldots, X_n\big)$$

then $grecv(i,j,\mu,id) \in filter_\epsilon^{\leq f}(h, X_\epsilon, X_1, \ldots, X_n) \subset X_\epsilon$ and for it to remain, one of the following options must be fulfilled

- $gsend(j,i,\mu,id) \in h_\epsilon$ or $fake(j, gsend(j,i,\mu,id) \mapsto A) \in h_\epsilon$ for some $A \in \{\text{📅}\} \sqcup \overline{GActions_j}$.
- $gsend(j,i,\mu,id) \in X_j$ and $go(j) \in filter_\epsilon^{\leq f}(h, X_\epsilon, X_1, \ldots, X_n) \subset X_\epsilon$.
- $\underline{fake(j, gsend(j,i,\mu,id) \mapsto A)} \in filter_\epsilon^{\leq f}(h, X_\epsilon, X_1, \ldots, X_n) = X_\epsilon$ for some $A \in \{\text{📅}\} \sqcup \overline{GActions_j}$ (the equality in this case follows from the fact that there are Byzantine events left after the filtering).

In each of the options, the same reasoning applies to $filter_\epsilon^B(h, X_\epsilon, X_1, \ldots, X_n)$ equally well because the same h and X_j are used whereas X_ϵ can only become larger. □

Corollary 1.7.13. *All statements from Lemma 1.3.15, Cor. 1.3.16, Cor. 1.3.18, and Lemma 1.6.10 hold also for contexts $\gamma = (P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi)$.*

So far we have not postulated that Byzantine events must be present in P_ϵ. We will now define several types of protocols that ensure the possibility of particular types of errors, up to the case of fully Byzantine agents, i.e., agents that are in principle capable of any malfunction imaginable.

Definition 1.7.14 (Types of agents)**.** Given an agent-context

$$\Big((P_\epsilon, \mathscr{G}(0), \tau, FS), P\Big), \tag{1.122}$$

an agent $i \in \mathcal{A}$ in this agent-context is called

- **fallible** if for any $X \in P_\epsilon(t)$,

$$X \cup \{fail(i)\} \quad \in \quad P_\epsilon(t). \tag{1.123}$$

  In other words, an agent is fallible if it can be branded Byzantine at any moment;

- **infallible** if for any $X \in P_\epsilon(t)$,

$$X \cap \big(BEvents_i \sqcup \{sleep(i), hibernate(i)\}\big) = \varnothing. \tag{1.124}$$

  An infallible agent cannot become Byzantine;

- **degradable** if for any $Y \subset BEvents_i$ and any $X \in P_\epsilon(t)$,

$$X \cup Y \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.125}$$
$$(X \setminus SysEvents_i) \cup Y \sqcup \{sleep(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.126}$$
$$(X \setminus SysEvents_i) \cup Y \sqcup \{hibernate(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent.} \tag{1.127}$$

  In other words, an agent is degradable if it can always make more mistakes;

- **correctable** if for any $X \in P_\epsilon(t)$,

$$X \setminus \big(BEvents_i \sqcup \{sleep(i), hibernate(i)\}\big) \quad \in \quad P_\epsilon(t). \tag{1.128}$$

  In other words, an agent is correctable if it can always refrain from all mistakes;

- **error-prone** if for any $Y \subset BEvents_i$ and any $X \in P_\epsilon(t)$,

$$\big(X \setminus (BEvents_i \sqcup \{sleep(i), hibernate(i)\})\big) \sqcup Y \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.129}$$
$$\big(X \setminus (BEvents_i \sqcup SysEvents_i)\big) \sqcup Y \sqcup \{sleep(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.130}$$
$$\big(X \setminus (BEvents_i \sqcup SysEvents_i)\big) \sqcup Y \sqcup \{hibernate(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent.} \tag{1.131}$$

  In other words, an agent is error-prone if it can commit any combination of Byzantine actions/events in any round;

- **delayable** if for any $X \in P_\epsilon(t)$,

$$X \setminus GEvents_i \quad \in \quad P_\epsilon(t). \tag{1.132}$$

  In other words, an agent is delayable if all its activities can be correctly postponed in any round, forcing its local state to remain unchanged after such a round (note that the absence of $go(i)$ also prevents the agent from acting on its own);

- **gullible** if for any $Y \subset BEvents_i$ and any $X \in P_\epsilon(t)$,

$$(X \setminus GEvents_i) \sqcup Y \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.133}$$
$$(X \setminus GEvents_i) \sqcup Y \sqcup \{sleep(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent;} \tag{1.134}$$
$$(X \setminus GEvents_i) \sqcup Y \sqcup \{hibernate(i)\} \in P_\epsilon(t) \text{ whenever it is } t\text{-coherent.} \tag{1.135}$$

  In other words, an agent is gullible if all its activities during a round can be replaced with an arbitrary set of Byzantine events;

- **isolatable** if for any $X \in P_\epsilon(t)$,

$$X \setminus \{grecv(i, j, \mu, id) \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \quad \in \quad P_\epsilon(t). \tag{1.136}$$

  In other words, an agent is isolatable if all correct message deliveries to it can be postponed at any round;

- **distractible** if for any $X \in P_\epsilon(t)$,

$$X \setminus \overline{GEvents_i} \quad \in \quad P_\epsilon(t). \tag{1.137}$$

  In other words, an agent is distractible if it can miss all external events, including all incoming messages;

- **impotent** if for any $X \in P_\epsilon(t)$,

$$X \setminus \{go(i)\} \quad \in \quad P_\epsilon(t). \tag{1.138}$$

  In other words, an agent is impotent if it is always possible it does not try to act;

- **fully Byzantine** if it is error-prone and gullible. In other words, whatever combination of correct and faulty events can happen to i, the same correct events are compatible with any other collection of faulty events (error-proneness) and, at the same time, any such collection of faulty events could happen without any correct events whatsoever (gullibility).

In all cases where the new set is not explicitly required to be t-coherent, it can be shown to be t-coherent whenever X is.

Remark 1.7.15. Note that error-proneness, degradability, gullibility, and full Byzanteneity means that the agent can become Byzantine in some runs (for instance, adding $fail(i)$ never violates t-coherency); hence, such agents are not infallible. On the other hand, delayability, isolatability, distractibility, and impotence do not necessarily imply any wrongdoing.

Corollary 1.7.16.

- *Any agent that is degradable, error-prone, or gullible is fallible.*
- *Any agent that is error-prone is degradable and correctable.*
- *An agent that is both error-prone and delayable is also gullible.*
- *An agent that is gullible is delayable.*
- *An agent that is fallible, degradable, error-prone, or gullible cannot be infallible.*
- *A fully Byzantine agent is gullible, error-prone, fallible, degradable, correctable, and delayable.*

Definition 1.7.17. For an upper bound $f \geq 0$, an agent-context (1.120) is called **fully f-Byzantine**, or simply **fully Byzantine**, if all agents are fully Byzantine.

Lemma 1.7.18. *All fully f-Byzantine agent-contexts are non-excluding.*

Proof. It is sufficient to issue $sleep(i)$ in every round. □

Chapter 2

The Byzantine Asynchronous Extension

2.1 Run Modifications

Definition 2.1.1. An **intervention** for an agent $i \in \mathcal{A}$, or **i-intervention** is a function

$$\rho\colon R \longrightarrow \overline{GActions}_i \times GEvents_i.$$

The set of all i-interventions is denoted by

$$Intervs\,(i) := \left\{\rho \mid \rho\colon R \longrightarrow \overline{GActions}_i \times GEvents_i\right\}.$$

One intervention $\rho(r)$ is intended to modify the behavior of one agent in one round of a given run r in a desired way. Whether this modification relies on the agent's original behavior in r or is a complete departure from it, can be encoded in the function ρ. The output

$$(X_i, X_{\epsilon_i}) = \rho(r)$$

is intended to represent a pair of sets $\beta_i^t(r') = X_i$ of actions by i and $\beta_{\epsilon_i}^t(r') = X_{\epsilon_i}$ of events imposed on i in the same round of a modified run r'. For ease of notation we also define

$$\mathfrak{a}\rho(r) := \pi_1\rho(r);$$
$$\mathfrak{e}\rho(r) := \pi_2\rho(r).$$

In other words, if $\rho(r) = (X_i, X_{\epsilon_i})$, then $\mathfrak{a}\rho(r) = X_i$ and $\mathfrak{e}\rho(r) = X_{\epsilon_i}$.

Definition 2.1.2. A **joint intervention** is a collection of i-interventions for all agents $i \in \mathcal{A}$. We denote the set of all joint interventions

$$Intervs := \prod_{i\in\mathcal{A}} Intervs\,(i). \tag{2.1}$$

Now, let us define an adjustment of a run as a timewise list of joint interventions. Each joint intervention is to be performed at the corresponding timestamp. It is defined as follows

Definition 2.1.3. An **adjustment**

$$[B_t; \ldots; B_0]$$

is a sequence of joint interventions $B_0 \ldots, B_t \in Intervs$ to be performed at successive timestamps from 0 to some $t \in \mathbb{N}$, which is called the **extent** of the adjustment. We denote the set of adjustments

$$Adjusts := \bigcup_{t\in\mathbb{N}} \left\{[B_t; \ldots; B_0] \mid B_0 \ldots, B_t \in Intervs\right\}. \tag{2.2}$$

Definition 2.1.4. Let $adj \in Adjusts$ be an adjustment

$$adj = [B_t; \ldots; B_0] \tag{2.3}$$

of extent $t \in \mathbb{N}$ where

$$B_m = (\rho_1^m, \ldots, \rho_n^m) \tag{2.4}$$

for each $0 \leq m \leq t$ (recall that $\mathcal{A} = \{1, \ldots, n\}$), where each ρ_i^m is an i-intervention. Let a run r be a $\tau_{P_\epsilon,P}$-transitional run. We say that a run r' is **obtained from** r **by adjustment** adj, or simply is adj**-adjusted variant of** r iff

$$r'(0) = r(0), \tag{2.5}$$

$$(\forall i \in \mathcal{A})(\forall t' \leq t) \quad \left(\beta_i^{t'}(r'), \beta_{\epsilon_i}^{t'}(r')\right) = \rho_i^{t'}(r), \tag{2.6}$$

$$(\forall i \in \mathcal{A})(\forall t' \leq t) \quad r_i'(t'+1) = update_i\left(r_i'(t'), \beta_i^{t'}(r'), \bigcup_{i \in \mathcal{A}} \beta_{\epsilon_i}^{t'}(r')\right), \tag{2.7}$$

$$(\forall t' \leq t) \quad r_\epsilon'(t'+1) = update_\epsilon\left(r_\epsilon'(t'), \quad \left(\bigcup_{i \in \mathcal{A}} \beta_{\epsilon_i}^{t'}(r'), \beta_1^{t'}(r'), \ldots, \beta_n^{t'}(r')\right)\right), \tag{2.8}$$

$$(\forall t' > t) \quad r'(t'+1) \in \tau_{P_\epsilon,P}(r'(t')). \tag{2.9}$$

We denote by $R(\tau_{P_\epsilon,P}, r, adj)$ the set of all adj-adjusted variants of the run r, computed under the transition function $\tau_{P_\epsilon,P}$.

Remark 2.1.5. The first property ensures that both runs start from the same initial state. The second one means that the β-sets of the new run for each agent i, i.e., the actions and events affecting i in rounds 0.5 through t.5, are fully determined by the adjustment. In the absence of global events, this means that everything happening during these rounds is controlled by adj. The third and fourth ones faithfully implement the updating phase of the round (see Figure 1.1) for local and environment states respectively for the extent of the adjustment. Finally, the last property ensures that beyond the adjusted segment, the new run extends in a $\tau_{P_\epsilon,P}$-transitional manner.

Remark 2.1.6. Though r is assumed to be $\tau_{P_\epsilon,P}$-transitional, *a priori* its adjusted variants need not be. Indeed, as noted above, the local and global histories are always updated in a consistent manner, i.e., in accordance with (1.57) and (1.59), but the artificial β-sets imposed by the adjustment adj may not follow the rules of the protocol, adversary, labelling, and filtering phases (see Figure 1.1).

Lemma 2.1.7. *For any run r, any transition function $\tau_{P_\epsilon,P}$, and any adjustment adj*

$$R(\tau_{P_\epsilon,P}, r, adj) \neq \varnothing.$$

Proof. The initial t-prefix of the desired adjusted run, more precisely the behavior up till and including the round t.5, is fully determined by adj and properties (2.5)–(2.8) and exists due to the totality of all the functions involved. The intended behavior starting from the round $(t+1)$.5 is governed by (2.9). Since both P_ϵ and P satisfy the no-apocalypse clause (see Remark 1.2.11), there is always at least one option for continuing the run in every round. Once again, the existence of adjusted runs does not generally imply that they are strongly consistent or even transitional. □

The primary method we use to show that an agent does not/cannot know some fact φ is taking a(n) existing/arbitrary run and adjusting it in a way that is imperceptible for this agent but makes φ false. Note that it is generally not sufficient to intervene with the behavior of only this agent because its local state might be affected by correct messages received from other agents. Thus, the behavior of other agents generally needs to be modified too. There are several types of interventions useful to achieve the needed adjustments.

The interventions needed for the agent who should not distinguish the original and adjusted runs replace each correct action or event it experienced in the original run with a Byzantine event that looks the same to the agent. Given that the agent's perception can deviate from the real actions/events, there is a range of choices regarding what really happens in the adjusted run. We present two straightforward options representing passive and active interventions $PFake_i$ and $AFake_i$ for agent i respectively.

Definition 2.1.8. For an agent $i \in \mathcal{A}$ and a run $r \in R$, we define i-interventions

$$PFake_i^t, AFake_i^t \quad : \quad R \to \overline{GActions_i} \times GEvents_i$$

as follows:

$$\begin{aligned} PFake_i^t(r) := \Big(&\varnothing, \\ &\beta_{b_i}^t(r) \quad \cup \quad \Big\{fake(i,E) \mid E \in \overline{\beta}_{\epsilon_i}^t(r)\Big\} \quad \cup \quad \big\{fake(i, \text{📅} \mapsto A) \mid A \in \beta_i^t(r)\big\} \quad \sqcup \\ &\{sleep(i) \mid aware(i, \beta_{\epsilon_i}^t(r))\} \quad \sqcup \quad \{hibernate(i) \mid unaware(i, \beta_{\epsilon_i}^t(r))\}\Big) \end{aligned} \tag{2.10}$$

$$\begin{aligned} AFake_i^t(r) := \Big(&\varnothing, \\ &\beta_{b_i}^t(r) \quad \cup \quad \Big\{fake(i,E) \mid E \in \overline{\beta}_{\epsilon_i}^t(r)\Big\} \quad \cup \quad \big\{fake(i, A \mapsto A) \mid A \in \beta_i^t(r)\big\} \quad \sqcup \\ &\{sleep(i) \mid aware(i, \beta_{\epsilon_i}^t(r))\} \quad \sqcup \quad \{hibernate(i) \mid unaware(i, \beta_{\epsilon_i}^t(r))\}\Big) \end{aligned} \tag{2.11}$$

Remark 2.1.9. The only difference between these two i-interventions lies in what the agent actually does while erroneously thinking that it did an action A. In case of the passive version $PFake_i$, agent i does not do anything, whereas in the active version $AFake_i$, agent i does perform action A, albeit in a Byzantine fashion. The two i-interventions coincide on Byzantine events.

Lemma 2.1.10. *Let $\rho \in \{PFake_i^t, AFake_i^t \mid t \in \mathbb{N}\}$ be an intervention and r and r' be arbitrary runs. Then*

1. *$\mathfrak{a}\rho(r) = \varnothing$, i.e., these interventions always produce the empty set of actions.*
2. *$go(i) \notin \mathfrak{e}\rho(r)$, i.e., these interventions never let agent i act.*
3. *$|\mathfrak{e}\rho(r) \cap \{sleep(i), hibernate(i)\}| = 1$, i.e., these interventions always intend to make agent i Byzantine by means of exactly one of commands $sleep(i)$ or $hibernate(i)$.*
4. *$\sigma\big(\mathfrak{a}\rho(r) \cup \mathfrak{e}\rho(r)\big) = \sigma\big(\mathfrak{e}\rho(r)\big) = \sigma\left(\beta_i^t(r) \cup \beta_{\epsilon_i}^t(r)\right)$, where $\rho \in \{PFake_i^t, AFake_i^t\}$, i.e., events and actions intended to be appended to the local history of agent i as a result of round t.5 after the intervention $PFake_i^t$ or $AFake_i^t$ are the same as before the intervention in the same round of the original run r.*[1]
5. *$aware(i, \mathfrak{e}\rho(r)) = aware(i, \beta_{\epsilon_i}^t(r))$, where $\rho \in \{PFake_i^t, AFake_i^t\}$, i.e., the awareness of agent i of the passing of round t.5 is not changed by $PFake_i^t$ or $AFake_i^t$.*[2]

Proof. The only properties that do not directly follow from the definition are the last three.

Property 3 follows from the fact that $unaware(i, Z)$ is defined to be the negation of $aware(i, Z)$ (see Def. 1.2.17). Hence, exactly one of them always holds.

[1] In some cases, the local history remains unaffected by these sets: namely, if there is no events/actions to add and the agent is unaware of the passing round, but the statement is true in this case too.

[2] In some cases, the local history does not depend on such awareness: namely, if there are events/actions to add, but the statement is true in this case too.

Property 4 follows from

$$\sigma(\{\mathit{fake}\,(i,E)\}) = \sigma(\{E\}) \qquad \text{and} \qquad \sigma(\{\mathit{fake}\,(i,\text{📅} \mapsto A)\}) = \sigma(\{\mathit{fake}\,(i,A \mapsto A)\}) = \sigma(\{A\})$$

which are a direct consequence of (1.32). Note that in the latter case $A \in \beta_i^t\,(r) \subset \overline{\mathit{GActions}_i}$.

For Property 5, note that $|\mathfrak{e}\rho(r) \cap \{\mathit{go}(i), \mathit{sleep}\,(i), \mathit{hibernate}\,(i)\}| = 1$ by Properties 2 and 3. Hence, there are exactly two possibilities: either $\mathit{unaware}(i, \mathfrak{e}\rho(r))$ due to the presence of $\mathit{hibernate}\,(i)$ or $\mathit{aware}i\mathfrak{e}\rho(r)$ due to the presence of $\mathit{sleep}\,(i)$, equivalently, due to the absence of $\mathit{hibernate}\,(i)$. More precisely,

$$\mathit{aware}(i, \mathfrak{e}\rho(r)) = t \qquad \iff \qquad \mathit{hibernate}\,(i) \notin \mathfrak{e}\rho(r) \qquad \iff$$
$$\mathit{sleep}\,(i) \in \mathfrak{e}\rho(r) \qquad \iff \qquad \mathit{aware}(i, \beta_{\epsilon_i}^t\,(r)) = t.$$

□

Another common construction is freezing an agent, i.e., allowing no actions or events update its local history. Such a behavior is captured by an i-intervention *CFreeze* if the agent is to remain correct or by $\mathit{BFreeze}_i$ if the agent is to become Byzantine. It is defined as follows.

Definition 2.1.11. For a run $r \in R$, we define

$$\mathit{CFreeze}\,(r) := (\varnothing, \varnothing). \tag{2.12}$$

It can serve as an i-intervention for any agent i.

Definition 2.1.12. For an agent $i \in \mathcal{A}$ and a run $r \in R$, we define

$$\mathit{BFreeze}_i\,(r) := (\varnothing, \{\mathit{fail}\,(i)\}). \tag{2.13}$$

Note that interventions *CFreeze* and $\mathit{BFreeze}_i$ are constant, in other words, the modifications they initiate are run-independent.

Sometimes we want an intervention that preserves the exact behavior of an agent $i \in \mathcal{A}$ during round $t.5$.

Definition 2.1.13. For an agent $i \in \mathcal{A}$, a run $r \in R$, and a timestamp $t \in \mathbb{N}$,

$$\mathit{Copy}_i^t\,(r) := (\beta_i^t\,(r), \beta_{\epsilon_i}^t\,(r)). \tag{2.14}$$

Sometimes we want an agent $i \in \mathcal{A}$ during round $t.5$ to concentrate on important messages originating from a specific set X of nodes ignoring the chatter from outside of this set but otherwise to carry on as without the intervention. We will often use a causal cone as X.

Definition 2.1.14. For an agent $i \in \mathcal{A}$, a run $r \in R$, a timestamp $t \in \mathbb{N}$, and a set $X \subset \mathcal{A} \times \mathbb{N}$ of nodes,

$$X\text{-}\mathit{Focus}_i^t\,(r) := \Big(\beta_i^t\,(r), \qquad \beta_{\epsilon_i}^t\,(r) \setminus \{\mathit{grecv}(i,j,\mu,\mathit{id}(j,i,\mu,k,m)) \mid (j,m) \notin X, k \in \mathbb{N}\}\Big). \tag{2.15}$$

Finally, sometimes we do not care about agent i itself but only need it to produce the same communication as in a given round.

Definition 2.1.15. For an agent $i \in \mathcal{A}$, a run $r \in R$, and a timestamp $t \in \mathbb{N}$,

$$\mathit{FakeEcho}_i^t\,(r) := \Big(\varnothing,$$
$$\{\mathit{fail}\,(i)\} \sqcup \{\mathit{fake}\,(i, \mathit{gsend}(i,j,\mu,\mathit{id}) \mapsto \text{📅}) \mid$$
$$\mathit{gsend}(i,j,\mu,\mathit{id}) \in \beta_i^t\,(r) \text{ or } (\exists A \in \overline{\mathit{GActions}_i} \sqcup \{\text{📅}\})\, \mathit{fake}\,(i, \mathit{gsend}(i,j,\mu,\mathit{id}) \mapsto A) \in \beta_{b_i}^t\,(r)\}\Big). \tag{2.16}$$

Using some of the assumptions on agents defined at the end of Chapter 1, we can prove the following lemma.

Lemma 2.1.16 (Brain-in-the-Vat Lemma)**.** *Let $\mathcal{A} = [\![1; n]\!]$ be the set of agents with protocols $P = (P_1, \ldots, P_n)$, let P_ϵ be the protocol of the environment, let r be a $\tau^B_{P_\epsilon,P}$-transitional run, let i be an agent, let $t > 0$ be a timestamp, and let $adj = [B_{t-1}; \ldots; B_0]$ be an adjustment of extent $t-1$ satisfying* (2.4) *for all $0 \leq m \leq t-1$ with*

$$\rho_i^m = PFake_i^m \qquad \text{and} \qquad \text{for all } j \neq i \quad \rho_j^m \in \{CFreeze, BFreeze_j\}.$$

If the protocol P_ϵ makes

- *agent i gullible,*
- *every agent $j \neq i$ delayable and fallible if $\rho_j^m = BFreeze_j$ for some m,*
- *all remaining agents delayable,*

then each run $r' \in R\left(\tau^B_{P_\epsilon,P}, r, adj\right)$ satisfies the following properties:

1. *r' is $\tau^B_{P_\epsilon,P}$-transitional;*
2. *$(\forall m \leq t)\, r'_i(m) = r_i(m)$;*
3. *$(\forall m \leq t)\,(\forall j \neq i)\; r'_j(m) = r'_j(0)$.*
4. *$(i,1) \in Bad(r',1)$ and, consequently, $(i,m) \in Failed(r',m')$ for all $m' \geq m > 0$;*
5. *$\mathcal{A}(Failed(r'(t))) = \{i\} \cup \{j \neq i \mid (\exists m \leq t-1)\, \rho_j^m = BFreeze_j\}$.*

Proof. We prove all these statements alongside the following properties:

6. $(\forall m < t)\,(\forall j \neq i)\; \beta^m_{\epsilon_j}(r') \subset \{fail(j)\}$.
   More precisely, $\beta^m_{\epsilon_j}(r') = \varnothing$ iff $\rho_j^m = CFreeze$ and $\beta^m_{\epsilon_j}(r') = \{fail(j)\}$ iff $\rho_j^m = BFreeze_j$;
7. $(\forall m < t)\, \beta^m_{\epsilon_i}(r') \setminus \beta^m_{f_i}(r') = \varnothing$;
8. $(\forall m < t)(\forall j \in \mathcal{A})\, \beta^m_j(r') = \varnothing$.

Consider an arbitrary $r' \in R\left(\tau^B_{P_\epsilon,P}, r, adj\right)$.

Property 4 follows from Lemma 2.1.10(3).

Property 5 follows from Property 4 and the fact that the only event assigned other agents $j \neq i$ is $fail(j)$ and it is only assigned by $BFreeze_j$, making all agents with $BFreeze_j$ interventions Byzantine while leaving all agents without $BFreeze_j$ interventions correct.

Property 6 follows from (2.12) and (2.13).

Property 7 follows from (2.10).

Property 8 follows from Lemma 2.1.10(1) for i and from (2.12) and (2.13) for $j \neq i$.

The remaining three properties except for the first depend solely on the first t rounds of r' and the first property starting from round $t.5$ directly follows from (2.9). Thus, it remains to show Properties 1–3 for $m \leq t$ by induction on m.

Base: $m = 0$**.** Properties 1 and 3 are trivial, whereas Property 2 follows from (2.5).

Step from m **to** $m+1$**.** We prove Property 1 based on the gullibility of i and delayability (and fallibility) of all other $j \neq i$. In order to show that $r'(m+1) \in \tau^B_{P_\epsilon,P}(r'(m))$, we need to demonstrate that the β-sets prescribed by adj can be obtained in a regular round. Since the adversary's choice of actions $\alpha^m_j(r)$ is immaterial due to the absence of $go(j)$ by Lemma 2.1.10(2) for i and by (2.12)/(2.13) for other $j \neq i$, we concentrate on showing which α-sets of events the adversary needs to choose. Consider $\alpha^m_\epsilon(r) \in P_\epsilon(m)$ from the original run r. It must be m-coherent because r is transitional. By the delayability of all $j \neq i$,

$$\alpha^m_\epsilon(r) \setminus \bigsqcup_{j \neq i} GEvents_j \quad = \quad \alpha^m_{\epsilon_i}(r) \quad \in \quad P_\epsilon(m).$$

Note that for any $Z \subset BEvents_i \sqcup \{sleep\,(i), hibernate\,(i)\}$,

$$\left(\alpha^m_{\epsilon_i}(r) \setminus GEvents_i\right) \sqcup Z \quad = \quad \varnothing \sqcup Z \quad = \quad Z$$

because $\alpha^m_{\epsilon_i}(r) \subset GEvents_i$. Thus, by the gullibility of i,

$$\begin{aligned} Y_0 \quad := \quad & \left(\beta^m_{\epsilon_i}(r) \cap BEvents_i\right) \quad \cup \\ & \left\{fake(i,E) \mid E \in \beta^m_{\epsilon_i}(r) \cap \overline{GEvents_i}\right\} \quad \cup \quad \{fake\,(i, \text{📅} \mapsto A) \mid A \in \beta^m_i(r)\} \quad \sqcup \\ & \{sleep\,(i) \mid aware(i, \beta^m_{\epsilon_i}(r))\} \quad \sqcup \quad \{hibernate\,(i) \mid unaware(i, \beta^m_{\epsilon_i}(r))\} \qquad \in \qquad P_\epsilon(m) \end{aligned}$$

if it is m-coherent. Note that exactly one of $sleep\,(i)$ and $hibernate\,(i)$ is added to Y_0 meaning that condition 2 of m-coherency is fulfilled (see Def. 1.2.1). Conditions 3–5 of m-coherency are trivially fulfilled because Y_0 contains no correct events. Finally, condition 1 is fulfilled because all fake actions in Y_0 either have 📅 actually performed or originate from $\beta^m_{\epsilon_i}(r) \subset \alpha^m_{\epsilon_i}(r) \subset \alpha^m_\epsilon(r)$, the latter being an m-coherent set. Thus, $Y_0 \in P_\epsilon(m)$. Finally, by the fallibility of all agents $j \neq i$ with $\rho^m_j = BFreeze_j$,

$$Y \qquad := \qquad Y_0 \cup \{fail\,(j) \mid \rho^m_j = BFreeze_j\} \qquad \in \qquad P_\epsilon(m)\,.$$

This Y is m-coherent and unaffected by filtering there are no correct events in Y to be filtered out. We conclude that these choices by the adversary result, after the filtering phase, in

$$\begin{aligned} &\beta^m_\epsilon(r) = Y, \\ &\beta^m_j(r) = \varnothing \qquad\qquad\qquad\qquad \text{for all } j \in \mathcal{A}. \end{aligned}$$

The latter is due to $go(j) \notin Y$ for any $j \in \mathcal{A}$. It remains to note that

$$\begin{aligned} &\beta^m_{\epsilon_i}(r) = \beta^m_\epsilon(r) \cap GEvents_i = Y_0, \\ &\beta^m_{\epsilon_j}(r) = \beta^m_\epsilon(r) \cap GEvents_j = \begin{cases} \varnothing & \text{if } \rho^m_j = CFreeze, \\ \{fail\,(j)\} & \text{if } \rho^m_j = BFreeze_j \end{cases} \qquad \text{for other } j \neq i. \end{aligned}$$

This completes the induction step for Property 1.

For Property 2, the induction step follows from Lemma 2.1.10(4)–(5). More precisely, given that $\sigma(Y) = \sigma(Y_0)$, we have the following cases:

- if $\sigma\left(\beta^m_{\epsilon_i}(r)\right) \neq \varnothing$, then

  $$r_i(m+1) = \sigma\left(\beta^m_i(r) \sqcup \beta^m_{\epsilon_i}(r)\right) : r_i(m) = \sigma\left(\beta^m_i(r) \sqcup \beta^m_{\epsilon_i}(r)\right) : r'_i(m)$$

  by the induction hypothesis. It remains to use Lemma 2.1.10(4) to see that it is the same as

  $$\sigma(Y) : r'_i(m) = \sigma(\varnothing \sqcup Y_0) : r'_i(m) = \sigma\left(\beta^m_i(r') \sqcup \beta^m_{\epsilon_i}(r')\right) : r'_i(m) = r'_i(m+1) \tag{2.17}$$

  because $\sigma\left(\beta^m_{\epsilon_i}(r')\right) = \sigma(Y_0) = \sigma\left(\beta^m_i(r) \sqcup \beta^m_{\epsilon_i}(r)\right) \supset \sigma\left(\beta^m_{\epsilon_i}(r)\right) \neq \varnothing$.

- if $\sigma\left(\beta^m_{\epsilon_i}(r)\right) = \varnothing$ but $aware(i, \beta^m_\epsilon(r)) = t$, then

  $$r_i(m+1) = \sigma\left(\beta^m_i(r) \sqcup \beta^m_{\epsilon_i}(r)\right) : r_i(m) = \sigma(\beta^m_i(r)) : r_i(m) = \sigma(\beta^m_i(r)) : r'_i(m) \tag{2.18}$$

  by the induction hypothesis. By Lemma 2.1.10(5), also

  $$aware(i, \beta^m_\epsilon(r')) = aware(i, \beta^m_{\epsilon_i}(r')) = aware(i, Y) = aware(i, \beta^m_{\epsilon_i}(r)) = aware(i, \beta^m_\epsilon(r)) = t.$$

  Thus, the last expression of (2.18) is equal to the first expression of (2.17) and this case can be concluded by continuing the series of equalities the same way as it was done in (2.17) in the preceding case.

- if $\sigma\left(\beta^m_{\epsilon_i}(r)\right) = \varnothing$ and $\mathit{unaware}(i, \beta^m_\epsilon(r)) = t$, then $\mathit{unaware}(i, \beta^m_\epsilon(r')) = t$ by the same reasoning we just applied to $\mathit{aware}(i, \beta^m_\epsilon(r'))$. In addition, by (1.23), $\mathit{passive}(i, \beta^m_\epsilon(r))$, meaning that $\beta^m_i(r) = \varnothing$ by (1.25). Finally,

$$\sigma\left(\beta^m_{\epsilon_i}(r')\right) = \sigma(Y) = \sigma\left(\beta^m_i(r) \sqcup \beta^m_{\epsilon_i}(r)\right) = \sigma\left(\beta^m_{\epsilon_i}(r)\right) = \varnothing.$$

  Thus, in this case,

$$r'_i(m+1) = r'_i(m) = r_i(m) = r_i(m+1).$$

This completes the proof of the induction step for Property 2.

For Property 3, the induction step is even simpler. Since $\beta^m_j(r') = \varnothing$ and $\beta^m_{\epsilon_j}(r') \subset \{\mathit{fail}(j)\}$ for any $j \neq i$, it follows that $\sigma(\beta^m_{\epsilon_j}(r')) = \varnothing$ and $\mathit{unaware}(j, \beta^m_\epsilon(r')) = \mathit{unaware}(j, \beta^m_{\epsilon_j}(r')) = t$, it follows that

$$r'_j(m+1) = r'_j(m) = r'_j(0)$$

by the induction hypothesis. □

Remark 2.1.17. The previous lemma states that for a designated agent $i \in \mathcal{A}$ at some local state $r_i(t)$ there is always an i-indistinguishable local state $r'_i(t)$ in an alternative transitional run r' such that all other agents are yet to leave their initial local states, with i definitely Byzantine while other agents can be made Byzantine or correct at will. We call this the Brain-in-the-Vat Lemma because agent i attains this indistinguishable local state by imagining that all actions and events from the original run happened to it without any participation of other agents.

Definition 2.1.18. For an adjustment $adj = [B_t; \ldots; B_0]$ of extent t satisfying (2.4), we denote

$$\mathit{Failed}(adj) := \left\{ j \mid (\exists m \leq t)\, \mathfrak{e}\rho^m_j \cap (\mathit{BEvents}_j \sqcup \{\mathit{sleep}(j), \mathit{hibernate}(j)\}) \neq \varnothing \right\}$$

the set of agents who are assigned Byzantine events, including $\mathit{sleep}(i)$ or $\mathit{hibernate}(i)$ instructions by this adjustment.

Corollary 2.1.19. *For the adjustment adj used in Lemma 2.1.16,*

$$\mathit{Failed}(adj) = \{i\} \cup \{j \neq i \mid (\exists m \leq t-1)\, \rho^m_j = \mathit{BFreeze}_j\}.$$

Hence, the number $|\mathit{Failed}(adj)| \geq 1$ of agents necessarily failed by this adjustment is always positive.

Corollary 2.1.20. *Lemma 2.1.16 also holds if r is a $\tau^{B_f}_{P_\epsilon,P}$-transitional run for any*

$$f \geq |\mathit{Failed}(adj)|$$

(in particular, if all $\rho^m_j = \mathit{CFreeze}$ for $j \neq i$, it is sufficient to have $f \geq 1$). Moreover, in this case Property 1 can be replaced by

1′. r' is $\tau^{B_f}_{P_\epsilon,P}$-transitional.

Proof. The construction is exactly the same. The additional filtering of too many Byzantine agents introduced in a round will not be used (until timestamp t) because the constructed part of the new run only fails allowable number of agents, including agent i. □

Corollary 2.1.21. *For an agent $i \in \mathcal{A}$, for a set $BD \subset \mathcal{A} \setminus \{i\}$ of agents, for a non-excluding agent-context $\chi = \left((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\right)$ such that $f \geq 1 + |BD|$ and P_ϵ makes i gullible, all other agents $j \neq i$ delayable, and additionally all agents from BD fallible, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, and for a timestamp $t > 0$, there is a run $r' \in R^\chi$ that satisfies Properties 2–8 from Lemma 2.1.16 and such that $\mathcal{A}(\mathit{Failed}(r'(t))) = \{i\} \cup BD$.*

Proof. Consider an tadjustment $adj = [B_{t-1}; \dots; B_0]$ with (2.4) such that for all $m < t$,

$$\rho_i^m = PFake_i,$$
$$\rho_j^m = \begin{cases} BFreeze_j & \text{if } j \in BD, \\ CFreeze & \text{if } j \in \mathcal{A} \setminus (\{i\} \sqcup BD). \end{cases}$$

Clearly, $|Failed\,(adj)| = 1 + |BD|$. By Cor. 2.1.20, there exists a τ^{B_f}-transitional run r'' satisfying all the required conditions. Clearly, $r'' \in R^{w\chi}$ because, by (2.5) we have $r''(0) = r(0) \in \mathscr{G}(0)$. It remains to note that the the initial prefix of r'' up to timestamp t can be extended to a run $r' \in R^\chi$ because χ is non-excluding. Since all required properties depend only on this initial prefix, they are also satisfied for r'. □

Corollary 2.1.22. *For an agent $i \in \mathcal{A}$, a set $BD \subset \mathcal{A} \setminus \{i\}$ of agents, a fully f-Byzantine agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, FS), P\big)$ with $f \geq 1 + |BD|$, a χ-based interpreted system $I = (R^\chi, \pi)$, a run $r \in R^\chi$, and a timestamp $t > 0$, there is a run $r' \in R^\chi$ that satisfies Properties 2–8 from Lemma 2.1.16.*

Proof. f-fully Byzantine agent-contexts are non-excluding by Lemma 1.7.18. Since all agents in such agent-contexts are fully Byzantine, they are gullible by definition. In addition to i being gullible, it follows from Cor. 1.7.16 that all other $j \neq i$ are delayable. Finally, since fully Byzantine agents are error-prone by definition, it follows from the same Cor. 1.7.16 that all $j \in BD$ are fallible. □

Corollary 2.1.23. *For the χ-based interpreted system $I = (R^\chi, \pi)$ and run r' from Cor. 2.1.21 or Cor. 2.1.22 with $BD = \varnothing$ and, accordingly, with $f \geq 1$, for any timestamp $t \in \mathbb{N}$,*

$$(\forall o \in \overline{Actions} \sqcup \overline{Events})\,(\forall t' \leq t)\,(I, r', t') \not\models \overline{occurred}(o). \tag{2.19}$$

Proof. Recall that by (1.84) and (1.83)

$$\overline{occurred}(o) = \bigvee_{j \in \mathcal{A}} \overline{occurred}_j(o).$$

Thus, we need to show that

$$(I, r', t') \not\models \overline{occurred}_j(o)$$

for each $j \in \mathcal{A}$, each $t' \leq t$, and each $o \in \overline{Actions} \sqcup \overline{Events}$. By (1.81), we need to show that

$$(\forall t'' < t')\, o \notin label^{-1}\left(\beta_j^{t''}(r') \sqcup \overline{\beta}_{\epsilon_j}^{t''}(r')\right).$$

Since all such $t'' < t$, it is sufficient to show

$$(\forall t'' < t)\, o \notin label^{-1}\left(\beta_j^{t''}(r') \sqcup \overline{\beta}_{\epsilon_j}^{t''}(r')\right).$$

For $t = 0$, this is vacuously true. For $t > 0$, we can use Cor. 2.1.21 or Cor. 2.1.22 respectively. Then this statement follows from the fact that for all agents $j \in \mathcal{A}$,

$$\beta_j^{t''}(r') = \overline{\beta}_{\epsilon_j}^{t''}(r') = \varnothing.$$

Regarding $\beta_j^{t''}(r')$, this follows from Lemma 2.1.16(8). Regarding $\overline{\beta}_{\epsilon_j}^{t''}(r')$ for $j \neq i$, it follows from Lemma 2.1.16(6). Finally, regarding $\overline{\beta}_{\epsilon_i}^{t''}(r')$ for agent i it follows from Lemma 2.1.16(7) and $\overline{\beta}_{\epsilon_i}^{t''}(r') \subset \beta_{\epsilon_i}^{t''}(r') \setminus \beta_{f_i}^{t''}(r')$. □

2.2 Introspection

By introspection we understand the ability of agents to reason about their own state and their own knowledge. The agent is primarily interested in its own correctness and the reliability of data it receives.

2.2.1 Local Introspection

We will now use the brain-in-the-vat construction to show that fully Byzantine agents can never be sure that a particular action/event definitively took place or, barring the initial state, that they are correct.

Lemma 2.2.1. *For an agent $i \in \mathcal{A}$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that $f \geq 1$ and P_ϵ makes i gullible and all other agents $j \neq i$ delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, for a timestamp $t \in \mathbb{N}$, and for an action or event $o \in \overline{Actions} \sqcup \overline{Events}$,*

$$(I, r, t) \not\models K_i \overline{occurred}(o).$$

In particular, this statement holds for fully f-Byzantine agent-contexts.

Proof. For $t = 0$, the statement is obvious. For $t > 0$, the statement follows directly from (2.19) of Cor. 2.1.23 and Lemma 2.1.16(2). □

Lemma 2.2.2. *For an agent $i \in \mathcal{A}$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that $f \geq 1$ and P_ϵ makes i gullible and all other agents $j \neq i$ delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, and for a timestamp $t > 0$,*

$$(I, r, t) \not\models K_i correct_i.$$

In particular, this statement holds for fully f-Byzantine agent-contexts.

Proof. Consider the run r' constructed in Cor. 2.1.21 (respectively, Cor. 2.1.22) for $BD = \varnothing$. Recall that by (1.78)

$$(I, r', t) \models correct_i \qquad \Longleftrightarrow \qquad (i, t) \notin Failed\,(r', t).$$

Thus, the statement follows directly from Properties 2 and 4 of Lemma 2.1.16. □

Remark 2.2.3. It is clear that within any agent-context based on a transition template τ^{B_0}, i.e., for $f = 0$, all agents always know that they are correct and they can learn about real actions/events from observation because no Byzantine events can ever happen in such runs. Conversely, such agents can never learn that they are faulty because it will never be true.

Unlike the knowledge of own correctness, it is in principle possible for a fully Byzantine agent to learn of its own defectiveness (provided, as just noted, that mistakes are, in fact, allowed).

Lemma 2.2.4. *For some agent $i \in \mathcal{A}$, an agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that $f \geq 1$, it is possible that for some (weakly) χ-based interpreted system I, some (weakly) χ-consistent run r, and some timestamp $t > 0$,*

$$(I, r, t) \models K_i faulty_i.$$

Proof. This happens whenever there is a mismatch between actions recorded in the agent's local history and actions prescribed by the agent's protocol for the preceding local state. □

2.2.2 Global Introspection

Similarly to the virtual impossibility for the agent to ascertain its own correctness, it is similarly almost impossible for an agent to learn the Byzantine status of another agent.

Lemma 2.2.5. *For some agents $i \neq j$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that $f \geq 1$ and P_ϵ makes agent i gullible and all other agents delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, and for a timestamp $t \in \mathbb{N}$,*

$$(I, r, t) \not\models K_i faulty_j.$$

In particular, this statement holds for fully f-Byzantine agent-contexts.

Proof. For $t = 0$, the statement is obvious because no agent can be faulty in the initial state. For $t > 0$, the statement follows directly from Properties 2 and 6 of Lemma 2.1.16 applied to $\rho_j^m = \mathit{CFreeze}$ for all $j \neq i$. □

Lemma 2.2.6. *For agents $i \neq j$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that $f \geq 2$ and P_ϵ makes agent i gullible, agent j delayable and fallible, and all other agents delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, and for a timestamp $t > 0$,*

$$(I, r, t) \not\models K_i correct_j.$$

In particular, this statement holds for fully f-Byzantine agent-contexts.

Proof. The statement follows directly from Properties 2 and 5 of Lemma 2.1.16 applied to

$$\rho_k^m = \begin{cases} \mathit{CFreeze} & \text{if } k \notin \{i, j\} \\ \mathit{BFreeze}_j & \text{if } k = j. \end{cases}$$

It is also sufficient to set $\rho_j^0 = \mathit{BFreeze}_j$ and all other $\rho_k^m = \mathit{CFreeze}$ for $k \neq i$. □

In this proof, $BD = \{j\}$ and $Failed\,(adj) = \{i, j\}$, which is why it was necessary to allow at least two agents to become Byzantine.

Given that typical Byzantine agents can never be sure that they are correct, or that another agent is correct (faulty), or that a particular action/event happened, their behavior cannot rely on knowledge but should be governed by a weaker epistemic state. We define the following operators:

$$B_i\varphi := K_i(correct_i \to \varphi) \tag{2.20}$$

$$H_i\varphi := correct_i \to B_i\varphi = correct_i \to K_i(correct_i \to \varphi). \tag{2.21}$$

Thus, a belief in φ means that φ follows from the local state of the agent provided the agent is correct, whereas hope that φ states that this belief need only hold if the agent is correct.

Lemma 2.2.7 (Properties of belief and hope)**.** *For any formula φ, any agent i, the following formulas are propositional tautologies and, hence, are valid in every interpreted system.*

- $\models B_i\varphi \to H_i\varphi$
- $\models correct_i \to (B_i\varphi \leftrightarrow H_i\varphi)$
- $\models faulty_i \to H_i\varphi$

Proof. The first two statements are obvious. The last statement follows from the definition of $faulty_i$ as $\neg correct_i$. □

As can be seen from the preceding lemma, belief is stronger than hope in general, but equivalent to it for correct agents. Hope provides no information for faulty agents, whereas belief can, which can be used for designing algorithms for malfunctioning agents. In fact, given that

$$\models (correct_i \to faulty_i) \leftrightarrow faulty_i$$

is also a propositional tautology, it is the case that

$$\models B_i faulty_i \leftrightarrow K_i faulty_i$$

due to the normality of the K_i modality. In other words, in order to react to its own faults, the agent should know of them, which can be formulated in terms of the belief modality. On the other hand, $\not\models H_i faulty_i \to K_i faulty_i$ because a faulty agent always hopes to be faulty but may not know it is.

On the other hand, the hope modality will be technically convenient in future proofs by affording more elegant formulations for iterated modalities.

2.3 The Role of the Causal Cone

In the preceding two sections, we constructed a minimal run that is *subjectively* identical to a given run at a given node. In this section, we construct a similarly minimal run that is identical *objectively* as far as this node is concerned. In doing so, we reaffirm the intuitive view of the causal cone as the minimal causal foundations of the node.

Before formulating this statement, we prove a technical auxiliary lemma to the effect that using $filter_\epsilon^{\leq f}$ the second time to make sure the maximum of f Byzantine agents is respected—doing it the second time has no effect even if some events (not necessarily Byzantine) are removed in between. Let us denote

$$X_{\epsilon_k}^f := X_\epsilon \cap (BEvents_k \sqcup \{sleep\,(k), hibernate\,(k)\}), \tag{2.22}$$

$$Failed\,(X_\epsilon) := \{k \mid X_{\epsilon_k}^f \neq \varnothing\}. \tag{2.23}$$

Lemma 2.3.1. *For arbitrary $X_\epsilon, Y_\epsilon, Z_\epsilon \subset GEvents$, arbitrary $X_i, X_i' \subset \overline{GActions_i}$ for each $i \in \mathcal{A}$, any $f \in \mathbb{N}$, and arbitrary global histories h and h' such that*

$$\begin{aligned} &|\,\mathcal{A}\,(Failed\,(h))\,| \leq f \\ &\mathcal{A}\,(Failed\,(h')) \subset \mathcal{A}\,(Failed\,(h))\,, \\ &\quad Failed\,(Z_\epsilon) \subset \mathcal{A}\,(Failed\,(h)) \cup Failed\left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n)\right), \end{aligned}$$

we have

$$filter_\epsilon^{\leq f}\left(h',\quad \left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) \setminus Y_\epsilon\right) \cup Z_\epsilon,\quad X_1',\quad \ldots,\quad X_n'\right) = \left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) \setminus Y_\epsilon\right) \cup Z_\epsilon. \tag{2.24}$$

Proof. There are two possibilities:

$$filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) = \begin{cases} X_\epsilon & \text{if } |\,\mathcal{A}\,(Failed\,(h)) \cup Failed\,(X_\epsilon)| \leq f, \\ X_\epsilon \setminus \bigsqcup\limits_{k \in \mathcal{A}} X_{\epsilon_k}^f & \text{if } |\,\mathcal{A}\,(Failed\,(h)) \cup Failed\,(X_\epsilon)| > f. \end{cases}$$

Case 1. $filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) = X_\epsilon$ because $|\,\mathcal{A}\,(Failed\,(h)) \cup Failed\,(X_\epsilon)| \leq f$. Clearly,

$$\begin{aligned} Failed\left(\left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) \setminus Y_\epsilon\right) \cup Z_\epsilon\right) \quad &= \quad Failed\,((X_\epsilon \setminus Y_\epsilon) \cup Z_\epsilon) \quad \subset \\ &Failed\,(X_\epsilon) \cup Failed\,(Z_\epsilon) \quad \subset \\ Failed\,(X_\epsilon) \cup \Big(\mathcal{A}\,(Failed\,(h)) \cup Failed\left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n)\right)\Big) \quad &= \\ &\mathcal{A}\,(Failed\,(h)) \cup Failed\,(X_\epsilon). \end{aligned}$$

Thus,

$$\begin{aligned}\left|\mathcal{A}\left(Failed\left(h'\right)\right)\right. \quad \cup \quad & Failed\left(\left(filter_{\epsilon}^{\leq f}\left(h,\, X_{\epsilon},\, X_1,\, \ldots,\, X_n\right) \setminus Y_{\epsilon}\right) \cup Z_{\epsilon}\right)\Big| \quad \leq \\ & \left|\,\mathcal{A}\left(Failed\left(h\right)\right) \cup Failed\left(X_{\epsilon}\right)\right| \quad \leq \quad f,\end{aligned}$$

and (2.24) is fulfilled.

Case 2. $filter_{\epsilon}^{\leq f}\left(h,\, X_{\epsilon},\, X_1,\, \ldots,\, X_n\right) = X_{\epsilon} \setminus \bigsqcup\limits_{k \in \mathcal{A}} X_{\epsilon_k}^f$ because $\left|\,\mathcal{A}\left(Failed\left(h\right)\right) \cup Failed\left(X_{\epsilon}\right)\right| > f$.

In this case,

$$Failed\left(filter_{\epsilon}^{\leq f}\left(h,\, X_{\epsilon},\, X_1,\, \ldots,\, X_n\right)\right) = \varnothing,$$

meaning that

$$Failed\left(Z_{\epsilon}\right) \subset \mathcal{A}\left(Failed\left(h\right)\right).$$

Let us denote

$$U_{\epsilon} \quad := \quad \left(filter_{\epsilon}^{\leq f}\left(h,\, X_{\epsilon},\, X_1,\, \ldots,\, X_n\right) \setminus Y_{\epsilon}\right) \cup Z_{\epsilon}.$$

Then

$$\begin{aligned}U_{\epsilon_k}^f \quad = \quad & U_{\epsilon} \cap \Big(BEvents_k \sqcup \left\{sleep\left(k\right), hibernate\left(k\right)\right\}\Big) \quad = \\ & \left(\left(\left(X_{\epsilon} \setminus \bigsqcup_{k \in \mathcal{A}} X_{\epsilon_k}^f\right) \setminus Y_{\epsilon}\right) \cup Z_{\epsilon}\right) \cap \Big(BEvents_k \sqcup \left\{sleep\left(k\right), hibernate\left(k\right)\right\}\Big) \quad = \\ & \left(\left(\left(X_{\epsilon} \setminus \bigsqcup_{k \in \mathcal{A}} X_{\epsilon_k}^f\right) \setminus Y_{\epsilon}\right) \cap \Big(BEvents_k \sqcup \left\{sleep\left(k\right), hibernate\left(k\right)\right\}\Big)\right) \quad \cup \\ & \Big(Z_{\epsilon} \cap \Big(BEvents_k \sqcup \left\{sleep\left(k\right), hibernate\left(k\right)\right\}\Big)\Big) \quad = \\ & \varnothing \quad \cup \quad Z_{\epsilon_k}^f \quad = \quad Z_{\epsilon_k}^f.\end{aligned}$$

Since $U_{\epsilon_k}^f = Z_{\epsilon_k}^f$ for all agents $k \in \mathcal{A}$, it immediately follows that

$$Failed\left(U_{\epsilon}\right) = Failed\left(Z_{\epsilon}\right) \subset \mathcal{A}\left(Failed\left(h\right)\right).$$

and, consequently

$$\begin{aligned}\left|\,\mathcal{A}\left(Failed\left(h'\right)\right) \cup Failed\left(U_{\epsilon}\right)\right| \quad \leq & \\ \left|\,\mathcal{A}\left(Failed\left(h\right)\right) \cup \mathcal{A}\left(Failed\left(h\right)\right)\right| \quad = \quad & \left|\,\mathcal{A}\left(Failed\left(h\right)\right)\right| \quad \leq \quad f.\end{aligned}$$

Consequently, (2.24) is fulfilled again.

Thus, we have demonstrated (2.24) in both cases. □

Definition 2.3.2. Let $X \subset \mathcal{A} \times \mathbb{N}$ be a set of nodes. We define

$$X \Rsh \quad := \quad X \cup \left\{(i, t+1) \mid (i,t) \in X\right\}$$

to contain X along with all immediate futures of nodes in X. Applying this to a causal cone we define

$$V_{(j,t')}^{\Rsh}\left(r\right) \quad := \quad \left(V_{(j,t')}\left(r\right)\right) \Rsh \setminus \left\{(j, t'+1)\right\}$$

excluding only the immediate future of the tip of the cone.

Lemma 2.3.3. *For a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$ with $t > 0$, for a non-excluding agent-context $\chi = \left((P_{\epsilon}, \mathscr{G}(0), \tau^{B_f}, \Psi), P\right)$ such that P_{ϵ} makes all agents j both delayable and correctable,[3] for*

[3]Strictly speaking, it is sufficient to demand delayability only for all agents j such that $(j, t-1) \notin V_{\theta}\left(r\right)$, whereas correctability is only necessary for all agents j such that $(j,0) \in V_{\theta}\left(r\right)$. Additionally, the condition of correctability can be dropped altogether if $f \geq n$.

a χ-based interpreted system $I = (R^{\chi}, \pi)$, and for a run $r \in R^{\chi}$, consider an adjustment $adj = [B_{t-1}; \ldots; B_0]$ satisfying (2.4) for all $0 \leq m \leq t-1$ with

$$\rho_j^m = \begin{cases} V_\theta(r)\text{-}Focus_j^m & \text{if } (j,m) \in V_\theta(r)\ , \\ CFreeze & \text{if } (j,m) \notin V_\theta(r). \end{cases}$$

Each run $r' \in R\left(\tau^{B_f}_{P_\epsilon,P}, r, adj\right)$ satisfies the following properties:

1. *For any agent $j \in \mathcal{A}$ and for any timestamp $m \leq t-1$*

$$\beta^m_{\epsilon_j}(r') \subset \beta^m_{\epsilon_j}(r) \tag{2.25}$$
$$\beta^m_{\epsilon_j}(r') = \beta^m_{\epsilon_j}(r) \qquad \text{if } (j,m+1) \in V_\theta(r), \tag{2.26}$$
$$\beta^m_{\epsilon_j}(r') = \varnothing \qquad \text{if } (j,m) \notin V_\theta(r); \tag{2.27}$$
$$\beta^m_{f_j}(r') = \beta^m_{f_j}(r) \qquad \text{if } (j,m) \in V_\theta(r), \tag{2.28}$$
$$\beta^m_{g_j}(r') = \beta^m_{g_j}(r) \qquad \text{if } (j,m) \in V_\theta(r), \tag{2.29}$$

2. $(\forall j \in \mathcal{A})\,(\forall m \leq t-1) \quad \beta^m_j(r') = \begin{cases} \beta^m_j(r) & \text{if } (j,m) \in V_\theta(r), \\ \varnothing & \text{if } (j,m) \notin V_\theta(r); \end{cases}$

3. *r and r' agree on $\{(j,m) \mid (j,m+1) \in V_\theta(r)\}$;*

4. *$r_j(m) = r'_j(m)$ whenever $(j,m) \in V_\theta(r)$. In particular, $r_i(t) = r'_i(t)$;*

5. *$Bad_{V^{\ni}_\theta(r)}(r') = Bad_{V^{\ni}_\theta(r)}(r)$;*

6. *$Failed(r'(t)) = Failed_{V^{\ni}_\theta(r)}(r') = Failed_{V^{\ni}_\theta(r)}(r)$;*

7. *$(\forall m \leq t) \quad Failed(r'(m)) \subset Failed(r(m))$;*

8. *for any node $\beta \in V^{\ni}_\theta(r)$ and any action/event $o \in \overline{Actions} \sqcup \overline{Events}$ that is not a receive event,*

$$(I,r,t) \models \overline{occurred}_\beta(o) \iff (I,r',t) \models \overline{occurred}_\beta(o),$$
$$(I,r,t) \models fake_\beta(o) \iff (I,r',t) \models fake_\beta(o)\,;$$

   in addition, if $\beta \in V_\theta(r)$ proper, then the equivalences hold also for receive events;

9. *r' is $\tau^{B_f}_{P_\epsilon,P}$-transitional.*

In particular, this statement holds for fully f-Byzantine agent-contexts.

Proof. Let $r' \in R\left(\tau^{B_f}_{P_\epsilon,P}, r, adj\right)$. Most parts of **Properties 1 and 2** directly follow from the definitions of $V_\theta(r)$-$Focus_j^m$ in (2.15), and *CFreeze* in (2.12). The only point we elaborate on is (2.26). Given (2.25), it is sufficient to prove that $\beta^m_{\epsilon_j}(r) \subset \beta^m_{\epsilon_j}(r')$ whenever $(j,m+1) \in V_\theta(r)$. Any non-receive event from $\beta^m_{\epsilon_j}(r)$ is automatically present in $\beta^m_{\epsilon_j}(r')$ because $V_\theta(r)$-$Focus_j^m$ only removes receive-events. If $grecv(j,k,\mu,id) \in \beta^m_{\epsilon_j}(r)$, then, by Corollaries 1.7.13 and 1.3.16 and Lemma 1.3.15, we have $id = id(k,j,\mu,s,t')$ for some $s \in \mathbb{N}$ such that $(k,t') \to^{r,t} (j,m+1)$. Thus, $(j,m+1) \in V_\theta(r)$ is sufficient to put $(k,t') \in V_\theta(r)$, which ensures that $grecv(j,k,\mu,id) \in \beta^m_{\epsilon_j}(r') = \mathfrak{e}V_\theta(r)$-$Focus_j^m(r)$.

According to Def. 1.3.20, **Property 3** states that

(a) $r_j(m) = r'_j(m)$,

(b) $\beta^m_j(r) = \beta^m_j(r')$, and

(c) $\beta^m_{\epsilon_j}(r) = \beta^m_{\epsilon_j}(r')$

whenever $(j, m+1) \in V_\theta(r)$. Thus, both (b) and (c) for all such nodes (j, m) follow directly from (2.26) part of Property 1 and from Property 2, in the latter case because $(j, m) \in V_\theta(r)$ whenever $(j, m+1) \in V_\theta(r)$. We now prove (a) by induction on timestamp $m \leq t-1$. For the base case $m = 0$, statement (a) follows from $r'(0) = r(0)$ by (2.5). For the induction step from m to $m+1$, consider $(j, m+2) \in V_\theta(r)$. Since $(j, m+1)$ must also be in the causal cone $V_\theta(r)$, by induction hypothesis, $r_j(m) = r'_j(m)$. Given (b) and (c) for (j, m), we have that r and r' agree on (j, m). Hence, by Lemma 1.3.22, $r_j(m+1) = r'_j(m+1)$, which completes the induction step, the induction proof of (a), and the proof of Property 3.

Property 4 now follows easily. Indeed, either $m = 0$ and $r_j(0) = r'_j(0)$ by (2.5). Otherwise, $m > 0$ and r and r' agree on $(j, m-1)$ by Property 3. The desired statement now follows by Lemma 1.3.22. It remains to note that $(i, t) \in V_{(i,t)}(r)$.

To prove **Property 5**, recall that by the definition of transitional runs for r and adjusted runs for r', the global histories in r and r' up till timestamp t are constructed from the β-sets according to the updating phase. By (2.28) part of Property 1, for every $(j, m) \in V_\theta(r)$ with $m < t$

$$\beta^m_{f_j}(r) = \beta^m_{f_j}(r').$$

This can be reformulated as for every $(j, m) \in V^{\ni}_\theta(r)$ with $m > 0$

$$\beta^{m-1}_{f_j}(r) = \beta^{m-1}_{f_j}(r').$$

It remains to note, by unfolding Definitions 1.3.3 and 1.3.5, that

$$\begin{aligned}
&(j,m) \in Bad_{V^{\ni}_\theta(r)}(r) \iff (j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad\left(r, T(V^{\ni}_\theta(r))\right) \iff \\
&(j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad(r,t) \iff (j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad(r(t)) \iff \\
&\qquad (j,m) \in V^{\ni}_\theta(r) \quad \wedge \quad m > 0 \quad \wedge \quad \beta^{m-1}_{f_j}(r) \neq \varnothing \iff \\
&\qquad (j,m) \in V^{\ni}_\theta(r) \quad \wedge \quad m > 0 \quad \wedge \quad \beta^{m-1}_{f_j}(r') \neq \varnothing \iff \\
&(j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad(r'(t)) \iff (j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad(r',t) \iff \\
&\qquad (j,m) \in V^{\ni}_\theta(r) \wedge (j,m) \in Bad\left(r', T(V^{\ni}_\theta(r))\right) \iff (j,m) \in Bad_{V^{\ni}_\theta(r)}(r')
\end{aligned}$$

Given that no (Byzantine) events happen outside $V^{\ni}_\theta(r)$, **Property 6** is a direct consequence of Property 5 since $Failed_{V^{\ni}_\theta(r)}(r)$ is fully determined by $Bad_{V^{\ni}_\theta(r)}(r)$, as also $Failed_{V^{\ni}_\theta(r)}(r')$ by $Bad_{V^{\ni}_\theta(r)}(r')$ (see Definitions 1.3.3 and 1.3.5).

To show **Property 7**, consider any $(l, m') \in Failed(r'(m))$. Then $0 < m' \leq m \leq t$ and $(l, m'') \in Bad(r'(m))$ for some $0 < m'' \leq m' \leq m \leq t$, i.e., that $\beta^{m''-1}_{f_l}(r') \neq \varnothing$, meaning that $\beta^{m''-1}_{\epsilon_l}(r') \neq \varnothing$, where $m'' - 1 \leq t - 1$. By (2.27) part of Property 1, this implies that $(l, m''-1) \in V_\theta(r)$. Hence, by (2.28) part of Property 1, $\beta^{m''-1}_{f_l}(r) = \beta^{m''-1}_{f_l}(r') \neq \varnothing$ and, consequently, $(l, m'') \in Bad(r(m))$, meaning that $(l, m') \in Failed(r(m))$.

Property 8 follows either from Property 3 or directly from part (2.26) of Property 1 and from Property 2. Indeed, for $\beta = (j, m)$ with $m \leq t$, according to (1.79) and (1.80), the truth values of both $\overline{occurred}_{(j,m)}(o)$ and $fake_{(j,m)}(o)$ are fully determined by the $\beta^{m-1}_{\epsilon_j}$ and β^{m-1}_j sets, which are identical for r and r' within the causal cone $V_\theta(r)$ by part (2.26) of Property 1 and by Property 2 (note that $m-1 \leq t-1$ and that for $m = 0$ the atomic propositions are trivially false). Similarly, these sets are identical modulo receive events in rounds immediately following the latest node in the causal cone.

It remains to prove **Property 9**, i.e., the transitionality of the run r'. As already discussed in the proof of Lemma 2.1.16(1), it is sufficient to show that all the β-sets produced by the adjustment *adj* can be obtained in a regular manner from appropriately chosen α-sets allowed by the protocol.

First, we define the requisite α-sets explaining why they can be issued by the relevant protocols. Consider any $m \leq t-1$. For each agent j such that $(j, m) \in V_\theta(r)$ we have

$$r_j(m) = r'_j(m) \qquad \text{for all } (j, m) \in V_\theta(r) \tag{2.30}$$

by the already proven Property 4. Hence, the set

$$\alpha_j^m(r) \in P_j(r_j(m)) = P_j\left(r'_j(m)\right)$$

can be chosen by the adversary to be $\alpha_j^m(r')$. For $(j,m) \notin V_\theta(r)$, it does not matter what the adversary chooses as $\alpha_j^m(r')$ (recall that there is always at least one choice available by the no-apocalypse clause). For the environment, by the correctability of all agents

$$filter_\epsilon^{\leq f}\left(r(m), \alpha_\epsilon^m(r), \alpha_1^m(r), \ldots, \alpha_n^m(r)\right) \quad \in \quad P_\epsilon(m)$$

because $\alpha_\epsilon^m(r) \in P_\epsilon(m)$. Further, by the delayability of all agents, the set

$$filter_\epsilon^{\leq f}\left(r(m), \alpha_\epsilon^m(r), \alpha_1^m(r), \ldots, \alpha_n^m(r)\right) \setminus \bigsqcup_{(l,m)\notin V_\theta(r)} GEvents_l \quad \in \quad P_\epsilon(m).$$

Thus,

$$\alpha_\epsilon^m(r') \quad := \quad filter_\epsilon^{\leq f}\left(r(m), \alpha_\epsilon^m(r), \alpha_1^m(r), \ldots, \alpha_n^m(r)\right) \setminus \bigsqcup_{(l,m)\notin V_\theta(r)} GEvents_l \tag{2.31}$$

can be chosen by the adversary. Note that

$$\alpha_\epsilon^m(r') \subset \alpha_\epsilon^m(r).$$

It remains to show that the filtering employed by the transition function $\tau_{P_\epsilon,P}^{B_f}$ turns these sets $\alpha_\epsilon^m(r')$, $\alpha_1^m(r'), \ldots, \alpha_n^m(r')$ into the exact β-sets prescribed by the adjustment *adj*. Given that $\beta_\epsilon^m(r')$ is partitioned into $\beta_{\epsilon_j}^m(r')$ for each $j \in \mathcal{A}$, we can verify the correctness for each node (j,m) separately. Let us abbreviate

$$\begin{array}{lcl} \Upsilon & := & filter_\epsilon^{B_f}\left(r'(m), \quad \alpha_\epsilon^m(r'), \quad \alpha_1^m(r'), \quad \ldots, \quad \alpha_n^m(r')\right), \\ \Upsilon_j & := & \Upsilon \quad \cap \quad GEvents_j \\ \Xi_j & := & filter_j^B\left(\alpha_1^m(r'), \quad \ldots, \quad \alpha_n^m(r'), \quad \Upsilon\right) \end{array}$$

Our goal is to show that $\Upsilon_j = \beta_{\epsilon_j}^m(r')$ and $\Xi_j = \beta_j^m(r')$ for each $j \in \mathcal{A}$.

- We start with nodes $(j,m) \notin V_\theta(r)$

  $$\Upsilon_j = \Upsilon \cap GEvents_j \subset \alpha_\epsilon^m(r') \cap GEvents_j = \varnothing = \mathfrak{e}\, CFreeze(r) = \beta_{\epsilon_j}^m(r').$$

  In particular, $go(j) \notin \Upsilon$, resulting in

  $$\Xi_j = \varnothing = \mathfrak{a}\, CFreeze(r) = \beta_j^m(r').$$

- We now turn to $(j,m) \in V_\theta(r)$. We abbreviate

  $$Y_\epsilon := \bigsqcup_{(l,m)\notin V_\theta(r)} GEvents_l, \tag{2.32}$$

  $$\overline{\alpha^m(r)} := \alpha_1^m(r), \ldots, \alpha_n^m(r), \tag{2.33}$$

  $$\overline{\alpha^m(r')} := \alpha_1^m(r'), \ldots, \alpha_n^m(r'). \tag{2.34}$$

  By (1.119)

  $$\Upsilon \quad = \quad filter_\epsilon^{B_f}\left(r'(m), \quad filter_\epsilon^{\leq f}\left(r(m), \alpha_\epsilon^m(r), \overline{\alpha^m(r)}\right) \setminus Y_\epsilon, \quad \overline{\alpha^m(r')}\right) \quad =$$
  $$filter_\epsilon^{B}\left(r'(m), \quad filter_\epsilon^{\leq f}\left(r'(m), \quad filter_\epsilon^{\leq f}\left(r(m), \alpha_\epsilon^m(r), \overline{\alpha^m(r)}\right) \setminus Y_\epsilon, \quad \overline{\alpha^m(r')}\right), \quad \overline{\alpha^m(r')}\right).$$

By the already proven Property 7, $Failed\,(r'(m)) \subset Failed\,(r(m))$ (recall that $m \leq t-1$). Hence, by Lemma 2.3.1 with

$$X_\epsilon = \alpha^m_\epsilon\,(r), \qquad Z_\epsilon = \varnothing, \qquad h = r(m), \qquad h' = r'(m), \qquad X_i = \alpha^m_i\,(r), \qquad X'_i = \alpha^m_i\,(r'),$$

given that $|\,\mathcal{A}\,(Failed\,(r(m)))\,| \leq f$ for the original run, we have

$$\Upsilon \quad = \quad filter^B_\epsilon\left(r'(m), \quad filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\,(r),\, \overline{\alpha^m\,(r)}\right) \setminus Y_\epsilon, \quad \overline{\alpha^m\,(r')}\right). \quad (2.35)$$

At the same time,

$$\beta^m_\epsilon\,(r) \quad = \quad filter^{B_f}_\epsilon\left(r(m),\ \alpha^m_\epsilon\,(r),\ \overline{\alpha^m\,(r)}\right) \quad = \\ filter^B_\epsilon\left(r(m),\ filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\,(r),\, \overline{\alpha^m\,(r)}\right),\ \overline{\alpha^m\,(r)}\right). \quad (2.36)$$

Let us further abbreviate

$$\Omega := filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\,(r), \overline{\alpha^m\,(r)}\right), \qquad \Omega' := \Omega \setminus Y_\epsilon,$$
$$\Omega_j := \Omega \cap GEvents_j, \qquad \Omega'_j := \Omega' \cap GEvents_j$$

making

$$\Upsilon = filter^B_\epsilon\left(r'(m), \quad \Omega', \quad \overline{\alpha^m\,(r')}\right), \quad (2.37)$$
$$\beta^m_\epsilon\,(r) = filter^B_\epsilon\left(r(m), \quad \Omega, \quad \overline{\alpha^m\,(r)}\right). \quad (2.38)$$

Given that $(j,m) \in V_\theta\,(r)$, by (2.32) we have $GEvents_j \cap Y_\epsilon = \varnothing$. In other words,

$$\Omega_j = \Omega'_j. \quad (2.39)$$

This, in particular, means that

$$active(j,\Omega) \iff active(j,\Omega'), \qquad passive(j,\Omega) \iff passive(j,\Omega') \quad (2.40)$$
$$aware(j,\Omega) \iff aware(j,\Omega'), \qquad unaware(j,\Omega) \iff unaware(j,\Omega') \quad (2.41)$$

Let us first prove that $\Xi_j = \beta^m_j\,(r')$ whenever $(j,m) \in V_\theta\,(r)$.

– If $passive(j,\Omega')$, then $passive(j,\Omega)$ and, hence, by the already proven Property 2,

$$\Xi_j = \varnothing = \beta^m_j\,(r) = \beta^m_j\,(r').$$

– If $active(j,\Omega')$, then $active(j,\Omega)$, and we have, again using Property 2,

$$\Xi_j = \alpha^m_j\,(r') = \alpha^m_j\,(r) = \beta^m_j\,(r) = \beta^m_j\,(r').$$

It remains to show that $\Upsilon_j = \beta^m_{\epsilon_j}\,(r')$ whenever $(j,m) \in V_\theta\,(r)$,

– If U does not have the form $grecv(j,k,\mu,id)$, then $V_\theta\,(r)$-$Focus^m_j$ and (2.39) ensure that

$$U \in \beta^m_{\epsilon_j}\,(r') \quad \Leftrightarrow \quad U \in \mathfrak{e}V_\theta\,(r)\text{-}Focus^m_j\,(r) \quad \Leftrightarrow \\ U \in \beta^m_{\epsilon_j}\,(r) \quad \Leftrightarrow \quad U \in \Omega_j \quad \Leftrightarrow \quad U \in \Omega'_j \quad \Leftrightarrow \quad U \in \Upsilon_j.$$

– Let now assume $U = grecv(j,k,\mu,id)$ and denote

$$V := gsend(k,j,\mu,id) \qquad \text{and} \qquad W_A := fake\,(k, gsend(k,j,\mu,id) \mapsto A).$$

Then

$$U \in \beta^m_{\epsilon_j}(r') \quad \Leftrightarrow \quad U \in \mathfrak{e}V_\theta(r)\text{-}Focus^m_j(r) \quad \Leftrightarrow$$
$$U \in \beta^m_{\epsilon_j}(r) \text{ and } (\exists t' \le m)\left((\exists s) id = id(k,j,\mu,s,t') \text{ and } (k,t') \in V_\theta(r)\right) \quad \Leftrightarrow$$
$$U \in \Omega_j \text{ and } \Big((\exists t' < m)\Big(\Big(V \in \beta^{t'}_k(r) \text{ or } (\exists A) W_A \in \beta^{t'}_{f_k}(r)\Big) \text{ and } (k,t') \in V_\theta(r)\Big) \text{ or}$$
$$\Big(\big((V \in \alpha^m_k(r) \& active(\Omega,k)) \text{ or } (\exists A) W_A \in \Omega_k\big) \text{ and } (k,m) \in V_\theta(r)\Big)\Big) \quad \Leftrightarrow$$
$$U \in \Omega'_j \text{ and } \Big((\exists t' < m)\Big(\Big(V \in \beta^{t'}_k(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{f_k}(r')\Big) \text{ and } (k,t') \in V_\theta(r)\Big) \text{ or}$$
$$\Big(\big((V \in \alpha^m_k(r') \& active(\Omega',k)) \text{ or } (\exists A) W_A \in \Omega'_k\big) \text{ and } (k,m) \in V_\theta(r)\Big)\Big) \quad \Leftrightarrow$$
$$U \in \Upsilon_j \quad \square$$

Corollary 2.3.4. *Property 8 implies that*

$$\left(\forall \beta \in V_\theta(r)\right)\left(\forall o \in \overline{Actions} \sqcup \overline{Events}\right) \quad (I,r,t) \models occurred_\beta(o) \Leftrightarrow (I,r',t) \models occurred_\beta(o).$$

Corollary 2.3.5. *Lemma 2.3.3 also implies that for any node $\eta \in \mathcal{A} \times \mathbb{N}$ and any path ξ,*

$$\eta \rightsquigarrow^{r,t}_\xi \theta \qquad \Longleftrightarrow \qquad \eta \rightsquigarrow^{r',t}_\xi \theta.$$

In particular,

$$V_\theta(r) = V_\theta(r'). \tag{2.42}$$

Proof. From left to right, if ξ connects η to θ in r, then all the nodes along ξ are in the causal cone, as are their immediate pasts. Thus all sends, correct or Byzantine, from these nodes are preserved in the new run r'. All receives forming the path are also preserved because they match sends originating from the causal cone. Therefore, the whole causal path ξ is preserved in r'.

From right to left, the argument is even simpler since all sends and receives in r' are present in the original run r. $\square$

Lemma 2.3.6. *For a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, agent $j \in \mathcal{A}$, timestamp $t' \ge t > 0$, for a non-excluding agent-context $\chi = \left((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\right)$ such that P_ϵ makes all agents gullible, correctable, and delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, and for a run $r \in R^\chi$, and the local action or event $o \in \overline{Actions} \sqcup \overline{Events}$,*

$$(I,r,t') \models correct_j \wedge K_j(correct_j \to occurred_{(i,t)}(o)) \qquad \Longrightarrow \qquad (i,t-1) \rightsquigarrow^{r,t'} (j,t')$$

Proof. Let

$$(I,r,t') \models correct_j \wedge K_j(correct_j \to occurred_{(i,t)}(o))$$

By Lemma 2.3.3 applied to the node (j,t'), there is an alternative run $r' \in R^\chi$ such that $\beta^m_{\epsilon_i}(r') = \beta^m_i(r') = \varnothing$ for all $(i,m) \notin V_{(j,t')}(r)$ with $m < t'$, and $r'_j(t') = r_j(t')$, and

$$(I,r,t') \models correct_j \quad \Longleftrightarrow \quad (j,t') \notin Failed(r,t') \quad \Longleftrightarrow$$
$$(j,t') \notin Failed(r',t') \quad \Longleftrightarrow \quad (I,r',t') \models correct_j.$$

From $r'_j(t') = r_j(t')$ and $(I,r,t') \models K_j(correct_j \to occurred_{(i,t)}(o))$, it follows that

$$(I,r',t') \models correct_j \to occurred_{(i,t)}(o)$$

and, given $(I,r',t') \models correct_j$, also

$$(I,r',t') \models occurred_{(i,t)}(o).$$

Therefore, at least one of $\beta^{t-1}_{\epsilon_i}(r')$ and $\beta^{t-1}_i(r')$ must not be empty. Given that $t-1 < t'$, this means that $(i,t-1) \in V_{(j,t')}(r)$, i.e., that $(i,t-1) \rightsquigarrow^{r,t'} (j,t')$. $\square$

Remark 2.3.7. Note that the requirement that $(I, r, t') \models correct_j$ is essential. Otherwise, the construction from the Brain-in-the-Vat Lemma 2.1.16 with an extra protocol-violating Byzantine action by j results in a situation where no causal links lead to j despite the fact that $(I, r, t') \models K_j(correct_j \to occurred_{(i,t)}(o))$ simply by virtue of knowing its faultiness, $(I, r, t') \models K_j \neg correct_j$.

2.4 The Byzantine Version of the Causal Cone

Before formulating the statement about the Byzantine causal cone, we reformulate Lemma 2.3.1 slightly:

Corollary 2.4.1. *For arbitrary $X_\epsilon, Y_\epsilon, Z_\epsilon \subset GEvents$, arbitrary $X_i, X_i' \subset \overline{GActions_i}$ for each $i \in \mathcal{A}$, any $f \in \mathbb{N}$, and arbitrary global histories h and h' such that*

$$\begin{aligned} &|\,\mathcal{A}\,(Failed\,(h))\,| \leq f \\ &\mathcal{A}\,(Failed\,(h')) \subset \mathcal{A}\,(Failed\,(h))\,, \\ &\quad Failed\,(Z_\epsilon) \subset \mathcal{A}\,(Failed\,(h)) \cup Failed\left(filter_\epsilon^{B_f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n)\right), \end{aligned}$$

we have

$$filter_\epsilon^{\leq f}\left(h', \quad \left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) \setminus Y_\epsilon\right) \cup Z_\epsilon, \quad X_1', \quad \ldots, \quad X_n'\right) = \left(filter_\epsilon^{\leq f}\,(h,\, X_\epsilon,\, X_1,\, \ldots,\, X_n) \setminus Y_\epsilon\right) \cup Z_\epsilon. \quad (2.43)$$

Proof. The only change in formulation compared to Lemma 2.3.1 is the replacement of the $\leq f$ filter with the B_f filter in the assumed properties of Z_ϵ. Fortunately, the B_f filter is defined by applying the B filter on top of the $\leq f$ filter. Moreover, the B filter does not remove any Byzantine events. Thus, as far as the number of agents being made faulty is concerned, the two filters have the same effect. □

Lemma 2.4.2. *For $f \in \mathbb{N}$, for a non-excluding agent-context $\chi = \left((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\right)$ such that all agents are gullible, correctable, and delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, for a node $\theta = (i, t) \in \mathcal{A} \times \mathbb{N}$ such that $(i, t) \notin Failed\,(r, t)$, let us partition the causal cone of θ into the following types of nodes depending on their causal relationship to θ:*

$$WellConnected := \left\{(j, m) \in V_\theta\,(r) \mid \overline{\Xi_\theta^{\{(j,m)\}}}\,(r) \neq \varnothing\right\} \quad (2.44)$$

$$FaultBuffer := \{(j, m) \in V_\theta\,(r) \mid (j, m+1) \in Failed\,(r, t)\} \quad (2.45)$$

$$SilentMasses := V_\theta\,(r) \setminus (FaultBuffer \sqcup WellConnected) \quad (2.46)$$

We define an adjustment $adj = [B_{t-1}; \ldots; B_0]$ satisfying (2.4) such that for each $m \leq t-1$:

$$\rho_j^m := \begin{cases} (V_\theta\,(r) \setminus SilentMasses)\text{-}Focus_j^m & \text{if } (j, m) \in WellConnected, \\ FakeEcho_j^m & \text{if } (j, m) \in FaultBuffer, \\ CFreeze & \text{if } (j, m) \in \left((\mathcal{A} \times \mathbb{N}) \setminus V_\theta\,(r)\right) \sqcup SilentMasses. \end{cases} \quad (2.47)$$

Then each run $r' \in R\left(\tau_{P_\epsilon, P}^{B_f}, r, adj\right)$, satisfies the following properties:

A. $\left(\forall (j, m) \in WellConnected\right) \quad r_j'\,(m) = r_j\,(m)$;

B. $(\forall m \leq t)\, r_i'(m) = r_i(m)$;

C. $Bad\,(r', t) = Bad_{V_\theta^\ni(r)}\,(r') = Failed_{V_\theta^\ni(r)}\,(r)$;
in particular, $(i, t') \notin Bad\,(r', t)$ for any $t' \leq t$ and $(i, t) \notin Failed\,(r', t)$;

D. $Failed\,(r', m) \subset Failed\,(r, m)$ *for all $m \leq t$;*

E. r' *is* $\tau^{B_f}_{P_\epsilon,P}$*-transitional.*

Corollary 2.4.3. *It immediately follows from Statement C. that*

F. $|\mathcal{A}(Failed\,(r',t))| = |\mathcal{A}(Failed\left(Failed_{V^{\ni}_{\theta}(r)}\,(r)\right))| \leq |\mathcal{A}(Failed\,(r,t))| \leq f$*;*

Proof. We start by stating several simple properties of these types of nodes.

- $\mathit{WellConnected} \cap \mathit{FaultBuffer} = \varnothing$. To prove this, it is sufficient to note that no correct path can lead from a node whose immediate future node is faulty, except, potentially, for the trivial path from the tip (i,t) of the causal cone $V_\theta\,(r)$ to itself. However, even if the immediate future $(i,t+1)$ of the tip is faulty, it cannot belong to $Failed\,(r,t)$, making it impossible for $(i,t) \in \mathit{FaultBuffer}$.

- If $(k,t') \to^{r,t} (j,m)$ and the node $(j,m) \in \mathit{WellConnected}$, then

$$(k,t') \in \mathit{WellConnected} \sqcup \mathit{FaultBuffer}. \tag{2.48}$$

  Indeed, for (j,m) to be in *WellConnected*, there must exist a correct path ξ such that $(j,m) \rightsquigarrow^{r,t}_{\xi} (i,t)$. First thing to note is that the path

$$(k,t') \to^{r,t} (j,m) \rightsquigarrow^{r,t}_{\xi} (i,t) \tag{2.49}$$

  places $(k,t') \in V_\theta\,(r)$. Thus, if $(k,t'+1) \in Failed\,(r,t)$, then $(k,t') \in \mathit{FaultBuffer}$. Otherwise, (2.49) is a correct path, meaning that $(k,t') \in \mathit{WellConnected}$.

- $(i,t) \in \mathit{WellConnected}$. This follows from the assumption $(i,t) \notin Failed\,(r,t)$, making the trivial path (i,t) a correct path from (i,t) to itself.

- If $(j,m) \in \mathit{WellConnected}$, then $(j,m') \in \mathit{WellConnected}$ for all $m' \leq m$. Indeed, for a correct path ξ from (j,m) to (i,t) that must exist for the *WellConnected* node (j,m), consider the path

$$(j,m') \to^{r,t} (j,m'+1) \to^{r,t} \ldots (j,m) \rightsquigarrow^{r,t}_{\xi} (i,t).$$

  If $j = i$ and $m = t$, then this path is correct because $(i,t) \notin Failed\,(r,t)$. Otherwise, this path is correct because $(j,m+1) \notin Failed\,(r,t)$ due to the correctness of ξ.

- $(i,m) \in \mathit{WellConnected}$ for all $m \leq t$. This is a direct corollary of the last two items. It also means that Statement B. follows from Statement A.

These considerations are sufficient to to prove Statement A. by induction on m. The base case $m = 0$ follows by (2.5). Assuming the statement holds for m, let us prove it for $m+1$. As we just proved,

$$(j,m+1) \in \mathit{WellConnected} \qquad \Longrightarrow \qquad (j,m) \in \mathit{WellConnected}.$$

Thus, $r_j(m) = r'_j(m)$ by IH. It is easy to see that

$$\mathfrak{e}(V_\theta\,(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j = \beta^m_{\epsilon_j}\,(r).$$

Indeed, the only $\beta^m_{\epsilon_j}\,(r)$ events that might be excluded are receive events $\mathit{grecv}(j,k,\mu,id)$ such that $id = id(k,j,\mu,s,t')$ for some $s,t' \in \mathbb{N}$ such that $(k,t') \notin V_\theta\,(r)$ or $(k,t') \in \mathit{SilentMasses}$. However, if $\mathit{grecv}(j,k,\mu,id) \in \beta^m_{\epsilon_j}\,(r)$, then $t' \leq m$ and there must be a causal link from (k,t') to $(j,m+1)$. Coupled with

$$(j,m+1) \in \mathit{WellConnected} \subset V_\theta\,(r)\,,$$

this places $(k,t') \in V_\theta\,(r)$. Further, (2.48) means that were $(k,t') \notin \mathit{SilentMasses}$. This guarantees the preservation of $\mathit{grecv}(j,k,\mu,id)$ in $\mathfrak{e}(V_\theta\,(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j$.

$$\mathfrak{a}(V_\theta\,(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j = \beta^m_j\,(r)$$

always holds. Thus, runs r and r' agree on (j,m). It follows from Lemma 1.3.22 that $r_j(m+1) = r'_j(m+1)$. This completes the proof of Statement A.

Statement A. implies Statement B. because $(i,m) \in \mathit{WellConnected}$ for all $m \le t$.

To prove Statement C., observe that by the definition of *adj*

- *CFreeze*, which is applied to *SilentMasses* and outside the causal cone, produces neither actions nor events, in particular, no Byzantine events; in particular, no node outside $V_\theta^{\ni}(r)$ can be bad;
- $(V_\theta(r) \setminus \mathit{SilentMasses})$-$\mathit{Focus}_j^m$, which is applied to *WellConnected*, produces no Byzantine events, exactly as in the original run, because for *WellConnected* nodes $(j,m+1) \notin \mathit{Failed}(r,t)$; and
- $\mathit{FakeEcho}_j^m$, which is applied to *FaultBuffer*, makes the next node *Bad* due, at the very least, to the $\mathit{fail}(j)$ event. In other words

$$\begin{aligned}\mathit{Bad}_{V_\theta^{\ni}(r)}(r') = \{(j,m) \in V_\theta^{\ni}(r) \mid (j,m-1) \in \mathit{FaultBuffer}\} = \\ \{(j,m) \in V_\theta^{\ni}(r) \mid (j,m-1) \in V_\theta(r) \,\&\, (j,m) \in \mathit{Failed}(r,t)\} = \\ \{(j,m) \in V_\theta^{\ni}(r) \mid (j,m) \in \mathit{Failed}(r,t)\} = \mathit{Failed}_{V_\theta^{\ni}(r)}(r).\end{aligned}$$

To show Statement D., we use the following chain of inclusions:

$$\begin{aligned}(j,m) \in \mathit{Failed}(r',t) \quad &\Longrightarrow \quad (\exists m' \le m \le t)\,(j,m') \in \mathit{Bad}(r',t) \quad \Longrightarrow \\ &(\exists m' \le m \le t)\left((j,m'-1) \in \mathit{FaultBuffer}\right) \quad \Longrightarrow \\ &(\exists m' \le m \le t)\left((j,m') \in \mathit{Failed}(r,t)\right) \quad \Longrightarrow \quad (j,m) \in \mathit{Failed}(r,t)\end{aligned}$$

which completes the proof of Statement D.

Thus, it remains only to show Statement E., i.e., the transitionality of the newly constructed run r'. As already discussed several times before, it is sufficient to show that all the β-sets produced by the adjustment *adj* can be obtained in a regular manner from appropriately chosen α-sets allowed by the protocols, we only need to check rounds from 0.5 to $(t-1).5$ by induction, and we can check each node (j,m) separately relying on the partition of $\beta_\epsilon^m(r')$ into subsets $\beta_{\epsilon_j}^m(r')$'s.

Consider any $m \le t-1$. We have

$$r_j(m) = r'_j(m) \qquad \text{for all } (j,m) \in \mathit{WellConnected} \tag{2.50}$$

by Statement A. Hence, for $(j,m) \in \mathit{WellConnected}$ the set

$$\alpha_j^m(r) \in P_j\left(r_j(m)\right) = P_j\left(r'_j(m)\right)$$

can be chosen by the adversary to be $\alpha_j^m(r')$. The adversary can choose anything as $\alpha_j^m(r')$ for $(j,m) \notin \mathit{WellConnected}$ (recall that there is always at least one choice available by the no-apocalypse clause).

By the correctability of all agents

$$\mathit{filter}_\epsilon^{\le f}\left(r(m),\, \alpha_\epsilon^m(r),\, \alpha_1^m(r),\, \ldots,\, \alpha_n^m(r)\right) \quad \in \quad P_\epsilon(m)$$

because $\alpha_\epsilon^m(r) \in P_\epsilon(m)$. Further, by the delayability of all agents, the set

$$\mathit{filter}_\epsilon^{\le f}\left(r(m),\, \alpha_\epsilon^m(r),\, \alpha_1^m(r),\, \ldots,\, \alpha_n^m(r)\right) \setminus \bigsqcup_{(l,m) \in \mathit{SilentMasses} \vee (l,m) \notin V_\theta(r)} \mathit{GEvents}_l \quad \in \quad P_\epsilon(m).$$

Finally, by the gullibility of all agents,

$$\left(filter_{\epsilon}^{\leq f}\left(r(m),\, \alpha_{\epsilon}^{m}\left(r\right),\, \alpha_{1}^{m}\left(r\right),\, \ldots,\, \alpha_{n}^{m}\left(r\right)\right) \quad \setminus \bigsqcup_{(l,m)\in SilentMasses \sqcup FaultBuffer \vee (l,m)\notin V_{\theta}(r)} GEvents_{l} \right) \quad \sqcup$$

$$\{fail\,(l) \mid (l,m) \in FaultBuffer\} \quad \sqcup \quad \left\{ fake\,(l, gsend(l,j,\mu,id) \mapsto \text{🗓}) \,\middle|\, (l,m) \in FaultBuffer \quad \& \right.$$

$$\left. \left(gsend(l,j,\mu,id) \in \beta_{l}^{m}\left(r\right) \quad \vee \quad (\exists A \in \overline{GEvents_{l}} \sqcup \{\text{🗓}\})\, fake\,(l, gsend(l,j,\mu,id) \mapsto A) \in \beta_{b_l}^{m}\left(r\right)\right) \right\} \quad \in$$

$$P_{\epsilon}\left(m\right). \quad (2.51)$$

Thus, the adversary can choose $\alpha_{\epsilon}^{m}\left(r'\right)$ to be the set from (2.51). It remains to show that the filtering employed by the transition function $\tau_{P_{\epsilon},P}^{B_f}$ turns these sets $\alpha_{\epsilon}^{m}\left(r'\right)$, $\alpha_{1}^{m}\left(r'\right), \ldots, \alpha_{n}^{m}\left(r'\right)$ into the exact β-sets prescribed by the adjustment *adj*. Given that $\beta_{\epsilon}^{m}\left(r'\right)$ is partitioned into $\beta_{\epsilon_j}^{m}\left(r'\right)$ for each $j \in \mathcal{A}$, we can verify the correctness for each node (j,m) separately. Let us abbreviate

$$\Upsilon \quad := \quad filter_{\epsilon}^{B_f}\left(r'(m), \quad \alpha_{\epsilon}^{m}\left(r'\right), \quad \alpha_{1}^{m}\left(r'\right), \quad \ldots, \quad \alpha_{n}^{m}\left(r'\right)\right), \quad (2.52)$$

$$\Upsilon_{j} \quad := \quad \Upsilon \quad \cap \quad GEvents_{j} \quad (2.53)$$

$$\Xi_{j} \quad := \quad filter_{j}^{B}\left(\alpha_{1}^{m}\left(r'\right), \quad \ldots, \quad \alpha_{n}^{m}\left(r'\right), \quad \Upsilon\right) \quad (2.54)$$

To finish the step of induction, our goal is to show that for each $j \in \mathcal{A}$

$$\Upsilon_{j} = \beta_{\epsilon_j}^{m}\left(r'\right), \quad (2.55)$$

$$\Xi_{j} = \beta_{j}^{m}\left(r'\right). \quad (2.56)$$

- We start with nodes $(j,m) \notin V_{\theta}\left(r\right)$ and $(j,m) \in SilentMasses$.

  $$\Upsilon_{j} \overset{(2.53)}{=} \Upsilon \cap GEvents_{j} \overset{(2.52)}{\subset} \alpha_{\epsilon}^{m}\left(r'\right) \cap GEvents_{j} \overset{(2.51)}{=} \varnothing \overset{(2.12)}{=} \mathfrak{e} CFreeze\left(r\right) \overset{(2.47)}{=} \beta_{\epsilon_j}^{m}\left(r'\right)$$

  yiedling (2.55). In particular, $go(j) \notin \Upsilon$, resulting in

  $$\Xi_{j} \overset{(2.54)}{=} \varnothing \overset{(2.12)}{=} \mathfrak{a} CFreeze\left(r\right) \overset{(2.47)}{=} \beta_{j}^{m}\left(r'\right) \quad (2.57)$$

  yielding (2.55) and completing the case of $(j,m) \notin V_{\theta}\left(r\right)$ and $(j,m) \in SilentMasses$.

We now turn to the more complex case $(j,m) \in V_{\theta}\left(r\right) \setminus SilentMasses$, which requires some preparation. We abbreviate

$$Y_{\epsilon} := \bigsqcup_{(l,m)\notin V_{\theta}(r)} GEvents_{l} \quad \sqcup \quad \bigsqcup_{(l,m)\in SilentMasses \sqcup FaultBuffer} GEvents_{l}, \quad (2.58)$$

$$\overline{\alpha^{m}\left(r\right)} := \alpha_{1}^{m}\left(r\right), \ldots, \alpha_{n}^{m}\left(r\right), \quad (2.59)$$

$$\overline{\alpha^{m}\left(r'\right)} := \alpha_{1}^{m}\left(r'\right), \ldots, \alpha_{n}^{m}\left(r'\right). \quad (2.60)$$

In addition, we let

$$Z_{\epsilon} := \{fail\,(l) \mid (l,m) \in FaultBuffer\} \quad \sqcup$$

$$\left\{ fake\,(l, gsend(l,j,\mu,id) \mapsto \text{🗓}) \,\middle|\, (l,m) \in FaultBuffer \quad \& \right.$$

$$\left. \left(gsend(l,j,\mu,id) \in \beta_{l}^{m}\left(r\right) \quad \vee \quad (\exists A \in \overline{GEvents_{l}} \sqcup \{\text{🗓}\})\, fake\,(l, gsend(l,j,\mu,id) \mapsto A) \in \beta_{b_l}^{m}\left(r\right)\right) \right\}. \quad (2.61)$$

Note that

$$Z_\epsilon \overset{(2.61)}{\subset} \bigsqcup_{(l,m)\in FaultBuffer} BEvents_l \overset{(2.45)}{\subset} \bigsqcup_{(l,m+1)\in Failed(r,t)} BEvents_l. \tag{2.62}$$

Thus,

$$\begin{aligned} Failed\left(Z_\epsilon\right) &\overset{(2.62)}{\subset} \{l \mid (l,m+1)\in Failed\left(r,t\right)\} = \\ &\mathcal{A}\left(Failed\left(r(m)\right)\right) \cup \{l \mid \beta^m_{f_l}\left(r\right)\neq\varnothing\} \overset{(1.55)}{=} \\ &\mathcal{A}\left(Failed\left(r(m)\right)\right) \cup \\ &\left\{l \mid filter^{B_f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right) \cap \left(BEvents_l \sqcup \{sleep\left(l\right), hibernate\left(l\right)\}\right) \neq \varnothing\right\} \overset{(2.22),(2.23)}{=} \\ &\mathcal{A}\left(Failed\left(r(m)\right)\right) \cup Failed\left(filter^{B_f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right)\right) \end{aligned} \tag{2.63}$$

Further,

$$\begin{aligned} \Upsilon &\overset{(2.52)}{=} filter^{B_f}_\epsilon\left(r'(m),\quad \left(filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right) \setminus Y_\epsilon\right) \sqcup Z_\epsilon,\quad \overline{\alpha^m\left(r'\right)}\right) \overset{(1.119)}{=} \\ &filter^{B}_\epsilon\left(r'(m),\quad filter^{\leq f}_\epsilon\left(r'(m),\quad \left(filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right) \setminus Y_\epsilon\right) \sqcup Z_\epsilon,\quad \overline{\alpha^m\left(r'\right)}\right),\quad \overline{\alpha^m\left(r'\right)}\right). \end{aligned}$$

By the already proven Statement D., $Failed\left(r'(m)\right) \subset Failed\left(r(m)\right)$ (recall that $m \leq t-1$). Hence, by Cor. 2.4.1 with

$$X_\epsilon = \alpha^m_\epsilon\left(r\right), \qquad h = r(m), \qquad h' = r'(m), \qquad X_l = \alpha^m_l\left(r\right), \qquad X'_l = \alpha^m_l\left(r'\right),$$

given (2.63) and the fact that $\left|\,\mathcal{A}\left(Failed\left(r(m)\right)\right)\right| \leq f$ for the original run, we have

$$\Upsilon = filter^{B}_\epsilon\left(r'(m),\quad \left(filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right) \setminus Y_\epsilon\right) \sqcup Z_\epsilon,\quad \overline{\alpha^m\left(r'\right)}\right). \tag{2.64}$$

At the same time,

$$\begin{aligned} \beta^m_\epsilon\left(r\right) &\overset{(1.55)}{=} filter^{B_f}_\epsilon\left(r(m),\ \alpha^m_\epsilon\left(r\right),\ \overline{\alpha^m\left(r\right)}\right) \overset{(1.119)}{=} \\ &filter^{B}_\epsilon\left(r(m),\ filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right),\, \overline{\alpha^m\left(r\right)}\right),\ \overline{\alpha^m\left(r\right)}\right). \end{aligned} \tag{2.65}$$

Let us further abbreviate

$$\Omega := filter^{\leq f}_\epsilon\left(r(m),\, \alpha^m_\epsilon\left(r\right), \overline{\alpha^m\left(r\right)}\right), \tag{2.66}$$

$$\Omega' := \left(\Omega \setminus Y_\epsilon\right) \sqcup Z_\epsilon, \tag{2.67}$$

$$\Omega_j := \Omega \cap GEvents_j, \tag{2.68}$$

$$\Omega'_j := \Omega' \cap GEvents_j \tag{2.69}$$

making

$$\Upsilon = filter^{B}_\epsilon\left(r'(m),\quad \Omega',\quad \overline{\alpha^m\left(r'\right)}\right), \tag{2.70}$$

$$\beta^m_\epsilon\left(r\right) = filter^{B}_\epsilon\left(r(m),\quad \Omega,\quad \overline{\alpha^m\left(r\right)}\right). \tag{2.71}$$

As we know, $V_\theta\left(r\right) \setminus SilentMasses = WellConnected \sqcup FaultBuffer$. We consider these two cases separately

- For $(j,m) \in \mathit{WellConnected}$, by (2.58) we have $\mathit{GEvents}_j \cap Y_\epsilon = \varnothing$ and by (2.61) also $\mathit{GEvents}_j \cap Z_\epsilon = \varnothing$. In other words, by (2.67), for all $(j,m) \in \mathit{WellConnected}$

$$\Omega_j = \Omega'_j. \tag{2.72}$$

By (1.24) and (2.70) or (1.24) and (2.71) respectively, for all $(j,m) \in \mathit{WellConnected}$,

$$\mathit{active}(j,\Omega') \iff \mathit{active}(j,\Upsilon), \tag{2.73}$$
$$\mathit{active}(j,\Omega) \iff \mathit{active}(j,\beta^m_\epsilon(r)). \tag{2.74}$$

Consequently,

$$\mathit{active}(j,\Omega) \iff \mathit{active}(j,\Omega'). \tag{2.75}$$

Thus, given (2.72) and (1.18), for all $(j,m) \in \mathit{WellConnected}$,

$$\mathit{active}(j,\Upsilon) \iff \mathit{active}(j,\beta^m_\epsilon(r)). \tag{2.76}$$

Let us first prove (2.56) whenever $(j,m) \in \mathit{WellConnected}$.

- If $\mathit{passive}(j,\Upsilon)$, then $\mathit{passive}(j,\beta^m_\epsilon(r))$ by (2.76) and, hence,

$$\Xi_j \overset{(1.25),(2.54)}{=} \varnothing \overset{(1.25),(1.56)}{=} \beta^m_j(r)$$

- If $\mathit{active}(j,\Upsilon)$, then $\mathit{active}(j,\beta^m_\epsilon(r))$, and we have

$$\Xi_j \overset{(1.25),(2.54)}{=} \alpha^m_j(r') \overset{\text{def}}{=} \alpha^m_j(r) \overset{(1.25),(1.56)}{=} \beta^m_j(r).$$

In either case,

$$\Xi_j = \beta^m_j(r) \overset{(2.15)}{=} \mathfrak{a}(V_\theta(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j(r) \overset{(2.47)}{=} \beta^m_j(r'),$$

which completes the proof of (2.56) for $(j,m) \in \mathit{WellConnected}$.

We now show (2.55) whenever $(j,m) \in \mathit{WellConnected}$ by case analysis on the type of events U.

$$U \in \beta^m_{\epsilon_j}(r') \overset{(2.47)}{\iff} U \in \mathfrak{e}(V_\theta(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j(r) \tag{2.77}$$

- If U does not have the form $\mathit{grecv}(j,k,\mu,id)$, then

$$U \in \mathfrak{e}(V_\theta(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j(r) \overset{(2.15)}{\iff}$$
$$U \in \beta^m_{\epsilon_j}(r) \overset{(1.24),(2.71)}{\iff} U \in \Omega_j \overset{(2.72)}{\iff} U \in \Omega'_j \overset{(1.24),(2.70)}{\iff} U \in \Upsilon_j,$$

  which together with (2.77) guarantees the intended result for non-receive events.

- Let us now assume $U = \mathit{grecv}(j,k,\mu,id)$.
  - If $id \neq id(k,j,\mu,s,t')$ for some $s \in \mathbb{N}$ and $t' \leq m$, then

$$U \notin \beta^m_{\epsilon_j}(r) \quad \text{and} \quad U \notin \Upsilon_j \qquad \text{by (1.66)},$$

    (in case of Υ_j, the run upto timestamp $m+1$ can be considered transitional by IH (upto m) and by the assumed transitionality for the last round that produced Υ_j). It follows from the former statement by (2.15) that

$$U \notin \mathfrak{e}(V_\theta(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j(r),$$

    which together with (2.77) guarantees the intended result for receives with incorrect id.

$*$ Let us now assume that $id = id(k, j, \mu, s, t')$ and $t' \leq m$ (recall that s and t' are unique in this case) and denote

$$V := gsend(k, j, \mu, id) \qquad \text{and} \qquad W_A := fake\,(k, gsend(k, j, \mu, id) \mapsto A).$$

For the newly constructed run, we have

$$\begin{aligned} &U \in \mathfrak{e}(V_\theta\,(r) \setminus \mathit{SilentMasses})\text{-}\mathit{Focus}^m_j\,(r) \text{ and } t' \leq m \quad \overset{(2.15)}{\Longleftrightarrow} \\ &\qquad U \in \beta^m_{\epsilon_j}\,(r) \text{ and } (k, t') \in V_\theta\,(r) \setminus \mathit{SilentMasses} \text{ and } t' \leq m \quad \overset{(2.46)}{\Longleftrightarrow} \\ &\qquad U \in \beta^m_{\epsilon_j}\,(r) \text{ and } (k, t') \in \mathit{WellConnected} \sqcup \mathit{FaultBuffer} \text{ and } t' \leq m. \end{aligned} \tag{2.78}$$

Now we break the situation into further four subcases depending on whether $t' < m$ or $t' = m$ and separating (k, t') nodes in *WellConnected* from *FaultBuffer*.

· If $(k, t') \in \mathit{WellConnected}$ and $t' < m$, then

$$\begin{aligned} &U \in \beta^m_{\epsilon_j}\,(r) \text{ and } (k, t') \in \mathit{WellConnected} \text{ and } t' < m \quad \overset{(1.24),(2.71)}{\Longleftrightarrow} \\ &\qquad U \in \Omega_j \text{ and } (k, t') \in \mathit{WellConnected} \text{ and } t' < m \text{ and} \\ &\qquad \Big(V \in \beta^{t'}_k\,(r) \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}\,(r)\Big) \quad \overset{(2.15),(2.47),(2.72)}{\Longleftrightarrow} \\ &\qquad U \in \Omega'_j \text{ and } (k, t') \in \mathit{WellConnected} \text{ and } t' < m \text{ and} \\ &\qquad\qquad \Big(V \in \beta^{t'}_k\,(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}\,(r')\Big) \end{aligned} \tag{2.79}$$

· If $(k, t') \in \mathit{WellConnected}$ and $t' = m$, then

$$\begin{aligned} &U \in \beta^m_{\epsilon_j}\,(r) \text{ and } (k, t') \in \mathit{WellConnected} \text{ and } t' = m \quad \overset{(1.24),(2.71)}{\Longleftrightarrow} \\ &\qquad U \in \Omega_j \text{ and } (k, m) \in \mathit{WellConnected} \text{ and } t' = m \text{ and} \\ &\qquad ((V \in \alpha^m_k\,(r) \& \mathit{active}(k, \Omega)) \text{ or } (\exists A) W_A \in \Omega_k) \quad \overset{(2.15),(2.47),(2.72),(2.75)}{\Longleftrightarrow} \\ &\qquad U \in \Omega'_j \text{ and } (k, m) \in \mathit{WellConnected} \text{ and } t' = m \text{ and} \\ &\qquad\qquad ((V \in \alpha^m_k\,(r') \& \mathit{active}(k, \Omega')) \text{ or } (\exists A) W_A \in \Omega'_k) \end{aligned} \tag{2.80}$$

· If $(k, t') \in \mathit{FaultBuffer}$, and $t' < m$, then

$$\begin{aligned} &U \in \beta^m_{\epsilon_j}\,(r) \text{ and } (k, t') \in \mathit{FaultBuffer} \text{ and } t' < m \quad \overset{(1.24),(2.71)}{\Longleftrightarrow} \\ &\qquad U \in \Omega_j \text{ and } (k, t') \in \mathit{FaultBuffer} \text{ and } t' < m \text{ and} \\ &\qquad \Big(V \in \beta^{t'}_k\,(r) \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}\,(r)\Big) \quad \overset{(2.16),(2.47),(2.72)}{\Longleftrightarrow} \\ &U \in \Omega'_j \text{ and } (k, t') \in \mathit{FaultBuffer} \text{ and } t' < m \text{ and } W_{\text{🗓}} \in \beta^{t'}_{b_k}\,(r') \quad \overset{(2.16),(2.47)}{\Longleftrightarrow} \\ &\qquad U \in \Omega'_j \text{ and } (k, t') \in \mathit{FaultBuffer} \text{ and } t' < m \text{ and} \\ &\qquad\qquad \Big(V \in \beta^{t'}_k\,(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}\,(r')\Big) \end{aligned} \tag{2.81}$$

· If $(k,t') \in \mathit{FaultBuffer}$ and $t' = m$, then

$$U \in \beta^m_{\epsilon_j}(r) \text{ and } (k,t') \in \mathit{FaultBuffer} \text{ and } t' = m \overset{(1.24),(2.71)}{\iff}$$
$$U \in \Omega_j \text{ and } (k,m) \in \mathit{FaultBuffer} \text{ and } t' = m \text{ and}$$
$$((V \in \alpha^m_k(r) \& \mathit{active}(k,\Omega)) \text{ or } (\exists A) W_A \in \Omega_k) \overset{(1.24),(1.25),(2.71)}{\iff}$$
$$U \in \Omega_j \text{ and } (k,m) \in \mathit{FaultBuffer} \text{ and } t' = m \text{ and}$$
$$\left(V \in \beta^m_k(r) \text{ or } (\exists A) W_A \in \beta^m_{b_k}(r)\right) \overset{(2.58),(2.61),(2.67)}{\iff}$$
$$U \in \Omega'_j \text{ and } (k,m) \in \mathit{FaultBuffer} \text{ and } t' = m \text{ and } W_{\text{🗓}} \in \Omega'_k \overset{(2.58),(2.61),(2.67)}{\iff}$$
$$U \in \Omega'_j \text{ and } (k,m) \in \mathit{FaultBuffer} \text{ and } t' = m \text{ and}$$
$$((V \in \alpha^m_k(r') \& \mathit{active}(k,\Omega')) \text{ or } (\exists A) W_A \in \Omega'_k) \quad (2.82)$$

Combining the cases (2.79) and (2.81) when $t' < m$ we can see that

$$U \in \Omega'_j \text{ and } (k,t') \in \mathit{WellConnected} \sqcup \mathit{FaultBuffer} \text{ and } t' < m \text{ and}$$
$$\left(V \in \beta^{t'}_k(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}(r')\right) \overset{(2.46)}{\iff}$$
$$U \in \Omega'_j \text{ and } (k,t') \in V_\theta(r) \setminus \mathit{SilentMasses} \text{ and } t' < m \text{ and}$$
$$\left(V \in \beta^{t'}_k(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}(r')\right) \overset{(2.12)(2.47)}{\iff}$$
$$U \in \Omega'_j \text{ and } t' < m \text{ and } \left(V \in \beta^{t'}_k(r') \text{ or } (\exists A) W_A \in \beta^{t'}_{b_k}(r')\right) \overset{(1.24),(2.70)}{\iff}$$
$$U \in \Upsilon_j \text{ and } t' < m. \quad (2.83)$$

Similarly, combining the cases (2.80) and (2.82) when $t' = m$,

$$U \in \Omega'_j \text{ and } (k,t') \in \mathit{WellConnected} \sqcup \mathit{FaultBuffer} \text{ and } t' = m \text{ and}$$
$$((V \in \alpha^m_k(r') \& \mathit{active}(k,\Omega')) \text{ or } (\exists A) W_A \in \Omega'_k) \overset{(2.46)}{\iff}$$
$$U \in \Omega'_j \text{ and } (k,t') \in V_\theta(r) \setminus \mathit{SilentMasses} \text{ and } t' = m \text{ and}$$
$$((V \in \alpha^m_k(r') \& \mathit{active}(k,\Omega')) \text{ or } (\exists A) W_A \in \Omega'_k) \overset{(2.58),(2.61),(2.67)}{\iff}$$
$$U \in \Omega'_j \text{ and } t' = m \text{ and } ((V \in \alpha^m_k(r') \& \mathit{active}(k,\Omega')) \text{ or } (\exists A) W_A \in \Omega'_k) \overset{(1.24),(2.70)}{\iff}$$
$$U \in \Upsilon_j \text{ and } t' = m. \quad (2.84)$$

We can now continue the equivalences in (2.78) for $t' \leq m$ as follows:

$$U \in \beta^m_{\epsilon_j}(r) \text{ and } (k,t') \in \mathit{WellConnected} \sqcup \mathit{FaultBuffer} \text{ and } t' \leq m \overset{(2.79)\text{–}(2.84)}{\iff}$$
$$U \in \Upsilon_j \text{ and } t' \leq m \quad (2.85)$$

This together with (2.77) guarantees the intended result for receives with correct id with $t' \leq m$.

Thus, we have proved (2.55) for the case $(j,m) \in \mathit{WellConnected}$.

- The final case to consider is $(j,m) \in \mathit{FaultBuffer}$. Note that, in fact,

$$\Omega'_j \overset{(2.69)}{=} \Omega' \cap \mathit{GEvents}_j \overset{(2.67)}{=} ((\Omega \setminus Y_\epsilon) \sqcup Z_\epsilon) \cap \mathit{GEvents}_j = ((\Omega \setminus Y_\epsilon) \cap \mathit{GEvents}_j) \sqcup (Z_\epsilon \cap \mathit{GEvents}_j) \overset{(2.58)}{=}$$
$$\varnothing \sqcup (Z_\epsilon \cap \mathit{GEvents}_j) = Z_\epsilon \cap \mathit{GEvents}_j. \quad (2.86)$$

meaning that by (2.62)

$$\Omega'_j \subset \mathit{BEvents} \cap \mathit{GEvents}_j = \mathit{BEvents}_j. \tag{2.87}$$

In particular,

$$\mathit{go}(j) \notin \Upsilon_j \subset \Omega'_j$$

by (1.54) and (2.70), i.e.,

$$\mathit{passive}(j, \Upsilon)$$

ensuring that

$$\Xi_j \overset{(1.25),(2.54)}{=} \varnothing \overset{(2.16)}{=} \mathfrak{a}\mathit{FakeEcho}_j^m\,(r) \overset{(2.47)}{=} \beta_j^m\,(r'), \tag{2.88}$$

which completes the proof of (2.56) for $(j, m) \in \mathit{FaultBuffer}$.

Finally, the only thing outstanding is (2.55) for $(j, m) \in \mathit{FaultBuffer}$.

$$\begin{aligned} U \in \Upsilon_j \quad &\overset{(1.24),(2.52),(2.87)}{\Longleftrightarrow} U \in \Omega'_j \quad \overset{(2.86)}{\Longleftrightarrow} \\ &U \in Z_\epsilon \cap \mathit{GEvents}_j \quad \overset{(2.16),(2.61)}{\Longleftrightarrow} U \in \mathfrak{e}\mathit{FakeEcho}_j^m\,(r) \overset{(2.47)}{=} \beta_{\epsilon_j}^m\,(r'). \end{aligned} \tag{2.89}$$

This completes the case of $(j, m) \in \mathit{FaultBuffer}$.

Thus, (2.55) and (2.56) hold for all (j, m). This completes the proof of Statement E. by induction and, with it, the proof of the whole lemma. □

Corollary 2.4.4. *Lemma 2.4.2 implies that*

$$\big(\forall \beta \in \mathit{WellConnected}\big)\,(\forall o \in \overline{\mathit{Actions}} \sqcup \overline{\mathit{Events}}) \quad (I, r, t) \models \overline{\mathit{occurred}}_\beta(o) \Leftrightarrow (I, r', t) \models \overline{\mathit{occurred}}_\beta(o).$$

Proof. The reasoning from right to left is trivial because the new run r' does not introduce any new *correct* actions or events.

From left to right, note that if β is *WellConnected*, then so is its immediate past and both are correct nodes. This immediately guarantees that all correct actions and events other than correct receives are preserved in the new run r'. If β correctly receives a message in r, by (2.48), this message originates from $V_\theta\,(r) \setminus \mathit{SilentMasses}$, and, thus, the receive is also preserved in r' □

Corollary 2.4.5. *Lemma 2.4.2 also implies that for any node $\eta \in \mathcal{A} \times \mathbb{N}$ and any path ξ,*

$$\xi \in \overline{\Xi_\theta^{\{\eta\}}}\,(r) \qquad \Longleftrightarrow \qquad \xi \in \overline{\Xi_\theta^{\{\eta\}}}\,(r')$$

Proof. First, note that causal links along correct paths do not contain Byzantine sends. Thus, all send actions and receive events forming such paths are correct.

From left to right, if ξ is a correct path connecting η to θ in r, then all the nodes along ξ are *WellConnected*, as are their immediate pasts. Thus all correct sends from these nodes are preserved in the new run r'. All receives forming the path are also preserved by the preceding corollary. Therefore, the whole causal path ξ is preserved in r'.

From right to left, the argument is even simpler since all correct sends and receives in r' are present in the original run r. □

The Byzantine causal cone lemma shows how to modify a given run without changing the local state of agent i and without introducing any new Byzantine nodes (see Lemma 2.4.2(C.)). More precisely, any Byzantine behavior of an agent in the new run happens only when the agent became Byzantine in the original run. From the point of view agent i at t, this still leaves the possibility that additional agents are faulty, which affects i's level of certainty about the run.

Before exploring this in more detail, we formulate the following auxiliary lemma

Lemma 2.4.6. *For $f \in \mathbb{N}$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that all agents are correctable, for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, one can assume w.l.o.g. that this run was produced with all applications of the Byzantine threshold filter $filter_\epsilon^{\leq f}$ being benign, i.e., with this filter never removing any (Byzantine) events. In other words, in case of correctable agents, one can, without loss of generality assume that the environment filter is the causal filter $filter_\epsilon^B$.*

Proof. In order to achieve this, it is sufficient, for each round to choose a different set of events in each round, namely, the result of applying $filter_\epsilon^{\leq f}$ to the originally chosen set of events. For rounds where $filter_\epsilon^{\leq f}$ was benign and affected no changes, the same set of events is chosen. For all remaining rounds where $filter_\epsilon^{\leq f}$ removed all Byzantine events, the alternative choice is possible due to all agents being correctable. It is trivial to see that such choices produce the same run. □

Remark 2.4.7. The preceding lemma does not mean that the Byzantine threshold filter is redundant. Whenever a run is constructed by specifying α-sets, $filter_\epsilon^{\leq f}$ must be applied to make sure the number of Byzantine agents does not exceed f. However, in case of correctable agents, any run that that is assumed to satisfy the Byzantine threshold f can be obtained without using this filter by a clever choice of α-sets to match the given β-sets.

Lemma 2.4.8. *For $f \in \mathbb{N}$, for a non-excluding agent-context $\chi = \big((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\big)$ such that all agents are fallible and correctable for a χ-based interpreted system $I = (R^\chi, \pi)$, for a run $r \in R^\chi$, for a timestamp $t \in \mathbb{N}$, and for*

$$B \subset \mathcal{A} \setminus \mathcal{A}\left(Failed\left(r,t\right)\right) \qquad \textit{such that} \qquad \left|B \sqcup \mathcal{A}\left(Failed\left(r,t\right)\right)\right| \leq f,$$

there exists a run $r' \in R^\chi$ such that

- $\beta_j^m\left(r'\right) = \beta_j^m\left(r\right)$ *for all* $(j,m) \in \mathcal{A} \times [\![0; t-1]\!]$;
- $\beta_{\epsilon_j}^m\left(r'\right) = \beta_{\epsilon_j}^m\left(r'\right) \sqcup \{fail\left(j\right)\}$ *for all* $(j,m) \in B \times [\![0; t-1]\!]$;
- $\beta_{\epsilon_j}^m\left(r'\right) = \beta_{\epsilon_j}^m\left(r'\right)$ *for all* $(j,m) \in (\mathcal{A} \setminus B) \times [\![0; t-1]\!]$.

In particular, for this run r', we have

$$\begin{aligned} r'_j(m) = r_j(m) \qquad & \textit{for all } (j,m) \in \mathcal{A} \times [\![0;t]\!], \\ Bad\left(r',t\right) = Bad\left(r,t\right) \cup (B \times [\![1;t]\!]), & \\ Failed\left(r',t\right) = Failed\left(r,t\right) \cup (B \times [\![1;t]\!]), & \end{aligned}$$

Proof. By Lemma 2.4.6 for our correctable agents, we assume w.l.o.g. that $filter_\epsilon^{\leq f}$ was always benign. Thus, it is sufficient to add $fail\left(j\right)$ for agents in B in all rounds. After more complex transformations have been described in full detail, it should be clear that the only thing potentially affected by these additions is the Byzantine threshold filter $filter_\epsilon^{\leq f}$. However, we know that it was always benign in the original run and we know that adding all agents from B to those agents *made* Byzantine in the original run does not violate the Byzantine threshold f. Thus, $filter_\epsilon^{\leq f}$ remains benign in the modified run too. □

Given the minimal changes to the run introduced in this lemma, the following corollary is almost immediate:

Corollary 2.4.9. *For the run r' constructed in the preceding lemma from a given run r, for any node $\eta \in \mathcal{A} \times \mathbb{N}$ and any path ξ,*

$$\eta \rightsquigarrow_\xi^{r,t} \theta \qquad \Longleftrightarrow \qquad \eta \rightsquigarrow_\xi^{r',t} \theta.$$

Moreover, such a path ξ is a correct path in r' iff it is correct in r and $\mathcal{A}\left(\xi\right) \cap B = \varnothing$.

Lemma 2.4.10. *For a node $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, agent $j \in \mathcal{A}$, timestamp $t' \geq t > 0$, for a non-excluding agent-context $\chi = \left((P_\epsilon, \mathscr{G}(0), \tau^{B_f}, \Psi), P\right)$ such that P_ϵ makes all agents gullible, correctable, and delayable, for a χ-based interpreted system $I = (R^\chi, \pi)$, and for a run $r \in R^\chi$, and the local action or event $o \in \overline{Actions} \sqcup \overline{Events}$, if*

$$(I, r, t') \models correct_j \wedge K_j(correct_j \rightarrow \overline{occurred}_{(i,t)}(o))$$

then, for any

$$B \subset \mathcal{A} \setminus \mathcal{A}\left(Failed_{V^{\ni}_{(j,t')}(r)}(r)\right) \qquad \text{such that} \qquad |B \sqcup \mathcal{A}\left(Failed_{V^{\ni}_{(j,t')}(r)}(r)\right)| \leq f, \tag{2.90}$$

there exists a correct path $\xi \in \overline{\Xi^{\{(i,t-1)\}}_{(j,t')}}(r)$ from $(i,t-1)$ to (j,t') such that $\mathcal{A}(\xi) \cap B = \varnothing$.

Proof. Let

$$(I, r, t') \models correct_j \wedge K_j(correct_j \rightarrow \overline{occurred}_{(i,t)}(o))$$

By Lemma 2.3.3 for run r and node (j,t'), there is an alternative run $r' \in R^\chi$ such that $\beta^m_{\epsilon_a}(r') = \beta^m_a(r') = \varnothing$ for all $(a,m) \notin V_{(j,t')}(r)$ with $m < t'$, and $r'_j(t') = r_j(t')$, and

$$Failed(r', t') = Failed_{V^{\ni}_{(j,t')}(r)}(r). \tag{2.91}$$

Consider any set B satisfying (2.90). It follows from (2.91) that

$$B \subset \mathcal{A} \setminus \mathcal{A}(Failed(r', t')) \qquad \text{and} \qquad |B \sqcup \mathcal{A}(Failed(r', t'))| \leq f.$$

Thus, by Lemma 2.4.8 for run r', timestamp t', and set B, there exists a run $r'' \in R^\chi$ such that

$$r''_j(t') = r'_j(t') = r_j(t')$$

and

$$Failed(r'', t') = Failed(r', t') \cup (B \times [\![1; t']\!]) = Failed_{V^{\ni}_{(j,t')}(r)}(r) \cup (B \times [\![1; t']\!]).$$

Since

$$(I, r, t') \models correct_j \iff (j,t') \notin Failed(r, t') \iff (j,t') \notin Failed_{V^{\ni}_{(j,t')}(r)}(r) \iff$$
$$(j,t') \notin Failed(r', t') \iff (j,t') \notin Failed(r', t') \& j \notin B \iff (j,t') \notin Failed(r'', t'),$$

we can apply Lemma 2.4.2 for run r'' and node (j,t') to obtain $r''' \in R^\chi$ such that

$$r'''_j(t') = r''_j(t') = r_j(t'),$$

and

$$(j,t') \notin Failed(r''', t'),$$

and

$$\beta^m_a(r''') = \overline{\beta}^m_{\epsilon_a}(r''') = \varnothing \tag{2.92}$$

for all $(a,m) \notin \overline{WellConnected''}$ with $m < t'$ where $\overline{WellConnected''}$ is determined w.r.t. node (j,t') in run r''.

From $r'''_j(t') = r_j(t')$ and $(I, r, t') \models K_j(correct_j \rightarrow \overline{occurred}_{(i,t)}(o))$, it follows that

$$(I, r''', t') \models correct_j \rightarrow \overline{occurred}_{(i,t)}(o)$$

Further, $(j,t') \notin Failed(r''', t')$ means that $(I, r''', t') \models correct_j$, therefore,

$$(I, r', t') \models \overline{occurred}_{(i,t)}(o).$$

Therefore, at least one of $\overline{\beta}^{t-1}_{\epsilon_i}(r')$ and $\beta^{t-1}_i(r')$ must not be empty. Given that $t - 1 < t'$, this means that $(i, t-1) \in \overline{WellConnected''}$, i.e., there exists a correct path $\xi \in \overline{\Xi^{\{(i,t-1)\}}_{(j,t')}}(r'')$. By Corollary 2.4.9, $\xi \in \overline{\Xi^{\{(i,t-1)\}}_{(j,t')}}(r')$ and $\mathcal{A}(\xi) \cap B = \varnothing$. Finally, by Corollary 2.3.5 and Statement 6 of Lemma 2.3.3, $\xi \in \overline{\Xi^{\{(i,t-1)\}}_{(j,t')}}(r)$. □

Chapter 3

Conclusion

In this first installment of our report, we introduced a formal framework to study epistemic states of agents that may be Byzantine. We considered the resulting causality conditions for asynchronous agents and developed a Byzantine analog of the causal cone. The results provide a nice parallel to the well-known results for correct agents, with correct causal paths in the Byzantine case playing the role of causal paths in the traditional case.

In the next installment of this report, we plan to show how the framework can be extended to distributed scenarios other than asynchronous systems, including but not limited to synchronous agents, synchronous communication, broadcast, multicast, coordinated agents, etc.

Acknowledgements The authors would like to thank Yoram Moses for many fruitful discussions.

Bibliography

[AW04] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, Second edition, 2004.

[BL87] James E. Burns and Nancy A. Lynch. The Byzantine Firing Squad problem. In Franco P. Preparata, editor, *Parallel and Distributed Computing*, volume 4 of *Advances in Computing Research: A research annual*, pages 147–161. JAI Press, 1987. URL: `https://apps.dtic.mil/docs/citations/ADA154770`.

[BZM10] Ido Ben-Zvi and Yoram Moses. Beyond Lamport's *happened-before*: On the role of time bounds in synchronous systems. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13–15, 2010, Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2010. `doi:10.1007/978-3-642-15763-9_42`.

[BZM13] Ido Ben-Zvi and Yoram Moses. Agent-time epistemics and coordination. In Kamal Lodaya, editor, *Logic and Its Applications, 5th Indian Conference, ICLA 2013, Chennai, India, January 10–12, 2013, Proceedings*, volume 7750 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2013. `doi:10.1007/978-3-642-36039-8_9`.

[BZM14] Ido Ben-Zvi and Yoram Moses. Beyond Lamport's *happened-before*: On time bounds and the ordering of events in distributed systems. *Journal of the ACM*, 61(2:13), April 2014. `doi:10.1145/2542181`.

[CGM14] Armando Castañeda, Yannai A. Gonczarowski, and Yoram Moses. Unbeatable consensus. In Fabian Kuhn, editor, *Distributed Computing, 28th International Symposium, DISC 2014, Austin, TX, USA, October 12–15, 2014, Proceedings*, volume 8784 of *Lecture Notes in Computer Science*, pages 91–106. Springer, 2014. `doi:10.1007/978-3-662-45174-8_7`.

[DFP+14] Danny Dolev, Matthias Függer, Markus Posch, Ulrich Schmid, Andreas Steininger, and Christoph Lenzen. Rigorously modeling self-stabilizing fault-tolerant circuits: An ultra-robust clocking scheme for systems-on-chip. *Journal of Computer and System Sciences*, 80(4):860–900, June 2014. `doi:10.1016/j.jcss.2014.01.001`.

[Die17] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Fifth edition, 2017. `doi:10.1007/978-3-662-53622-3`.

[DM90] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a Byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, October 1990. `doi:10.1016/0890-5401(90)90014-9`.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[FHMV99] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Common knowledge revisited. *Annals of Pure and Applied Logic*, 96(1–3):89–105, March 1999. doi:10.1016/S0168-0072(98)00033-5.

[Fim18] Patrik Fimml. Knowledge in distributed systems with byzantine failures. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, 2018.

[FS12] Matthias Függer and Ulrich Schmid. Reconciling fault-tolerant distributed computing and systems-on-chip. *Distributed Computing*, 24(6):323–355, January 2012. doi:10.1007/s00446-011-0151-7.

[GM13] Yannai A. Gonczarowski and Yoram Moses. Timely common knowledge: Characterising asymmetric distributed coordination via vectorial fixed points. In Burkhard C. Schipper, editor, *Proceedings of TARK XIV (2013)*, pages 79–93, Chennai, India, January 7–9, 2013. URL: http://www.tark.org/proceedings/tark_jan7_13/p79-gonczarowski.pdf.

[GM18] Guy Goren and Yoram Moses. Silence. In *PODC '18, Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 285–294, Egham, United Kingdom, July 23–27, 2018. ACM. doi:10.1145/3212734.3212768.

[Hin62] Jaakko Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962.

[HM90] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, July 1990. doi:10.1145/79147.79161.

[HMW01] Joseph Y. Halpern, Yoram Moses, and Orli Waarts. A characterization of eventual Byzantine agreement. *SIAM Journal on Computing*, 31(3):838–865, 2001. doi:10.1137/S0097539798340217.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978. doi:10.1145/359545.359563.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982. doi:10.1145/357172.357176.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1996.

[Mic89] Ruben Michel. A categorical approach to distributed systems, expressibility and knowledge. In Piotr Rudnicki, editor, *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 129–143. ACM, 1989. doi:10.1145/72981.72990.

[MT86] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge: Preliminary version. In *27th Annual Symposium on Foundations of Computer Science*, pages 208–221, Toronto, ON, Canada, 27–29 October, 1986. IEEE. doi:10.1109/SFCS.1986.46.

[MT88] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3(1–4):121–169, November 1988. doi:10.1007/BF01762112.

[PS18] Daniel Pfleger and Ulrich Schmid. On knowledge and communication complexity in distributed systems. In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity, 25th International Colloquium,*

SIROCCO 2018, Ma'ale HaHamisha, Israel, June 18–21, 2018, Revised Selected Papers, volume 11085 of *Lecture Notes in Computer Science*, pages 312–330. Springer, 2018. doi:10.1007/978-3-030-01325-7_27.

[RS11] Peter Robinson and Ulrich Schmid. The Asynchronous Bounded-Cycle model. *Theoretical Computer Science*, 412(40):5580–5601, September 2011. doi:10.1016/j.tcs.2010.08.001.

[ST87a] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, July 1987. doi:10.1145/28869.28876.

[ST87b] T. K. Srikanth and Sam Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987. doi:10.1007/BF01667080.

[WS09] Josef Widder and Ulrich Schmid. The Theta-Model: achieving synchrony without clocks. *Distributed Computing*, 22(1):29–47, April 2009. doi:10.1007/s00446-009-0080-x.