Knowledge in Byzantine Message-Passing System

From February to July at Embedded Computing Systems Group, TU Wien
under the supervision of Ulrich Schmid

Laurent Prosperi
M1 student in Computer Science
at ENS Cachan in 2016–2017

September 3, 2017

Le contexte général

Ce travail s'inscrit dans la modélisation des systèmes distribués[AW04][Lyn96] - constitués d'une multitude d'agents (ex : ordinateur, processus ...) - de plus en plus présents dans notre quotidien (ex: la gestion d'un temps précis et cohérent sur le réseau, remarquons qu'une telle fonctionnalité est vitale). L'une des approches utilisées est l'étude du savoir des agents (ex : un réseau de capteurs détecte un orage) à travers la *logique épistemique* ; l'ouvrage de référence est [FHMV04]. Les deux principales applications sont : la tolérance aux pannes des protocoles distribués - un protocole est la spécification du comportement des agents - et dans une moindre mesure la *Knowledge-Based programming* ou programmation par savoir (ex : si le réseau de capteurs connaît l'existence - détecte - un orage alors il déclenche une alerte).

À cela, on a cherché à incorporer la notion de *panne Byzantine*, un agent Byzantin a un comportement arbitraire pouvant dévier de ses spécifications. La première pierre fut posée par Patrik Fimml[Fim17], l'étudiant m'ayant précédé.

[MT86] et [HMW01] traitent de la relation entre savoir et *Byzantine agreement*[LSP82] dans le cadre des *crashs* (un agent cesse de fonctionner pour l'éternité) [MT86] et des *omission failures* (un agent cesse momentanément de fonctionner). De plus, [Mic89] introduit un modèle peu extensible de *pannes Byzantines* assez faibles (les actions d'un agent Byzantin sont limitées par son état) et la notion de savoir n'est définie que pour les agents corrects ce qui empêche notamment l'introspection.

Le problème étudié

La majeure partie de mon travail porte sur la mise au point du premier modèle intégrant pleinement la toute puissance des pannes *Byzantines* à la notion de savoir dans un système distribué par passage de messages. Ce modèle permettra par la suite d'explorer, sur une base stable, les relations entre savoir et tolérance aux pannes.

Par ailleurs, j'ai établi une condition nécessaire et suffisante liant la structure des communications(ie un graphe) à la connaissance d'un fait. Un telle condition fut mise en exergue dans [BZM14] sans pannes d'aucune sorte. Ceci facilite l'étude de la tolérance aux pannes.

La contribution proposée

Je propose, dans un premier temps, un modèle modulaire, extensible et assez facile d'utilisation permettant d'exprimer le savoir d'un agent en présence de pannes (principalement Byzantines). Ce

modèle s'inscrit dans la continuation de [FHMV04]. De plus, il est suffisament stable et extensible pour pouvoir y incorporer bon nombre de modèles classiques en informatique distribuée. Enfin, il permet notamment une certaine forme d'introspection : un agent peut-il savoir, dans certains cas, s'il a (eu) un comportement correct.

Ce modèle a émergé de mes travaux sur la relation entre la structure des communication et savoir d'un fait. Dans un premier temps, j'ai travaillé directement avec le modéle de [Fim17] mais je ne pouvais pas décrire précisément les interactions entre agent et savoir. De fait, par itérations successives, une première version traitant du cas d'*agents ascynchrones* est apparue.

Par la suite, avec les commentaires et la précieuse relecture de Roman Kuznets, je l'ai généralisé et étendu.

Les arguments en faveur de sa validité

Le modèle semble intéressant car il est organisé en sous parties indépendantes, fonctionnant en boite noire et représentant des concepts différents. De plus, ce modèle se place dans la continuité de [FHMV04]. Notons qu'on l'a naturellement étendu à l'ensemble des modèles classiques que nous connaissions. On peut aussi incorporer naturellement, si nécéssaire, la notion d'oracle. Enfin, le pouvoir de nuisance donné aux agents *Byzantins* permet une execution arbitraire et un savoir arbitraire, on peut facilement se restreindre à des pannes plus bénignes.

La condition liant savoir et communications : on a retrouvé les conditions de tolérance aux pannes, en terme de proportion d'agents Byzantins, de certains algorithmes de synchronisation d'horloges.

Le bilan et les perspectives

En conclusion, j'ai mis au point un modèle générique, modulable permettant d'exprimer finement le savoir d'un agent, correct ou non, en présence de tous types de pannes (principalement *Byzantine*). Par ailleurs, j'ai réussi à exhiber une condition nécessaire et suffisante, dans le cas d'*agents asynchrones*, liant savoir et structure des communications.
Les travaux en cours sont dans un premier temps le nettoyage et la stabilisation du modèle. Ensuite, la réécriture des lemmes, preuves et thèorèmes dans la nouvelle version du modèle. Une publication dans un journal(probablement JCAM) est en préparation.

Plusieurs développements sont à prévoir, dans un premier temps établir une hierarchie entre les différentes extensions de notre framework afin de pouvoir transposer les propriétés, lemmes, théorèmes naturellement. Ensuite, on pourra faire cohéxister différentes sévéritées de pannes (ex : crash + Byzantine). Et enfin établir une hiérarchie de protocoles, en fonction du savoir qu'ils permettent d'atteindre.

Acknowledgement

This work evolved from the previous report of Patrik Fimml[Fim17] and it has been lead under the supervision of Ulrich Schmid. I would particularly like to thank Roman Kuznets for its comments and its precious reviewing and more generally the Embedded Computing Systems Group for hosting me.

Introduction

Fault-tolerance is one of the core stone in distributed computing, indeed in this field agents (processors, computers, ...) can be faulty due to external events (power loss, fire, ...), we will model

this with benign fault (ex : *crash*, *omission failure*). Or due to an arbitrary behaviour, for instance an agent can be able to do each time the worst actions for the integrity of the system, we will denote this arbitrary behaviour as *Byzantine failure* [LSP82]. In order to enhance fault-tolerance, we need to model finely the interactions between the environment, the agents and their behaviours. One of the way, to achieve this goal, is to use the *knowledge* of agents (knowledge of the other agents' actions, of the events of the environment or of the past), the reference book is [FHMV04]. There are some attempts to incorporate failure into the knowledge framework, for benign faults it has been done by [MT86] and [HMW01]. Moreover, there is a previous attempt to incorporate *Byzantine failures* in [Mic89], in this paper they have design *weak Byzantine faults* : Byzantine agents must behave according to their states and they can not restrict arbitrarily their knowledge of world(however they can extend it arbitrarily). Moreover, they can not express the knowledge of a faulty agent however in the general case every agent can behave arbitrary at any time (this can model some computers compromised by an attack in a network).

In the first part of this report, we present an extension of the epistemic runs and systems framework for modeling distributed system introduced in [FHMV04] to also incorporate Byzantine agents. Our framework is strictly modular : each pieces of the framework have a specific goal without overlapping. This modularity allows to cover all existing distributed computing models we are aware of (like lock-step synchronous system, all variants of partially synchronous systems, asynchronous system with or without oracles like failure detectors and even timed systems. Moreover, Byzantine agents may behave arbitrarily(and may or may not be aware of doing so), our framework allows epistemic reasoning also for faulty agents (they may have arbitrary knowledge).

In the second part, we show the utility of our framework by applying it for identifying necessary and sufficient communication structure for certain distributed computing problems in asynchronous systems with Byzantine faulty agents. We extend pivotal concepts introduced in [BZM14] for the fault-free case to Byzantine setting.

Nota bene : Some inconsistencies remains between the different parts of this report (and probably inside them). Indeed, both of them are not expressed in the same version of our framework. We are at the end of the cleaning of the general framework and soon we will be able to rewrite the second part.

Contents

Part I
The Byzantine Message-Passing Model

1 Agents and states

We consider distributed multi-agent systems with agents viewed as processes or humans.

Definition 1.1 (Agents)**.** We consider a non-empty finite set A of **agents**. Without loss of generality, we assume that $A = [\![1; n]\!]$ for some positive integer n. **Local timestamps**, or simply **points**, are identified by pairs $(i, t) \in A \times \mathbb{N}$ of an agent and a timestamp. For a set $X \subseteq A \times \mathbb{N}$ of local timestamps, we define the set $A(X) = \{i \mid (\exists t \in \mathbb{N})(i, t) \in X\}$ of involved agents.

In our model, non-negative integer timestamps $0, 1, 2, \ldots$ are used for snapshots of the system's state. All actions are performed in the open intervals in between: $]0; 1[$, $]1; 2[$, $]2; 3[$, $\ldots$.

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

Definition 1.2 (Initial states)**.** We denote by Σ_i the set of **local initial states** of agent $i \in A$. A **joint initial state**, or **global initial state**, is a tuple of local initial states from $\mathscr{G}(0) = \prod_{i \in A} \Sigma_i$.

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 A.

Definition 1.3 (Local internal actions)**.** We denote by Int_i the set of **local internal actions** of agent $i \in A$.

Definition 1.4 (Local external events)**.** We denote by Ext_i the set of **local external events** the agent $i \in A$ can be subjected to.

We consider a message-passing system whereby agents communicate exclusively by messages.

Definition 1.5 (Messages)**.** We denote by *Msgs* the (possibly infinite) set of **messages** that agents can send to each other. Let $i, j \in A$ be two agents. 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 in agent i's history as $send(j, \mu_k)$ for the copy k of the message; we consider $send(j, \mu_0)$ to be the master copy and in protocols where multiple copies are not necessary denote it simply by $send(j, \mu)$;

- 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 the receiving agent does not know which copy of the message it received).

We denote by $\Omega(Msgs) = \{send(j, \mu_k), \quad recv(j, \mu) \mid j \in A, \mu \in Msgs, k \in \mathbb{N}\}$ the set of all possible *send* and *recv* occurrences.

Remark 1.6 (Global ids and global view of messages)**.** Having multiple copies of the same message serves only one purpose : to enable sending several duplicates of the same message within one round. The environment, which plays the role of the delivery system, must be able to distinguish between identical copies of messages with identical contents but sent/received at different times. Further, while agent i only observes messages sent by itself, 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'. This is modelled by a global message identifier $id \in \mathbb{N}$, or simply GMI. Messages are in the *global format*(or *environment's view*) : $gsend(i, j, \mu, id)$ and $grecv(j, i, \mu, id)$; in opposition to the *local one* : $send(j, \mu_k)$ and $recv(i, \mu)$.

Remark 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 of the run. This is implemented by letting agents sleep through one or more rounds. The waking of agent i is performed by the event $go(i)$ initiated by the environment.

Thus, we are making a distinction between the agent *actively doing nothing* (observing the round passing without any action) and *passively not doing anything* (not noticing the round).

Remark 1.8 (Modelling Byzantine messages)**.** 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. In particular, Byzantine agents can freely choose the contents and recipient of their messages. However, the transfer of such a Byzantine message to its intended recipient is controlled by the environment and follows the standard delivery procedures.

Remark 1.9 (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 the decision to shift the control of Byzantine behavior from corrupted agents to the environment.

Remark 1.10 (How environment communicates)**.** The environment is considered to be omniscient and already knows everything the agents might want to communicate to it. If information has to be delivered to an agent from the environment, this is represented by an external event.

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

Definition 1.11 (Global message identifier function)**.** We fix a function for computing GMIs as any computable one-to-one total function $id\colon A \times A \times Msgs \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[1]

Definition 1.12 (Internal actions)**.** From the point of view of agent $i \in A$, its correct **internal actions**, which can be prescribed by its protocol, consist of the *send* actions from Def. 1.5 and local internal actions represented by $internal\,(i, a)$ for $a \in Int_i$:

$$\overline{Actions_i} = \{send(j, \mu_k) \mid \ j \in A, \mu \in Msgs, k \in \mathbb{N}\} \sqcup \{internal\,(i, a) \mid a \in Int_i\} \tag{1}$$

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

$$\overline{GActions_i} = \{gsend(i, j, \mu, id) \mid \ j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{internal\,(i, a) \mid a \in Int_i\} \tag{2}$$

We also define the sets of (correct) internal actions available to all agents, which only makes sense from the global point of view:

$$\overline{GActions} = \bigsqcup_{i \in A} \overline{GActions_i} \tag{3}$$

[1] A simple though not necessarily the most efficient possibility is to use $2^i \cdot 3^j \cdot 5^{\lceil \mu \rceil} \cdot 7^k \cdot 11^t$.

Definition 1.13 (External events)**.** From the point of view off agent $i \in A$, **correct external events** that it can observe consist of the *recv* actions from Def. 1.5 and local external events represented by $external\,(i,o)$ for $o \in Ext_i$:

$$\overline{Events}_i = \{recv(j,\mu) \mid j \ \in A, \mu \in Msgs\} \sqcup \{external\,(i,o) \mid o \in Ext_i\} \tag{4}$$

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

$$\overline{GEvents}_i = \{grecv(i,j,\mu,id) \mid j \ \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external\,(i,o) \mid o \in Ext_i\} \tag{5}$$

For each correct internal action or correct external event $u \in \overline{GActions}_i \sqcup \overline{GEvents}_i$ of agent i, there is a matching **Byzantine external event**

$$fake\,(i,u).$$

Thus, for received fake messages, the environment creates the message "out of thin air" as if it has been delivered with a particular GMI. Similarly, a fake sent message is created already with a GMI, unlike the correctly sent messages, which are supplied with a GMI in a separate step after creation. However, the messages with GMI are delivered independent of whether they originate as fake or correct ones.

In addition, the environment can trigger two more events that control i's behavior :

- $go(i)$ approves i's actions prescribed by its protocol and
- $fail(i)$ makes i Byzantine irrespective of whether any wrongdoing has already occurred.

Both types of events are not (directly[2]) observable by agents and, hence, are not part of $\overline{Events}_i$.

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

$$GEvents_i = \overline{GEvents}_i \sqcup \{fake\,(i,u) \mid u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} \sqcup \{go(i), fail(i)\} \tag{6}$$

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

$$\overline{GEvents} = \bigsqcup_{i \in A} \overline{GEvents}_i, \qquad GEvents = \bigsqcup_{i \in A} GEvents_i \tag{7}$$

Remark 1.14. The primary purpose of $fail\,(i)$ is to record Byzantine behavior arising form inaction. Accordingly, it may not be necessary in the models where inaction is always considered to be correct behavior(asynchronous agents). We still use it as a bookkeeping device.

We consider agents with perfect recall. Hence, the state of an agent is defined as its local *history*:

Definition 1.15 (Agent's history)**.** A **history** h **of agent** $i \in A$, or its **local state**, over initial states Σ_i, local internal actions Int_i, local external events Ext_i, and messages $Msgs$ is a non-empty sequence

$$h = [E_m, \ldots, E_1, E_0]$$

such that $E_0 \in \Sigma_i$ and $\forall j \in [\![1;m]\!]$ we have $E_j \subset \overline{Actions}_i \sqcup \overline{Events}_i$. In this case m is called the **length of history** h and denoted $|h|$. We say that a set $E \subset \overline{Actions}_i \sqcup \overline{Events}_i$ **happens** in the history h and write $E \subset h$ iff $E = E_j$ for some $j \in [\![1;m]\!]$. We say that $o \in \overline{Actions}_i \sqcup \overline{Events}_i$ **happens** in the history h and write $o \in h$ iff $o \in E$ for some set $E \subset h$.

[2]Events $go(i)$ can sometimes be detected by the acting agent based on the fact that it is acting.

Definition 1.16 (Environment's history)**.** A **history** h **of the system**, or the **global state**, for agents $A = [\![1; n]\!]$ over the global initial states $\mathscr{G}(0) = \prod_{i \in A} \Sigma_i$, local internal actions Int_i for each agent $i \in A$, local external events Ext_i for each agent $i \in A$, and messages $Msgs$ is a tuple

$$h = (h_\epsilon, h_1, \ldots, h_n)$$

where the history of the environment is a (possibly empty) sequence

$$h_\epsilon = [E_m, \ldots, E_1]$$

such that $\forall j \in [\![1; m]\!]$ we have $E_j \subseteq \overline{GActions} \sqcup GEvents$ and h_i is a local state of agent $i \in A$ over Σ_i, Int_i, Ext_i and $Msgs$. In this case m is called the **length of history** h and denoted $|h|$. We say that a set $E \subset \overline{GActions} \sqcup GEvents$ **happens** in the history h_ϵ or in the system h and write $E \subset h_\epsilon$ iff $E = E_j$ for some $j \in [\![1; m]\!]$. We say that $o \in \overline{GActions} \sqcup GEvents$ **happens** in the history h_ϵ or system h and write $o \in h$ iff $o \in E$ for some set $E \subset h$.

Since h_i's are intended to be local histories that are also reflected in h_ϵ, there are various properties one can expect from h. We should require $|h_i| \leq |h_\epsilon|$ and any action occurring locally in h_i should also occur, in its global form, in h_ϵ. These consistency properties will be eventually ensured in the next section.

Definition 1.17 (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 over Σ_i, Int_i, Ext_i, $Msgs$. $\mathscr{L} = \prod_{i \in A} \mathscr{L}_i$ 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.

2 Transition relation

There are multiple consistency restrictions to be imposed on the histories to ensure that information from one of h_i does not contradict what is recorded 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, and the responsible part is the agent's protocol. The environment plays a triple role. Firstly, it is the 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: the protocol of the environment does not restrict which events and in which combinations can occur beyond basic coherency checks such as ensuring that the GMI's have correct timestamps or beyond faithfully describing the underlying distributed system. Thirdly, part of the environment that performs non-deterministic choice is designated the *adversary*. It is by rigging this part to choose the worst reactions by both agents and the environment that we will be achieving the worst-case scenarios.

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, P_i cannot use time t as a parameter. Conversely, the protocol P_ϵ of the omniscient environment can use time t, in fact using t is necessary to correctly forge GMIs for sending 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.

Definition 2.1 (Protocol)**.** For the set $A = [\![1; n]\!]$ of agents, agent $i \in A$, and time moment $t \in \mathbb{N}$:

1. A (non-deterministic) **protocol for agent** i over the set of initial states Σ_i, set of internal actions Int_i, set of external events Ext_i, and set of messages $Msgs$ of the agents is any function
$$P_i \colon \mathscr{L}_i \to 2^{2^{\overline{Actions}_i}} \setminus \{\varnothing\}.$$
If $h \in \mathscr{L}_i$ is a local state of agent i, then each member $S \in P_i(h)$ is a subset of $\overline{Actions}_i$ and represents one of non-deterministic choices prescribing how i should act in this local state.

2. Given individual agents' protocols $P_1, \ldots, P_n$ over their respective sets of initial states Σ_i, sets of internal actions Int_i, sets of external events Ext_i, and set of messages $Msgs$, their **joint protocol** is a function of global states returning a tuple of agent's protocols: for a global state $h = (h_\epsilon, h_1, \ldots, h_n)$ we define
$$P(h) = \big(P_1(h_1), \ldots, P_n(h_n)\big)$$

3. A (non-deterministic) **protocol for the environment** over the sets of internal actions Int_i, sets of external events Ext_i, and set of messages $Msgs$ is any function
$$P_\epsilon \colon \mathbb{N} \longrightarrow 2^{2^{GEvents}} \setminus \{\varnothing\}. \tag{8}$$
In other words, for each $t \in \mathbb{N}$, each member $S \in P_\epsilon(t)$ is a subset of $GEvents$, i.e.,
$$\begin{aligned} S \quad \subset \quad & \{grecv(i,j,\mu,id) \mid i,j \in A, \mu \in Msgs, id \in \mathbb{N}\} \quad \sqcup \quad \{go(i) \mid i \in A\} \quad \sqcup \\ & \{external(i,e) \mid i \in A, e \in Ext_i\} \quad \sqcup \quad \{fail(i) \mid i \in A\} \quad \sqcup \\ & \{fake(i,u) \mid i \in A, u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\}, \end{aligned}$$
and represents one of non-deterministic possibilities for what can happen in the system at time t.5.

Remark 2.2 (Time sensitive actions)**.** The dependence of the environment's protocol on time enables modelling of time-sensitive actions.

Remark 2.3 (Life must go on)**.** The environment and the agents always have at least one (possibly empty) set of events at its disposal (no-apocalypse clause). The situation when the agents crash can still be represented.

Remark 2.4 (Adversary)**.** In addition to allowing the non-deterministic choice to make the worst choices among agent's actions and environment's events, we give the adversary another small degree of freedom in that it can always make the agent do nothing. This ability is needed to ensure the agent can always break its protocol by doing nothing and still register that a round has passed.

When a woken up agent sends a message, it must first be transformed from the local to the global view. This is performed by the *labelling functions* $label_i$ for each $i \in A$. Similarly, when a message is delivered or a fake event occurs, 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 2.5 (Labeling functions)**.** For the set $A = [\![1; n]\!]$ of agents, an agent $i \in A$, and time $t \in \mathbb{N}$, 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:
$$label_i(u,t) = \begin{cases} gsend(i,j,\mu,id(i,j,\mu,k,t)) & \text{if } u = send(j,\mu_k) \\ u & \text{otherwise} \end{cases}$$

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 by a separate function. Thus, this function does not deal with fake events.

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

as follows:

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

The functions $label^{-1}$ and $label_i$ extend to sets in the standard way.

Remark 2.6. The injectivity of the function $id(\cdot)$ used by $label_i$ ensures that each message is unique from the point of view of the environment.

Definition 2.7 (Non-deterministic choice for protocols)**.** For the set $A = [\![1; n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and protocols P_ϵ for the environment and $P_1, \ldots, P_n$ for the agents, we obtain for agent $i \in A$ and time $t \in \mathbb{N}$ 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 set
$$\alpha_\epsilon^t = X_\epsilon \tag{9}$$
for some set $X_\epsilon \in P_\epsilon(t)$ non-deterministically chosen by the adversary.

2. Actions agent $i \in A$ would perform if woken up are a set
$$\alpha_i^{h,t} = label_i(X_i, t) \tag{10}$$
for some set $X_i \in P_i(h_i) \cup \{\varnothing\}$ non-deterministically chosen by the adversary.

3. All these choices are combined in
$$\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t}).$$

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

1. *regular events* for agent i:
$$\overline{\alpha}_{e_i}^t = \alpha_\epsilon^t \cap \overline{GEvents}_i = \{grecv(i, j, \mu, id) \in \alpha_\epsilon^t \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external(i, o) \in \alpha_\epsilon^t \mid o \in Ext_i\} \tag{11}$$

2. instruction to wake agent i:
$$\alpha_{g_i}^t = \alpha_\epsilon^t \cap \{go(i)\} \tag{12}$$

3. Fake events for agent i, including those mimicking the agent's actions:
$$\begin{aligned} \alpha_{b_i}^t = \{fake(i, u) \in \alpha_\epsilon^t \mid u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} = \\ \{fake(i, gsend(i, j, \mu, id)) \in \alpha_\epsilon^t \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake(i, grecv(i, j, \mu, id)) \in \alpha_\epsilon^t \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake(i, internal(i, a)) \in \alpha_\epsilon^t \mid a \in Int_i\} \sqcup \{fake(i, external(i, o)) \in \alpha_\epsilon^t \mid o \in Ext_i\} \end{aligned} \tag{13}$$

4. instructions making agent i Byzantine:

$$\alpha^t_{f_i} = \alpha^t_{b_i} \sqcup \left(\alpha^t_\epsilon \cap \{fail\,(i)\}\right) \tag{14}$$

Finally, we define

$$\overline{\alpha}^t_e = \bigsqcup_{i\in A} \overline{\alpha}^t_{e_i} \qquad \alpha^t_g = \bigsqcup_{i\in A} \alpha^t_{g_i} \qquad \alpha^t_b = \bigsqcup_{i\in A} \alpha^t_{b_i} \qquad \alpha^t_f = \bigsqcup_{i\in A} \alpha^t_{f_i}$$

It is easy to see that

Lemma 2.8. *For the set of agents agent $A = [\![1; n]\!]$, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, and protocols P_ϵ for the environment and $P_1, \ldots, P_n$ for the agents, for agent $i \in A$ and for time $t \in \mathbb{N}$ we have that*

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

To make the environment's protocol independent of the global history, which is crucial for many proofs, we had to sacrifice the ability to weed out actions that are "physically" impossible. This comprises the following types of events:

- a message delivered without being previously[3] sent;
- faking a receipt of a message that was actually received from the same source;
- faking an external event that actually happened;
- faking sending a message that was actually sent to the same destination;
- faking an internal action that was actually performed;
- faking a GMI.

The protocol P_ϵ cannot check many of these conditions as they depend on the global history and actions of agents. Thus, in our model we first allow these events in the P_ϵ sets but neutralizing their effect using the following filter:

Definition 2.9 (Event and action filter functions)**.** Let $a \in GActions$ and $o \in GEvents$. We define an **event filter function**

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

as follows. Given a global history, a set of events attempted by the environment (chosen by the adversary) and sets of actions to be performed by the agents (also chosen by the adversary), the function returns the set of events to be performed by the environment, those attempted that are "physically" possible. Formally, for the set $A = [\![1; n]\!]$ of agents, a set $X \subset GEvents$ of events attempted by the environment, sets $X_i \subset \overline{GActions}_i$ of actions to be performed by each agent $i \in A$, and a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, we define

$$\begin{aligned} filter_\epsilon\,(h, X, X_1, \ldots, X_n) \quad = \quad & X \quad \setminus \\ \Big(\big\{ grecv(j,i,\mu,id) \quad | \quad & gsend(i,j,\mu,id) \notin h_\epsilon \quad \wedge \quad fake\,(i, gsend(i,j,\mu,id)) \notin h_\epsilon \quad \wedge \\ & (gsend(i,j,\mu,id) \notin X_i \vee go(i) \notin X) \quad \wedge \quad fake\,(i, gsend(i,j,\mu,id)) \notin X \big\} \quad \sqcup \\ \big\{ fake\,(i,a) \quad | \quad & a \in X_i \quad \wedge \quad go(i) \in X \big\} \quad \sqcup \quad \big\{ fake\,(i,o) \quad | \quad o \in X \big\} \quad \sqcup \\ & \big\{ fake\,(i, gsend(i,j,\mu,id)) \quad | \quad \forall k \in \mathbb{N}, id \neq id(i,j,\mu,k,|h|) \big\}\Big) \end{aligned} \tag{15}$$

[3]Here previously means in one of the preceding rounds or in the same round, correctly or in a Byzantine fashion.

In addition we define **action filter functions** for agents $A = [\![1; n]\!]$

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

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

$$filter_i\left(X_1, \ldots, X_n, X_\epsilon\right) = \begin{cases} X_i & \text{if } go(i) \in X_\epsilon \\ \varnothing & \text{otherwise} \end{cases}$$

It is obvious from the definition that

$$filter_\epsilon\left(h, X, X_1, \ldots, X_n\right) \subset X \tag{16}$$

$$filter_i\left(X_i, X_\epsilon\right) \subset X_i \tag{17}$$

Thus, after protocols $P_\epsilon\left(t\right)$ and $P_i(h_i)$ provided a range of possible event/action collections 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 be performed in reality. The resulting sets are called β-sets by analogy with α-sets.

Definition 2.10. For the set of agents agent $A = [\![1; n]\!]$, a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t})$, agent $i \in A$, and time $t \in \mathbb{N}$,

1. $\beta_\epsilon^{h,\alpha^{h,t}} = filter_\epsilon\left(h, \quad \alpha_\epsilon^t, \quad \alpha_1^{h,t}, \quad \ldots, \quad \alpha_n^{h,t}\right)$
2. $\beta_i^{h,\alpha^{h,t}} = filter_i\left(\alpha_1^{h,t}, \ldots, \alpha_n^{h,t}, \beta_\epsilon^{h,\alpha^{h,t}}\right)$
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 with α_ϵ^t we also distinguish the following subsets of $\beta_\epsilon^{h,\alpha^{h,t}}$ for each agent $i \in A$:

1. *regular events* for agent i:

$$\begin{aligned} \overline{\beta}_{e_i}^{h,\alpha^{h,t}} = \beta_\epsilon^{h,\alpha^{h,t}} \cap \overline{GEvents}_i = \\ \{grecv(i,j,\mu,id) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{external\left(i,o\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid o \in Ext_i\} \subset \overline{\alpha}_{e_i}^t \end{aligned} \tag{18}$$

2. instruction to wake agent i:

$$\beta_{g_i}^{h,\alpha^{h,t}} = \beta_\epsilon^{h,\alpha^{h,t}} \cap \{go(i)\} = \alpha_{g_i}^t \tag{19}$$

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

$$\begin{aligned} \beta_{b_i}^{h,\alpha^{h,t}} = \{fake\left(i,u\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} = \\ \{fake\left(i, gsend(i,j,\mu,id)\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake\left(i, internal\left(i,a\right)\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid a \in Int_i\} \sqcup \\ \{fake\left(i, grecv(i,j,\mu,id)\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in A, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake\left(i, external\left(i,o\right)\right) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid o \in Ext_i\} \subset \alpha_{b_i}^t \end{aligned} \tag{20}$$

4. instructions making agent i Byzantine:

$$\beta_{f_i}^{h,\alpha^{h,t}} = \beta_{b_i}^{h,\alpha^{h,t}} \sqcup \left(\beta_{\epsilon}^{h,\alpha^{h,t}} \cap \{fail\,(i)\}\right) \subset \alpha_{f_i}^{t} \tag{21}$$

Finally, we define

$$\overline{\beta}_{e}^{h,\alpha^{h,t}} = \bigsqcup_{i\in A} \overline{\beta}_{e_i}^{h,\alpha^{h,t}} \qquad \beta_{g}^{h,\alpha^{h,t}} = \bigsqcup_{i\in A} \beta_{g_i}^{h,\alpha^{h,t}}$$
$$\beta_{b}^{h,\alpha^{h,t}} = \bigsqcup_{i\in A} \beta_{b_i}^{h,\alpha^{h,t}} \qquad \beta_{f}^{h,\alpha^{h,t}} = \bigsqcup_{i\in A} \beta_{f_i}^{h,\alpha^{h,t}}$$

Remark 2.11. The filtering is split in two steps first one for events $\beta_{\epsilon}^{h,\alpha^{h,t}} = \ldots$ and secondly one for the agents $\beta_{i}^{h,\alpha^{h,t}} = filter_i\left(\alpha_1^{h,t},\ldots,\alpha_n^{h,t},\boldsymbol{\beta_{\epsilon}^{t}}\right)$. In order to allow more freedom of filtering, for instance we can filter *go* event (and propagate this to agents) otherwise filtering *go* will not affect agents' behaviour.

Remark 2.12. Consider a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t})$, $i \in A$, time $t \in \mathbb{N}$. Then

$$\beta_{\epsilon}^{h,\alpha^{h,t}} = \overline{\beta}_{e}^{h,\alpha^{h,t}} \sqcup \beta_{g}^{h,\alpha^{h,t}} \sqcup \beta_{f}^{h,\alpha^{h,t}}.$$

Remark 2.13. Consider a global history $h \in \mathscr{G}$, an agent $i \in A$ and time $t \in \mathbb{N}$, the actions filtering function $filter_i$ for agent i ensures that

$$\beta_{i}^{h,\alpha^{h,t}} \neq \varnothing \qquad \Longrightarrow \qquad \beta_{g_i}^{h,\alpha^{h,t}} \neq \varnothing$$

Lemma 2.14. *For the set $A = [\![1;n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t})$, for agent $i \in A$ and for time $t \in \mathbb{N}$ we have that*

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

Each agent initially starts off in a correct state and may become Byzantine either by violating its protocol or by designation of the environment. Thus, it is more precise to talk about Byzantine states of agents, about Byzantine local timestamps $(i,t) \in A \times \mathbb{N}$ instead of announcing the agents themselves to be universally Byzantine.

Definition 2.15. For the set of agents $A = [\![1;n]\!]$, consider a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ of length $|h| = M$, so that $h_\epsilon = [E_M, \ldots, E_1]$. We define the sets of *Bad* and *Failed* local timestamps

$$Bad\,(h) = \left\{(i,t) \in A \times \mathbb{N} \mid \left(\exists u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\right) fake\,(i,u) \in E_t \right\}$$
$$Failed\,(h) = \left\{(i,t) \in A \times \mathbb{N} \mid \exists t' \leq t \left(fail\,(i) \in E_{t'} \text{ or } (i,t') \in Bad\,(h)\right)\right\}$$

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. Formally, this means that the local histories must be purged of the (1) *fake* instances and GMIs and of (2) controlling commands $go(i)$ and $fail\,(i)$. Both tasks are performed by the "cover-up" function σ: the former on the action/event level and the latter on the set level:

Definition 2.16 (Cover-up function)**.** The function $\sigma\colon 2^{GActions \sqcup GEvents} \longrightarrow 2^{\overline{Actions} \sqcup \overline{Events}}$ is defined as follows

$$\sigma\big(X\big) = label^{-1}\left((X \cap \overline{GEvents}) \sqcup \{s \mid (\exists i) fake\,(i,s) \in X\}\right)$$

If $o \in GActions \sqcup GEvents$ but o is neither $go(i)$ nor $fail\,(i)$, we also write $\sigma\big(o\big)$ to denote the only element of $\sigma\big(\{o\}\big)$.

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

Definition 2.17 (State update functions)**.** For the set $A = [\![1; n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, given a tuple of performed actions and events $X = (X_\epsilon, X_1, \ldots, X_n) \in 2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}$, for agent $i \in A$, we use the following abbreviation $X_{\epsilon_i} = X_\epsilon \cap GEvents_i$. Agents i's update function $update_i$ outputs a new local history from $\mathscr{L}_i$ based on i's actions X_i and environment's control X_ϵ as follows:

$$update_i\left(h_i, X_i, X_\epsilon\right) = \begin{cases} h_i & \text{if } X_{\epsilon_i} \subset \{fail\left(i\right)\} \\ \left[\sigma\left(X_{\epsilon_i} \sqcup X_i\right)\right] : h_i & \text{otherwise} \end{cases} \tag{22}$$

where : represents sequence concatenation. The environment's state update function $update_\epsilon$ outputs a new state of the environment based on X_ϵ:

$$update_\epsilon\left(h_\epsilon, X\right) = \left(X_\epsilon \sqcup X_1 \sqcup \cdots \sqcup X_n\right) : h_\epsilon$$

Thus, the state of the system overall is modified as follows:

$$update\left(h, X\right) = \Big(update_\epsilon\left(h_\epsilon, X\right), update_1\left(h_1, X_1, X_\epsilon\right), \ldots, update_n\left(h_n, X_n, X_\epsilon\right)\Big) \tag{23}$$

Remark 2.18. The first clause corresponds to the situation when the agent is not woken up and, hence, does not change its local state.

Definition 2.19 (Transition relation)**.** For the set of agents $A = [\![1; n]\!]$ with protocols $P = (P_1, \ldots, P_n)$ and the protocol P_ϵ of the environment, we define a **transition relation** $\tau_{P_\epsilon, P} \subseteq \mathscr{G}^2$ between a current global state and possible next global states. For global states $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and $h' = (h'_e, h'_1, \ldots, h'_n) \in \mathscr{G}$, we say that $h \tau_{P_\epsilon, P} h'$, or $(h, h') \in \tau_{P_\epsilon, P}$, or $h' \in \tau_{P_\epsilon, P}(h)$ if there exists a tuple

$$\alpha^{h,|h|} = (\alpha_\epsilon^{|h|}, \alpha_1^{h,|h|}, \ldots, \alpha_n^{h,|h|}) \in 2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}$$

with $\alpha_\epsilon^{|h|} \in P_\epsilon\left(|h|\right)$ and $\alpha_i^{h,|h|} \in P_i\left(h_i\right)$ for every $i \in A$ such that

$$h' = \Big(update_\epsilon\Big(h_\epsilon, \quad \beta^{h,\alpha^{h,|h|}}\Big), \quad update_1\Big(h_1, \quad \beta_1^{h,\alpha^{h,|h|}}, \quad \beta_\epsilon^{h,\alpha^{h,|h|}}\Big), \\ \ldots, \quad update_n\Big(h_n, \quad \beta_n^{h,\alpha^{h,|h|}}, \quad \beta_\epsilon^{h,\alpha^{h,|h|}}\Big)\Big) \tag{24}$$

3 Runs and contexts

As already mentioned integer times are used exclusively to take snapshots of the local and global states. The set 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 times.

Definition 3.1 (Run)**.** A **run** is a function that assigns a global state to each integer time.

$$r : \mathbb{N} \longrightarrow \mathscr{G} \tag{25}$$

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 time.

$$r_i : \mathbb{N} \longrightarrow \mathscr{L}_i \tag{26}$$

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

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

where π_j is the jth projection function for tuples/sequences.

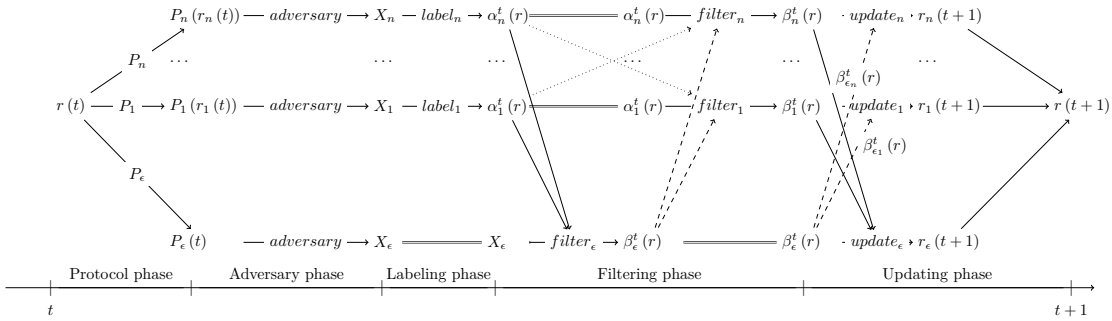

Figure 1: The evolution of states from time $t \in \mathbb{N}$ to $t+1$ inside a run r constructed according to the transition relation $\tau_{P_\epsilon,P}$, where $A = [\![1;n]\!]$.

Definition 3.2. For a set of agents $A = [\![1;n]\!]$ and a set $X \subset A \times \mathbb{N}$ of local timestamps, or simply points, we define its upper **time bound** $T(X)$ to be the largest $T \in \mathbb{N}$ such that $(i,T) \in X$ for some $i \in A$ if such a T exists. $T(\varnothing)$ is defined to be 0. A set X without a time bound is called **unbounded**.

Definition 3.3. For a set of agents $A = [\![1;n]\!]$, a run $r \in R$, time $t \in \mathbb{N}$, and a bounded set $X \subset A \times \mathbb{N}$ of local timestamps, we define

$$Bad\left(r,t\right) = Bad\left(r\left(t\right)\right) \qquad Bad_X\left(r\right) = X \cap Bad\left(r, T(X)\right)$$
$$Failed\left(r,t\right) = Failed\left(r\left(t\right)\right) \qquad Failed_X\left(r\right) = X \cap Failed\left(r, T(X)\right)$$

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

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

Remark 3.4. For the set $A = [\![1;n]\!]$ of agents, given a run $r \in R$ and time $t \in \mathbb{N}$, given a tuple of performed actions and events

$$\beta^{r(t),\alpha^{r(t),t}} = \left(\beta_\epsilon^{r(t),\alpha^{r(t),t}}, \beta_1^{r(t),\alpha^{r(t),t}}, \ldots, \beta_n^{r(t),\alpha^{r(t),t}}\right),$$

for agent $i \in A$, the next global and local states are computed based on $\beta^{r(t),\alpha^{r(t),t}}$ according to the transition relation $\tau_{P_\epsilon,P}$ from Def. 2.19 iff, for all $i \in A$,

$$r_i\left(t+1\right) = update_i\left(r_i\left(t\right), \quad \beta_i^{r(t),\alpha^{r(t),t}}, \quad \beta_\epsilon^{r(t),\alpha^{r(t),t}}\right) \tag{27}$$

$$r_\epsilon\left(t+1\right) = update_\epsilon\left(r_e\left(t\right), \quad \beta^{r(t),\alpha^{r(t),t}}\right) \tag{28}$$

$$\tag{29}$$

In this case, we always have

$$r\left(t+1\right) \in \tau_{P_\epsilon,P}\left(r\left(t\right)\right) \tag{30}$$

This detailed diagram shows that runs that comply with a transition relation $\tau_{P_\epsilon,P}$ are constructed based on $\alpha^t\left(r\right) = \left(\alpha_1^t\left(r\right), \ldots, \alpha_n^t\left(r\right)\right)$ chosen by the adversary at each round based on the joint agents' protocol P and on X_ϵ chosen by the adversary at each round based on the environment's protocol P_ϵ. However, these choices are not part of the run and, generally, cannot be uniquely restored from the run. What is recorded in the global history and can be restored from it are the actually performed actions and events $\beta^t\left(r\right) = \left(\beta_\epsilon^t\left(r\right), \beta_1^t\left(r\right), \ldots, \beta_n^t\left(r\right)\right)$, which correspond to $\beta^{r(t),\alpha^{r(t),t}}$ in Rem. 3.4.

Runs r complying with the transition relation $\tau_{P_\epsilon,P}$ have several obvious properties:

Remark 3.5 (Total recall)**.** Not only $\beta^t(r)$ but also all $\beta^{t'}(r)$ for $t' \leq t$ can be extracted from $r(t)$.

Remark 3.6 (GMIs are unique)**.** The GMI *id* complete determines the sender, the recipient, the sent message and the time of sending, whether the message is sent correctly or faultily.

Remark 3.7 (Send-receive causality)**.** 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.

In Def. 2.19, we defined the basic transition relation $\tau_{P_\epsilon,P}$ based on the protocols of the agents and the environment. We will sometimes need to change the filtering phase of the transition relation. Hence, we leave the exact details of the transition as a parameter τ that converts the protocols into a transition relation.

Definition 3.8 (Transition template)**.** A **transition template** is a two-place function that takes the protocol P_ϵ of the environment and the joint agents' protocol P and outputs a transition relation

$$\tau(P_\epsilon)(P) = \tau_{P_\epsilon,P} \in \mathscr{G}^2 \tag{31}$$

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. To enforce such properties we are forced to restrict the set of runs being considered.

Definition 3.9 (Admissibility condition)**.** An **admissibility condition** Ψ is any subset of the set R of all runs.

Now we have all ingredients to define sets of runs for particular communication models.

Definition 3.10 (Context)**.** A **context** consists of an environment protocol P_ϵ, a set of global initial states $\mathscr{G}(0)$, a transition template τ and an admissibility condition Ψ.

$$(P_\epsilon, \mathscr{G}(0), \tau, \Psi) \tag{32}$$

We also consider sets of contexts varying only in the environment's protocol:

$$\Gamma = \{\ (P_\epsilon, \mathscr{G}(0), \tau, \Psi) \mid P_\epsilon \text{ is an environment protocol } \} \tag{33}$$

We can view a **context** as an extended environment where a **joint protocol** is executed.

Definition 3.11 (Agent-context)**.** An **agent-context**, for a context γ and joint protocol P is the pair (γ, P).

Definition 3.12 (Consistency)**.** For a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau, \Psi) \in \Gamma$ and a joint protocol P, we define the set of runs **weakly consistent** with P in γ and call it the system $R^{w(\gamma,P)}$ as the set of runs that are built by composing the **transition relation** $\tau_{P_\epsilon,P}$ starting from a **global initial state** from $\mathscr{G}(0)$:

$$R^{w(\gamma,P)} \quad = \quad \{r \in R \quad | \quad r(0) \in \mathscr{G}(0) \quad \text{and} \quad (\forall t \in \mathbb{N})\, r(t+1) \in \tau_{P_\epsilon,P}(r(t))\} \tag{34}$$

A run r weakly consistent with P in γ is called **strongly consistent** or simply **consistent** if, additionally, $r \in \Psi$. We denote the system of all consistent runs

$$R^{(\gamma,P)} = R^{w(\gamma,P)} \cap \Psi \tag{35}$$

Now, we will add properties and definitions related to agents' knowledge. We start with runs indistinguishable for a group $G \subset A$ of agents:

Definition 3.13 (Local equivalence)**.** For an agent-context χ, a group of agents $G \subset A$ and two runs $r, r' \in R^\chi$ we say that r and r' are **locally equivalent** for G, written $r \sim^G r'$, iff

$$\Big(\forall (j,t) \in G \times \mathbb{N}\Big)\, r_j(t) = r'_j(t)$$

We also write $\sim^i$ instead of $\sim^{\{i\}}$.

Remark 3.14. For any group $G \subset A$ of agents, the relation $\sim^G$ is an equivalence relation.

4 Syntax and Semantics

In the previous sections, we have defined how to specify a behaviour, abiding by some arbitrary rules, of the environment and of the agents. Here, we will define a formal language and its semantics in order to express knowledge of an agent in a such a system.

Definition 4.1 (Atomic propositions)**.** Π is the countable set of **atomic propositions**.

Definition 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 of an atomic proposition $p \in \Pi$ is $\pi(h)(p)$.

Definition 4.3 (Interpreted agent-context)**.** An **interpreted agent-context** is a triple $\xi = (\gamma, \pi, P)$ for an interpretation π, a context γ and joint protocol P.

Definition 4.4 (Interpreted system)**.** An **interpreted system**, for an interpreted agent-context (γ, π, P), is the tuple $I^\gamma_P = (R^P, \pi)$

For agent i's point of view two global states are indistinguishable when the local state of i is the same in both global states. Then, we will naturally define an indistinguishability relation R_i over global states for agent i, called **possible worlds relation**.

Definition 4.5 (Possible worlds relation)**.** For agent $i \in A = [\![1; n]\!]$, the **possible worlds relation** $R_i \subset \mathscr{G}^2$ is formally defined as follows:

$$R_i = \{\ (h, h') \mid \pi_{i+1} h = \pi_{i+1} h' \} \tag{36}$$

In other words, in the global history h agent i considers the global history h' possible if i's local histories, which are the $(i+1)$th projections of the global ones, are the same in the two histories.

With this relation we can build a Kripke structure over the global states $\mathscr{G}$. Therefore, we will be able to define epistemic modal operators in the following.

Definition 4.6. For agents $A = [\![1; n]\!]$, and an agent-context $\chi = (\gamma, P)$, we define the related Kripke structure as follows:

$$D^\chi = \{r(t) \mid r \in R^P, t \in \mathbb{N}\} \tag{37}$$

$$R^\chi_i = R_i \cap D^\chi \tag{38}$$

$$F^\chi = \Big(D^\chi, \quad R^\chi_1, \ldots, R^\chi_n\Big) \tag{39}$$

Remark 4.7. F^χ is an S5 model with domain $D^\chi \subseteq \mathscr{G}$.

Now, we will define a **language** $\mathscr{L}$ to deal with the expression of knowledge in a *system*. For this we will extend the propositional logic with :

1. three modal operators :

- K_i operator for agent $i \in A$, in order to deal with the "knowledge of a fact by agent i." For a global state $h \in \mathscr{G}$, $K_i(\varphi)$ formula can be read as "the agent i knows φ holds": this means that, in every global state indistinguishable from h for i, the proposition φ holds.
- $E_G()$ operator for a group of agents $G \subset A$, in order to express "everyone in the group of agents G knows." It is naturally defined from the operators K_i.
- $C_G()$ operator for a group of agents $G \subset A$, in order to express "something is common knowledge among the agents of G." Common knowledge of φ refers to the fact that everyone in G knows that everyone in G knows ... that everyone in G knows φ.

2. and two temporal operators :

- $\Box$ operator, to express things like "the sender will remember for ever that he has sent *Hello*."
- And its dual operator $\Diamond$, to express things like "he will eventually see that this is a fake news."

Definition 4.8. For an agent $i \in A$, a group of agents $G \subset A$ and an atomic proposition $p \in \Pi$, $\mathscr{L}$ is generated by the following BNF specification

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

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

Definition 4.9. For an interpreted agent-context $\chi = (\gamma, P)$, an agent $i \in A$, a group of agents $G \subset A$, an integer $0 < k \in \mathbb{N}$, a Kripke structure F^χ, a run $r \in R^\chi$, time $t \in \mathbb{N}$, an interpretation function π and the Kripke model $M = (F^\chi, \pi)$:

$$\begin{array}{ll}
(M,r,t) \models p & \text{iff } \pi(r(t))(p) = \top \\
(M,r,t) \models \varphi \vee \varphi' & \text{iff } (M,r,t) \models \varphi \text{ or } (M,r,t) \models \varphi' \\
(M,r,t) \models \varphi \wedge \varphi' & \text{iff } (M,r,t) \models \varphi \text{ and } (M,r,t) \models \varphi' \\
(M,r,t) \models \neg\varphi & \text{iff } (M,r,t) \not\models \varphi \\
(M,r,t) \models K_i(\varphi) & \text{iff } (\forall r' \in R^P)(\forall t' \in \mathbb{N})\big(r'_i(t') = r_i(t) \Rightarrow (M^\chi, r', t') \models \varphi\big) \\
(M,r,t) \models E_G(\varphi) & \text{iff } (\forall i \in G)\,(M,r,t) \models K_i(\varphi) \\
(M,r,t) \models E^0_G(\varphi) & \text{iff } (M,r,t) \models \varphi \\
(M,r,t) \models E^k_G(\varphi) & \text{iff } (M,r,t) \models E_G\left(E^{k-1}_G(\varphi)\right) \\
(M,r,t) \models C_G(\varphi) & \text{iff } (\forall m \in \mathbb{N})\,(M,r,t) \models E^m_G(\varphi) \\
(M,r,t) \models \Diamond\varphi & \text{iff } (\exists t' \geq t)\,(M,r,t') \models \varphi \\
(M,r,t) \models \Box\varphi & \text{iff } (\forall t' \geq t)\,(M,r,t') \models \varphi
\end{array}$$

5 Atomic Propositions

We have the language $\mathscr{L}$ to make statements and the associated semantics to tell the truth value of a formula for a given joint protocol P, for a given run $r \in R$ and time $t \in \mathbb{N}$. Now, we will designate some atomic propositions from Π as special and consider their truth values fixed according to $r(t)$ rather than arbitrary. In other words, we will restrict interpretations π so as to adhere to the following intended meanings:

- $correct_{(i,t)}$ states that by time instant $t \in \mathbb{N}$ agent $i \in A$ did not violate its protocol in any way, in particular, did not exhibit any Byzantine actions or events, from time 0 to time t, whether i was marked with $fail(i)$ in this time period or not.

- $fake_{(i,t+1)}(o)$ states that o is present in round $t.5$ of $r_i(t+1)$ for agent $i \in A$, time $t \in \mathbb{N}$, and event/action $o \in \overline{Actions}_i \sqcup \overline{Events}_i$ due to Byzantine behavior, i.e., $o \in \sigma\big(\beta^t_{b_i}(r)\big)$.
- $\overline{occurred}_{(i,t+1)}(o)$ states that o is present in round $t.5$ of $r_i(t+1)$ for $i \in A$, $t \in \mathbb{N}$, and $o \in \overline{Actions}_i \sqcup \overline{Events}_i$ due to correct behavior, i.e., $o \in label^{-1}\left(\beta^t_i(r) \sqcup \overline{\beta}^t_{e_i}(r)\right)$.

Definition 5.1. For an interpreted agent-context χ, an agent $i \in A$, two time instants $t, t' \in \mathbb{N}$ such that $t' \geq t$, and for an internal action or a correct external event $o \in \overline{Actions}_i \sqcup \overline{Events}_i$, we require for the Kripke model $M = (F^\chi, \pi)$ that the interpretation π satisfy the following properties:

$$(M, r, t') \models correct_{(i,t)} \text{ iff } (i,t) \notin Bad(r, t') \tag{40}$$
$$(M, r, t') \models fake_{(i,t)}(o) \text{ iff } o \in \sigma\left(\beta^{t-1}_{b_i}(r)\right) \tag{41}$$

The environment is, by designed, omniscient then it remembers the precise behaviour of an agent for eternity.

Remark 5.2. Let an agent-context χ, $o \in \overline{Actions} \sqcup \overline{Events}$, $r \in R^\chi, \theta = (i,t) \in A \times \mathbb{N}, t' \geq t$ and a Kripke model $M = (F^\chi, \pi)$

$$(M, r, t') \models correct_\theta \Leftrightarrow (M, r, t') \models \Box correct_\theta$$
$$(M, r, t') \models fake_\theta(o) \Leftrightarrow (M, r, t') \models \Box fake_\theta(o)$$

Now, we will define the **absolute occurrence** of an internal action or a correct external event $o \in \overline{Actions} \sqcup \overline{Events}$ at some point $(i,t) \in A \times \mathbb{N}$. It describes the non Byzantine happening of o. In fact, the **absolute occurrence** corresponds to the point of view of the system, because it is aware of the deep self of an action (resp event).

Definition 5.3 (Absolute occurred)**.** For an agent-context χ, an agent $i \in A$, two time instants $(t, t') \in \mathbb{N}, t' \geq t$, a Kripke model $M = (F^\chi, \pi)$ and an internal action $a \in \overline{Actions}$ or a correct external event $e \in \overline{Events}$

$$(M, r, t') \models \overline{occurred}_{(i,t)}(e) \text{ iff } e \in label^{-1}\left(\overline{\beta}^{t-1}_{e_i}(r)\right) \tag{42}$$
$$(M, r, t') \models \overline{occurred}_{(i,t)}(a) \text{ iff } a \in label^{-1}\left(\beta^{t-1}_i(r)\right) \tag{43}$$

The **absolute occurred** is part of the behaviour of an agent therefore the environment will remember this forever

Remark 5.4. Let an agent-context χ, $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^\chi, (i,t) \in A \times \mathbb{N}, t' \geq t$, and the Kripke model $M = (F^\chi, \pi)$

$$(M, r, t') \models \overline{occurred}_{(i,t)}(o) \Leftrightarrow (M, r, t') \models \Box\overline{occurred}_{(i,t)}(o)$$

The **absolute occurrence** of an internal action or a correct external event $o \in \overline{Actions} \sqcup \overline{Events}$ at some point $(i,t) \in A \times \mathbb{N}$ ought and the Byzantine occurrence of o at the same point ought to be disjoint. More formally

Lemma 5.5. *Let an agent-context χ , $r \in R^\chi, t \in \mathbb{N}, \theta = (i,t) \in A \times \mathbb{N}, o \in \overline{Actions} \sqcup \overline{Events}, t' \geq t$, and a Kripke model $M = (F^\chi, \pi)$*

- $(M, r, t') \models \big(\overline{occurred}_\theta(o) \Rightarrow \neg fake_\theta(o)\big)$
- $(M, r, t') \models \big(fake_\theta(o) \Rightarrow \neg\overline{occurred}_\theta(o)\big)$

We can deal with the non-Byzantine actions (or events) hence we will define the tool - the **relative occurrence** - needed to describe the occurrence of action and events (in a Byzantine way or not). In fact the **relative occurrence** corresponds to the point of view of an agent, because he can not at first glance tell if a behaviour is Byzantine or not.

Definition 5.6 (Relative occurred)**.** For an agent-context χ, agent $i \in A$ and two time instants $(t,t') \in \mathbb{N}, t' \geq t$, a Kripke model $M = (F^{\chi}, \pi)$ and for an internal action or a correct external event $o \in \overline{Actions} \sqcup \overline{Events}$

$$(M,r,t') \models occurred_{(i,t)}(o) \text{ iff } (M,r,t') \models \overline{occurred}_{(i,t)}(o) \vee fake_{(i,t)}(o) \tag{44}$$

$$(M,r,t') \models occurred_i(o) \text{ iff } \exists t \leq t', (M,r,t') \models occurred_{(i,t)}(o) \tag{45}$$

$$(M,r,t') \models occurred(o) \text{ iff } \exists i \in A, (M,r,t') \models occurred_i(o) \tag{46}$$

Now, will extend the previous **relative occurrence** not only to a point but to an anonymous set of agent-disjoint points

$$(M,r,t') \models \varepsilon occurred(o) \text{ iff } \exists i_1, \ldots, i_\varepsilon, \begin{cases} j \neq l \Rightarrow i_j \ \neq i_l \\ (M,r,t') \models \bigwedge_{l \in [\![1;\varepsilon]\!]} occurred_{i_l}(o) \end{cases} . \tag{47}$$

Lemma 5.7. *Let an agent-context χ, $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^{\chi}, (i,t) \in A \times \mathbb{N}, t' \geq t$ and a Kripke model $M = (F^{\chi}, \pi)$*

$$(M,r,t') \models occurred_{(i,t)}(o) \text{ iff } \begin{cases} r_i(t) = E : r_i(t-1) \\ o \in E \end{cases}$$

6 Past and causality relationship

In this section, we will introduce the **causality relationship**(*Lamport's Happened Before relation*), it represents the dependence between points i.e when an agent i send a message to j at time 0, received at time 1, then all the future states of j after time 1 are related to the receipt of the message or more generally to the point $(i,0)$.

Definition 6.1 (Lamport's Happened Before relation with Byzantine Failure)**.** For a run r, two points $(i,j) \in A^2$, two timestamps $(t,t') \in \mathbb{N}^2$, a message $\mu \in Msgs$ and an GID $id \in \mathbb{N}$ we define the **Lamport's Happened Before relation** $\rightarrow_r^{Lb}$ as the smallest relation, over points, satisfying :

1. Locality relation :
$$\text{If } t \leq t' \text{ then } (i,t) \rightarrow_r^{Lb} (i,t') \tag{48}$$

2. Message relation :
$$\text{If } \begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \ \beta_i^t(r) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id)) \in \beta_{b_i}^t(r) \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{e_j}^{t'-1}(r) \end{cases} \quad \text{then } \ (i,t) \rightarrow_r^{Lb} (i,t') \tag{49}$$

3. Transitivity relation :
$$\text{If } \begin{cases} (i,t) \rightarrow_r^{Lb} (j,t') \\ (j,t') \rightarrow_r^{Lb} (k,t'') \end{cases} \quad \text{then } (i,t) \rightarrow_r^{Lb} (k,t'') \tag{50}$$

Definition 6.2 (Causal graph)**.** For a run r, we define the causal graph of r as $G(r) = (V, E^r)$, representing all the causality dependences of nodes in r, such that

1. $V = A \times \mathbb{N}$
2. $E^r = E^r_{loc} \cup E^r_{msg}$
   - $E^r_{loc} = \{((i,t),(i,t+1)) \mid (i,t) \in A \times \mathbb{N}\}$
   - $E^r_{msg} = \left\{ ((i,t),(j,t')) \mid \left\{ \begin{array}{l} \left\{ \begin{array}{l} gsend(i,j,\mu,id) \in \ \beta^t_i(r) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id)) \in \beta^t_{b_i}(r) \end{array} \right. \\ grecv(j,i,\mu,id) \in \overline{\beta}^{t'-1}_{e_j}(r) \end{array} \right. \right\}$

Definition 6.3 (Path)**.** For a run r, two points $(\beta,\theta) \in (A \times \mathbb{N})^2$, we define the following notations

- $\beta \to \theta$ iff $(\beta,\theta) \in E^r$
- ξ is a path from β to θ iff $\beta \rightsquigarrow^{\xi} \theta$ iff $\xi : \beta = \eta_0 \to \eta_1 \to \cdots \to \eta_{|\xi|-1} = \theta$

The **causal cone** of a point $\theta \in A \times \mathbb{N}$ in a run r is the points that are part of the past of θ i.e points that impact the state of θ.

Definition 6.4 (Causal Cone)**.** For a run r, a point $\theta \in A \times \mathbb{N}$, we define the causal cone of θ in r as follows $V^r_\theta = \{\beta \in A \times \mathbb{N} \mid \beta \to^{Lb}_r \theta\}$

Definition 6.5 (Past graph)**.** For a run r and a point $\theta \in A \times \mathbb{N}$, $Past(r,\theta)$ is the subgraph of G(r) induced by V^r_θ

Lemma 6.6. *For a run r and a point $\theta \in A \times \mathbb{N}$, $\beta \in V^r_\theta \Rightarrow V^r_\beta \subset V^r_\theta$*

Lemma 6.7. *For a run r and two points $(\beta,\theta) \in (A \times \mathbb{N})^2$*

$$\beta \to^{Lb}_r \theta \text{ iff there is a path from } \beta \text{ to } \theta \text{ in } Past(r,\theta)$$

.

Part II

The Byzantine Asynchronous Agents

7 The framework extension

In this section, we use the previous framework to define the **Byzantine Asynchronous Agents extension**. In this model, any agent can become Byzantine at any time, however the total number of Byzantine agents is bounded. Moreover, we add a sanity condition : no correct agent can be frozen forever by the environment.

A **Byzantine Asynchronous Environment protocol** is an environment protocol which allows to fail any agent or to fake any action or event at any time. It is formally defined as follows

Definition 7.1 (Byzantine Agents Environment Protocols)**.**

$$\begin{aligned} \mathscr{C}^{BA}_\epsilon = \{P_\epsilon \text{ an environment protocol} \mid \forall t \in \mathbb{N}, \forall S \in P_\epsilon(t), \\ \forall X \subset \{fake(j,u), fail(j) \mid j \in A, u \in \overline{GActions} \sqcup \overline{GEvents}\}, \\ S \cup X \in P_\epsilon(t) \wedge S \backslash X \in P_\epsilon(t)\} \end{aligned} \quad (51)$$

Definition 7.2. For agents A, the **Fair schedule** condition ensures that no correct agent can be frozen for eternity by the environment i.e that within in a finite delay the environment will issue a $go(i)$ for agent $i \in A$ if i is correct. It is formally defined as follows

$$FS = \left\{ r \in R \mid \forall (i,t) \in A \times \mathbb{N}, \exists t' \geq t, \begin{cases} go(i) \in \beta_{g_i}^{t'}(r) \\ \text{ or} \\ (i,t') \in Failed(r,t') \end{cases} \right\} \tag{52}$$

The **Maximal number of Byzantine Agents** condition ensures that the number of Byzantine agents in a run will be bounded by some constant $f \in [\![0; |A|]\!]$. This condition is useful because a lot of results holds only if the number of Byzantine agents is bounded. It is formally defined as follows

$$MB = \{ r \in R \mid |A(Failed(r))| \leq f \} \tag{53}$$

The **admissibility conditions** for the *Byzantine Asynchronous Context* are *Faire schedule* and *Maximal number of Byzantine Agents*

$$\Psi^{BA} = FS \cap MB \tag{54}$$

Definition 7.3 (Byzantine Asynchronous Context)**.** The set of **Byzantine Asynchronous Contexts** is defined as follows

$$\Gamma^{BA} = \left\{ (P_\epsilon, \mathscr{G}(0), \tau, \Psi^{ba}) \mid P_\epsilon \in \mathscr{C}_\epsilon^{BA} \right\} \tag{55}$$

Then we define the related **Byzantine Asynchronous agent-contexts** as follows

$$X^{BA} = \left\{ (\gamma, P) \mid \gamma \in \Gamma^{BA}, P \text{ a joint protocol} \right\} \tag{56}$$

We have defined the extension, of the framework describe in Part 1, that we will use until the end of this report. Now, we will need to define a notion, the agreement, in order to guarantee the same execution in two different runs.

Definition 7.4 (Agreement on a point)**.** For a agent-context χ and two runs $r, r' \in R^\chi$, we say that r and r' **agree on** a point $(i,t) \in A \times \mathbb{N}$ iff

1. $r_i(t) = r'_i(t)$ — the local states of i at t are identical
2. $\overline{\beta}_{e_i}^t(r) = \overline{\beta}_{e_i}^t(r')$ — correct external actions imposed on i in round $t.5$ are identical
3. $\beta_i^t(r) = \beta_i^t(r')$ — intended protocol actions of i in round $t.5$ are identical
4. $\beta_{g_i}^t(r) = \beta_{g_i}^t(r')$ — no difference is observed as to whether i is awoken for round $t.5$
5. $\beta_{b_i}^t(r) = \beta_{b_i}^t(r')$ — faulty behavior of i in round $t.5$ is identical

Now, let us extend the agreement to a set of points $S \subset A \times \mathbb{N}$. We say that r and r' agree on S iff

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

Lemma 7.5. *For an agent-context χ, two runs $(r, r') \in R^{\chi 2}$ and a point $(i,t) \in A \times \mathbb{N}$*

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

Furthermore, i have naturally instantiated this framework in order to represent the following models : *Synchronous agents*, *Reliable communication*, *Synchronous communication*, *Multicast communication*, *Rendezvous communication*, *Coordinated agents*, *Partially Synchronous agents*, *Time-bounded communication*, *Lock-step synchronous agents.*

8 From Past toward Knowledge

In this section, we link the knowledge of an occurrence of a non-Byzantine fact to the communication structure. However, i will present only a small subset of results we have because they have been expressed(and proved) in a past version of the model(that differs from the current one by at least fifty versions). First, according to the intuition, we state that the knowledge of an agent i at time t only depends on the past of the point (i,t). More formally :

Lemma 8.1. *Let P a joint protocol, $r \in R^P$, $\theta = (i,t) \in A \times \mathbb{N}$. Then there is a $r' \in R^P$ such that:*

A. r and r' agree on V_θ^r

B. $\forall (j,t') \in A \times \mathbb{N}, t' \leq t, (j,t') \notin V_\theta^r \Rightarrow \beta_{\epsilon_j}^{t'}(r') = \varnothing$

C. $Bad_{V_\theta^r}(r) = Bad_{V_\theta^{r'}}(r')$

D. $A\left(Failed_{V_\theta^{r'}}(r')\right) = A\left(Bad_{V_\theta^{r'}}(r')\right)$

E. $\forall \beta \in V_\theta^r, o \in \overline{Actions} \sqcup \overline{Events}$:

$$(M^P, r, t) \models \overline{occurred}_\beta(o) \Leftrightarrow (M^P, r', t) \models \overline{occurred}_\beta(o)$$
$$(M^P, r, t) \models fake_\beta(o) \Leftrightarrow (M^P, r', t) \models fake_\beta(o)$$

Now, we will define a group of convenient notations in order to deal with the communication structure.

Definition 8.2. For an action or an event $o \in \overline{Actions} \sqcup \overline{Events}$, for a point $\theta = (i,t) \in A \times \mathbb{N}$ and a run r, we denote by $\overline{\Theta_\theta^r}(o)$ the points where o occurred, in a non-Byzantine fashion, inside the causal cone of θ in r. It is formally defined as follows

$$\overline{\Theta_\theta^r}(o) = \{\eta \in V_\theta^r | (M^P, r, t) \models \overline{occurred}_\eta(o)\} \tag{57}$$

We denote by $\Xi_\theta^r(o)$ all the paths from the set of points $\overline{\Theta_\theta^r}(o)$ to the point θ.

$$\Xi_\theta^r(o) = \left\{ \xi : \text{path in } Past(r,\theta) \mid \exists \eta \in \Theta_\theta^r(o), t' \leq t, \begin{cases} \eta \mapsto^\xi (i,t') \\ i \notin A(\xi \backslash \{(i,t')\}) \end{cases} \right\} \tag{58}$$

And we define the related notion $E_\theta^r(o)$ which represents the dependencies between agents independently of the time.

$$E_\theta^r(o) = \{\ i1 \to \cdots \to ik \mid \xi : (i1,t1) \to \cdots \to (ik,tk) \in \Xi_\theta^r(o)\} \tag{59}$$

Eventually, we denote by $Barrier_\theta^r(o)$ the smallest set of agents needed to disconnect $\{i\}$ from $A\left(\overline{\Theta_\theta^r}(o)\right)$ in $E_\theta^r(o)$.

$$Barrier_\theta^r(o) = \left\{ (j,t) \mid t \in \mathbb{N}, j \in \arg\min_{X \subset A \backslash \{i\}} Xseparing\{i\} \text{ from } A(\Theta_\theta^r(o)) \text{ in } E_\theta^r(o) \right\} \tag{60}$$

Now, we will present the main lemma, it states that in order to gain knowledge of an occurrence of a non Byzantine action (resp event) we need that $|Barrier_\theta^r(o)| > f$, where f is the maximal number of Byzantine agents in a run. The intuition behind this lemma is the following one : there exists at least one non Byzantine path from one agent of $\overline{\Theta_\theta^r}(o)$ to i in order to propagate the knowledge of o.

Lemma 8.3. *For a non Byzantine action or event* $o \in \overline{Agents} \cup \overline{Events}$*, a run* r *and* $\varepsilon = f - |\{j \in A(Bad(V_\theta^r))|\forall m, (j,m) \notin \xi \in \Xi_\theta^r\}| + 1$

$$\begin{cases} (M^P, r, t) \models \neg fail(i) \wedge K_i(fail(i) \vee \overline{occurred}(o)) \\ (M^P, r, t) \not\models occurred_i(o) \end{cases} \Rightarrow |Barrier_\theta^r(o)| \geq \varepsilon$$

In order to gain knowledge of a non Byzantine action(resp event), we need at least ε different sources for the action(resp event). Where ε is a run specific bound, and the upper bound is f.

Corollary 8.3.1. *For a non Byzantine action or event* $o \in \overline{Agents} \cup \overline{Events}$*, a run* r *and* $\varepsilon = f - |\{j \in A(Bad(V_\theta^r))|\forall m, (j,m) \notin \xi \in \Xi_\theta^r\}| + 1$

$$\begin{cases} (M^P, r, t) \models \neg fail(i) \wedge K_i(fail(i) \vee \overline{occurred}(o)) \\ (M^P, r, t) \not\models occurred_i(o) \end{cases} \Rightarrow |\overline{\Theta_\theta^r}(o)| \geq \varepsilon$$

Theorem 8.4 (Knowledge Gain Theorem)**.** *Let* $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, \theta = (i,t) \in A \times \mathbb{N}, (M^P, r, t) \models \neg occurred_i(o)$,

$$(M^P, r, t) \models K_i(fail(i) \vee \overline{occurred}(o)) \text{ then } \forall r' \in R, r_i(t) = r'_i(t), |Barrier_\theta^r(o)| \geq \varepsilon$$

I strongly believe that we can prove the reciprocal in the current model because i have done it in a previous one and i have design the current model in order to keep the same ethos. However, for now, the previous proof can not hold and adapting it need extra care and a finalized framework.

Part III

Conclusion

This report extends the notion of epistemic reasoning for modeling distributed systems into Byzantine settings. Our framework relies on a strict separation of concerns of various actors, mainly a strict separation of the environment where each part enhances a specific constrain. In fact, each actor acts as a black box for the others therefore the behaviour of one of them can be changed easily without altering the whole framework. As a result, our framework can cover all existing distributed computing models we are aware of like the lock-step synchronous agents, all variants of partially synchronous systems, asynchronous ones with or without oracles and even timed systems. However, we have not investigated how to add several distinct thresholds of faults as in the *Hybrid fault model*. Moreover, Byzantine agents may behave arbitrary and may have an arbitrary knowledge indeed this framework allows us to deal about the knowledge of all agents, faulty or not.

Reasoning about the knowledge of an agent can be arduous and tricky, a way to simplify this is to only study the structure of the past of an agent at some time moment(i.e the communication structure). We have identified a necessary and sufficient communication structure ensuring the knowledge of the occurrence of a non Byzantine event or a non Byzantine action in asynchronous systems. We have not studied the extension of thus structure to other models yet, such a study would be a natural continuation of this work.

In the current work, we are polishing the framework and after we will start diving into the proofs of communication structure condition. There is a draft of the current research report at the end of the appendix(B).

In the future work, we plan first to extend the framework in order to cover heterogeneous agents i.e agent that will not be process the same way by the environment. A perfect example of such heterogeneous processing is the *timely-f-source* model where agents are synchronous and communication are reliable, moreover there is a dynamic non-faulty agent(called *source*) that has

time-bound communication and multicast with at least f distinct agents. And also to cover different thresholds of failure i.e mix benign faults (ex : crash failure) with Byzantine one. Secondly, we plan to study a possible hierarchy of the extensions, in order to provide transparent property conservation from an extension to another one.

References

[AW04] Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.

[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 (JACM)*, 61(2):13, 2014.

[Die00] Reinhard Diestel. *Graph Theory*. Springer, 2000.

[DM90] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, 1990.

[FHMV04] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[Fim17] Patrik Fimml. Knowledge in distributed systems with byzantine failures. Master's thesis, Faculty of Informatics at the TU Wien, 2017. In preparation.

[HMW01] Joseph Y Halpern, Yoram Moses, and Orli Waarts. A characterization of eventual byzantine agreement. *SIAM Journal on Computing*, 31(3):838–865, 2001.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[Lyn96] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[Mic89] Ruben Michel. A categorical approach to distributed systems expressibility and knowledge. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 129–143. ACM, 1989.

[MT86] Yoram Moses and Mark R Tuttle. Programming simultaneous actions using common knowledge: Preliminary version. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 208–221. IEEE, 1986.

Part IV

Annexes

A Proofs

Proof of 5.5. Let $\theta = (i,t) \in A \times \mathbb{N}, t' \geq t, (M, r, t') \models \overline{occurred}_\theta(o)$,

Let us prove $(M, r, t') \models \left(\overline{occurred}_\theta(o) \Rightarrow \neg fake_\theta(o)\right)$:

Case I: $o \in \overline{Actions}$. $o \ \in \sigma\left(\beta_i^{t-1}(r)\right)$

Case 1: $o = internal\,(i,u)$.
By definition of $filter_\epsilon$(Def2.9), we have $fake\,(i,o) \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$.
Case 2: $o = send(j,\mu_k)$.
$send(j,\mu_k)$ and $fake\,(i, gsend(i,j,\mu,t-1))$ will share the same id in the global format due to the labeling function $filter_\epsilon$(Def2.9).
Therefore with $filter_\epsilon$, we have $fake\,(i,o) \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$.

Case II: $o \in \overline{Events}$. $o \in \overline{\beta}_{e_i}^{t-1}(r)$
By definition of $filter_\epsilon$, we have $fake\,(i,o) \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$
By definition of $fake_\theta\,(o)$ we have $(M,r,t') \not\models fake_\theta\,(o)$

Conclusion : $(M,r,t') \models \overline{occurred}_\theta(o) \Rightarrow \neg fake_\theta\,(o)$

Proof of 5.7. Let $t' \geq t, r_i(t) = E : r_i(t-1)$

Case I: $(M,r,t') \models occurred_{(i,t)}(o)$.

Case 1: $(M,r,t') \models \overline{occurred}_{(i,t)}(o)$. ie $o \in label^{-1}\left(\beta_i^{t-1}(r)\right)$ or $o \in label^{-1}\left(\overline{\beta}_{e_i}^{t-1}(r)\right)$
So by τ(Def2.19) we have $o \in E$
Case 2: $(M,r,t') \models fake_{(i,t)}\,(o)$. i.e $fake\,(i,o) \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$
So by τ(Def2.19) we have $o \in E$

Conclusion : $o \in E$

Case II: $o \in E$. By definition of τ

Case 1: $o \in label^{-1}\left(\beta_i^{t-1}(r)\right)$. We have $(M,r,t') \models \overline{occurred}_{(i,t)}(o)$
hence $(M,r,t') \models occurred_{(i,t)}(o)$
Case 2: $o \in label^{-1}\left(\overline{\beta}_{e_i}^{t-1}(r)\right)$. We have $(M,r,t') \models \overline{occurred}_{(i,t)}(o)$
hence $(M,r,t') \models occurred_{(i,t)}(o)$
Case 3: $fake\,(i,o) \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. So by τ we have $(M,r,t') \models fake_{(i,t)}\,(o)$
hence $(M,r,t') \models occurred_{(i,t)}(0)$

Conclusion : $(M,r,t') \models occurred_{(i,t)}(o)$

Proof of 6.6. Let $\theta \in A \times \mathbb{N}, \beta \in V_\theta^r$, let us prove that $\eta \in V_\beta^r \Rightarrow \eta \in V_\theta^r$,
Let $\eta \in V_\theta^r$, $\begin{cases} \beta \in V_\theta^r, \text{ so there is a } \xi_2 : \beta \rightsquigarrow^{\xi_2} \theta \\ \eta \in V_\beta^r, \text{ so there is a } \xi_1 : \eta \rightsquigarrow^{\xi_1} \beta \end{cases}$ hence $\xi : \xi_1.\xi_2$, in G(r), therefore $\eta \in V_\theta^r$

Proof of 6.7. Let a run r, $(\theta_1 = (i,t), \theta_2 = (j,t')) \in (A \times \mathbb{N})^2$

Case I: $\beta \rightarrow_r^{Lb} \theta$. We will reason by induction on the structure of $\beta \rightarrow_r^{Lb} \theta$.

Case 1: Locality.
Case a: $\theta_1 = \theta_2$. ξ is the path of length 0 between θ and θ
Case b: $i = j$ and $t' = t+1$. so by definition $(\theta_1,\theta_2) \in E_{loc}^r$
Case 2: Message relation.
so by definition $(\theta_1,\theta_2) \in E_{msg}^r$
Case 3: Transitivity.
so there is $\eta = (k,t'') \in V^r$ so that $\theta_1 \rightarrow_r^{Lb} \eta \rightarrow_r^{Lb} \theta_2$
Hence by induction :
1.with $\theta_1 \rightarrow_r^{Lb} \eta$: we get $\theta_1 \rightsquigarrow^{\xi_1} \eta$ in $Past(r,\eta) \subset_{\text{with Lem6.6}} Past(r,\theta)$
2.with $\eta \rightarrow_r^{Lb} \theta$: we get $\eta \rightsquigarrow^{\xi_2} \theta_2$ in $Past(r,\theta)$

So $\xi = \xi_1.\xi_2, \beta \rightsquigarrow^{\xi} \theta$ in $Past(r, \theta)$

Conclusion : $\exists \xi, \beta \rightsquigarrow^{\xi} \theta$ in $Past(r, \theta)$

Case II: $\xi : \beta \rightsquigarrow^{\xi} \eta$ in $Past(r, \theta)$. We will reason by induction on the structure of ξ.

Case 1: $\xi = \theta_1 \to \theta_2, (\theta_1, \theta_2) \in E^r_{loc}$. by *Locality* we have $\theta_1 \to^{Lb}_r \theta_2$

Case 2: $\xi = \theta_1 \to \theta_2, (\theta_1, \theta_2) \in E^r_{msg}$. by *Message* we have $\theta_1 \to^{Lb}_r \theta_2$

Case 3: $\xi = \xi_1.\xi_2,$ where $\theta_1 \rightsquigarrow^{\xi_1} \eta, \eta \rightsquigarrow^{\xi_2} \theta_2$. Hence by induction :

1.with $\theta_1 \rightsquigarrow^{\xi_1} \eta$: we get $\theta_1 \to^{Lb}_r \eta$

2.with $\eta \rightsquigarrow^{\xi_1} \theta_2$: we get $\eta \to^{Lb}_r \theta$

So by *Transitivity* we have $\theta_1 \to^{Lb}_r \theta_2$

Conclusion : $\theta_1 \to^{Lb}_r \theta_2$

Proof of 7.5. Let $(r, r') \in R^{X^2}$, r and r' agree on $(i, t) \in A \times \mathbb{N}$

$$\begin{cases} \overline{\beta}^t_{e_i}(r') & =_{2)} \overline{\beta}^t_{e_i}(r') \\ \beta^t_{g_i}(r') & =_{3)} \beta^t_{g_i}(r') \\ \beta^t_i(r') & =_3 \beta^t_i(r') \\ \beta^t_{b_i}(r') & =_{4)} \beta^t_{b_i}(r') \end{cases} \text{ therefore by (2.17) } r_i(t+1) = r'_i(t+1)$$

Proof of 8.1. Let $r \in R^P, \theta = (i, t) \in A \times \mathbb{N}$,
Let us build such a $r' \in R^P$

Part A. Building rules for $\alpha^m_i(r)$ and $\alpha^m_{\epsilon_i}(r)$:

(a) *Init* :
$r'(0) = r(0)$

(b) *Out* for $m \le t$: $\overline{\beta}^m_{e_j}(r') = \begin{cases} \text{If } (j,m) \in V^r_\theta \text{ then } \overline{\beta}^m_{e_j}(r) \\ \text{Else } \varnothing \end{cases}$

(c) *In* for $m \le t$: $\begin{cases} \beta^m_j(r') = \begin{cases} \text{If } (j,m) \in V^r_\theta \text{ then } \beta^m_j(r) \\ \text{Else } \varnothing \end{cases} \\ \beta^m_{g_j}(r') = \begin{cases} \text{If } (j,m) \in V^r_\theta \text{ then } \beta^m_{g_j}(r) \\ \text{Else } \varnothing \end{cases} \end{cases}$

(d) *Fake* for $m \le t$: $\beta^m_{f_j}(r') = \begin{cases} \text{If } (j,m) \in V^r_\theta \text{ then } \beta^m_{b_j}(r) \\ \text{Else } \varnothing \end{cases}$

(e) *Extending* for $m > t$:

Extends the previous partial run r' by iterating the transition function τ.

Part B. Let us prove A. :
Let $(j, m) \in V^r_\theta$, let prove by induction of m that r, r' agree on (j,m)

Case I: $m = 0$. A. holds with *Init*

Case II: $m > 0$. By *Locality* $(j, m-1) \in V^r_\theta$

Let us prove *agreement 1.*
By induction on $(j, m-1) \in V^r_\theta$ we have r and r' agree on $(j, m-1)$.
Then with Rq7.5, $r_j(m) = r'_j(m)$

Let us prove *agreement 2.* $\begin{cases}(j,m)\in V_\theta^r\\ Out\end{cases}$ therefore $\overline{\beta}_{e_j}^m(r)=\overline{\beta}_{e_j}^m(r')$

Let us prove *agreement 3.* $\begin{cases}(j,m)\in V_\theta^r\\ In\end{cases}$ therefore $\beta_j^m(r)=\beta_j^m(r')$

Let us prove *agreement 4.* $\begin{cases}(j,m)\in V_\theta^r\\ Fake\end{cases}$ therefore $\beta_{b_j}^m(r)=\beta_{b_j}^m(r')$

Conclusion : r,r' agree on (j,m)

Part C. Let us prove B. :

Let $(j,m)\notin V_\theta^r, m\le t$

$\begin{cases}In\Rightarrow\beta_{g_j}^m(r')=\varnothing\\ Out\Rightarrow\overline{\beta}_{e_j}^m(r')=\varnothing\\ Fake\Rightarrow\beta_{f_j}^m(r')=\varnothing\end{cases}$ therefore $\beta_{\epsilon_j}^m(r')=\varnothing$

Part D. Let us prove C. :

(a) Let prove $Bad_{V_\theta^r}(r)\subset Bad_{V_\theta^{r'}}(r')$

Case I: $Bad_{V_\theta^r}(r)=\varnothing$. Trivial

Case II: $Bad_{V_\theta^r}(r)\neq\varnothing$.

Let $(j,m)\in Bad_{V_\theta^r}(r)$, $\beta_{b_j}^{m-1}(r)\neq\varnothing$ then with *Fake* $\beta_{b_j}^{m-1}(r')\neq\varnothing$.

Hence $(j,m)\in Bad_{V_\theta^r}(r')=_{\text{with ??}}Bad_{V_\theta^{r'}}(r')$

Conclusion : $Bad_{V_\theta^r}(r)\subset Bad_{V_\theta^{r'}}(r')$

(b) Let prove $Bad_{V_\theta^{r'}}(r')\subset Bad_{V_\theta^r}(r)$ same proof than (a)

Conclusion : $Bad_{V_\theta^{r'}}(r')=Bad_{V_\theta^r}(r)$

Part E. Let us prove D. :

Fake ensures that $\forall m\le t, j\in A, fail(j)\notin\beta_{\epsilon_j}^m(r')$.

Therefore $A\left(Failed_{V_\theta^r}(r',t)\right)=A\left(Bad_{V_\theta^r}(r',t)\right)$

Part F. Let us prove E. :

Let $\eta=(j,m)\in V_\theta^r, 0<m\le t$,

(a) Let prove $(M^P,r,t)\models\overline{occurred}_\eta(o)\Leftrightarrow(M^{r'},t,\models)\overline{occurred}_\eta(o)$

Case I: $(M^P,r,t)\models\overline{occurred}_\eta(o)$.

Case 1: $o\in\overline{Actions}$. i.e $o\in\beta_j^{m-1}(r)$ and $\beta_{g_j}^{m-1}(r)\neq\varnothing$

With *In* : $\begin{cases}o\in\beta_j^{m-1}(r)=\beta_j^{m-1}(r')\\ \beta_{g_j}^{m-1}(r')=\beta_{g_j}^{m-1}(r)\neq\varnothing\end{cases}$

Therefore $(M^{r'},t,\models)\overline{occurred}_\eta(o)$

Case 2: $o\in\overline{Events}$. i.e $o\in\overline{\beta}_{e_j}^{m-1}(r)$

With *Out* : $o\in\overline{\beta}_{e_j}^{m-1}(r)=\overline{\beta}_{e_j}^{m-1}(r')$.

Therefore $(M^{r'},t,\models)\overline{occurred}_\eta(o)$

Case II: $(M^{r'},t,\models)\overline{occurred}_\eta(o)$. Same proof as Case I

(b) Let prove $(M^P,r,t)\models fake_\eta(o)\Leftrightarrow(M^{r'},t,\models)fake(\beta,o)$

Case I: $(M^P, r, t) \models fake_\beta(o)$. i.e $fake(j,o) \in \beta_{b_j}^{m-1}(r)$
With *Fake* : $fake(j,o) \in \beta_{b_j}^{m-1}(r) = \beta_{b_j}^{m-1}(r')$
Therefore $(M^{r'}, t, \models) fake_\eta(o)$
Case II: $(M^{r'}, t, \models) fake_\eta(o)$. Same proof as Case I

Part G. Let us check that there is no conflict between the previous rules and τ(Def 2.19) :
In fact, we will check for $\mathbf{m} \leq \mathbf{t}$(*Extending* ensures this for $m > t$) that :

(a) The rules are reachable form the protocol

- $\exists X \in label(P_\epsilon(m), m) \cup \{\varnothing\}, \beta_\epsilon^m(r') = filter_\epsilon(r'(m-1), X, \alpha_1^m(r'), \ldots, \alpha_n^m(r'))$:

Case I: $(j,m) \in V_\theta^r$.
Let $X_{e_j} = \alpha_{\epsilon_j}^m(r) \backslash \{fail(j)\}$ and $\forall k \in A, X_k = \begin{cases} \text{If } (k,m) \in V_\theta^r \text{ then } \alpha_j^m(r) \\ \text{Else } \varnothing \end{cases}$
We have $X_{e_j} \subset \alpha_{\epsilon_j}^m(r) \cup \{fail(j)\} \in_{r \in R^P} label(P_\epsilon(m), m) \cup \{\varnothing\}$.
Moreover $\alpha_{\epsilon_j}^m(r) \in_{r \in R^P} label(P_\epsilon(m), m) \cup \{\varnothing\}$.
Therefore with (**??**) we have $X_{e_j} \in label(P_\epsilon(m), m) \cup \{\varnothing\}$.

So let $\alpha_{\epsilon_j}^m(r') = X_{e_j}$, and now let us prove than it cross $filter_\epsilon$ according to r.

Case 1: $grecv(j,k,\mu,id) \in \overline{\alpha}_{e_j}^m(r') =_{In} \overline{\alpha}_{e_j}^m(r)$.

Case a: $grecv(j,k,\mu,id) \in \overline{\beta}_{e_j}^m(r)$.

$$\exists m' \leq m, \begin{cases} gsend(k,j,\mu,id) \in \beta_k^{m'}(r) \text{ and } \beta_{g_k}^{m'}(r) \neq \varnothing \\ \text{or} \\ fake(k, gsend(k,j,\mu,id)) \in \beta_{b_k}^{m'}(r) \end{cases}$$

Therefore $(k,m') \in V_\theta^r$ and then with *In* and *Fake* :

$$\begin{cases} gsend(k,j,\mu,id) \in \beta_k^{m'}(r') \text{ and } \beta_{g_k}^{m'}(r') \neq \varnothing \\ \text{or} \\ fake(k, gsend(k,j,\mu,id)) \in \beta_{b_k}^{m'}(r') \end{cases}$$

Hence $grecv(j,k,\mu,id) \in \overline{\beta}_{e_j}^m(r')$

Case b: $grecv(j,k,\mu,id) \notin \overline{\beta}_{e_j}^m(r)$.

$$\nexists m' \leq m, \begin{cases} gsend(k,j,\mu,id) \in \beta_k^{m'}(r) \text{ and } \beta_{g_k}^{m'}(r) \neq \varnothing \\ \text{or} \\ fake(k, gsend(k,j,\mu,id)) \in \beta_{b_k}^{m'}(r) \end{cases}$$

Therefore $(k,m') \notin V_\theta^r$ hence with *In* and *Fake* :

$$\begin{cases} \forall m' \leq m, \beta_k^{m'}(r') = \varnothing \\ \forall m' \leq m, \beta_{\epsilon_k}^{m'}(r') = \varnothing \end{cases}$$

Hence $grecv(j,k,\mu,id) \notin \overline{\beta}_{e_j}^m(r')$

Case 2: $fake(j,u) \in \overline{\alpha}_{e_j}^m(r') =_{In} \overline{\alpha}_{e_j}^m(r)$.
With *In* and *Out* we have :

$$fake(j,u) \in \overline{\beta}_{e_j}^m(r') \Leftrightarrow fake(j,u) \in \overline{\beta}_{e_j}^m(r)$$

Case II: $(j,m) \notin V_\theta^r$. i.e $\alpha_{\epsilon_j}^m(r') =_{m \leq t} \varnothing$

- $\exists X \in label_j\left(P_j\left(r'_j(m-1)\right), m\right) \cup \{\varnothing\}, \beta_j^m(r') = filter_i\left(X, \alpha_{\epsilon_i}^m(r)\right)$:

Let $X_j = \alpha_j^m(r) \in_{r \in R^P} label_j\left(P_j\left(r'_j(m-1)\right), m\right) \cup \{\varnothing\}$

$$\begin{aligned} filter_i\left(X_j, \alpha_{\epsilon_i}^m(r)\right) &= filter_i\left(X_j, \alpha_{\epsilon_i}^m(r) \backslash \{fail(i)\}\right) \\ &= filter_i\left(X_j, \alpha_{\epsilon_i}^m(r')\right) \\ &= \beta_j^m(r') \end{aligned}$$

Part H. Let us check that $r' \in R^{ba}$:

- $\dfrac{r'(0) \in \mathscr{G}(0) :}{r'(0) =_{Init} r(0) \in \mathscr{G}(0)}$

Part I. Let us check that $r' \in R^P$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by Extending

(b) $r \in FS$: ensured by Extending

(c) $r \in MB$: we will check this for $m \leq t$(*Extending* ensures this for $m > t$)

$$\begin{aligned} |A\left(Failed_{V_\theta^{r'}}(r')\right)| &=_{D.} |A\left(Bad_{V_\theta^{r'}}('r)\right)| \\ &=_{C.} |A\left(Bad_{V_\theta^{r}}(r)\right)| \\ &\leq_{r \in R^P} f \end{aligned}$$

Conclusion : $r' \in R^P$ verifies A. B. C. D. E.

Proof of 8.3. Let $r \in R^P, \theta = (i,t) \in A \times \mathbb{N}$,
Let us build $r' \in R^P$ such that $\begin{cases} r'_i(t) = r_i(t) \\ (M^{r'}, t, \not\models)\overline{occurred}(o) \\ (M^{r'}, t, \models)\neg fail_i \end{cases}$ and let $(j,m) \in A \times \mathbb{N}$,

N.B : In the whole proof we will only deal about V_θ^r and ignore all the points outside thanks to Lem8.1.

Part A. $V_\theta^r = NotConcerned \sqcup DMZ \sqcup Barrier_\theta^r(o) \sqcup After$:

$$NotConcerned = \left\{\eta \in V_\theta^r \mid V_\eta^r \cap \Theta_\theta^r(o) = \varnothing\right\} \setminus Barrier_\theta^r(o)$$

$$After = \left\{ \eta \in V_\theta^r \setminus Barrier_\theta^r(o) \mid \begin{cases} \forall \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta, \exists \lambda \in Barrier_\theta^r(o), \beta \mapsto^\xi \lambda \mapsto^\xi \eta \\ \exists \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta \end{cases} \right\}$$

$$DMZ = V_\theta^r \setminus (After \sqcup Barrier_\theta^r(o) \sqcup NotConcerned)$$

(a) First let us prove $V_\theta^r = NotConcerned \sqcup DMZ \sqcup Barrier_\theta^r(o) \sqcup After$ is a disjoint union

$$NotConcerned \cap After = \varnothing \tag{61}$$

Let $x \in NotConcerned$ therefore by (6.4) we have $\nexists \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta$
Hence $x \notin After$

(b) Nodes in $Barrier_\theta^r(o)$ are stable in time, i.e if they are in $Barrier_\theta^r(o)$ then they were in from the beginning and they will belong to this set for eternity.

$$\text{If } (j,m) \in Barrier_\theta^r(o) \text{ then } \forall m', (j,m') \in Barrier_\theta^r(o) \tag{62}$$

By definition of $Barrier_\theta^r(o)$

(c) An agent in $After$ is previously in $After$ or $NotConcerned$

$$\text{If } (j,m) \in After \text{ then } \forall m' \leq m, (j,m') \in After \sqcup NotConcerned \tag{63}$$

Let $(j,m) \in After$

Case I: $(j,m-1) \in NotConcerned$. Trivial
Case II: *otherwise*.
Let ξ a path : $\beta \in \Theta_\theta^r(o) \mapsto (j,m-1)$
Let $\xi^{ext} = \xi \to (j,m)$ therefore $\exists \lambda \in \xi^{ext}, \lambda \in Barrier_\theta^r(o)$.
Hence by (62) we know $\lambda \neq (j,m-1)$
Then $\forall \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta, \exists \lambda \in Barrier_\theta^r(o), \beta \mapsto^\xi \lambda \mapsto^\xi \eta$
Therefore $(j,m-1) \in After$

(d) An agent in $NotConcerned$ is previously in $After$ or $NotConcerned$

$$\text{If } (j,m) \in NotConcerned \text{ then } \forall m' \leq m, (j,m') \in NotConcerned \tag{64}$$

Due to Lem6.6

(e) An agent in DMZ will be in DMZ for eternity

$$\text{If } (j,m) \in DMZ \text{ then } \forall (j,m') \in V_\theta^r, m' \geq m, (j,m') \in DMZ \tag{65}$$

Let $(j,m) \in DMZ \cup After$:
$(j,m+1) \notin_{(64)} NotConcerned$
$(j,m+1) \notin_{(62)} Barrier_\theta^r(o)$
$(j,m+1) \notin_{(63)} After$

(f) $\theta \in After$ by def

(g) $\Theta_\theta^r(o) \subset DMZ \cup Barrier_\theta^r(o)$

Part B. Building rules for $\beta_i^m(r)$ and $\beta_{\epsilon_i}^m(r)$:

(a) *Init* : $r'(0) = r(0)$

(b) *Conservation* for $(j,m) \in After \sqcup NotConcerned$ (i.e $m \leq t$):
$$\begin{cases} \beta_j^m(r') = \beta_j^m(r) \\ \beta_{\epsilon_j}^m(r') = \beta_{\epsilon_j}^m(r') \end{cases}$$

(c) *Barrier* for $(j,m) \in Barrier_\theta^r(o), m \leq t$:
$$\begin{cases} \beta_{b_j}^m(r') & = \beta_{b_j}^m(r) \cup \{fake(j,u) \mid \begin{cases} \text{If } \beta_{g_j}^m(r) = \varnothing \text{ then } u \in \overline{\beta}_{e_j}^m(r) \} \\ \text{else } u \in \beta_j^m(r) \cup \overline{\beta}_{e_j}^m(r) \} \end{cases} \\ \overline{\beta}_{e_j}^m(r') \cup \beta_{g_j}^m(r') & = \varnothing \\ \beta_j^m(r') & = \varnothing \end{cases}$$

(d) *Freeze* for $(j,m) \in DMZ, m \leq t$:
$$\begin{cases} \beta_{\epsilon_j}^m(r') & = \varnothing \\ \beta_j^m(r') & = \varnothing \end{cases}$$

(e) *Extending* for $m > t$:

Extends the previous partial run r' by iterating the transition function τ.

Part C. Let us prove $\forall (j,m) \in NotConcerned \sqcup Barrier_\theta^r(o) \sqcup After, r'_j(m) = r'_j(m)$:
Let $(j,m) \in NotConcerned \sqcup Barrier_\theta^r(o) \sqcup After$, let us do it by induction on m.

Case I: $m = 0$. It holds with *Init*
Case II: $m > 0$.

Case 1: $(j,m) \in NotConcerned$. i.e $(j,m-1) \in_{(64)} NotConcerned$
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \text{r,r' agree on (j,m-1) by } Conservation \end{cases}$
Then by Lem7.5 $r'_j(m) = r_j(m)$
Case 2: $(j,m) \in Barrier^r_\theta(o)$. i.e $(j,m-1) \in_{(62)} Barrier^r_\theta(o)$

Case a: $\beta^m_{\epsilon_j}(r) \subset \{fail(j)\}$. i.e $r_j(m) = r_j(m-1)$
$\beta^m_{\epsilon_j}(r') \subset_{Barrier} \{fail(j)\}$ i.e

$$\begin{aligned} r'_j(m) &= r'_j(m-1) \\ &=_{induction} r_j(m-1) \\ &= r_j(m) \end{aligned}$$

Case b: $\beta^m_{\epsilon_j}(r) \not\subset \{fail(j)\}$ and $\beta^m_{g_j}(r) = \varnothing$.
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \sigma\left(\beta^m_{b_j}(r')\right) =_{Barrier} \sigma\left(\beta^m_{b_j}(r)\right) \cup \overline{\beta}^m_{e_j}(r) \end{cases}$
Then by the transition function(2.19) $r'_j(m) = r_j(m)$
Case c: $\beta^m_{\epsilon_j}(r) \not\subset \{fail(j)\}$ and $\beta^m_{g_j}(r) \neq \varnothing$.
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \sigma\left(\beta^m_{b_j}(r')\right) =_{Barrier} \sigma\left(\beta^m_{b_j}(r)\right) \cup \overline{\beta}^m_{e_j}(r) \cup \beta^r_g(j)m \end{cases}$
Then by the transition function(2.19) $r'_j(m) = r_j(m)$
Case 3: $(j,m) \in After$. i.e $(j,m-1) \in_{(63)} After \sqcup NotConcerned$
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \text{r,r' agree on (j,m-1) by } Conservation \end{cases}$
Then by Lem7.5 $r'_j(m) = r_j(m)$

Part D. Let us prove $\forall (j,m) \in NotConcerned \sqcup Barrier^r_\theta(o) \sqcup After, r'_j(m+1) = r'_j(m+1)$:
By C. we have $r'_j(m) = r'_j(m)$.
Therefore by the same proof than in C.II we have $r'_j(m+1) = r'_j(m+1)$

Part E. Let us prove $\forall (j,m) \in V^r_\theta, (M^{r'}, t, \not\models) \overline{occurred}_{(j,m)}(o)$:
Let $(j,m) \in V^r_\theta$, let do this by induction on m.

Case I: $m = 0$. It holds, nothing happen at time 0.
Case II: $m > 0$.
Case 1: $(j,m-1) \in DMZ$. i.e $r'_j(m) =_{Freeze} r'_j(m-1)$
Therefore $\forall u \in \overline{Actions} \sqcup \overline{Events}, (M^{r'}, t, \not\models) \overline{occurred}_{(j,m)}(u)$
Case 2: $(j,m-1) \in Barrier^r_\theta(o)$.
By *Barrier* we have $\begin{cases} \overline{\beta}^m_{e_j}(r') = \varnothing \\ \beta^m_j(r') = \varnothing \end{cases}$
Therefore $\forall u \in \overline{Actions} \sqcup \overline{Events}, (M^{r'}, t, \not\models) \overline{occurred}_{(j,m)}(u)$
Case 3: $(j,m-1) \in NotConcerned \sqcup After$.
By *Conservation* we have $\begin{cases} \overline{\beta}^m_{e_j}(r') = \overline{\beta}^m_{e_j}(r) \\ \beta^m_j(r') = \beta^m_j(r) \end{cases}$
And by def of $NotConcerned$ (resp $After$) $(M^P, r, t) \not\models \overline{occurred}_{(j,m)}(u)$
Therefore $(M^{r'}, t, \not\models) \overline{occurred}_{(j,m)}(u)$

Part F. Let us prove than $(M^{r'}, t, \models) \neg fail_i$:

By hyp $(M^P, r, t) \models \neg fail_i$ and $\begin{cases} i \notin_{(65)} A(DMZ) \\ i \notin_{def} A(Barrier^r_\theta(o)) \end{cases}$.

Therefore with *Conservation* we have $(M^{r'}, t, \models) \neg fail_i$

Part G. Let us check that there is no conflict between the previous rules and τ(Def 2.19) :

In fact, we will check for $m \leq t$(*Extending* ensures this for $m > t$) that :

(a) The rules are reachable form the protocol

- $\exists X \in label(P_\epsilon(m), m) \cup \{\varnothing\}, \beta^m_\epsilon(r') = filter_\epsilon(r'(m-1), X, \beta^m_1(r'), \ldots, \beta^m_n(r'))$:

  We will do this for each $\beta^{]}_{[}(j) nr'm$

  Case I: $(j, m-1) \in DMZ$. i.e $\beta^m_{\epsilon_j}(r') =_{Freeze} \varnothing$
  Therefore it holds

  Case II: $(j, m-1) \in Barrier^r_\theta(o)$. i.e $\begin{cases} \beta^m_j(r') =_{Barrier} \varnothing \\ \beta^m_{g_j}(r') =_{Barrier} \varnothing \\ \overline{\beta}^m_{e_j}(r') =_{Barrier} \varnothing \end{cases}$

  Therefore $\beta^m_{\epsilon_j}(r') =_{(2.9)} filter_\epsilon\left(r'(m-1), \beta^m_{\epsilon_j}(r'), \beta^m_1(r'), \ldots, \beta^m_n(r')\right)$

  Case III: $(j, m-1) \in NotConcerned$. i.e $\begin{cases} \beta^m_j(r') =_{Conservation} \beta^m_j(r) \\ \beta^m_{\epsilon_j}(r') =_{Conservation} \beta^m_{\epsilon_j}(r') \end{cases}$

  Therefore we only have to prove than if $grecv(j, k, \mu, id) \in \overline{\beta}^m_{e_j}(r')$ then $\exists m' \leq m, \begin{cases} gsend(k, j, \mu, id) \in \beta^{m'}_k(r') \text{ and } \beta^{m'}_{g_k}(r') \neq \varnothing \\ \text{ or } \\ fake(k, gsend(k, j, \mu, id)) \in \beta^{m'}_{b_k}(r') \end{cases}$.

  Let (k,m') such a point in r (exists because $r \in R^P$).
  By def of $NotConcerned$ we have $(k, m') \in NotConcerned$ therefore with D. it holds.

  Case IV: $(j, m-1) \in After$. i.e $\begin{cases} \beta^m_j(r') =_{Conservation} \beta^m_j(r) \\ \beta^m_{\epsilon_j}(r') =_{Conservation} \beta^m_{\epsilon_j}(r') \end{cases}$

  Therefore we only have to prove than if $grecv(j, k, \mu, id) \in \overline{\beta}^m_{e_j}(r')$ then $\exists m' \leq m, \begin{cases} gsend(k, j, \mu, id) \in \beta^{m'}_k(r') \text{ and } \beta^{m'}_{g_k}(r') \neq \varnothing \\ \text{ or } \\ fake(k, gsend(k, j, \mu, id)) \in \beta^{m'}_{b_k}(r') \end{cases}$.

  Let (k,m') such a point in r (exists because $r \in R^P$).

  Case 1: $(k, m') \in Barrier^r_\theta(o)$. With D. it holds
  Case 2: $(k, m') \in After$. With D. it holds
  Case 3: $(k, m') \in Concerned$. With D. it holds
  Case 4: $(k, m') \in DMZ$. This case can not be due to the definition of $After$.

- $\beta^m_j(r') \in label_j\left(P_\epsilon(j) r'_j(m-1), m\right) \cup \{\varnothing\}$:

  Case I: $(j, m-1) \in DMZ$. i.e $\beta^m_j(r') =_{Freeze} \varnothing$
  Therefore it holds
  Case II: $(j, m-1) \in Barrier^r_\theta(o)$. i.e $\beta^m_j(r') =_{Barrier} \varnothing$
  Therefore it holds

Case III: $(j, m-1) \in NotConcerned \sqcup After$.

$$\begin{aligned}\beta_j^m(r') &=_{Conservation} \beta_j^m(r)\\ &\in_{r\in R} label_j(P_\epsilon(j)\, r_j(m-1), m) \cup \{\varnothing\}\\ &\in_{C.} label_j\left(P_\epsilon(j)\, r'_j(m-1), m\right) \cup \{\varnothing\}\end{aligned}$$

(b) The rules pass through *update*(Def 2.17) without any damage

- If $\alpha_g^m(r') = \varnothing$ then $\alpha_j^m(r') = \varnothing$:

Case I: $(j, m) \in Barrier_\theta^r(o) \sqcup DMZ$. i.e $\beta_{g_j}^m(r') = \varnothing$
Therefore it holds
Case II: $(j, m) \in After \sqcup NotConcerned$.
It exactly follows r so we do not care.

Part H. Let us check that $r' \in R^{ba}$:

- $r'(0) \in \mathscr{G}(0)$:

$$\begin{aligned}r'(0) &=_{Init} r(0)\\ &\in_{r\in R} \mathscr{G}(0)\end{aligned}$$

Part I. Let us check that $r' \in R^P[ba]$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by Extending

(b) $r \in FS$: ensured by Extending

(c) $r \in MB$: we will check this for $m \leq t$(*Extending* ensures this for $m > t$)
$A(Failed(r', m)) \subset A(Failed(r, m)) \cup A(Barrier_\theta^r(o))$. It is due to *Conservation* for $A(Failed(r, m))$ and due to *Barrier* for $A(Barrier_\theta^r(o))$
Thanks to Lem8.1 (cf. N.B at the beginning of this proof) we have :

$$A(Failed(r', m)) \subset A\left(Bad_{V_\theta^r}(r, m)\right) \cup A(Barrier_\theta^r(o))$$

Therefore

$$\begin{aligned}|A(Failed(r', m))| &\leq |A\left(Bad_{V_\theta^r}(r, m)\right) \setminus A(Barrier_\theta^r(o))| + |A(Barrier_\theta^r(o))|\\ &\leq f\end{aligned}$$

$$\text{Conclusion : } r' \in R^P \text{ and } \begin{cases} r'_i(t) = r_i(t)\\ (M^{r'}, t, \models) \neg fail_i\\ (M^{r'}, t, \not\models) \overline{occurred}(o)\end{cases}$$

B Draft of the research report

Knowledge in Byzantine Message-Passing System

Laurent Prosperi
ENS Paris-Saclay

Roman Kuznets
ECS, TU Wien

Ulrich Schmid
ECS, TU Wien

August 19, 2017

Abstract

We present an extension of the epistemic runs and systems framework for modeling distributed systems introduced by Fagin et.al. to also incorporate Byzantine agents. Our framework relies on a careful separation of concerns of the various actors involved in the evolution of a message-passing distributed system, and their respective constraints : the agents' protocol, the underlying computation model, and the adversary that controls Byzantine faulty behavior. This modularization allows our framework to cover all existing distributed computing models we are aware of, like lock-step synchronous systems, all variants of partially synchronous systems, asynchronous systems with or without oracles like failure detectors and even timed systems. Albeit Byzantine faulty agents may behave arbitrarily (and may or may not be aware of doing so), our framework allows epistemic reasoning also about such agents (they may have arbitrary knowledge).

We demonstrate the utility of our framework by applying it for identifying necessary and sufficient communication structures for certain distributed computing problems (like variants of clock synchronization) in asynchronous systems with Byzantine faulty agents. We extend pivotal concepts introduced in [Ben-Zvi and Moses, JACM'14] for the fault-free case to the Byzantine setting, and identify ε-pedes as a crucial causal structure.

Contents

Chapter 1

The Byzantine Message-Passing Framework

1.1 Agents and states

We consider distributed multi-agent systems with agents viewed as processes or humans.

Definition 1.1.1 (Agents)**.** We consider a non-empty finite set $\mathcal{A}$ of **agents**. Without loss of generality, we assume that $\mathcal{A} = [\![1; n]\!]$ for some positive integer n.[1] **Local timestamps**, or simply **points**, are identified by pairs $(i, t) \in \mathcal{A} \times \mathbb{N}$ of an agent and a timestamp. 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 exlusively for snapshots of the system's state. All actions are performed in the open intervals in between: $]0; 1[$, $]1; 2[$, $]2; 3[$, $\ldots$.[2] Conceptually, we will assume that they occur at 0.5, 1.5, 2.5, ...

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

Definition 1.1.2 (Initial states)**.** We denote by Σ_i the set of **local initial states** of agent $i \in \mathcal{A}$. A **joint initial state**, or **global initial state**, is a tuple of local initial states from the set $\mathscr{G}(0) = \prod_{i \in \mathcal{A}} \Sigma_i$.

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 $\mathcal{A}$. We do not assume agents to be necessarily of the same type. Hence, their sets of initial states, available internal actions, and potential external events can differ. An example of a local internal action could be incrementing a local counter. An example of a local external event could be the input from a motion-detector sensor.

Definition 1.1.3 (Local internal actions)**.** We denote by Int_i the set of **local internal actions** of agent $i \in \mathcal{A}$. Local internal actions are denoted a, a', a_1 etc. We also distinguish one action available to all agents: $\clock \in Int_i$ for all $i \in \mathcal{A}$. The action $\clock$ is called **tick**.

Definition 1.1.4 (Local external events)**.** We denote by Ext_i the set of **local external events** the agent $i \in \mathcal{A}$ can be subjected to. Local external events are denoted e, e', e_1 etc.

We also use $o, o', o_1, \ldots, u, u', u_1, \ldots$ when unsure whether it is an event or an action.

We consider a message-passing system whereby agents communicate exclusively by messages.

Definition 1.1.5 (Messages)**.** We denote by *Msgs* the (possibly infinite) set of **messages** that agents can send to each other. Let $i, j \in \mathcal{A}$ be two agents. A message $\mu \in Msgs$ can be

[1] We use the notation $[\![k; m]\!]$ to denote the set of integer numbers 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., $\{x \in \mathbb{R} \mid k < x < m\}$.

- sent by agent i to agent j (possibly in multiple copies), which constitutes an internal action of i and is recorded in agent i's history as $send(j, \mu_k)$ for the copy k of the message; we consider $send(j, \mu_0)$ to be the master copy and in protocols where multiple copies are not necessary denote it simply by $send(j, \mu)$;
- received by agent j from agent i, which consitutes an external event for j and is recorded in agent j's history as $recv(i, \mu)$ (note that the receiving agent does not know which copy of the message it received).

We denote by

$$\Omega(Msgs) = \{send(j, \mu_k), \quad recv(j, \mu) \mid j \in \mathcal{A}, \mu \in Msgs, k \in \mathbb{N}\}$$

the set of all possible *send* and *recv* occurrences.

Remark 1.1.6 (Global ids and global view)**.** Having multiple copies of the same message serves only one purpose: to enable sending several duplicates of the same message at the same time. Generally, these copies need not be used and, conversely, their use does not have any effect on the sending agent's behavior. In particular, there is no obligation to use consecutive numbers for copies and there is no obligation to always use a fresh number of copy in the next rounds. Thus, despite having a possibility to distinguish all sent messages, agents are under no obligation to do so.

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, which also plays the role of the delivery system, must, however, be able to distinguish between 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 track the 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 happening to i and the same event e happening to j. Thus, the environment represents copy k of a message μ sent from i to j and assigned GMI id in the format $gsend(i, j, \mu, id)$ and $grecv(j, i, \mu, id)$ with the information of the copy number k transferred to the GMI id, action a' by agent i in the format $A' = internal\,(i, a)$, event e' hapenning to i in the format $E' = external\,(i, e)$. We will refer to this as a *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 generally assume that a'' and A'' or e_{13} and E_{13}, etc. represent the same action/event presented locally and globally respectively.

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 of the run. This is implemented by letting agents sleep through one or more rounds. For each round, the environment controls whether an agent will be awoken to act according to its protocol or to observe some external events or whether the agent will skip the round altogether.

The waking of agent i is performed by the event $go(i)$ initiated by the environment. Such an event leads to agent i performing internal actions prescribed by its protocol, nondeterministically chosen among all possible reactions to i's local state. Even when the protocol prescribes no actions at all, the agent would still know that another round has passed. By contrast, the agent is always aware of any external events, including received messages, imposed on it by the environment: even without $go(i)$ agent i will know of any round in which at least one external event from Ext_i

happened to it. However, an agent not woken up by $go(i)$ and not disturbed by external events would not change its state and would not be aware of the passed round.[3]

Thus, we are making a distinction between the agent *actively doing nothing* (observing the round passing without any action) and *passively not doing anything* (not noticing the passing of the round). In order to make this distinction formal, we treat the set of actions $\{\clubsuit\}$ as a possible formal representation of doing nothing actively: $\clubsuit$ is the action of turning the clock forward. This representation will be shown to be distinct from doing nothing passively both for the environment and for the agent itself.

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 to send a message, this action is thwarted by the environment not waking the agent up by a $go(i)$ command. Additionally, the arrival of a message should not happen unless the message was in fact sent earlier (i.e., no later than in the same round). This causally motivated restriction on the model will be implemented by a special filter (see Def. 1.2.12), which 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.

Remark 1.1.9 (Modelling Byzantine messages)**.** 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. In particular, Byzantine agents can freely choose the contents and recipients of their messages. However, the transfer of such a Byzantine message to its intended recipient is controlled by the environment and follows the standard delivery procedures. Hence, it is not the Byzantine sender but the environment that determines the GMI of such messages. This, in particular, means 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.

Remark 1.1.10 (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. The agent is unable to distinguish between itself performing some action a and the environment imposing the action a on 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.

Remark 1.1.11 (How environment communicates)**.** Our model does not provide for message passing between agents 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. On the other hand, if information has to be delivered to an agent from the environment, this is better represented by an external event rather than a message.

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

Definition 1.1.12 (Global message identifier function)**.** We fix a function for computing GMIs as any computable one-to-one total function $id\colon \mathcal{A} \times \mathcal{A} \times \mathit{Msgs} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[4]

Note that the function retrieving the arguments t and k from $id(i, j, \mu, k, t)$ 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 functions computable efficiently.

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

[4] 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.

Definition 1.1.13 (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. 1.1.5 and local internal actions $a \in Int_i$:

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

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

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

We also define the sets of (correct) internal actions available to all agents:

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

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

Remark 1.1.14. 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 perfomed by several agents, the representations $internal\,(i, a) \neq internal\,(j, a)$ of the action for these agents $i \neq j$ are distinct.

Definition 1.1.15 (External events)**.** From the point of view off agent $i \in A$, **correct external events** that it can observe consist of the *recv* actions from Def. 1.1.5 and local external events $e \in Ext_i$:

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

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 internal action or correct external event $U \in \overline{GActions}_i \sqcup \overline{GEvents}_i$ of agent i, there is a matching **Byzantine external event**

$$fake\,(i, U).$$

Thus, for received fake messages, the environment creates the message "out of thin air" as if it has been delivered with a particular GMI. Similarly, a fake sent message is created already with a GMI, unlike the correctly sent messages, which are supplied with a GMI in a separate step after creation. However, the delivery of messages with GMI does not depend on whether they originate as fake or correct ones.

In addition, the environment can trigger two more events that control i's behavior:

- $go(i)$ approves i's actions prescribed by its protocol and
- $fail(i)$ marks i as Byzantine from this moment on.

Both types of events are not (directly[5]) observable by agents and, hence, are not part of $\overline{Events}_i$.

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

$$GEvents_i = \overline{GEvents}_i \sqcup \{fake\,(i, U) \mid U \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} \sqcup \{go(i), fail(i)\} \tag{1.7}$$

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

$$\overline{Events} = \bigcup_{i \in \mathcal{A}} \overline{Events}_i, \qquad \overline{GEvents} = \bigsqcup_{i \in \mathcal{A}} \overline{GEvents}_i, \qquad GEvents = \bigsqcup_{i \in \mathcal{A}} GEvents_i \tag{1.8}$$

[5]Events $go(i)$ can sometimes be detected by the acting agent based on the fact that it is acting.

Remark 1.1.16 (Modelling Byzantine inaction)**.** One of the ways to record Byzantine inaction is by using $fail\,(i)$.

However, one subtle case of Byzantine failure through inaction is implemented with the help of ☃: when an agent does not perform any prescribed actions but nevertheless marks the passing of time. The difficulty here is that the agent is woken up either by its own action or by an external event, and either can be problematic if the agent is not supposed to do anything. Fortunately, we have a non-action action ☃, and hence, can use it to mark a correct time-observing inaction and use $fake\,(i, internal\,(i,☃))$ to mark Byzantine time-observing inaction on the part of the agent.

Remark 1.1.17. Although all external events are modelled as triggered by the environment, there is a significant difference between $grecv(i, j, \mu, id)$ and $fake\,(i, grecv(i, j, \mu, id))$, both visible for agent i as simply $recv(j, \mu)$. The former can only be triggered if μ was actually sent from j to i (rightfully or in a Byzantine way), whereas the latter can always be triggered and is understood as the agent only pretending to receive a message. There is little formal difference between $external\,(i, e)$ and $fake\,(i, external\,(i, e))$: both are imposed by the environment and visible as e only to agent i, who cannot distinguish between them. But the underlying intuition is that the former represents the agent actually observing e, whereas the latter has the agent pretending to observe e.

Remark 1.1.18. 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.

The formal model is closer to the latter case, where the malfunctions are imposed by the environment. However, the former case remain faithfully represented as the environment is free to implement any malicious intent by the agent. On the other hand, this exculpating view of agents facilitates modeling their epistemic states. In particular, the knowledge of a malicious agent about its actions is equated to the knowledge of an agent who performed the same actions due to malfunctions.

We consider agents with perfect recall. Hence, the state of an agent is defined as its local *history*:

Definition 1.1.19 (Agent's history)**.** A **history** h **of agent** $i \in \mathcal{A}$, or its **local state**, over initial states Σ_i, local internal actions Int_i, local external events Ext_i, and messages $Msgs$ is a non-empty sequence

$$h = [\lambda_m, \ldots, \lambda_1, \lambda_0]$$

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 and denoted $|h|$. We say that a set $\lambda \subset \overline{Actions_i} \sqcup \overline{Events_i}$ **happens** in the history h of agent i and write $\lambda \subset h$ iff $\lambda = \lambda_j$ for some $j \in [\![1; m]\!]$. We say that $o \in \overline{Actions_i} \sqcup \overline{Events_i}$ **happens** in the history h and write $o \in h$ iff $o \in \lambda$ for some set $\lambda \subset h$.

Definition 1.1.20 (Environment's history)**.** A **history** h **of the system**, or the **global state**, for agents $\mathcal{A} = [\![1; n]\!]$ over the global initial states $\mathscr{G}(0) = \prod_{i\in\mathcal{A}} \Sigma_i$, local internal actions Int_i for each agent $i \in \mathcal{A}$, local external events Ext_i for each agent $i \in \mathcal{A}$, and messages $Msgs$ is a tuple

$$h = (h_\epsilon, h_1, \ldots, h_n)$$

where the history of the environment is a (possibly empty) sequence

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

such that $\forall j \in [\![1; m]\!]$ we have $\Lambda_j \subseteq \overline{GActions} \sqcup GEvents$ and h_i is a local state of agent $i \in A$ over Σ_i, Int_i, Ext_i and $Msgs$. In this case m is called the **length of history** h and denoted $|h|$. 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 $\Lambda \subset h_\epsilon$.

Since h_i's are intended to be local histories that are also reflected in h_ϵ, there are various properties one can expect from h. For instance, since agents can skip a round but cannot perform a round of actions unbeknownst to the environment, we should require $|h_i| \leq |h_\epsilon|$. Similarly, any action occurring locally in h_i should also occur, in its global form, in h_ϵ. These consistency properties will be eventually ensured, but enforcing them now is a bit premature as they they are closely related to the way systems transition from one state to another, which will be discussed in the next section.

Definition 1.1.21 (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 over Σ_i, Int_i, Ext_i, $Msgs$. $\mathscr{L} = \prod_{i \in \mathcal{A}} \mathscr{L}_i$ 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, $\mu \in Msgs$ is a message, and $id \in \mathbb{N}$ is a GMI.

Correct internal actions and their Byzantine copies:

- $a \in Int_i$ 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;
  - in i's local history, this means that agent i performed a, but does not necessarily know whether it was performed according to the protocol or in a Byzantine fashion.
- $internal\,(i, a)$ for $a \in Int_i$ represents the following:
  - in the environment's history, this means that i performed a according to i's protocol.
- $fake\,(i, internal\,(i, a))$ for $a \in Int_i$ 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);
  - in the environment's history, this means that i performed a in a Byzantine fashion.

Sending messages:

- $send(j, \mu_k)$ represents the following:
  - in i's protocol, this prescribes agent i to to send copy k of 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;
  - in i's local history, this means that agent i sent copy k of message μ to j, but does not necessarily know 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 sent message μ to j according to i's protocol, and the message was assigned the GMI id (which contains information about the copy number).

- $fake\,(i, gsend(i, j, \mu, id))$ represents the following:
  - in the environment's protocol, this prescribes agent i to send 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;
  - in the environment's history, this means that i sent copy μ to j in a Byzantine fashion and the message was assigned the GMI id (which contains information about the copy number).

Correct external events and their Byzantine copies:

- $e \in Ext_i$ represents the following:
  - in i's local history, it means that i behaves as if e happened, but does not necessarily know whether it really happened or was a Byzantine event;

- $external\,(i, e)$ for $e \in Ext_i$ represents the following:
  - in the environment's protocol, it prescribes the environment to impose e on agent i;
  - in the environment's history, it means that i really observed e happening.

- $fake\,(i, external\,(i, e))$ for $e \in Ext_i$ represents the following:
  - in the environment's protocol, it prescribes agent i to pretend e happened in a Byzantine fashion;
  - in the environment's history, it means that i pretended to observe e in a Byzantine fashion.

Receiving messages:

- $recv(j, \mu)$ represents the following:
  - in i's local history, it means that i behaves as if it received message μ from agent j, but does not necessarily know whether the receipt of the message really happened or was a Byzantine event.

- $grecv(i, j, \mu, id)$ represents the following:
  - in the environment's protocol, it prescribes to deliver message μ with GMI id from j to i;
  - in the environment's history, it means that message μ with GMI id was really delivered from j to i.

- $fake\,(i, grecv(i, j, \mu, id))$ represents the following:
  - in the environment's protocol, it prescribes agent i to pretend that message μ was received from j in a Byzantine fashion (here the GMI id is not visible to the agent and, hence, plays no role; it is kept to keep notation more uniform);
  - in the environment's history, it means that i pretended to receive μ from j in a Byzantine fashion (the GMI id is not visible to the agent and, hence, plays no role; it is kept to keep notation more uniform).

Environment controlling agents' actions:

- $go(i)$ represents the following:
  - in the environment's protocol, it prescribes agent i to wake up and perform the actions prescribed by i's protocol;
  - in the environment's history, it means that i was woken up and performed the actions prescribed by i's protocol.
- $fail(i)$ represents the following:
  - in the environment's protocol, it prescribes to brand agent i Byzantine irrespective of any actions it has performed;
  - in the environment's history, it means that i was branded Byzantine irrespective of any actions it has performed.

1.2 Transition relation

As discussed at the end of the preceding section, there are multiple consistency restrictions to be imposed on the histories to ensure that information from one of h_i, incomplete as it might be, does not contradict what is recorded 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, and the responsible part is the agent's protocol. The environment plays a triple role. Firstly, it is the 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: the protocol of the environment does not restrict which events and in which combinations can occur beyond basic coherency checks such as ensuring that the GMI's have correct timestamps or beyond faithfully describing the underlying distributed system. An example of the latter case is, for instance, modelling of broadcast systems where the protocol of the environment would allow sending the same message either to all or to none, but not to some. Thirdly, part of the environment that performs non-deterministic choice is designated the *adversary*. It is by rigging this part to choose the worst reactions by both agents and the environment that we will be achieving the worst-case scenarios.

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, 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 sending 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.

Definition 1.2.1 (Protocol)**.** Given the set $\mathcal{A} = [\![1; n]\!]$ of agents, agent $i \in \mathcal{A}$, and timestamp $t \in \mathbb{N}$:

1. A (non-deterministic) **protocol for agent** i over the set of initial states Σ_i, set of internal actions Int_i, set of external events Ext_i, and set of messages $Msgs$ of the agents is any function
$$P_i \colon \mathscr{L}_i \to 2^{2^{\overline{Actions_i}}} \setminus \{\varnothing\}$$
If $h \in \mathscr{L}_i$ is a local state of agent i, then each member $S \in P_i(h)$ is a subset of $\overline{Actions_i}$ and represents one of non-deterministic choices prescribing how i should act in this local state. Note that $P_i(h) \neq \varnothing$.

2. Given individual agents' protocols $P_1, \ldots, P_n$ over their respective sets of initial states Σ_i, sets of internal actions Int_i, sets of external events Ext_i, and set of messages $Msgs$, their **joint protocol** is a function of global states returning a tuple of agent's protocols: for a global state $h = (h_\epsilon, h_1, \ldots, h_n)$ we define

$$P(h) = \big(P_1(h_1), \ldots, P_n(h_n)\big)$$

3. A (non-deterministic) **protocol for the environment** over the sets of internal actions Int_i, sets of external events Ext_i, and set of messages $Msgs$ is any function

$$P_\epsilon \colon \mathbb{N} \longrightarrow 2^{2^{GEvents}} \setminus \{\varnothing\}. \tag{1.9}$$

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

$$\begin{aligned} S \quad \subset \quad & \{grecv(i,j,\mu,id) \mid i,j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \quad \sqcup \quad \{go(i) \mid i \in \mathcal{A}\} \quad \sqcup \\ & \{external(i,e) \mid i \in \mathcal{A}, e \in Ext_i\} \quad \sqcup \quad \{fail(i) \mid i \in \mathcal{A}\} \quad \sqcup \\ & \{fake(i,u) \mid i \in \mathcal{A}, u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\}, \end{aligned}$$

and represents one of the non-deterministic possibilities for what can happen in the system at time t.5. Note that $P_\epsilon(t) \neq \varnothing$.

Definition 1.2.2. For $\sigma \in \{0,1\}$ and a set X we define

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

Definition 1.2.3. We denote by $\mathscr{C}$ the set of all joint protocols.

Definition 1.2.4. We denote by $\mathscr{C}_\epsilon$ the set of all environment protocols such that $P_\epsilon \in \mathscr{C}_\epsilon$ iff for each $t \in \mathbb{N}$ the following condition holds

$$\begin{aligned} X \ \in P_\epsilon(t) \quad \Longrightarrow \quad & \Big(\forall \sigma_1 \in \{0,1\} \ldots \ \forall \sigma_n \in \{0,1\} \\ & \forall Y_1 \subset \{fail(1)\} \sqcup \big\{fake(1,U) | U \in \overline{GActions}_1 \sqcup \overline{GEvents}_1\big\} \ldots \\ & \forall Y_n \subset \{fail(n)\} \sqcup \big\{fake(n,U) | U \in \overline{GActions}_n \sqcup \overline{GEvents}_n\big\}\Big) \\ & \big((X \cap \overline{GEvents}_1)^{\sigma_1} \sqcup \cdots \sqcup (X \cap \overline{GEvents}_n)^{\sigma_n} \sqcup Y_1 \sqcup \cdots \sqcup Y_n\big) \in P_\epsilon(t) \end{aligned} \tag{1.10}$$

In other words, whichever correct events can happen to an agent at a particular time, the environment can squash them all and additionally is completely free in assigning fake actions/inaction to this agent. In doing so, agents can be treated independently of each other.

It is easy to see that $\varnothing \in P_\epsilon(t)$ for every $t \in \mathbb{N}$ and any $P_\epsilon \in \mathscr{C}_\epsilon$.

Remark 1.2.5 (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 communication server maintenance.

Remark 1.2.6 (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., complete with the GMI, from the very beginning. In fact, correct receive events must contain a GMI to determine whether a message with such 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 the GMI, but the message is propagated to the recipient by the environment. Thus, when a woken up agent sends a message, it must first be transformed from the local to the global view. This task is performed by the *labelling 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 1.2.7 (Labeling functions)**.** For the set $\mathcal{A} = [\![1; n]\!]$ of agents, an agent $i \in \mathcal{A}$, and timestamp $t \in \mathbb{N}$, 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:

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

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 by a separate function. Thus, this function does not deal with fake events.

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

as follows:

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

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

Remark 1.2.8. The injectivity of the function $id(\cdot)$ used by $label_i$ ensures that each message is unique from the point of view of the environment.

Remark 1.2.9. The second clause in the definition $label^{-1}\,(U)$ is mostly cosmetic: we make GMI 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 1.2.10 (Non-deterministic choice for protocols)**.** For the set $\mathcal{A} = [\![1; n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and protocols P_ϵ for the environment and $P_1, \ldots, P_n$ 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 set

$$\alpha_\epsilon^t = X_\epsilon \tag{1.11}$$

for some set $X_\epsilon \in P_\epsilon\,(t)$ non-deterministically chosen by the adversary. (Recall that $\varnothing$ can always be chosen.)

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

$$\alpha_i^{h,t} = \ label_i\left(X_i, t\right) \tag{1.12}$$

for some set $X_i \in P_i\left(h_i\right)$ non-deterministically chosen by the adversary. (Recall that $\{\hat{\mathbb{U}}\}$ can always be chosen.)

3. All these choices are combined in

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

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

1. *Regular events* for agent i:

$$\overline{\alpha}_{\epsilon_i}^t = \alpha_\epsilon^t \cap \overline{GEvents}_i = \{grecv(i,j,\mu,id) \in \alpha_\epsilon^t \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external\left(i,e\right) \in \alpha_\epsilon^t \mid e \in Ext_i\} \tag{1.13}$$

2. Instruction to wake agent i:

$$\alpha_{g_i}^t = \alpha_\epsilon^t \cap \{go(i)\} \tag{1.14}$$

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

$$\begin{aligned}\alpha_{b_i}^t = \{fake\left(i,U\right) \in \alpha_\epsilon^t \mid U \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} = \\ \{fake\left(i, gsend(i,j,\mu,id)\right) \in \alpha_\epsilon^t \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake\left(i, grecv(i,j,\mu,id)\right) \in \alpha_\epsilon^t \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \\ \{fake\left(i, internal\left(i,a\right)\right) \in \alpha_\epsilon^t \mid a \in Int_i\} \sqcup \{fake\left(i, external\left(i,e\right)\right) \in \alpha_\epsilon^t \mid e \in Ext_i\}\end{aligned} \tag{1.15}$$

4. Instructions making agent i Byzantine:

$$\alpha_{f_i}^t = \alpha_{b_i}^t \sqcup \left(\alpha_\epsilon^t \cap \{fail\left(i\right)\}\right) \tag{1.16}$$

Finally, we define

$$\overline{\alpha}_\epsilon^t = \bigsqcup_{i \in A} \overline{\alpha}_{\epsilon_i}^t \qquad \alpha_g^t = \bigsqcup_{i \in A} \alpha_{g_i}^t \qquad \alpha_b^t = \bigsqcup_{i \in A} \alpha_{b_i}^t \qquad \alpha_f^t = \bigsqcup_{i \in A} \alpha_{f_i}^t$$

It is easy to see that

Lemma 1.2.11. *For the set of agents agent $\mathcal{A} = [\![1;n]\!]$, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, 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*

$$\alpha_\epsilon^t \subset GEvents \qquad \textit{and} \qquad \alpha_i^{h,t} \subset \overline{GActions}_i$$

To make the environment's protocol independent of the global history, which is crucial for many proofs, we had to sacrifice the ability to weed out actions that are "physically" impossible. This comprises the following types of events:

- a message delivered without being previously[6] sent;
- faking a receipt of a message that was actually received from the same source in the same round;

[6]Here previously means in one of the preceding rounds or in the same round, correctly or in a Byzantine fashion.

- faking an external event that actually happened in the same round;
- faking sending a message that was actually sent to the same destination in the same round;
- faking an internal action that was actually performed in the same round;
- faking a GMI;
- breaking the joint protocol by inaction without failing.

The protocol P_ϵ cannot check many of these conditions as they depend on the global history and actions of agents. Thus, in our model we first allow these events in the P_ϵ sets but immediately neutralize their effect using the following filter:

Definition 1.2.12 (Event and action filter functions)**.** Let $A \in \overline{GActions}$ and $E \in GEvents$. We define an **event filter function** for Byzantine agents

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

as follows. Given a global history, a set of events attempted by the environment (chosen by the adversary) and sets of actions to be performed by the agents (also chosen by the adversary), the function returns the set of events to be performed by the environment, those attempted that are "physically" possible. Formally, for the set $\mathcal{A} = [\![1; n]\!]$ of agents, a set $X \subset GEvents$ of events attempted by the environment, sets $X_i \subset \overline{GActions}_i$ of actions to be performed by each agent $i \in \mathcal{A}$, and a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, we define

$$\begin{aligned} filter_\epsilon^B(h, X_\epsilon, X_1, \ldots, X_n) \quad = \quad & X_\epsilon \quad \setminus \\ \Big(\big\{ grecv(j,i,\mu,id) \quad | \quad & gsend(i,j,\mu,id) \notin h_\epsilon \quad \wedge \quad fake\,(i, gsend(i,j,\mu,id)) \notin h_\epsilon \quad \wedge \\ & (gsend(i,j,\mu,id) \notin X_i \vee go(i) \notin X_\epsilon) \quad \wedge \quad fake\,(i, gsend(i,j,\mu,id)) \notin X_\epsilon \big\} \quad \sqcup \\ \big\{ fake\,(i,A) \quad | \quad & A \in X_i \quad \wedge \quad go(i) \in X_\epsilon \big\} \quad \sqcup \quad \big\{ fake\,(i,E) \quad | \quad E \in X_\epsilon \big\} \quad \sqcup \\ & \big\{ fake\,(i, gsend(i,j,\mu,id)) \quad | \quad (\forall k \in \mathbb{N}) id \neq id(i,j,\mu,k,|h|) \big\} \quad \sqcup \\ & \big\{ fake\,(i, grecv(j,i,\mu,id)) \quad | \quad (\exists id' \in \mathbb{N}) grecv(j,i,\mu,id') \in X_\epsilon \big\} \Big) \end{aligned} \tag{1.17}$$

In addition we define **action filter functions** for Byzantine agents $\mathcal{A} = [\![1; n]\!]$

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

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

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

It is obvious from the definition that

$$filter_\epsilon^B(h, X_\epsilon, X_1, \ldots, X_n) \subset X_\epsilon \tag{1.18}$$

$$filter_i^B(X_1, \ldots, X_n, X_\epsilon) \subset X_i \tag{1.19}$$

Thus, after protocols $P_\epsilon(t)$ and $P_i(h_i)$ provided a range of possible event/action collections 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 be performed in reality. The resulting sets are called β-sets by analogy with α-sets.

Remark 1.2.13. In this definition, we could directly define a local version of the **action filter function** $filter_i^B : 2^{\overline{GActions}_i} \times 2^{GEvents} \longrightarrow 2^{\overline{GActions}_i}$ where we disregard the choices of other agents. But we will need the definition of $filter_i$ as an $(n+1)$-ary function to refine the model to implement, for example, *rendez-vous communication*.

Definition 1.2.14. For the set $\mathcal{A} = [\![1; n]\!]$ of agents, a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha_\epsilon^t, \alpha_1^{h,t}, \ldots, \alpha_n^{h,t})$, agent $i \in \mathcal{A}$, and timestamp $t \in \mathbb{N}$,

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:

$$\overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} = \beta_\epsilon^{h,\alpha^{h,t}} \cap \overline{GEvents}_i = \{grecv(i,j,\mu,id) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{external\,(i,e) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid e \in Ext_i\} \subset \overline{\alpha}_{\epsilon_i}^t \quad (1.20)$$

2. Instruction to wake agent i:

$$\beta_{g_i}^{h,\alpha^{h,t}} = \beta_\epsilon^{h,\alpha^{h,t}} \cap \{go(i)\} = \alpha_{g_i}^t \quad (1.21)$$

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

$$\beta_{b_i}^{h,\alpha^{h,t}} = \{fake\,(i,U) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid U \in \overline{GActions}_i \sqcup \overline{GEvents}_i\} = \{fake\,(i, gsend(i,j,\mu,id)) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{fake\,(i, internal\,(i,a)) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid a \in Int_i\} \sqcup \{fake\,(i, grecv(i,j,\mu,id)) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid j \in \mathcal{A}, \mu \in Msgs, id \in \mathbb{N}\} \sqcup \{fake\,(i, external\,(i,e)) \in \beta_\epsilon^{h,\alpha^{h,t}} \mid e \in Ext_i\} \subset \alpha_{b_i}^t \quad (1.22)$$

4. Instructions making agent i Byzantine:

$$\beta_{f_i}^{h,\alpha^{h,t}} = \beta_{b_i}^{h,\alpha^{h,t}} \sqcup \left(\beta_\epsilon^{h,\alpha^{h,t}} \cap \{fail\,(i)\}\right) \subset \alpha_{f_i}^t \quad (1.23)$$

Finally, we define

$$\overline{\beta}_\epsilon^{h,\alpha^{h,t}} = \bigsqcup_{i \in A} \overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} \qquad \beta_g^{h,\alpha^{h,t}} = \bigsqcup_{i \in A} \beta_{g_i}^{h,\alpha^{h,t}}$$
$$\beta_b^{h,\alpha^{h,t}} = \bigsqcup_{i \in A} \beta_{b_i}^{h,\alpha^{h,t}} \qquad \beta_f^{h,\alpha^{h,t}} = \bigsqcup_{i \in A} \beta_{f_i}^{h,\alpha^{h,t}}$$
$$\beta_{\epsilon_i}^{h,\alpha^{h,t}} = \overline{\beta}_{\epsilon_i}^{h,\alpha^{h,t}} \sqcup \beta_{g_i}^{h,\alpha^{h,t}} \sqcup \beta_{f_i}^{h,\alpha^{h,t}}$$

Remark 1.2.15. 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 $\beta_i^{h,\alpha^{h,t}} = filter_i \left(\alpha_1^{h,t}, \ldots, \alpha_n^{h,t}, \beta_\epsilon^{h,\alpha^{h,t}}\right)$. 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 1.2.16. Consider a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha^t_\epsilon, \alpha^{h,t}_1, \ldots, \alpha^{h,t}_n)$, $i \in \mathcal{A}$, timestamp $t \in \mathbb{N}$. Then

$$\beta^{h,\alpha^{h,t}}_\epsilon = \overline{\beta}^{h,\alpha^{h,t}}_\epsilon \sqcup \beta^{h,\alpha^{h,t}}_g \sqcup \beta^{h,\alpha^{h,t}}_f.$$

Remark 1.2.17. Consider a global history $h \in \mathscr{G}$, an agent $i \in \mathcal{A}$ and timestamp $t \in \mathbb{N}$. Then actions filtering function $filter_i$ for agent i ensures that

$$\beta^{h,\alpha^{h,t}}_i \neq \varnothing \qquad \Longrightarrow \qquad \beta^{h,\alpha^{h,t}}_{g_i} \neq \varnothing$$

Lemma 1.2.18. *For the set $\mathcal{A} = [\![1; n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, a tuple of requested actions and events $\alpha^{h,t} = (\alpha^t_\epsilon, \alpha^{h,t}_1, \ldots, \alpha^{h,t}_n)$, for agent $i \in \mathcal{A}$ and for timestamp $t \in \mathbb{N}$ we have that*

$$\beta^{h,\alpha^{h,t}}_\epsilon \subset GEvents \qquad \textit{and} \qquad \beta^{h,\alpha^{h,t}}_i \subset \overline{GActions}_i$$

Each agent initially starts off in a correct state and may become Byzantine either by violating its protocol or by the designation of the environment. Thus, it is more precise to talk about Byzantine states of agents, about Byzantine local timestamps $(i,t) \in \mathcal{A} \times \mathbb{N}$ instead of announcing the agents themselves to be universally Byzantine. Local timestamps that are in direct violation of the protocol are called *Bad*. All local timestamps of an agent from the first *Bad* local timestamp or from the round the agent is designated Byzantine by the environment are called *Failed*. Recall that time in global histories h is represented by $|h|$.

Definition 1.2.19. For the set of agents $\mathcal{A} = [\![1; n]\!]$, consider a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ of length $|h| = M$, so that $h_\epsilon = [\Lambda_M, \ldots, \Lambda_1]$. We define the sets of *Bad* and *Failed* local timestamps

$$\begin{aligned} Bad\,(h) &= \big\{(i,t) \in \mathcal{A} \times \mathbb{N} \mid fail\,(i) \in \Lambda_t \text{ or } \big(\exists U \in \overline{GActions}_i \sqcup \overline{GEvents}_i\big) fake\,(i,U) \in \Lambda_t \ \big\} \\ Failed\,(h) &= \big\{(i,t) \in \mathcal{A} \times \mathbb{N} \mid \exists t' \leq t \,\big((i,t') \in Bad\,(h)\big)\big\} \end{aligned}$$

Remark 1.2.20 (Relationship between $Failed$ and Bad)**.** For any global history h, $Bad\,(h) \subset Failed\,(h)$

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 the proposed framework that agents should not be able to tell the difference between an external event that did occur and a forged external event, nor between their own action and a Byzantine action imposed by the environment. 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 the resulting actions and events but cannot directly detect the environment's meddling.

Formally, this means that the local histories must be purged of the (1) *fake* instances and GMIs and of (2) controlling commands $go(i)$ and $fail\,(i)$. Both tasks are performed by the "cover-up" function σ: the former on the action/event level and the latter on the set level:

Definition 1.2.21 (Cover-up function)**.** The function $\sigma\colon 2^{\overline{GActions} \sqcup GEvents} \longrightarrow 2^{\overline{Actions} \sqcup \overline{Events}}$ is defined as follows

$$\sigma\big(X\big) = label^{-1}\Big(\big(X \cap (\overline{GActions} \sqcup \overline{GEvents})\big) \sqcup \{U \mid (\exists i) fake\,(i,U) \in X\}\Big) \tag{1.24}$$

If $U \in \overline{GActions} \sqcup GEvents$ but U is neither $go(i)$ nor $fail\,(i)$, we also write $\sigma\big(U\big)$ to denote the only element of $\sigma\big(\{U\}\big)$.

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

Definition 1.2.22 (State update functions)**.** For the set $\mathcal{A} = [\![1; n]\!]$ of agents, given a global history $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$, given a tuple of performed actions and events $X = (X_\epsilon, X_1, \ldots, X_n) \in 2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}$, for agent $i \in \mathcal{A}$, we use the following abbreviation $X_{\epsilon_i} = X_\epsilon \cap GEvents_i$. Agents i's update function $update_i$ outputs a new local history from $\mathscr{L}_i$ based on i's actions X_i and environment's control X_ϵ as follows:

$$update_i\left(h_i, X_i, X_\epsilon\right) = \begin{cases} h_i & \text{if } X_{\epsilon_i} \subset \{fail\,(i)\} \\ \left[\sigma\big(X_{\epsilon_i} \sqcup X_i\big)\right] : h_i & \text{otherwise} \end{cases} \tag{1.25}$$

where : represents sequence concatenation. The environment's state update function $update_\epsilon$ outputs a new state of the environment based on X_ϵ:

$$update_\epsilon\left(h_\epsilon, X\right) = \left(X_\epsilon \sqcup X_1 \sqcup \cdots \sqcup X_n\right) : h_\epsilon \tag{1.26}$$

Thus, the state of the system overall is modified as follows:

$$update\left(h, X\right) = \Big(update_\epsilon\left(h_\epsilon, X\right), update_1\left(h_1, X_1, X_\epsilon\right), \ldots, update_n\left(h_n, X_n, X_\epsilon\right)\Big) \tag{1.27}$$

Remark 1.2.23. The first clause in (1.25) corresponds to the situation when the agent is not woken up and, hence, does not change its local state. This can happen if the environment imposes no events on i or when the only event is branding i Byzantine from this moment on.

Definition 1.2.24 (Transition relation)**.** For the set of agents $\mathcal{A} = [\![1; n]\!]$ with protocols $P = (P_1, \ldots, P_n)$ and the protocol P_ϵ of the environment, we define a **Byzantine transition relation** $\tau^B_{P_\epsilon, P} \subseteq \mathscr{G}^2$ between a current global state and possible next global states. For global states $h = (h_\epsilon, h_1, \ldots, h_n) \in \mathscr{G}$ and $h' = (h'_\epsilon, h'_1, \ldots, h'_n) \in \mathscr{G}$, we say that $h \tau^B_{P_\epsilon, P} h'$, or $(h, h') \in \tau^B_{P_\epsilon, P}$, or $h' \in \tau^B_{P_\epsilon, P}(h)$ if there exists a tuple

$$\alpha^{h,|h|} = (\alpha^{|h|}_\epsilon, \alpha^{h,|h|}_1, \ldots, \alpha^{h,|h|}_n) \in 2^{GEvents} \times 2^{\overline{GActions_1}} \times \cdots \times 2^{\overline{GActions_n}}$$

with $\alpha^{|h|}_\epsilon \in P_\epsilon\left(|h|\right)$ and $\alpha^{h,|h|}_i = label_i\left(X_i, |h|\right)$ for some $X_i \in P_i\left(h_i\right) \cup \{\varnothing\}$ for every $i \in A$ such that

$$h' = \Big(update_\epsilon\Big(h_\epsilon, \quad \beta^{h,\alpha^{h,|h|}}\Big), \quad update_1\Big(h_1, \quad \beta_1^{h,\alpha^{h,|h|}}, \quad \beta_\epsilon^{h,\alpha^{h,|h|}}\Big), \\ \ldots, \quad update_n\Big(h_n, \quad \beta_n^{h,\alpha^{h,|h|}}, \quad \beta_\epsilon^{h,\alpha^{h,|h|}}\Big)\Big) \tag{1.28}$$

Remark 1.2.25 (Transition relation as an instance of transition template)**.** The transition relation $\tau^B_{P_\epsilon, P}$ will be used as the main and basic transition relation, but we will also use other $\tau_{P_\epsilon, P}$. For instance, in modelling rendezvous communications, we will strengthen the filtering functions to filter out unwanted *send* commands. Accordingly, we can also view a $\tau_{P_\epsilon, P}$ as a result of applying a *transition template* τ to an environment's protocol P_ϵ and a joint agents' protocol P (this idea is formalized in Def. 1.3.6). Thus, the same protocols may produce different transition relations under different transition templates.

1.3 Runs and contexts

As already mentioned, integer timestamps are used exclusively to take snapshots of the local and global states. The set 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 times.

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

$$r : \mathbb{N} \longrightarrow \mathscr{G} \tag{1.29}$$

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.

$$r_i : \mathbb{N} \longrightarrow \mathscr{L}_i \tag{1.30}$$

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

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

Definition 1.3.2. For a set $\mathcal{A} = [\![1; n]\!]$ of agents and a set $X \subset \mathcal{A} \times \mathbb{N}$ of local timestamps, or simply points, we define its upper **time bound** $T(X)$ to be the largest $T \in \mathbb{N}$ such that $(i, T) \in X$ for some $i \in \mathcal{A}$ 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.

Definition 1.3.3. For a set $\mathcal{A} = [\![1; n]\!]$ of agents, run $r \in R$, timestamp $t \in \mathbb{N}$, and bounded set $X \subset \mathcal{A} \times \mathbb{N}$ of local timestamps, we define

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

For an unbounded set $X \subset \mathcal{A} \times \mathbb{N}$, 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 1.3.4. The definition of $Bad_X(r)$ and $Failed_X(r)$ for unbounded sets is compatible with that for bounded sets, when applied to bounded sets X. The benefit of the latter is that it is efficiently computable.

Remark 1.3.5. For the set $\mathcal{A} = [\![1; n]\!]$ of agents, agents' protocols P and the environment's protocol P_ϵ, we are for now mostly interested in runs $r \in R$ such that for each timestamp $t \in \mathbb{N}$,

$$r(t+1) \in \tau^B_{P_\epsilon, P}(r(t)) \tag{1.31}$$

Sometimes we call such runs $\tau^B_{P_\epsilon,P}$-*transitional*, or simply *transitional*.

In the interests of generality and modularity of concepts, we defined the transition relation 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 runs (see also Fig. 1.1) with the dual purpose: to give a compact summary of the procedure and introduce the 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. 1.3.14 after we introduce all parts comprising a context.

In Def. 1.2.24, we defined the basic transition relation $\tau^B_{P_\epsilon,P}$ based on the 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 relation. Hence, we leave the exact details of the transition as a parameter τ that converts protocols into a transition relation.

Definition 1.3.6 (Transition template)**.** A **transition template** is a two-place function that takes the protocol P_ϵ of the environment and the joint agents' protocol P and outputs a transition relation

$$\tau(P_\epsilon)(P) = \tau_{P_\epsilon,P} \in \mathscr{G}^2 \tag{1.32}$$

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

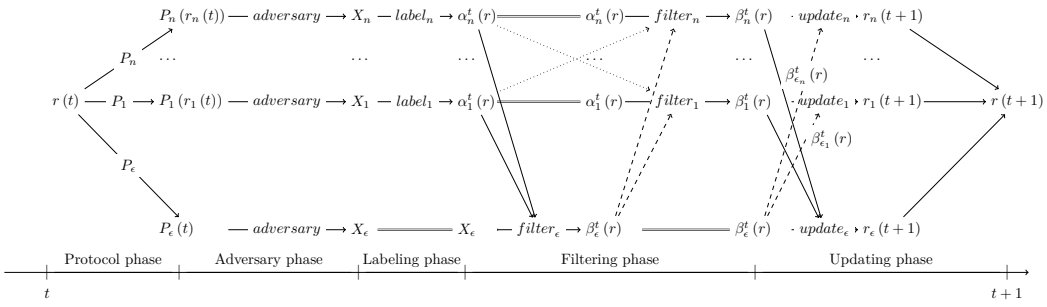

Figure 1.1: The evolution of states in round $t.5$ (from timestamp $t \in \mathbb{N}$ to $t+1$) inside a run r constructed according to the transition relation $\tau_{P_\epsilon,P}$, where $\mathcal{A} = [\![1;n]\!]$. Different communication models require changes to the filtering functions $filter_\epsilon$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

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

1. *Protocol phase*
   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 sets of events in the round based on time t.

2. *Adversary phase*
   From these ranges, the adversary non-deterministically picks one set
   $$X_i \in P_i(r_i(t)) \cup \{\varnothing\} \tag{1.33}$$
   of actions for each agent i and a set
   $$X_\epsilon \in P_\epsilon(t) \tag{1.34}$$
   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*
   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
   $$\alpha_i^t(r) = label_i(X_i, t). \tag{1.35}$$
   The set of environmental events X_ϵ is already in the global format and requires no modifications:
   $$\alpha_\epsilon^t(r) = X_\epsilon. \tag{1.36}$$
   Note that $\alpha_i^t(r) \subset \overline{GActions}_i$ and $\alpha_\epsilon^t(r) \subset GEvents$.

4. *Filtering phase*
   In this phase, intended actions and events that are deemed "physically 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 relations. The crucial and stable part of this filtering is that no actions by agent i are allowed by the environment if $go(i)$ is not among the chosen environment's events (no computation can be performed if processor time is denied). Note that some environment's events may also be "physically 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:

$$\beta_\epsilon^t(r) = filter_\epsilon\left(r(t), \quad \alpha_\epsilon^t(r), \quad \alpha_1^t(r), \quad \ldots, \quad \alpha_n^t(r)\right) \tag{1.37}$$

(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 $j \in A$. The resulting sets of actions actually performed by agents are denoted $\beta_i^t(r)$:

$$\beta_i^t(r) = filter_i\left(\alpha_1^t(r), \quad \ldots, \quad \alpha_n^t(r), \quad \beta_\epsilon^t(r)\right) \tag{1.38}$$

Note that $\beta_i^t(r) \subset \alpha_i^t(r) \subset \overline{GActions}_i$ and $\beta_\epsilon^t(r) \subset \alpha_\epsilon^t(r) \subset GEvents$.

5. *Updating phase*
The events $\beta_\epsilon^t(r)$ and actions $\beta_i^t(r)$ actually happenning 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 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 $\beta_{\epsilon_i}^t(r)$ imposed on i, whereas the global history is modified based on the complete information about all events and actions performed in the round.

$$r_i(t+1) = update_i\left(r_i(t), \quad \beta_i^t(r), \quad \beta_\epsilon^t(r)\right) \tag{1.39}$$

$$\beta^t(r) = \left(\beta_\epsilon^t(r), \quad \beta_1^t(r), \quad \ldots, \quad \beta_n^t(r)\right), \tag{1.40}$$

$$r_\epsilon(t+1) = update_\epsilon\left(r_\epsilon(t), \quad \beta^t(r)\right) \tag{1.41}$$

We will routinely use (1.33)–(1.41) 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 (1.41), all β sets can be easily retrieved from a given transitional run. We do not even have to assume a global history to result from a transitional run to define βs, though this definition would make sense mostly for transitional runs. In compliance with (1.26), for an arbitrary global history h, we define

$$\beta_i(h) = \pi_1 h_\epsilon \cap \overline{GActions}_i \tag{1.42}$$

$$\beta_\epsilon(h) = \pi_1 h_\epsilon \cap GEvents \tag{1.43}$$

For the case of runs

$$\beta_i^t(r) = \pi_1 r_\epsilon(t+1) \cap \overline{GActions}_i \tag{1.44}$$

$$\beta_\epsilon^t(r) = \pi_1 r_\epsilon(t+1) \cap GEvents \tag{1.45}$$

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: $\overline{\beta}_\epsilon^t(r)$ imposed on all agents and $\overline{\beta}_{\epsilon_i}^t(r)$ imposed specifically on agent i;
- *go* events: $\beta_g^t(r)$ imposed on all agents and $\beta_{g_i}^t(r)$ imposed specifically on agent i;
- Byzantine behavior: $\beta_b^t(r)$ imposed on all agents and $\beta_{b_i}^t(r)$ imposed specifically on agent i;
- all external events that make the agent Byzantine until the end of the run, including faulty inaction $fail(i)$: $\beta_f^t(r)$ 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 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 (1.33)–(1.34) that eventually generate $\beta_1^t(r), \ldots, \beta_n^t(r), \beta_\epsilon^t(r)$ from (1.44)–(1.45) according to (1.35)–(1.38). 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 the transition relation $\tau_{P_\epsilon,P}$.

Transitional runs have several useful properties:

Remark 1.3.7 (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: for instance,

$$\beta_\epsilon^{t'}(r) = \pi_1 r_\epsilon(t'+1) \cap GEvents = \pi_1(\pi_{t-t'+2} r_\epsilon(t+1)) \cap GEvents$$

Lemma 1.3.8 (Objective global time)**.** *For $\tau_{P_\epsilon,P}$-transitional runs, $|r(t)| = t$.*

It easily follows from (1.33)–(1.41) that

Lemma 1.3.9 (GMIs are correct)**.** *For a $\tau_{P_\epsilon,P}^B$-transitional run r, for $i, j \in \mathcal{A}$, $\mu \in Msgs$, $t \in \mathbb{N}$, and $id \in \mathbb{N}$*

$$gsend(i,j,\mu,id) \in \beta_i^t(r) \implies id = id(i,j,\mu,a,t) \text{ for some } a \in \mathbb{N} \tag{1.46}$$

$$fake(i, gsend(i,j,\mu,id)) \in \beta_{b_i}^t(r) \implies id = id(i,j,\mu,a,t) \text{ for some } a \in \mathbb{N} \tag{1.47}$$

$$grecv(i,j,\mu,id) \in \overline{\beta}_{\epsilon_i}^t(r) \implies id = id(j,i,\mu,a,t') \text{ for some } a \in \mathbb{N} \text{ and } t' \leq t \tag{1.48}$$

Proof. By (1.38), Def. 1.2.12, (1.35), and Def. 1.2.7, the id for correct *gsend* instructions is supplied by the function $label_i$, which guarantees the first statement.

Similarly, for the second statement, by (1.37) and (1.17), all Byzantine *gsend* instructions that feature an incorrect *id* are filtered out by the function $filter_\epsilon$.

Finally, for the third statement, by the same (1.37) and (1.17), if a *grecv* command was not filtered out by $filter_\epsilon$, the matching *gsend* command or its Byzantine version must have occurred at the latest by the same round. In other words, taking into account (1.37)–(1.38), there are four possibilities:

1. $gsend(j,i,\mu,id) \in r_\epsilon(t) \implies gsend(j,i,\mu,id) \in \beta_j^{t'}(r)$ for some $t' < t$
2. $gsend(j,i,\mu,id) \in \beta_j^t(r)$
3. $fake(j, gsend(j,i,\mu,id)) \in r_\epsilon(t) \implies fake(j, gsend(j,i,\mu,id)) \in \beta_{b_j}^{t'}(r)$ for some $t' < t$
4. $fake(j, gsend(j,i,\mu,id)) \in \beta_{b_j}^t(r)$

In other words,

$$\text{either} \quad gsend(j,i,\mu,id) \in \beta_j^{t'}(r) \quad \text{or} \quad fake(j, gsend(j,i,\mu,id)) \in \beta_{b_j}^{t'}(r)$$

for some $t' \leq t$. It remains to use the already proved (1.46) or (1.47) respectively. □

Corollary 1.3.9.1. *For a $\tau_{P_\epsilon,P}^B$-transitional run r, for $i, j \in \mathcal{A}$, $\mu \in Msgs$, $t \in \mathbb{N}$, and $id \in \mathbb{N}$*

$$grecv(i,j,\mu,id) \in \overline{\beta}_{\epsilon_i}^t(r) \implies$$
$$gsend(j,i,\mu,id) \in \beta_j^{t'}(r) \quad or \quad fake(j, gsend(j,i,\mu,id)) \in \beta_{b_j}^t(r) \quad for\ some\ t' \leq t \tag{1.49}$$

Corollary 1.3.9.2 (GMIs are unique)**.** *For a $\tau^B_{P_\epsilon,P}$-transitional run r, for $i,j,k,l \in \mathcal{A}$, $\mu,\mu' \in Msgs$, $t,t' \in \mathbb{N}$, and $id \in \mathbb{N}$:*

$$\begin{cases} gsend(i,j,\mu,id) \in \beta^t_i\,(r) \\ gsend(k,l,\mu',id) \in \beta^{t'}_k\,(r) \end{cases} \implies \begin{cases} k=i \\ l=j \\ t'=t \\ \mu'=\mu \end{cases}$$

$$\begin{cases} fake\,(i,gsend(i,j,\mu,id)) \in \beta^t_{b_i}\,(r) \\ gsend(k,l,\mu',id) \in \beta^{t'}_k\,(r) \end{cases} \implies \begin{cases} k=i \\ l=j \\ t'=t \\ \mu'=\mu \end{cases}$$

$$\begin{cases} fake\,(i,gsend(i,j,\mu,id)) \in \beta^t_{b_i}\,(r) \\ fake\,(k,gsend(k,l,\mu',id)) \in \beta^{t'}_{b_k}\,(r) \end{cases} \implies \begin{cases} k=i \\ l=j \\ t'=t \\ \mu'=\mu \end{cases}$$

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.

Proof. The statements follow from Lemma 1.3.9 and the injectivity of $id(\cdot)$ from Def. 1.1.12. □

Corollary 1.3.9.3 (Send-receive causality)**.** *For a $\tau^B_{P_\epsilon,P}$-transitional run r, for $i,j,k,l \in \mathcal{A}$, $\mu,\mu' \in Msgs$, $t,t' \in \mathbb{N}$, and $id \in \mathbb{N}$:*

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

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

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 1.3.9 and the injectivity of $id(\cdot)$ from Def. 1.1.12. □

Remark 1.3.10. Property (1.46) actually holds for any $\tau_{P_\epsilon,P}$-transitional run because all GMIs for correct *gsend*s are created by $label_i$, whether they are later filtered or not. The other two properties depend on filtering. It could be argued that Byzantine behavior can be strengthened and/or reliability of the communication channel can be weakened so much that these properties would be violated.

While it is preferrable to directly build desired properties beign modeled into the transition relations, so-to-say 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 property of the whole infinite run. Therefore, to enforce such properties we are forced to restrict the set of runs being considered.

Definition 1.3.11 (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 1.3.12 (Context)**.** A **context**

$$\gamma = (P_e, \mathscr{G}(0), \tau, \Psi) \tag{1.50}$$

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

Definition 1.3.13 (Agent-context)**.** Given a context γ and joint protocol P, we can combine them in an **agent-context** $\chi = (\gamma, P)$.

Definition 1.3.14 (Consistency)**.** For a context $\gamma = (P_e, \mathscr{G}(0), \tau, \Psi)$ and a joint protocol P, we define the set of runs **weakly consistent** with P in γ and call it the system $R^{w(\gamma,P)}$ to be set of $\tau_{P_e,P}$-transitional runs that start at some **global initial state** from $\mathscr{G}(0)$:

$$R^{w(\gamma,P)} \quad = \quad \{r \in R \quad | \quad r(0) \in \mathscr{G}(0) \quad \text{and} \quad (\forall t \in \mathbb{N})\, r(t+1) \in \tau_{P_e,P}(r(t))\} \tag{1.51}$$

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

$$R^{(\gamma,P)} = R^{w(\gamma,P)} \cap \Psi. \tag{1.52}$$

In order to define *coherence* of local states, we introduce the fundamental part of a logical formalism to reason about agents' knowledge during (consistent) runs. 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 the 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:

Definition 1.3.15 (Local equivalence)**.** For an agent-context χ, a group of agents $G \subset \mathcal{A}$ and two runs $r, r' \in R^\chi$ we say that r and r' are **locally equivalent** for G, written $r \approx^G r'$, iff

$$\Big(\forall (j,t) \in G \times \mathbb{N}\Big)\, r_j(t) = r'_j(t)$$

We also write $\approx^i$ instead of $\approx^{\{i\}}$.

The full formalism will be introduced in Sect. 1.4.

Remark 1.3.16. For any group $G \subset \mathcal{A}$ of agents, the relation $\approx^G$ is an equivalence relation.

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 1.3.17 (Coherence)**.** For an agent-context χ, a run $r \in R^\chi$ and a point $(i,t) \in \mathcal{A} \times \mathbb{N}$, we say $r_i(t)$ is χ-**coherent w.r.t.** i, or simply **coherent w.r.t.** i, iff

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

1.4 Syntax and Semantics

In the previous sections, we described several agent-contexts modelling various communication scenarios. Independent of a scenario chosen, we define a formal language and its semantics in order to express knowledge of an agent in a such a system.

As stated in Def. 1.3.15, two global states are indistinguishable for agent $i \in \mathcal{A}$ if and only if i's local state in both these states is the same (i.e., the two states are exactly the same from the point of view of agent i). Given this indistinguishability relation we will define an epistemic Kripke model with worlds being global states of runs.

For multiple reasons, we consider the communication model, in the form of (agent-)context to be common knowledge among agents. In other words, the only possibilities an agent in a run from R^χ should consider are global states from various runs from R^χ. For instance, a syncronous agent who determined that it had skipped a round should conclude that it is compromised rather than imagining itself in an asynchronous context. It is necessary to distinguish knowledge state of different types of agents, given that consensus with Byzantine failures can be solved in the synchronous context but not in the asynchronous one. It is also ultimately reasonable since the lack of knowledge of communication robustness clearly precludes the opportunity to exploit the benefits of the particular communication scheme.

Definition 1.4.1 (Atomic propositions)**.** Π is the countable set 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 of an atomic proposition $p \in \Pi$ is $\pi(h)(p)$.

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 $R' = R^\chi$.

Interpreted systems are close relatives of Kripke models, tailored to the applications to runs in distributed environments, with global states playing the role of possible worlds.

From agent i's point of view two global states are indistinguishable when the local state of i is the same in both global states. Then, we will define in a natural way an indistinguishability relation $\sim_i$ over global states for agent i, called **possible worlds relation**.

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

$$\sim_i = \{\ (h, h') \mid \pi_{i+1} h = \pi_{i+1} h' \} \tag{1.53}$$

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'.

Now, we define a **language** $\mathscr{L}$ to deal with the expression of knowledge in a *system*. For this we extend the propositional logic with :

1. three modal operators :
   - K_i operator for agent $i \in \mathcal{A}$, in order to deal with the "knowledge of a fact by agent i." For a global state $h \in \mathscr{G}$, $K_i\varphi$ formula can be read as "the agent i knows φ holds": this means that, in every global state indistinguishable from h for i, the proposition φ holds.
   - E_G operator for a group of agents $G \subset \mathcal{A}$, in order to express "everyone in the group of agents G knows." It is naturally defined from the operators K_i.
   - C_G operator for a group of agents $G \subset \mathcal{A}$, in order to express "something is a common knowledge among the agents of G." Common knowledge of φ refers to the fact that everyone in G knows that everyone in G knows ... that everyone in G knows φ.

2. and two temporal operators :

   - always in the future operator $\Box$ to express things like "the sender will remember for ever that he has sent *Hello*." In temporal logic, it is usually denoted G.
   - And its dual operator $\Diamond$, sometime in the future, to express things like "he will eventually see that this is a fake news." In temporal logic, it is usually denoted F.

Definition 1.4.5. For an agent $i \in \mathcal{A}$, a group of agents $G \subset \mathcal{A}$ and an atomic proposition $p \in \Pi$, $\mathscr{L}$ is generated by the following BNF specification

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

We define the remaining Boolean connectives such as $\rightarrow$ and $\leftrightarrow$ in the standard way. In addition, we define $E_G^0\varphi = \varphi$ and $E_G^{n+1}\varphi = E_G E_G^n\varphi$.

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

Definition 1.4.6. For a set $R' \subset R$ of runs, an agent $i \in \mathcal{A}$, a group of agents $G \subset \mathcal{A}$, an interpretation π, the interpreted system $I = (R', \pi)$, a run $r \in R'$, and a timestamp $t \in \mathbb{N}$:

$$\begin{array}{lll}
(I,r,t) \models p & \text{iff} & \pi(r(t))(p) = \top \\
(I,r,t) \models \neg\varphi & \text{iff} & (I,r,t) \not\models \varphi \\
(I,r,t) \models \varphi \vee \varphi' & \text{iff} & (I,r,t) \models \varphi \ \text{ or } (I,r,t) \models \varphi' \\
(I,r,t) \models \varphi \wedge \varphi' & \text{iff} & (I,r,t) \models \varphi \ \text{ and } (I,r,t) \models \varphi' \\
(I,r,t) \models K_i\varphi & \text{iff} & (\forall r' \in R')(\forall t' \in \mathbb{N})\Big(r'(t') \sim_i r(t) \ \Rightarrow \ (I,r',t') \models \varphi\Big) \\
(I,r,t) \models E_G\varphi & \text{iff} & (\forall i \in G)\,(I,r,t) \models K_i\varphi \\
(I,r,t) \models C_G\varphi & \text{iff} & (\forall m \in \mathbb{N})\,(I,r,t) \models E_G^m\varphi \\
(I,r,t) \models \Box\varphi & \text{iff} & (\forall t' \geq t)\,(I,r,t') \models \varphi \\
(I,r,t) \models \Diamond\varphi & \text{iff} & (\exists t' \geq t)\,(I,r,t') \models \varphi
\end{array}$$

1.5 Atomic Propositions

We have the language $\mathscr{L}$ to make statements and the associated semantics to tell the truth value of a formula for a given interpreted system I, given run $r \in R$ and timestamp $t' \in \mathbb{N}$. Now, we will 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 $i \in A$:

- $correct_{(i,t)}$ states that by time instant $t \in \mathbb{N}$, 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 $fail\,(i)$ from timestamp 0 to timestamp t.
- $fake_{(i,t)}\,(o)$ states that the Byzantine version of $o \in \overline{Actions} \sqcup \overline{Events}$ occurred for agent $i \in \mathcal{A}$ in round $(t-1).5$, i.e., $o \in \sigma\big(\beta_{b_i}^{t-1}\,(r)\big)$, or equivalently $fake\,(i,o) \in \beta_{b_i}^{t-1}\,(r)$. In particular, the truth of this atomic proposition implies that $o \in \overline{Actions}_i \sqcup \overline{Events}_i$.
- $\overline{occurred}_{(i,t)}(o)$ states that the correct version of $o \in \overline{Actions} \sqcup \overline{Events}$ occurred for $i \in \mathcal{A}$ in round $(t-1).5$, i.e., $o \in label^{-1}\left(\beta_i^{t-1}\,(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}\,(r)\right)$. In particular, the truth of this atomic proposition implies that $o \in \overline{Actions}_i \sqcup \overline{Events}_i$.

- $occurred_i(o)$ states that agent i has experienced some version of $o \in \overline{Actions} \sqcup \overline{Events}$, be it correct or Byzantine, of which the agent is unaware.

We now give formal definitions and discuss properties of these atomic propositions. First, we will define for each agent $i \in \mathcal{A}$ at time instant $t \in \mathbb{N}$ and for each internal action or correct external event $o \in \overline{Actions} \sqcup \overline{Events}$.

Definition 1.5.1. A weakly χ-based interpreted system $I = (R^{w(\chi)}, \pi)$ and its interpretation π are called **proper** if, for 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 for $fake_{(i,t)}(o) \in \Pi$, $\overline{occurred}_{(i,t)}(o) \in \Pi$, and for $correct_{(i,t)} \in \Pi$:

$$\pi(r(t'))\left(correct_{(i,t)}\right) = \top \quad \text{iff} \quad (i,t) \notin Failed(r,t') \tag{1.54}$$

$$\pi(r(t'))\left(fake_{(i,t)}(o)\right) = \top \quad \text{iff} \quad t \geq 1 \text{ and } o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right) \tag{1.55}$$

$$\pi(r(t'))\left(\overline{occurred}_{(i,t)}(o)\right) = \top \quad \text{iff} \quad t \geq 1 \text{ and } o \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right) \tag{1.56}$$

$$\pi(r(t'))\left(occurred_i(o)\right) = \top \quad \text{iff} \quad o \in r_i(t') \tag{1.57}$$

Remark 1.5.2. Although the first three of these atomic propostions are objective properties, which are typically imperceptible for agents, they are formulated for actions/events o in the local representation.

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 infinite runs. The global history $r(t')$ does not contain any information about $t > t'$. Indeed, there are generally multiple $\tau_{P_\epsilon,P}$-transitional runs continuing from the global history $r(t')$ possible, 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, while we could give equivalent definitions for arbitrary runs, the restriction to transitional runs enables us to use β-sets instead of redefining them all yet again.

Remark 1.5.3.

$$\pi(r(t'))\left(fake_{(i,t)}(o)\right) = \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i$$

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

$$\pi(r(t'))\left(occurred_i(o)\right) = \top \quad \text{implies} \quad o \in \overline{Actions}_i \sqcup \overline{Events}_i$$

The omniscient environment does not forget.

Lemma 1.5.4. *Consider an agent-context χ, $o \in \overline{Actions} \sqcup \overline{Events}$, $r \in R^{w(\chi)}$, $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, $t' \geq t$, and a weakly χ-based proper interpreted system $I = (R^{w(\chi)}, \pi)$:*

$$(I,r,t') \models correct_\theta \iff (I,r,t') \models \Box correct_\theta$$

$$(I,r,t') \models fake_\theta(o) \iff (I,r,t') \models \Box fake_\theta(o)$$

$$(I,r,t') \models \overline{occurred}_\theta(o) \iff (I,r,t') \models \Box\overline{occurred}_\theta(o)$$

$$(I,r,t') \models occurred_i(o) \iff (I,r,t') \models \Box occurred_i(o)$$

Proof. The direction from right to left is trivial in all four 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 all four cases, the truth is based on a particular event, correct or Byzantine occurring in the global run at round $(t-1).5)$ in the first three cases or by timestamp t' in the last case. Since all futures of t' lie to the future of this event (due to $t \leq t'$ for the first three cases) and the round enumeration remains stable, the event remains in the global run. $\square$

Note that this property can easily be violated for $correct_{(i,t)}$, which is based on certain kinds of event *not* occurring by timestamp t.

In our framework, a correct action/event cannot be simultaneously faked. More formally,

Lemma 1.5.5. *Consider a Byzantine context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau^B, \Psi)$, an agent-context $\chi = (\gamma, P)$, some $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R^{w(\chi)}$, a local timestamp $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, a timestamp $t' \geq t$, and a weakly χ-based proper interpreted system $I = (R^{w(\chi)}, \pi)$:*

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

Proof. To prove the first implication, assume $(I, r, t') \models \overline{occurred}_\theta(o)$. By the definition of properness, this means that $o \in label^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. There are two possibilities:

Case I: $o \in label^{-1}\left(\beta_i^{t-1}(r)\right)$ is an internal action. In this case, $o \in \overline{Actions}_i$ and there must exist $o' \in \beta_i^{t-1}(r) \subset \overline{GActions}_i$ such that $o = label^{-1}(o')$. Note that $\beta_i^{t-1}(r) \neq \varnothing$ implies $go(i) \in \beta_{g_i}^{t-1}(r)$ by Def. 1.2.12. The situation further splits into two cases. If $o = internal(i,u)$ it can only be obtained from $o' = internal(i,u)$; if $o = send(j, \mu_k)$, it can only be obtained from $o' = gsend(i, j, \mu, id(i, j, \mu, k, t-1))$.[7] We need to show that $o \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$, which by definition (1.24) of σ, given $\beta_{b_i}^{t-1}(r) \cap \overline{GEvents} = \varnothing$, means $fake(i,o') \notin \beta_{b_i}^{t-1}(r)$. This indeed follows by Def. 1.2.12 of $filter_\epsilon^B$ from $o' \in \beta_i^{t-1}(r)$ and $go(i) \in \beta_{g_i}^{t-1}(r)$.

Case II: $o \in label^{-1}\left(\overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$ is a correct external event. In this case, $o \in \overline{Events}_i$ and there must exist $o' \in \overline{\beta}_{\epsilon_i}^{t-1}(r) \subset \overline{GEvents}_i$ such that $o = label^{-1}(o')$. The situation again splits into two cases, this time somewhat different:

> Case 1: $o = external(i,u)$ is an external event. Then o it can only be obtained from $o' = external(i,u)$. The reasoning here is the same as for actions above: there is only one global event triggering such a response of the agent, and every fake global event is filtered out in the presence of its correct counterpart.
>
> Case 2: $o = recv(j,\mu)$ is a message receipt. Then o can be obtained from $o' = grecv(i,j,\mu,id')$ for some $id' \in \mathbb{N}$ (this id' must be correctly built but it is irrelevant for the proof). We need to show that $o \notin \sigma\left(\beta_{b_i}^{t-1}(r)\right)$, which again means $fake(i,o'') \notin \beta_{b_i}^{t-1}(r)$ for some $o'' = grecv(i,j,\mu,id'')$ for some potentially different id''. However, whether $id' = id''$, the event $fake(i,o'')$ will be filtered out according to Def. 1.2.12 of $filter_\epsilon^B$, which removes duplicate fake receipts independent of the ids.

We have demonstrated that $(I, r, t') \models \overline{occurred}_\theta(o) \to \neg fake_\theta(o)$.

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

Using these atomic propositions with fixed evaluations, we can define derived concepts with similarly fixed meanings. For instance, the **absolute occurrence** $\overline{occurred}_\theta(o)$ represents information about local actions and events accessible only for the environment. We now define several notions related to **relative occurrence** $occurred_{(i,t)}(o)$ that represents agents' information about the same events:

Definition 1.5.6 (Relative occurred)**.** Consider any agent $i \in A = [\![1; n]\!]$, any timestamp $t \in \mathbb{N}$, and any integer $k \geq 1$. For any $o \in \overline{Actions} \sqcup \overline{Events}$,

$$occurred_{(i,t)}(o) = \overline{occurred}_{(i,t)}(o) \vee fake_{(i,t)}(o) \tag{1.58}$$

$$occurred^{(k)}(o) = \bigvee_{\substack{S \subset A \\ |S| = k}} \bigwedge_{i \in S} occurred_i(o) \tag{1.59}$$

$$occurred(o) = occurred^{(1)}(o) \tag{1.60}$$

[7] All correctly sent messages have correct GMIs.

Informally speaking,

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

In all these cases agents are not aware whether o was correct or Byzantine; in case of k agents, a mixture of correct and Byzantine entries also satisfies the conditions.

Remark 1.5.7. Note that $\mathit{occurred}^{(k)}(o)$ requires the existence of *some* k pairwise distinct agents. In particular, in order to fulfill $K_i\mathit{occurred}^{(k)}(o)$, 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.

It is a direct corollary of Lemma 1.5.4 that

Remark 1.5.8. Consider an agent-context χ, $o \in \overline{Actions} \sqcup \overline{Events}$, $r \in R^{w(\chi)}$, $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, $t' \geq t$, and a weakly χ-based proper interpreted system $I = (R^{w(\chi)}, \pi)$

$$\begin{aligned}
(I,r,t') \models \mathit{occurred}_{(i,t)}(o) \quad &\Leftrightarrow \quad (I,r,t') \models \Box\mathit{occurred}_{(i,t)}(o) \\
(I,r,t') \models \mathit{occurred}(o) \quad &\Leftrightarrow \quad (I,r,t') \models \Box\mathit{occurred}(o) \\
(I,r,t') \models \mathit{occurred}^{(k)}(o) \quad &\Leftrightarrow \quad (I,r,t') \models \Box\mathit{occurred}^{(k)}(o)
\end{aligned}$$

Lemma 1.5.9. *Consider a context $\gamma = (P_e, \mathscr{G}(0), \tau^B, \Psi)$, an agent-context $\chi = (\gamma, P)$, $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R^\chi$, a local timestamp $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$ with $t \geq 1$, a timestamp $t' \geq t$, and a weakly χ-based proper interpreted system $I = (R^{w(\chi)}, \pi)$.*

$$(I,r,t') \models \mathit{occurred}_\theta(o) \qquad \textit{iff} \qquad r_i(t) = \lambda : r_i(t-1) \ \textit{ and } o \in \lambda$$

Proof. Case I: from left to right. Assume $(I,r,t') \models \mathit{occurred}_\theta(o) = \overline{\mathit{occurred}}_\theta(o) \vee \mathit{fake}_\theta(o)$

Case 1: $(I,r,t') \models \overline{\mathit{occurred}}_\theta(o)$. Then $o \in \mathit{label}^{-1}\left(\beta_i^{t-1}(r) \sqcup \overline{\beta}_{\epsilon_i}^{t-1}(r)\right)$. Thus, $o = \mathit{label}^{-1}(o')$ 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, as demonstrated in the proof of Lemma 1.5.5, if o' is an action from $\beta_i^{t-1}(r)$, then $go(i) \in \beta_{\epsilon_i}^{t-1}(r)$, meaning that $\beta_{\epsilon_i}^{t-1}(r) \not\subset \{fail(i)\}$. So by (1.39), definition (1.25) of update_i, and definition (1.24) of σ we conclude that $r_i(t) = \lambda : r_i(t-1)$ and $o = \mathit{label}^{-1}(o') \in \lambda$.

Case 2: $(I,r,t') \models \mathit{fake}_\theta(o)$. Then $o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. It remains to note that $\beta_{b_i}^{t-1}(r) \neq \varnothing$ implies that $\beta_{\epsilon_i}^{t-1}(r) \not\subset \{fail(i)\}$ and $r_i(t) = \lambda : r_i(t-1)$. Once again, the definition (1.25) of update_i implies $o \in \lambda$.

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

Case II: from right to left. Assume $r_i(t) = \lambda$: $r_i(t-1)$ and $o \in \lambda$. By definition (1.25) of update_i it means that $\beta_{\epsilon_i}^{t-1}(r) \not\subset \{fail(i)\}$ and one of the following happens:

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

Case 2: $o \in \sigma\left(\beta_{b_i}^{t-1}(r)\right)$. We have $(I,r,t') \models \mathit{fake}_\theta(o)$, and, hence, $(I,r,t') \models \mathit{occurred}_\theta(0)$

We proved that in either case $(I,r,t') \models \mathit{occurred}_\theta(o)$.

□

Lemma 1.5.10. *Consider a context $\gamma = (P_\epsilon, \mathscr{G}(0), \tau^B, \Psi)$, an agent-context $\chi = (\gamma, P)$, $o \in \overline{Actions} \sqcup \overline{Events}$, a run $r \in R^{w(\chi)}$, an agent $i \in A$, a timestamp $t \in \mathbb{N}$, and a weakly or strongly χ-based proper interpreted system $I = (R', \pi)$ with $R' \in \{R^{w(\chi)}, R^\chi\}$.*

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

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 $(r',t') \in R'$ such that $r(t) \sim_i r'(t')$, i.e., $r_i(t) = r'_i(t')$, we have

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

Therefore $(I,r,t) \models K_i\, occurred_i(o)$. □

1.6 Past and causality relationship

In this section, we will introduce the **causality relationship**(*Lamport's Happened Before relation*), it represents the dependence between points i.e when an agent i send a message to j at time 0, received at time 1, then all the futur states of j after time 1 are related to the receipt of the message or more generally to the point $(i,0)$. In the next chapter, we will link the knowledge of a node to the structure of the causality relationships.

Definition 1.6.1 (Lamport's Happened Before relation with Byzantine Failure)**.** For a run r, two points $(i,j) \in \mathcal{A}^2$, two timestamps $(t,t') \in \mathbb{N}^2$, a message $\mu \in Msgs$ and an GID $id \in \mathbb{N}$ we define the **Lamport's Happened Before relation** $\rightarrow_r^{Lb}$ as the smallest relation, over points, statisfying :

1. Locality relation :
$$\text{If } t \leq t' \text{ then } (i,t) \rightarrow_r^{Lb} (i,t') \tag{1.61}$$

2. Message relation :
$$\text{If } \begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \beta_i^t(r) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id)) \in \beta_{b_i}^t(r) \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t'-1}(r) \end{cases} \text{ then } (i,t) \rightarrow_r^{Lb} (i,t') \tag{1.62}$$

3. Transitivity relation :
$$\text{If } \begin{cases} (i,t) \rightarrow_r^{Lb} (j,t') \\ (j,t') \rightarrow_r^{Lb} (k,t'') \end{cases} \text{ then } (i,t) \rightarrow_r^{Lb} (k,t'') \tag{1.63}$$

Now, for a run r the related *causal graph* represents all the causality dependences of nodes in r. We will see that we can build the *causal graph* by using the *Lamport's Happened Before relation* (and repespectively)

Definition 1.6.2 (Causal graph)**.** For a run r, we define the causal graph of r as $G(r) = (V, E^r)$ such that

1. $V = \mathcal{A} \times \mathbb{N}$
2. $E^r = E^r_{loc} \cup E^r_{msg}$

- $E^r_{loc} = \{((i,t),(i,t+1)) \mid (i,t) \in \mathcal{A} \times \mathbb{N}\}$
- $E^r_{msg} = \left\{ ((i,t),(j,t')) \mid \left\{ \begin{array}{l} \left\{ \begin{array}{l} gsend(i,j,\mu,id) \in \ \beta^t_i\,(r) \\ \text{or} \\ fake\,(i, gsend(i,j,\mu,id)) \in \beta^t_{b_i}\,(r) \end{array} \right. \\ grecv(j,i,\mu,id) \in \overline{\beta}^{t'-1}_{\epsilon_j}\,(r) \end{array} \right. \right\}$

Definition 1.6.3 (Path)**.** For a run r, two points $(\beta,\theta) \in (\mathcal{A} \times \mathbb{N})^2$, we define the following notations

- $\beta \to \theta$ iff $(\beta,\theta) \in E^r$
- ξ is a path from β to θ $\quad$ iff $\beta \rightsquigarrow^{\xi} \theta$
 iff $\xi : \beta = \eta_0 \to \eta_1 \to \cdots \to \eta_{|\xi|-1} = \theta$

The **causal cone** of a point $\theta \in \mathcal{A} \times \mathbb{N}$ in a run r is the points that are part of the past of θ i.e points that impact the state of θ.

Definition 1.6.4 (Causal Cone)**.** For a run r, a point $\theta \in \mathcal{A} \times \mathbb{N}$, we define the causal cone of θ in r as follow $V^r_\theta = \{\beta \in \mathcal{A} \times \mathbb{N} \mid \beta \to^{Lb}_r \theta\}$

Definition 1.6.5 (Past graph)**.** For a run r and a point $\theta \in \mathcal{A} \times \mathbb{N}$, $Past(r,\theta)$ is the subgraph of G(r) induced by V^r_θ

Lemma 1.6.6. *For a run r and a point $\theta \in \mathcal{A} \times \mathbb{N}$, $\beta \in V^r_\theta \Rightarrow V^r_\beta \subset V^r_\theta$*

Proof. Let $\theta \in \mathcal{A} \times \mathbb{N}, \beta \in V^r_\theta$, let us prove that $\eta \in V^r_\beta \Rightarrow \eta \in V^r_\theta$,
Let $\eta \in V^r_\theta$, $\left\{ \begin{array}{l} \beta \in V^r_\theta, \text{ so there is a } \xi_2 : \beta \rightsquigarrow^{\xi_2} \theta \\ \eta \in V^r_\beta, \text{ so there is a } \xi_1 : \eta \rightsquigarrow^{\xi_1} \beta \end{array} \right.$ hence $\xi : \xi_1.\xi_2$, in G(r), therefore $\eta \in V^r_\theta$ $\square$

Lemma 1.6.7. *For a run r and two points $(\beta,\theta) \in (\mathcal{A} \times \mathbb{N})^2$*

$$\beta \to^{Lb}_r \theta \text{ iff there is a path from } \beta \text{ to } \theta \text{ in } Past(r,\theta)$$

.

Proof. Let a run r, $(\theta_1 = (i,t), \theta_2 = (j,t')) \in (\mathcal{A} \times \mathbb{N})^2$

Case I: $\beta \to^{Lb}_r \theta$. We will reason by induction on the structure of $\beta \to^{Lb}_r \theta$.

Case 1: Locality.

Case a: $\theta_1 = \theta_2$. ξ is the path of lenght 0 between θ and θ

Case b: $i = j$ and $t' = t+1$. so by definition $(\theta_1,\theta_2) \in E^r_{loc}$

Case 2: Message relation.
so by definition $(\theta_1,\theta_2) \in E^r_{msg}$

Case 3: Transitivity.
so there is $\eta = (k,t'') \in V^r$ so that $\theta_1 \to^{Lb}_r \eta \to^{Lb}_r \theta_2$
Hence by induction :

1.with $\theta_1 \to^{Lb}_r \eta$: we get $\theta_1 \rightsquigarrow^{\xi_1} \eta$ in $Past(r,\eta) \subset_{\text{with Lem1.6.6}} Past(r,\theta)$

2.with $\eta \to^{Lb}_r \theta$: we get $\eta \rightsquigarrow^{\xi_2} \theta_2$ in $Past(r,\theta)$

So $\xi = \xi_1.\xi_2, \beta \rightsquigarrow^{\xi} \theta$ in $Past(r,\theta)$

Conclusion : $\exists \xi, \beta \rightsquigarrow^{\xi} \theta$ in $Past(r,\theta)$

Case II: $\xi : \beta \rightsquigarrow^{\xi} \eta$ in $Past(r,\theta)$. We will reason by induction on the structure of ξ.

Case 1: $\xi = \theta_1 \to \theta_2, (\theta_1,\theta_2) \in E^r_{loc}$. by *Locality* we have $\theta_1 \to^{Lb}_r \theta_2$

Case 2: $\xi = \theta_1 \to \theta_2, (\theta_1,\theta_2) \in E^r_{msg}$. by *Message* we have $\theta_1 \to^{Lb}_r \theta_2$

Case 3: $\xi = \xi_1.\xi_2,$ where $\theta_1 \rightsquigarrow^{\xi_1} \eta, \eta \rightsquigarrow^{\xi_2} \theta_2$. Hence by induction :

1.with $\theta_1 \rightsquigarrow^{\xi_1} \eta$: we get $\theta_1 \rightarrow_r^{Lb} \eta$

2.with $\eta \rightsquigarrow^{\xi_1} \theta_2$: we get $\eta \rightarrow_r^{Lb} \theta$

So by *Transitivity* we have $\theta_1 \rightarrow_r^{Lb} \theta_2$

Conclusion : $\theta_1 \rightarrow_r^{Lb} \theta_2$

□

1.7 About the power of Byzantine agents

1. Thanks to $fake(,)$, a Byzantine agent can simulate any actions or events without considering its local state neither the wake up from the environment($go()$). With the following restrictions :

   (a) Let $i \in \mathcal{A}, a \in \overline{Actions}, fake(i,a)$:
   by definition of $\mathscr{L}_i$: $a \in \overline{Actions_i}$
   This restriction does not restrict the nuisance power on the other agents indeed the messages sent by a Byzantine agents do not depend on the localstate.

   (b) Let $i \in \mathcal{A}, e \in \overline{Events}, fake(i,e)$:
   by definition of $\mathscr{L}_i$: $a \in \overline{Events_i}$
   The external events' restriction does not restrict the nuisance power on the other agents(same arguments than for $\overline{Actions}$).
   However we have to deal about $fake\,(i, send(l,k)\mu id)$: in the current model, the definition of $\mathscr{L}_i$ restricts l to $l = i$ but this is a restriction of the power of nuisance. Therefore, it would be interesting to make an extension of the current model where l is not necessary equal to i.

2. Thanks to the *adversary*(and its nondeterministic choice) each agent - Byzantine or not - is allowed to do less actions than its protocol requires(same thing for events).

3. Thanks to the transition function τ each agent - Byzantine or not - remembers every actions he has done, and every events that have occured to him. But he does not remember the distincion between a correct one and a fake one. By the same argument as before, this will not restrict the nuisance power on the other agents but it will alterate the knowledge of the Byzantine agents.

1.8 Extensions

In the previous sections, we have define the general framework. Now, in order to implement a specific behaviour, for instance to make agents synchronous, we will introduce **framework extensions** which are instantiations and restrictions of the general framework. It is formally defined as follows :

Definition 1.8.1 (Extension)**.** For a set of pairs of protocols $PP^A \subset \mathscr{C}_\epsilon \times \mathscr{C}$, a transition template τ^A, an admissibility condition Ψ^A and a set of sets of global initial states IS^A, we define the framework extension

$$\mathscr{E}^A = (PP^A, IS^A, \tau^A, \Psi^A)$$

Definition 1.8.2. We say that an agent-context $\chi = ((P_\epsilon, \mathscr{G}(0), \tau, \Psi^b), P)$ is part of an extension $\mathscr{E}^A = (PP^A, IS^A, \tau^A, \Psi^A)$, denoted $\chi \in \mathscr{E}^A$ iff

1. $(P_\epsilon, P) \in PP^A$
2. $\mathscr{G}(0) \subset IS^A$

3. $\tau = \tau^A$

4. $\Psi^b = \Psi^A$

Two extensions are said *incompatible* if the properties of the agents or of the environment they provide are themselves incompatible. It is formally defined as follows

Definition 1.8.3. For two sets of pairs of protocols $PP^\alpha, PP^\beta \subset \mathscr{C}_\epsilon \times \mathscr{C}$, two transition templates τ^α, τ^β, two admissibility conditions Ψ^α, Ψ^β and two sets of sets of global initial states IS^α, IS^β, the extensions $\mathscr{E}^\alpha = (PP^\alpha, IS^\alpha, \tau^\alpha, \Psi^\alpha)$ and $\mathscr{E}^\beta = (PP^\beta, IS^\beta, \tau^\beta, \Psi^\beta)$ are incompatible iff

$$PP^\alpha \cap PP^\beta = \varnothing \bigvee IS^\alpha \cap IS^\beta = \varnothing \bigvee \tau^{\alpha+\beta} = \varnothing \bigvee \Psi^\alpha \cap \Psi^\beta = \varnothing$$

Reciprocally we define the compatibility relation, $\mathscr{E}^\alpha$ and $\mathscr{E}^\beta$ are compatible iff they are not incompatible.

In order to merge the properties of two extensions in a third one, we will define the intersection of extensions as follows

Definition 1.8.4. For two transition templates τ^A, τ^B we define their intersection τ^{A+B} such that for each environment protocol P_ϵ and each joint protocol P :

$$\tau^{A+B}(P_\epsilon)(P) = \tau^A_{P_\epsilon,P} \cap \tau^B_{P_\epsilon,P}$$

Definition 1.8.5. For two sets pairs of protocols $PP^\alpha, PP^\beta \subset \mathscr{C}_\epsilon \times \mathscr{C}$, two transition templates τ^α, τ^β, two admissibility conditions Ψ^α, Ψ^β and two sets of sets of global initial states IS^α, IS^β, we define the intersection of two compatible extensions $\mathscr{E}^\alpha = (PP^\alpha, IS^\alpha, \tau^\alpha, \Psi^\alpha)$ and $\mathscr{E}^\beta = (PP^\beta, IS^\beta, \tau^\beta, \Psi^\beta)$ as follows :

$$\mathscr{E}^{\alpha+\beta} = (PP^\alpha \cap PP^\beta, IS^\alpha \cap IS^\beta, \tau^{\alpha+\beta}, \Psi^\alpha \cap \Psi^\beta)$$

In the following of this chapter, we will cover different existing distributed computing models as extensions of our general framework. In order to do so, we will have to change sometimes the behaviour of the transition function, this changes will only affect the filtering part in order to represent the alteration of the physical laws. However, changing the filtering correspond to change the physical law therefore their is no natural conservation of the properties of the Byzantine framework. In the future work, we plan to establish a hierarchy of extension based on the preservation of their intrinsic properties. For now, we have only separate extensions that keep the same physical laws from the others.

1.8.1 Extensions preserving the general properties of Byzantine framework

Byzantine Asynchronous Agents

The **Byzantine Asynchronous Extension** is an extension of the general framework in which any agent can become Byzantine, by Byzantine action($fake(i,o)$) or by Byzantine inaction($fail(i)$), at any time however we will restrict the number of Byzantine agents in a whole run to some upper bound f.

We restrict the number of Byzantine agent in a run thanks to the admissibility condition MB, because we need to look at the entire history in order to count the Byzantine agents. It is defined as follows

Definition 1.8.6. For and upper bound $f \leq |\mathcal{A}|$, $MB_f = \{r \in R \mid \forall t \in \mathbb{N}, |\mathcal{A}(Failed(r,t))| \leq f\}$

Moreover, we add the **Fair Schedule** admissibility condition to ensure that no agents can be stalled for sure by the environment, i.e, that within in a finite delay the environment will issue a $go(i)$ for each correct agent i. A Byzantine agent have the possibility to sleep for eternity in order to give him full power on his own knowledge. It is formally defined as follows :

Definition 1.8.7. $FS = \left\{ r \in R \mid \forall (i,t) \in \mathcal{A} \times \mathbb{N}, \exists t' \geq t, \begin{cases} go(i) \in \beta_{g_i}^{t'}(r) \\ \text{or} \\ (i,t') \in Failed(r,t') \end{cases} \right\}$

Definition 1.8.8. For and upper bound $f \leq |\mathcal{A}|$, we denote by $\mathscr{E}^{BA_f} = (\mathscr{C}_\epsilon \times \mathscr{C}, \mathscr{G}, \tau^B, FS \cap MB_f)$ the **Byzantine Asynchronous Extension**.

Synchronous Agents

A **Synchronous agent** is aware of the global time(n.b in the global framework each agent have its own local time that is less or equal than the global time). Therefore, we will wake up each agent each time in order to synchronize the local time with the global time. In order, to allow an arbitrary knowledge in the case of faulty agent - for instance allowing them not to know time - we do not ensure their wake up.

All of this properties will be provided by the environment protocol, and therefore will embedded this into the class $\mathscr{C}_\epsilon^S$ of environment protocols.

Definition 1.8.9 (Synchronous Agents Protocols)**.**

$$\mathscr{C}_\epsilon^S = \{P_\epsilon \in \mathscr{C}_\epsilon \mid \forall t \in \mathbb{N}, \forall S \in P_\epsilon(t), \forall i \in \mathcal{A}, \{go(i), fail(i)\} \cap S \neq \varnothing\}$$

Remark 1.8.10. We allow Byzantine agents not to record the time, thanks to $fail(i)$, in order to give them full power on their knowledge. Moreover, we need this if we want to simulate crash failure for synchronous agents.

Definition 1.8.11. We denote by $\mathscr{E}^S = (\mathscr{C}_\epsilon^S \times \mathscr{C}, \mathscr{G}, \tau^B, R)$ the **Synchronous Agent Extension**.

Reliable Communication

In the **Reliable Communication Extension**, agents can behave arbitrary and the communication - the transmission of message by the environment - is reliable i.e a message send will be delivered by the environment in a finite time. However, in order to allow arbitrary knowledge for faulty agent, we will allow the environment not to deliver a message if the receiver is faulty in order to preserve its knowledge.

We will ensure this properties thanks to the admissibility condition $EDel$ because we need to scan the entire history to know which messages have already been sent.

Definition 1.8.12 (Eventual Message Delivery)**.**

$$EDel = \left\{ r \in R \mid \begin{cases} gsend(i,j,\mu,id) \in r_e(t) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id)) \in r_e(t) \end{cases} \Longrightarrow \exists t', \begin{cases} grecv(j,i,\mu,id) \in r_e(t') \\ \text{or} \\ (j,t') \in Failed(r,t') \end{cases} \right\}$$

Remark 1.8.13.

$$EDel = \left\{ r \in R \mid \begin{cases} gsend(i,j,\mu,id) \in \overline{\beta}_{\epsilon_i}^t(r) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id)) \in \beta_{b_i}^t(r) \end{cases} \Longrightarrow \exists t', \begin{cases} grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t'}(r) \\ \text{or} \\ fail(j) \in \beta_{f_j}^{t'}(r) \end{cases} \right\} \tag{1.64}$$

Definition 1.8.14. We define by $\mathscr{E}^{RC} = (\mathscr{C}_\epsilon \times \mathscr{C}, \mathscr{G}, \tau^B, EDel)$ the **Reliable Communication Extension**.

Synchronous Communication

Synchronous Communication means that when a message is received, it has been sent during the same round. Furthermore, this is a property of channels therefore a Byzantine agent can not force a send however he can still fake a receive because it does not involve message transmission.

Definition 1.8.15. Synchronous Communication Protocols

$$\mathscr{C}_\epsilon^{SC} = \{P_\epsilon \text{ an environment protocol } \mid \forall X \in P_\epsilon(t),\ \text{if } grecv(i,j,\mu,id(j,i,\mu,k,t')) \in X \text{ then } t' = t\} \tag{1.65}$$

Definition 1.8.16. We denote by $\mathscr{E}^{SC} = (\mathscr{C}_\epsilon^{SC} \times \mathscr{C}, \mathscr{G}, \tau^B, R)$ the **Synchronous Communication Extension**.

Multicast Communication

Multicast Communication paradigm means that when a message is sent, it is sent to a group of agents in the same round. In this section, we provide a software based multicast - the protocol of the agents has to guarantee this behaviour - and in the next one we will provide an hardware based multicast - the environment has to guarantee the multicast behaviour.

First, we define a **multicast communication problem** for a group G of multicast agents. For each agent $i \in G$ a set of groups of related agents Mc_i. When i sends a message to an agent j and j is part of a group of Mc_i then the group becomes the multicast group for this message.

Definition 1.8.17. For a group of agents $G \subset \mathcal{A}$, for each agent $i \in G$ we define the set of related agents for agent i as

$$Mc_i \subset 2^{\mathcal{A}} \tag{1.66}$$

And then we define the **Multicast Communication** problem as follows :

$$C = \bigcup_{i \in G} \{(i, Mc_i)\} \tag{1.67}$$

Since, we implement a software based multicast we will naturally use a restriction of the join protocol to do so

Definition 1.8.18. Multicast join protocols For a Multicast Communication problem C and for the group of agents $G \subset \mathcal{A}$

$$\mathscr{C}^{MC_C} = \{P = (P_1, \ldots, P_n) \text{ a joint protocol } \mid \forall i \in G,\ h \text{ a history}, X \in P_i(h), \text{ if } \begin{cases} send(j, \mu_k) \in X \\ \exists Y \in Mc_{i,j} \in Y \end{cases} \text{ then } \forall k \in Y, send(k, \mu_k) \in X\} \tag{1.68}$$

Definition 1.8.19. For a Multicast communication problem C, we denote by $\mathscr{E}^{MC_C} = (\mathscr{C}_\epsilon \times \mathscr{C}^{MC_C}, \mathscr{G}, \tau^B, R)$ the **Multicast Communication Extension**.

Definition 1.8.20. For the Broadcast problem $B = \bigcup_{i \in \mathcal{A}} \{(i, \{\mathcal{A}\})\}$, we denote by $\mathscr{E}^{BC} = (\mathscr{C}_\epsilon \times \mathscr{C}^{MC_B}, \mathscr{G}, \tau^B, R)$ the **Broadcast Communication Extension**.

Coordinated Agents

Coordinated Agents means that there are groups of agents that must be coordinated i.e such agents can wake up at round $t.5$ only if the whole group wakes up $t.5$.

First, we define a **coordinated agents problem** for a group G of coordinated agents. For each agent $i \in G$ a set of groups of related agents Co_i. When agent i is waken up at least one of the coordinated group of Co_i must be too.

Definition 1.8.21. For a group of agents $G \subset \mathcal{A}$, for each agent $i \in G$ we define the set of coordinated agents for agent i as

$$Co_i \subset 2^{\mathcal{A}} \tag{1.69}$$

And then we define the **Coordinated Agents** problem as follows :

$$C = \bigcup_{i \in G} \{(i, Co_i)\} \tag{1.70}$$

To satisfy such a problem we reinforce the rules of agent's wakening through the environment protocol, formally defined as follows :

Definition 1.8.22. For a Coordinated agents problem C and the related group of agents $G \subset \mathcal{A}$

$$\mathscr{C}_\epsilon^{CA_C} = \{P_\epsilon \text{ an environment protocol} \mid \forall t, \forall X \in P_\epsilon(t), \forall i \in G \\ \text{if } go(i) \in X \text{ then } \exists Y \in Co_i, \{go(j) \mid j \in Y\} \subset X\} \tag{1.71}$$

Definition 1.8.23. For a coordinated actions problem C, we denote by $\mathscr{E}^{CA_C} = (\mathscr{C}_\epsilon^{CA_C} \times \mathscr{C}, \mathscr{G}, \tau^B, R)$ the **Coordinated Actions Extension**.

Partially Synchronous Agents

In this extension, each agent must be active at least once each ϕ rounds. To ensure this we need to look into the global history. Therefore, we will define a related admissibility condition PSA_ϕ.

Definition 1.8.24. For a delay $\phi \in \mathbb{N}$, we use the following admissibility condition :

$$PSA_\phi = \left\{ r \in R \mid \forall (i,t) \in \mathcal{A} \times \mathbb{N}, \beta_{g_i}^t(r) = \varnothing \Rightarrow \exists t' \in [\![t+1; t+\phi]\!], \begin{cases} \beta_{g_i}^{t'}(r) \neq \varnothing \\ \text{or} \\ (i,t') \in Failed(r,t') \end{cases} \right\}$$

Definition 1.8.25. For a delay $\phi \in \mathbb{N}$, we denote by $\mathscr{E}^{PSA_\phi} = (\mathscr{C}_\epsilon \times \mathscr{C}, \mathscr{G}, \tau^B, PSA_\phi)$ the **Partially Synchronous Agents Extension**.

Time-bounded Communication

We say that communications are time-bounded if for every channels and for every messages there is an upper-bound(possibly infinite) on the transmission time. Nota bene, the transmission are not reliable, therefore the time-bounded only specify the time window during which the delivering of a message can be done. In order to gain flexibility, bounds can be change depending on the sending time and depending on the message too - for instance a byte of data and picture will not have the same time bound. We encode this bounds in an upper-bound structure defined as follows

Definition 1.8.26. For the first ordinal number ω, the agents $(i,j) \in \mathcal{A}^2$ and the channel $i \mapsto j$, we define the message transmission upper-bound for the channel $i \mapsto j$ as follow

$$\delta_{i \mapsto j} : \quad Msgs \times \mathbb{N} \quad \longrightarrow \quad \mathbb{N} \cup \{\omega\}$$

Now we extend to the message transmission upper-bound for agents $\mathcal{A}$

$$\Delta = \bigcup_{(i,j) \in \mathcal{A}^2} \{\delta_{i \mapsto j}\}$$

Such a time-bounded communication is a physical law of the transmission system therefore we will ensure it thanks to the environment behaviour.

Definition 1.8.27. For an upper-bound structure Δ,

$$\mathscr{C}_\epsilon^{TC_\Delta} = \{P_\epsilon \text{ an environment protocol } \mid \forall t \in \mathbb{N}, \forall X \in P_\epsilon(t), \\ \text{if } grecv(j,i,\mu,id(i,j,\mu,t')) \in X \text{ then } t - \delta_{i\mapsto j}(\mu,t) \leq t' \begin{cases} gsend(i,j,\mu,id(i,j,\mu,t')) \ \in \overline{\beta}_{\epsilon_i}^{t'}(r) \\ \text{or} \\ fake(i, gsend(i,j,\mu,id(i,j,\mu,t'))) \in \beta_{b_i}^{t'}(r) \end{cases} \} \tag{1.72}$$

Definition 1.8.28. For an upper-bound structure Δ, we denote by $\mathscr{E}^{TC_\Delta} = (\mathscr{C}_\epsilon^{TC_\Delta} \times \mathscr{C}, \mathscr{G}, \tau^B, R)$ the **Time-bounded Communication Extension**.

Lock-step synchronous agents

In the **Lock-step Synchronous Agents Extension** agents are synchronous, communication are also synchronous and reliable. Moreover communications are broadcast (physical or not). To implement this extension we only have to combine some previous one(*Synchronous Agents, Synchronous Communication, Broadcast, Physical Broadcast*) as follows :

In case of a software broadcast we have the following extension :

Definition 1.8.29. We denote by $\mathscr{E}^{LSS} = \left((\mathscr{C}_\epsilon^S \cap \mathscr{C}_\epsilon^{SC}) \times \mathscr{C}^{MC_B}, \mathscr{G}, \tau^B, EDel\right)$ the **Lock-step Synchronous Agents Extension** where $B = \bigcup_{i\in\mathcal{A}} \{(i, \{\mathcal{A}\})\}$.

Otherwise, in case of a physical broadcast we have

Definition 1.8.30. We denote by $\mathscr{E}^{PLSS} = \left((\mathscr{C}_\epsilon^S \cap \mathscr{C}_\epsilon^{SC}) \times \mathscr{C}, \mathscr{G}, \tau^{PMC_B}, EDel\right)$ the **Physical Lock-step Synchronous Agents Extension** where $B = \bigcup_{i\in\mathcal{A}} \{(i, \{\mathcal{A}\})\}$.

1.8.2 Other extensions

Here we presents some extensions expressed in our general framework. However, the properties are not naturally transferred from the general one to the extension. Indeed, in the following we will have to change the physical laws (i.e the filtering) function.

Physical Multicast Communication

Physical Multicast Communication paradigm means that when a message is sent, it is sent to a subset of agents in the same round. And this property is ensured by the transmission channels(i.e the environment). For instance it represents the behaviour of an hardware broadcast. Therefore, we will use the filtering function to implement this physical law.

Definition 1.8.31. Formally, for the set $\mathcal{A}$ of agents, a set $X_\epsilon \subset GEvents$ of events attempted by the environment, sets $X_i \subset \overline{GActions_i}$ of actions to be performed by each agent $i \in \mathcal{A}$ and a Multicast problem C

$$filter_i^{PMC_C}(X_1,\ldots,X_n,X_\epsilon) = filter_i^B(X_1,\ldots,X_n,X_\epsilon) \setminus \\ \{send(j,\mu_k) \mid \forall Y \in Mc_{i,j}, \exists x \in Y, send(x,\mu_k) \notin X_i\} \tag{1.73}$$

Definition 1.8.32. For a Multicast communication problem C, we denote by τ^{PMC_C} the transition template τ^B where the $filter^B$ is replaced by $filter^{PMC_C}$.

Definition 1.8.33. For a Multicast communication problem C, we denote by $\mathscr{E}^{PMC_C} = (\mathscr{C}_\epsilon \times \mathscr{C}, \mathscr{G}, \tau^{PMC_C}, R)$ the **Physical Multicast Communication Extension**.

Definition 1.8.34. For the Broadcast problem $B = \bigcup_{i\in\mathcal{A}} \{(i, \{\mathcal{A}\})\}$, we denote by $\mathscr{E}^{PBC} = (\mathscr{C}_\epsilon \times \mathscr{C}, \mathscr{G}, \tau^{PMC_B}, R)$ the **Physical Broadcast Communication Extension**.

Rendezvous Communication

Rendezvous communication refers to a system where sender and receiver must synchronize themselves. This implies that a message sent by a correct agent must be received in the same round. This behaviour is a physical law of the transmission system therefore it is managed by the environment.

First, we define a **Rendezvous problem** for a group G of rendezvous channels. For each directed rendezvous channel $(i,j) \in G$ a set of groups of related channels $rdv_{(i,j)}$. When the channel (i,j) is activated at least one of its related groups must be activated in a rendezvous fashion.

Definition 1.8.35. For a group of channels $G \subset \mathcal{A}^2$, for each directed channel $(i,j) \in G$, from sender i to receiver j, we define the set of coordinated channels for agent i as follows :

$$rdv_{(i,j)} \subset 2^{\mathcal{A}^2} \tag{1.74}$$

And then we define the **Rendezvous** problem as follow

$$C = \bigcup_{x \in G} \{(x, rdv_x)\} \tag{1.75}$$

Thanks to a restriction of the environment protocol, we enforce the rendezvous behaviour of the channels of G i.e related agents can only receive a message during the round of the sending.

Definition 1.8.36.

$$\mathscr{C}_\epsilon^{RdV} = \{P_\epsilon \text{ an environment protocol } \mid \forall X \in P_\epsilon(t), \text{ if } \begin{cases} grecv(j,i,\mu,id(i,j,\mu,t')) \in X \\ (i,j) \in G \end{cases} \text{ then } t = t'\} \tag{1.76}$$

We implement the rules of a *rendezvous problem* C in the filtering function, because C is discerned as a physical law for agents.

Definition 1.8.37. Formally, for the set $\mathcal{A}$ of agents, a set $X_\epsilon \subset GEvents$ of events attempted by the environment, sets $X_i \subset \overline{GActions_i}$ of actions to be performed by each agent $i \in \mathcal{A}$ and a Rendezvous problem C

$$filter_i^{RdV_C}(X_1,\ldots,X_n,X_\epsilon) = filter_i^B(X_1,\ldots,X_n,X_\epsilon) \setminus \{gsend(i,j,\mu,id) \mid \forall X \in rdv_{i\mapsto j}, \exists (k \mapsto l) \in X, \begin{cases} gsend(k,l,\mu',id') \notin X_j \\ \text{or} \\ fake(k, gsend(k,l,\mu',id')) \notin X_\epsilon \end{cases} \tag{1.77}$$

Definition 1.8.38. For a rendezvous problem C, we denote by τ^{RdV_C} the transition template τ^B where $filter^B$ is replaced by $filter^{RdV_C}$.

Definition 1.8.39. For a rendezvous problem C, we denote by $\mathscr{E}^{RdV_C} = (\mathscr{C}_\epsilon^{RdV_C} \times \mathscr{C}, \mathscr{G}, \tau^{RdV_C}, R)$ the **Rendezvous Communication Extension**.

Chapter 2

The Byzantine Asynchronous Extension

2.1 Introspection

Lemma 2.1.1. *For an upper bound on the number of Byzantine agents in a run $1 \leq f \leq |\mathcal{A}|$, an agent context $\chi = ((P_\epsilon, \mathscr{G}(0), \tau, \Psi), P) \in \mathscr{E}^{BA_f}$, the χ-based interpreted system $I = (R^\chi, \pi)$, a run $r \in R^\chi$ and a point $(i,t) \in \mathcal{A} \times \mathbb{N}$ such that $t > 0$, there is a run $r' \in R^\chi$ and a timestamp t' that :*

1. $(\forall m \leq t) r'_i(m) = r_i(m)$
2. $(\forall t' \geq 1)(I, r', t') \models fail_i$
3. $(\forall (j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t, i \neq j)\, r'_j(m) = r'_j(0)$
4. $(\forall (j,m) \in \mathcal{A} \times \mathbb{N}, m < t)\, \beta^m_{\epsilon_j}(r') = \varnothing$

Proof. In order to prove the existence of such a run r', let us build it thanks to the four following rules. First, the new run start in the same configuration than the initial one, we denote this rule by *Init* which formally correspond to $r'(0) = r(0)$. Then we will define two distinct behaviours : one before time t and one after. Before time t for agent i, any action and event in r will become a Byzantine one in r' : this is formally for $m < t$ *Fake*$_i$ $\begin{cases} \beta^m_{b_i}(r') & = \beta^m_{b_i}(r) \cup \{fake(i,u) \mid u \in \beta^m_i(r) \cup \overline{\beta}^m_{\epsilon_i}(r)\} \\ \beta^m_{f_i}(r') & = \{fail(i)\} \cup \beta^m_{b_i}(r') \\ \overline{\beta}^m_{\epsilon_i}(r') & = \varnothing \\ \beta^m_{g_i}(r') & = \varnothing \end{cases}$.

And an other one for agent $j \neq i$ before time t, j will remain frozen by the Environment until time t : this formally defined for $m < t$ by *Freeze* $\beta^m_{\epsilon_j}(r') = \varnothing$. And finally, for time $m \geq t$, we will just apply the transition relation in order to construct the end of the run in a consistent way, this operation is denoted by *Extending*.

Now, let us prove that the previous rules do not contradict some properties of the transition relation. In order to do so, we need to prove first that for each $m \in \mathbb{N}, \beta^m_\epsilon(r') \subset P_\epsilon(m)$. For $m \geq t$ it is trivial due to *Extending*, for $m < t$ for all $j \neq i$ we have $\beta^m_{\epsilon_j}(r') = \varnothing$ therefore we have only to prove that $\beta^m_{\epsilon_i}(r') \subset P_\epsilon(m)$. Moreover thanks to $Fake_i$ $\beta^m_{\epsilon_i}(r') \subset \{fail(i), fake(i,u) \mid u \in \overline{GActions}_i \sqcup \overline{GEvents}_i\}$ then by (1.2.4) we have $\beta^m_{\epsilon_i}(r') \subset P_\epsilon(m)$. Eventually, we have to check that $\forall X_1 \in GActions_1, \ldots, X_n \in GActions_n, \forall j \in \mathcal{A}, \forall m < t$, $\beta^m_{\epsilon_j}(r') = X =_{def} filter_\epsilon\left(r'(m), \beta^m_{\epsilon_j}(r'), X_1, \ldots X_n\right)$. For $m < t$ and $j \neq i$ it is trivial since $\beta^m_{\epsilon_j}(r') = \varnothing$, now let us deal with i by case disjunction on $u \in \beta^m_{\epsilon_i}(r')$:

Case I: $u = fake(i, internal(i,o))$. $u \in X$ because $\beta^m_{g_i}(r') = \varnothing$.

Case II: $u = fake(i, external(i,o))$. $u \in X$ because $\overline{\beta}^m_{\epsilon_i}(r') = \varnothing$.

Case III: $u = fake(i, gsend(i,j,\mu, id(a,b,c,d,e)))$. $gsend(i,j,\mu,id(a,b,c,d,e)) \in \beta^m_i(r)$ therefore $a = i$, $b = j$, $c = \mu$ and $e = m$ thanks to 1.2.12. Hence, $u \in X$.

Case IV: $u = fake(i, grecv(i,j,\mu,id))$. $u \in X$ because $\overline{\beta}^m_{\epsilon_i}(r') = \varnothing$.

Case V: $u = fail(i)$. $u \in X$ because filter does not affect $fail()$.

Now, let us prove that $r' \in R^\chi$, thanks to *Init* $r'(0) = r(0) \in \mathscr{G}(0)$ then we have only to take care of the admissibility condition. The Fair Schedule(1.8.7) it is ensured by *Extending*, now we will take care of the MB_f condition for $m < t$ thanks to *Extending*. The only Byzantine agent is i by construction and $f \geq 1$ hence $r' \in MB$. In conclusion of this first part we have $r' \in R^\chi$.

Let us prove the first property of the lemma by induction on $m \in [\![0; t]\!]$.

Case I: $m = 0$. It holds trivially thanks to *Init*

Case II: $m >$ for agent $j \neq i$. $\beta^m_{\epsilon_j}(r') = \varnothing$ then thanks to (1.2.22) $r'_j(m) = r'_j(m-1)$. Then by induction property $r'_j(m) = r'_j(m-1)$.

Case III: $m >$ for agent i. $\beta^m_{\epsilon_j}(r') = \{fail(i)\}$ $r'_j(m) = r'_j(m-1)$ thanks to (1.2.22). Otherwise, thanks to *Fake* $\sigma\left(\beta^m_{\epsilon_j}(r')\right) = \sigma\left(\beta^m_{\epsilon_j}(r') \cup \beta^m_j(r')\right)$ therefore $r'_j(m) = r'_j(m-1)$. Then by induction property $r'_j(m) = r'_j(m-1)$.

since $\beta^m_g(r') = \varnothing$ by $Fake_i$.

Part A. Building rules for $\alpha^m_i(r)$ and $\alpha^m_{\epsilon_i}(r)$:

(a) *Init* : $\begin{cases} \alpha^0_{f_i}(r') & = \{fail(i)\} \cup \alpha^0_{b_i}(r') \\ r'(0) & = r(0) \end{cases}$

(b) *Fake*$_i$ for $m < t$: $\begin{cases} \alpha^m_{b_i}(r') & = \alpha^m_{b_i}(r) \cup \{fake(i,u) \mid u \in \alpha^m_i(r) \cup label^{-1}\left(\overline{\alpha}^m_{\epsilon_i}(r)\right)\} \\ \overline{\alpha}^m_{\epsilon_i}(r') \cup \alpha^m_{g_i}(r') & = \varnothing \\ \alpha^m_i(r') & = \varnothing \end{cases}$

(c) *Freeze* for $m < t, j \in \mathcal{A}\backslash\{i\}$: $\begin{cases} \alpha^m_{\epsilon_j}(r') & = \varnothing \\ \alpha^m_j(r') & = \varnothing \end{cases}$

(d) *Extending* for $m \geq t$:

Extends the previous partial run r' by iterating the transition function τ according to γ.

Part B. Let us check that there is no conflict between the previous rules and τ(Def 1.2.24) :
In fact, we will check for $\mathbf{m<t}$(*Extending* ensures this for $m \geq t$) that :

(a) The rules are reachable form the protocol

- $\alpha^m_\epsilon(r') \in label(P_\epsilon(m), m) \cup \{\varnothing\}$:

  Case I: $j \neq i$. $\alpha^m_{\epsilon_j}(r') =_{Freeze} \varnothing$
  Case II: $i = j$. $\alpha^m_{\epsilon_i}(r') \subset_{Fake_i} \{fake(i,u) \mid u \in \overline{GActions} \cup \overline{Events}\}$
  Then it holds with (**??**).
- $\alpha^m_j(r') \in label_j\left(P_j\left(r'_j(m)\right), m\right) \cup \{\varnothing\}$:

  $\begin{cases} \forall j \in \mathcal{A}\backslash\{i\}, \alpha^m_j(r') =_{Freeze} \varnothing \\ \alpha^m_i(r') =_{Fake_i} \varnothing \end{cases}$ therefore it holds.

(b) The rules pass through $filter_\epsilon$(Def 1.2.12) withthout any damage

i. If $fake(k,a) \in \alpha^m_{b_k}(r'), a \in \overline{GActions}$ then $a \notin \alpha^m_k(r')$:

Case I: $k \neq i$. $\alpha^m_{\epsilon_k}(r') =_{Freeze} \varnothing$
Case II: $k = i$. $\alpha^m_k(r') =_{Fake_i} \varnothing$

ii. If $fake(k, external(k,e)) \in \alpha^m_{b_k}(r')$ then $external(k,e) \notin \overline{\alpha}^m_{\epsilon_k}(r')$:

Case I: $k \neq i$. $\alpha^m_{\epsilon_k}(r') =_{Freeze} \varnothing$
Case II: $k = i$. $\overline{\alpha}^m_{\epsilon_k}(r') =_{Fake_i} \varnothing$

iii. If $fake(k, recv(j,\mu)) \in \alpha^m_{b_k}(r')$ then $\forall id \in \mathbb{N}, grecv(k,j,\mu,id) \notin \overline{\alpha}^m_{\epsilon_k}(r')$:

Case I: $k \neq i$. $\alpha^m_{\epsilon_k}(r') =_{Freeze} \varnothing$
Case II: $k = i$. $\overline{\alpha}^m_{\epsilon_k}(r') =_{Fake_i} \varnothing$

iv. If $grecv(k,j,\mu,id) \in \overline{\alpha}^m_{\epsilon_k}(r')$ then $\begin{cases} \exists m' \leq m, gsend(j,k,\mu,id) \in \beta^{m'}_j(r') \neq \varnothing \\ \text{or} \\ \exists m' \leq m, fake(j, gsend(j,k,\mu,id)) \in \beta^{m'}_{b_j}(r') \end{cases}$:

Case I: $k \neq i$. $\alpha^m_{\epsilon_k}(r') =_{Freeze} \varnothing$
Case II: $k = i$. $\overline{\alpha}^m_{\epsilon_k}(r') =_{Fake_i} \varnothing$

(c) The rules pass through $filter_j$ for each agent $j \in \mathcal{A}$ (Def 1.2.12) withthout any damage (ie $\beta^m_j(r') = \alpha^m_j(r')$)

- If $\alpha^m_g(r') = \varnothing$ then $\alpha^m_j(r') = \varnothing$:

$$\begin{cases} \forall j \neq i, \alpha^m_j(r') =_{Freeze} \varnothing \\ \alpha^m_i(r') =_{Fake_i} \varnothing \end{cases}$$

Part C. Let us check that $r' \in R$:

- $r'(0) \in \mathscr{G}(0)$:
  $r'(0) =_{Init} r(0) \in_{r \in R^P} \mathscr{G}(0)$

Part D. Let us check that $r' \in R^P$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by Extending

(b) $r \in FS$: ensured by Extending

(c) $r \in MB$: we will check this for $m \leq t$ (*Extending* ensures this for $m > t$)
It holds because $\mathcal{A}(Failed(r,m)) \subset \{i\}$:

i. With *Init* $i \in \mathcal{A}(Failed(r,1))$

ii. $\forall j \in \mathcal{A}\backslash\{i\}, \alpha^m_{f_j}(r') =_{Freeze} \varnothing$

Part E. Lets us prove $\forall j \in \mathcal{A}\backslash\{i\}, m \leq t, r'_j(m) = r_j(0)$:
Let do this by induction on m :

Case I: $m = 0$. It holds with *Init*
Case II: $0 \leq m < t$.
With *Freeze* we have $\beta^m_{\epsilon_j}(r') = \varnothing$ therefore :

$$\begin{aligned} r'_j(m+1) &=_{Def1.2.22} r'_j(m) \\ &=_{\text{induction hyp}} r_j(0) \end{aligned}$$

Part F. Lets us prove $\forall m \leq t, r'_i(m) = r_i(m)$:
Let do this by induction on m :

Case I: $m = 0$. It holds with *Init*
Case II: $0 \leq m < t$.
If $\beta^m_{\epsilon_i}(r') = \varnothing$ it holds by the same proof as for $j \neq i$ else :

$$\begin{aligned} r'_i(m+1) &=_{Def1.2.22} \left[\sigma\left(\overline{\beta}^m_{\epsilon_i}(r') \cup \beta^m_{b_i}(r')\right) : r'_i(m)\right. \\ &=_{Freeze} \sigma(\beta^m_{b_i}(r')) : r'_i(m) \\ &=_{\text{induction hyp}} \sigma(\beta^m_{b_i}(r')) : r_i(m) \\ &=_{Fake_i} r_i(m+1) \end{aligned}$$

Part G. $\forall t' \geq 1, (I^P, r', t') \models fail_i$:
It holds by *Init*.

Conclusion : $r' \in R^P$ verifies *1., 2., 3., 4.*

□

Lemma 2.1.2. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, f \geq 1, r \in R^P, (i,t) \in \mathcal{A} \times \mathbb{N}, (I^P, r, t) \nvDash K_i \overline{occurred}(o)$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, (i,t) \in \mathcal{A} \times \mathbb{N}$, and for the sake of contradiction $(I^P, r, t) \models K_i \overline{occurred}(o)$,

With Lem2.1.1 we have $r' \in R^P$ such that r,r' verify the properties of Lem2.1.1 for the point (i,t).

Lets us prove $\forall u \in \overline{Actions} \sqcup \overline{Events}, (I^P, r', t) \nvDash \overline{occurred}(u)$:
Let $m \leq t, u \in \overline{Actions} \sqcup \overline{Events}$,

Case I: $j \in \mathcal{A} \backslash \{i\}$.
Then $r'_j(m) =_{3.} r_j(0)$ therefore $(I^P, r', t) \nvDash \overline{occurred}_{(j,m)}(u)$
Case II: $j = i$.
$\overline{\beta^m_{\epsilon_i}}(r') \cup \beta^m_{g_i}(r') =_{4.} \varnothing$ therefore $(I^P, r', t) \nvDash \overline{occurred}_{(i,m)}(u)$

Let us conclude :
$\begin{cases} (I^P, r', t) \nvDash \overline{occurred}(o) \\ (I^P, r', t) \models_{r'_i(t) = r_i(t)} \overline{occurred}(o) \end{cases}$ Contradiction.

Conclusion : $(I^P, r, t) \nvDash K_i \overline{occurred}(o)$

□

Lemma 2.1.3. *Let* $r \in R^P, f \geq 1, (i,t) \in \mathcal{A} \times \mathbb{N}$, *then* $(I^P, r, t) \nvDash K_i correct_i$

Proof. Let $r \in R^P, f \geq 1, (i,t) \in \mathcal{A} \times \mathbb{N}$,
With Lem2.1.1 we have $r' \in R^P$ such that r,r' verify the properties of Lem2.1.1 for the point (i,t).
And the current lemma holds with *1.* and *2.*. □

Remark 2.1.4. Let $r \in R^P, (i,t) \in \mathcal{A} \times \mathbb{N}, t' \leq t, r_i(t) = \cdots : E : r_i(t'-1)$,

$$\{internal(\dots,\dots), send(\dots,\dots) \in E\} \notin P_i(r_i(t'-1)) \cup \{\varnothing\} \Rightarrow (I^P, r, t') \models K_i fail_i$$

Lemma 2.1.5. *Let* $r \in R^P, f \geq 1, (i,t) \in \mathcal{A} \times \mathbb{N}, t \geq 1$, *then* $(I^P, r, t) \nvDash K_i fail_j$

Proof. Same proof than for (2.1.3) □

Lemma 2.1.6. *Let* $r \in R^P, f \geq 2, (i,t) \in \mathcal{A} \times \mathbb{N}, t \geq 1$, *then* $(I^P, r, t) \nvDash K_i correct_j$

Proof. Same kind of proof than for (2.1.3) except for *Init* we add $\beta^0_{f_j}(r') = \{fail(j)\}$ □

Definition 2.1.7 (Agreement on a point)**.** For a agent-context χ and two runs $r, r' \in R^{\chi}$, we say that r and r' **agree on** a point $(i,t) \in \mathcal{A} \times \mathbb{N}$ iff

1. $r_i(t) = r'_i(t)$ — the local states of i at t are identical
2. $\overline{\beta}^t_{\epsilon_i}(r) = \overline{\beta}^t_{\epsilon_i}(r')$ — correct external actions imposed on i in round $t.5$ are identical
3. $\beta^t_i(r) = \beta^t_i(r')$ — intended protocol actions of i in round $t.5$ are identical
4. $\beta^t_{g_i}(r) = \beta^t_{g_i}(r')$ — no difference is observed as to whether i is awoken for round $t.5$
5. $\beta^t_{b_i}(r) = \beta^t_{b_i}(r')$ — faulty behavior of i in round $t.5$ is identical

Now, let us extend the agreement to a set of points $S \subset \mathcal{A} \times \mathbb{N}$. We say that r and r' agree on S iff

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

Remark 2.1.8.

- Points 1–4 correspond to the definition of agreement in [2].
- To handle the Byzantine nodes we have added 5 in order to ensure identical round execution. However, in this model we allow the difference in whether the node is failed.

Lemma 2.1.9. *For an agent-context χ, two runs $(r, r') \in R^{\chi 2}$ and a point $(i,t) \in \mathcal{A} \times \mathbb{N}$*

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

Proof. Let $(r, r') \in R^{\chi 2}$, r and r' agree on $(i,t) \in \mathcal{A} \times \mathbb{N}$

$$\begin{cases} \overline{\beta}^t_{\epsilon_i}(r') & =_{2)}\ \overline{\beta}^t_{\epsilon_i}(r') \\ \beta^t_{g_i}(r') & =_{3)}\ \beta^t_{g_i}(r') \\ \beta^t_i(r') & =_{3}\ \beta^t_i(r') \\ \beta^t_{b_i}(r') & =_{4)}\ \beta^t_{b_i}(r') \end{cases} \text{ therefore by (1.2.22) } r_i(t+1) = r'_i(t+1)$$

□

Remark 2.1.10. For an agent-context χ, two runs $(r, r') \in R^{\chi 2}$ and a point $(i,t) \in \mathcal{A} \times \mathbb{N}$,

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

Proof. Let $\mathcal{A} = \{i\}, Msgs = \varnothing, Int_i = \varnothing, Ext_i = \{Start\}, \Sigma_i = \{s_0\}$,

Let r such that $\begin{cases} r_i(0) = s_0 \\ \overline{\beta}^0_{\epsilon_i}(r) = \{external(i, Start)\} \end{cases}$ and r' such that $\begin{cases} r'_i(0) = s_0 \\ \beta^0_{b_i}(r') = \{fake(i, external(i, Start))\} \end{cases}$

Therefore $\begin{cases} r_i(0) = r'_i(0) \\ r_i(1) = r'_i(1) \end{cases}$ and r, r' does not agree on $(i,0)$ □

Chapter 3

From Past toward Knowledge

DO NOT USE α but β We are looking for a necessary and sufficient condition on the structure of $Past(r,(i,t))$ to ensure that agent i knows at time t that some action(event) have already happened in a non-faulty way. Due to Lemma2.1.2, "knows" means $fail_i \vee K_i\overline{occurred}()$.

3.1 Runs' transmutation lemmas

Lemma 3.1.1 (Lemma 7.5)**.** *Let P a joint protocol, $r \in R^P$, $\theta = (i,t) \in \mathcal{A} \times \mathbb{N}$. Then there is a $r' \in R^P$ such that:*

A. r and r' agree on V_θ^r

B. $\forall (j,t') \in \mathcal{A} \times \mathbb{N}, t' \leq t,\ (j,t') \notin V_\theta^r \Rightarrow \beta_{\epsilon_j}^{t'}(r') = \varnothing$

C. $Bad_{V_\theta^r}(r) = Bad_{V_\theta^{r'}}(r')$

D. $\mathcal{A}\left(Failed_{V_\theta^{r'}}(r')\right) = \mathcal{A}\left(Bad_{V_\theta^{r'}}(r')\right)$

E. $\forall \beta \in V_\theta^r, o \in \overline{Actions} \sqcup \overline{Events}$:

$$(I^P, r, t) \models \overline{occurred}_\beta(o) \Leftrightarrow (I^P, r', t) \models \overline{occurred}_\beta(o)$$
$$(I^P, r, t) \models fake_\beta(o) \Leftrightarrow (I^P, r', t) \models fake_\beta(o)$$

Proof. Let $r \in R^P, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$,
Let us build such a $r' \in R^P$

Part A. Building rules for $\alpha_i^m(r)$ and $\alpha_{\epsilon_i}^m(r)$:

(a) *Init* :
$r'(0) = r(0)$

(b) *Out* for $m \leq t$: $\overline{\beta}_{\epsilon_j}^m(r') = \begin{cases} \text{If } (j,m) \in V_\theta^r \text{ then } \overline{\beta}_{\epsilon_j}^m(r) \\ \text{Else } \varnothing \end{cases}$

(c) *In* for $m \leq t$: $\begin{cases} \beta_j^m(r') = \begin{cases} \text{If } (j,m) \in V_\theta^r \text{ then } \beta_j^m(r) \\ \text{Else } \varnothing \end{cases} \\ \beta_{g_j}^m(r') = \begin{cases} \text{If } (j,m) \in V_\theta^r \text{ then } \beta_{g_j}^m(r) \\ \text{Else } \varnothing \end{cases} \end{cases}$

(d) *Fake* for $m \leq t$: $\beta_{f_j}^m(r') = \begin{cases} \text{If } (j,m) \in V_\theta^r \text{ then } \beta_{b_j}^m(r) \\ \text{Else } \varnothing \end{cases}$

(e) *Extending* for $m > t$:

Extends the previous partial run r' by iterating the transition function τ.

Part B. Let us prove A. :
Let $(j,m) \in V_\theta^r$, let prove by induction of m that r, r' agree on (j,m)

Case I: $m = 0$. A. holds with *Init*
Case II: $m > 0$. By *Locality* $(j, m-1) \in V_\theta^r$

Let us prove *agreement 1.*
By induction on $(j, m-1) \in V_\theta^r$ we have r and r' agree on $(j, m-1)$.
Then with Rq2.1.9, $r_j(m) = r'_j(m)$

Let us prove *agreement 2.* $\begin{cases} (j,m) \in V_\theta^r \\ Out \end{cases}$ therefore $\overline{\beta}_{\epsilon_j}^m(r) = \overline{\beta}_{\epsilon_j}^m(r')$

Let us prove *agreement 3.* $\begin{cases} (j,m) \in V_\theta^r \\ In \end{cases}$ therefore $\beta_j^m(r) = \beta_j^m(r')$

Let us prove *agreement 4.* $\begin{cases} (j,m) \in V_\theta^r \\ Fake \end{cases}$ therefore $\beta_{b_j}^m(r) = \beta_{b_j}^m(r')$

Conclusion : r,r' agree on (j,m)

Part C. Let us prove B. :
Let $(j,m) \notin V_\theta^r, m \leq t$

$\begin{cases} In \Rightarrow \beta_{g_j}^m(r') = \varnothing \\ Out \Rightarrow \overline{\beta}_{\epsilon_j}^m(r') = \varnothing \\ Fake \Rightarrow \beta_{f_j}^m(r') = \varnothing \end{cases}$ therefore $\beta_{\epsilon_j}^m(r') = \varnothing$

Part D. Let us prove C. :

(a) Let prove $Bad_{V_\theta^r}(r) \subset Bad_{V_\theta^{r'}}(r')$

Case I: $Bad_{V_\theta^r}(r) = \varnothing$. Trivial
Case II: $Bad_{V_\theta^r}(r) \neq \varnothing$.
Let $(j,m) \in Bad_{V_\theta^r}(r)$, $\beta_{b_j}^{m-1}(r) \neq \varnothing$ then with *Fake* $\beta_{b_j}^{m-1}(r') \neq \varnothing$.
Hence $(j,m) \in Bad_{V_\theta^r}(r') =_{\text{with 3.1.2.1}} Bad_{V_\theta^{r'}}(r')$
Conclusion : $Bad_{V_\theta^r}(r) \subset Bad_{V_\theta^{r'}}(r')$

(b) Let prove $Bad_{V_\theta^{r'}}(r') \subset Bad_{V_\theta^r}(r)$ same proof than (a)

Conclusion : $Bad_{V_\theta^{r'}}(r') = Bad_{V_\theta^r}(r)$

Part E. Let us prove D. :
Fake ensures that $\forall m \leq t, j \in \mathcal{A}, fail(j) \notin \beta_{\epsilon_j}^m(r')$.
Therefore $\mathcal{A}\left(Failed_{V_\theta^r}(r',t)\right) = \mathcal{A}\left(Bad_{V_\theta^r}(r',t)\right)$

Part F. Let us prove E. :
Let $\eta = (j,m) \in V_\theta^r, 0 < m \leq t$,

(a) Let prove $(I^P, r, t) \models \overline{occurred}_\eta(o) \Leftrightarrow (I^{r'}, t, \models)\overline{occurred}_\eta(o)$

Case I: $(I^P, r, t) \models \overline{occurred}_\eta(o)$.

Case 1: $o \in \overline{Actions}$. ie $o \in \beta_j^{m-1}(r)$ and $\beta_{g_j}^{m-1}(r) \neq \varnothing$

With In : $\begin{cases} o \in \beta_j^{m-1}(r) = \beta_j^{m-1}(r') \\ \beta_{g_j}^{m-1}(r') = \beta_{g_j}^{m-1}(r) \neq \varnothing \end{cases}$

Therefore $(I^{r'}, t, \models) \overline{occurred}_\eta(o)$

Case 2: $o \in \overline{Events}$. ie $o \in \overline{\beta}_{\epsilon_j}^{m-1}(r)$

With Out : $o \in \overline{\beta}_{\epsilon_j}^{m-1}(r) = \overline{\beta}_{\epsilon_j}^{m-1}(r')$.

Therefore $(I^{r'}, t, \models) \overline{occurred}_\eta(o)$

Case II: $(I^{r'}, t, \models) \overline{occurred}_\eta(o)$. Same proof as Case I

(b) Let prove $(I^P, r, t) \models fake_\eta(o) \Leftrightarrow (I^{r'}, t, \models) fake(\beta, o)$

Case I: $(I^P, r, t) \models fake_\beta(o)$. ie $fake(j, o) \in \beta_{b_j}^{m-1}(r)$

With $Fake$: $fake(j, o) \in \beta_{b_j}^{m-1}(r) = \beta_{b_j}^{m-1}(r')$

Therefore $(I^{r'}, t, \models) fake_\eta(o)$

Case II: $(I^{r'}, t, \models) fake_\eta(o)$. Same proof as Case I

Part G. Let us check that there is no conflict between the previous rules and τ(Def 1.2.24) :
In fact, we will check for $\mathbf{m} \leq \mathbf{t}$(*Extending* ensures this for $m > t$) that :

(a) The rules are reachable form the protocol

- $\exists X \in label(P_\epsilon(m), m) \cup \{\varnothing\}, \beta_\epsilon^m(r') = filter_\epsilon(r'(m-1), X, \alpha_1^m(r'), \ldots, \alpha_n^m(r'))$:

Case I: $(j, m) \in V_\theta^r$.

Let $X_{e_j} = \alpha_{\epsilon_j}^m(r) \backslash \{fail(j)\}$ and $\forall k \in \mathcal{A}, X_k = \begin{cases} \text{If } (k, m) \in V_\theta^r \text{ then } \alpha_j^m(r) \\ \text{Else } \varnothing \end{cases}$

We have $X_{e_j} \subset \alpha_{\epsilon_j}^m(r) \cup \{fail(j)\} \in_{r \in R^P} label(P_\epsilon(m), m) \cup \{\varnothing\}$.
Moreover $\alpha_{\epsilon_j}^m(r) \in_{r \in R^P} label(P_\epsilon(m), m) \cup \{\varnothing\}$.
Therefore with (**??**) we have $X_{e_j} \in label(P_\epsilon(m), m) \cup \{\varnothing\}$.

So let $\alpha_{\epsilon_j}^m(r') = X_{e_j}$, and now let us brove than it travers $filter_\epsilon$ according to r.

Case 1: $grecv(j, k, \mu, id) \in \overline{\alpha}_{\epsilon_j}^m(r') =_{In} \overline{\alpha}_{\epsilon_j}^m(r)$.

Case a: $grecv(j, k, \mu, id) \in \overline{\beta}_{\epsilon_j}^m(r)$.

$\exists m' \leq m, \begin{cases} gsend(k, j, \mu, id) \in \beta_k^{m'}(r) \text{ and } \beta_{g_k}^{m'}(r) \neq \varnothing \\ \text{or} \\ fake(k, gsend(k, j, \mu, id)) \in \beta_{b_k}^{m'}(r) \end{cases}$

Therefore $(k, m') \in V_\theta^r$ and then with In and $Fake$:

$\begin{cases} gsend(k, j, \mu, id) \in \beta_k^{m'}(r') \text{ and } \beta_{g_k}^{m'}(r') \neq \varnothing \\ \text{or} \\ fake(k, gsend(k, j, \mu, id)) \in \beta_{b_k}^{m'}(r') \end{cases}$

Hence $grecv(j, k, \mu, id) \in \overline{\beta}_{\epsilon_j}^m(r')$

Case b: $grecv(j, k, \mu, id) \notin \overline{\beta}_{\epsilon_j}^m(r)$.

$\nexists m' \leq m, \begin{cases} gsend(k, j, \mu, id) \in \beta_k^{m'}(r) \text{ and } \beta_{g_k}^{m'}(r) \neq \varnothing \\ \text{or} \\ fake(k, gsend(k, j, \mu, id)) \in \beta_{b_k}^{m'}(r) \end{cases}$

Therefore $(k, m') \notin V_\theta^r$ hence with In and $Fake$:

$$\begin{cases}\forall m' \leq m, \beta_k^{m'}(r') = \varnothing \\ \forall m' \leq m, \beta_{\epsilon_k}^{m'}(r') = \varnothing\end{cases}$$

Hence $grecv(j,k,\mu,id) \notin \overline{\beta}_{\epsilon_j}^m(r')$

Case 2: $fake(j,u) \in \overline{\alpha}_{\epsilon_j}^m(r') =_{In} \overline{\alpha}_{\epsilon_j}^m(r)$.
With *In* and *Out* we have :

$$fake(j,u) \in \overline{\beta}_{\epsilon_j}^m(r') \Leftrightarrow fake(j,u) \in \overline{\beta}_{\epsilon_j}^m(r)$$

Case II: $(j,m) \notin V_\theta^r$. ie $\alpha_{\epsilon_j}^m(r') =_{m \leq t} \varnothing$

- $\dfrac{\exists X \in label_j\left(P_j\left(r'_j(m-1)\right),m\right) \cup \{\varnothing\}, \beta_j^m(r') = filter_i\left(X, \alpha_{\epsilon_i}^m(r)\right) :}{\text{Let } X_j = \alpha_j^m(r) \in_{r \in R^P} label_j\left(P_j\left(r'_j(m-1)\right),m\right) \cup \{\varnothing\}}$

$$\begin{aligned} filter_i\left(X_j, \alpha_{\epsilon_i}^m(r)\right) &= filter_i\left(X_j, \alpha_{\epsilon_i}^m(r) \backslash \{fail(i)\}\right) \\ &= filter_i\left(X_j, \alpha_{\epsilon_i}^m(r')\right) \\ &= \beta_j^m(r') \end{aligned}$$

Part H. Let us check that $r' \in R^{ba}$:

- $\dfrac{r'(0) \in \mathscr{G}(0) :}{r'(0) =_{Init} r(0) \in \mathscr{G}(0)}$

Part I. Let us check that $r' \in R^P$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by Extending

(b) $r \in FS$: ensured by Extending

(c) $r \in MB$: we will check this for $m \leq t$(*Extending* ensures this for $m > t$)

$$\begin{aligned} |\mathcal{A}\left(Failed_{V_\theta^{r'}}(r')\right)| &=_{D.} |\mathcal{A}\left(Bad_{V_\theta^{r'}}('r)\right)| \\ &=_{C.} |\mathcal{A}\left(Bad_{V_\theta^r}(r)\right)| \\ &\leq_{r \in R^P} f \end{aligned}$$

Conclusion : $r' \in R^P$ verifies A. B. C. D. E.

□

Remark 3.1.2. D. implies that $\forall \beta \in V_\theta^r, (I^P, r, t) \models occurred_\beta(o) \Leftrightarrow (I^{r'}, t, \models) occurred_\beta(o)$

Corollary 3.1.2.1. *Let* $(r,r') \in {R^P}^2, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$,

$$\begin{cases} \textit{1) r,r' agree on } V_\theta^r \\ \textit{2) } \forall (j,t') \in \mathcal{A} \times \mathbb{N}, t' \leq t, (j,t') \notin V_\theta^r \Rightarrow \beta_{\epsilon_j}^{t'}(r') = \varnothing \end{cases} \Rightarrow Past(r,\theta) = Past(r',\theta)$$

Proof. Let $(r,r') \in {R^P}^2, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}, Past(r,\theta) = (V_\theta^r, E_\theta^r)$,

1. $\dfrac{\text{Let us prove that } V_\theta^r \subset V_\theta^{r'} \text{ and } E_\theta^r \subset E_\theta^{r'} :}{\text{Let } (\eta_j = (j,t_j), \eta_k = (k,t_k)) \in E_\theta^r}$

Case I: $(\eta_j, \eta_k) \in E^r_{loc}$. therefore $(\eta_j, \eta_k) \in E^{r'}_{loc}$
Case II: $(\eta_j, \eta_k) \in E^r_{msg}$.

Then $\begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \ \beta_i^{t_j}(r) \text{ and } \beta_{g_i}^{t_j}(r) \neq \varnothing \\ \text{or} \\ fake\left(i, gsend(i,j,\mu,id)\right) \in \beta_{b_i}^{t_j}(r) \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t_k-1}(r) \end{cases}$

And r,r' agree on η_j and η_k hence $\begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \ \beta_i^{t_j}(r') \text{ and } \beta_{g_i}^{t_j}(r') \neq \varnothing \\ \text{or} \\ fake\left(i, gsend(i,j,\mu,id)\right) \in \beta_{b_i}^{t_j}(r') \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t_k-1}(r') \end{cases}$

Eventually $(\eta_j, \eta_k) \in E^{r'}_{msg}$

2. Let us prove that $V^{r'}_\theta \subset V^r_\theta$ and $E^{r'}_\theta \subset E^r_\theta$:
Let $(\eta_j = (j, t_j), \eta_k = (k, t_k)) \in E^{r'}_\theta$

Case I: $(\eta_j, \eta_k) \in E^{r'}_{loc}$. therefore $(\eta_j, \eta_k) \in E^r_{loc}$
Case II: $(\eta_j, \eta_k) \in E^{r'}_{msg}$.

Then $\begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \ \beta_i^{t_j}(r') \text{ and } \beta_{g_i}^{t_j}(r') \neq \varnothing \\ \text{or} \\ fake\left(i, gsend(i,j,\mu,id)\right) \in \beta_{b_i}^{t_j}(r') \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t_k-1}(r') \end{cases}$

with 2) we have $(\eta_j, \eta_k) \in (V^r_\theta)^2$ then by 1) $\begin{cases} \begin{cases} gsend(i,j,\mu,id) \in \ \beta_i^{t_j}(r) \text{ and } \beta_{g_i}^{t_j}(r) \neq \varnothing \\ \text{or} \\ fake\left(i, gsend(i,j,\mu,id)\right) \in \beta_{b_i}^{t_j}(r) \end{cases} \\ grecv(j,i,\mu,id) \in \overline{\beta}_{\epsilon_j}^{t_k-1}(r) \end{cases}$

Eventually $(\eta_j, \eta_k) \in E^r_{msg}$

Conclusion : $Past(r, \theta) = Past(r', \theta)$

□

Lemma 3.1.3. *Let $r \in R^P$, $\theta = (i,t) \ \in \mathcal{A} \times \mathbb{N}, J \subset \mathcal{A} \times \mathbb{N}, |J \backslash \mathcal{A}\left(Bad_{V^r_\theta}(r)\right)| \leq \varepsilon = f - |\mathcal{A}\left(Bad_{V^r_\theta}(r)\right)|$. Then there is a $r' \in R^P$ such that:*

1. *r' followed Lem3.1.1 A. B. C. E.*
2. *$\forall j \ \in J, (j,0) \in Failed(r', 1)$*
3. *$\mathcal{A}(Failed(r')) = \mathcal{A}(Failed(r)) \cup J$*

Proof. Let $r \in R^P$,
Let us build such a $r' \in R^P$

Part A. Building rules for $\alpha^m_i(r)$ and $\alpha^m_{\epsilon_i}(r)$:

(a) All the rules of proof of Lem3.1.1

(b) *Init* : $\begin{cases} r(0) = r'(0) \\ \{fail(j) | j \in J\} \subset \alpha^0_f(r') \end{cases}$

Part B. Let us prove $r \in MB$:

$$\begin{aligned}
|\mathcal{A}\left(Failed_{V_\theta^{r'}}(r')\right)| &=_{\text{proof of Lem3.1.1 and } Init} |\mathcal{A}\left(Bad_{V_\theta^{r'}}(r')\right)| + |J \backslash \mathcal{A}\left(Bad_{V_\theta^{r'}}(r')\right)| \\
&=_{Lem(3.1.1)C.} |\mathcal{A}\left(Bad_{V_\theta^{r}}(r)\right)| + |J \backslash \mathcal{A}\left(Bad_{V_\theta^{r}}(r)\right)| \\
&\leq |\mathcal{A}\left(Bad_{V_\theta^{r}}(r)\right)| + f - |\mathcal{A}\left(Bad_{V_\theta^{r}}(r)\right)| \\
&\leq f
\end{aligned}$$

Part C. For the others properties it is the same than in the proof of Lem3.1.1

Conclusion : r' verify Lem3.1.3

□

Lemma 3.1.4 (Fake news lemma)**.** *Let* $r \in R^P, t \in \mathbb{N}, o \in \overline{Actions} \sqcup \overline{Events}$ *and* $(i, t_o) \in \mathcal{A} \times \mathbb{N}$ *s.t* $\begin{cases} (I^P, r, t) \models \overline{occurred}_{(i,t_o)}(o) \\ |\mathcal{A}(Failed(r)) \setminus \{i\} < f| \end{cases}$ *then there exists* $r' \in R^P$ *:*

A. $r \sim^A r'$

B. $\forall u \in \overline{Actions} \sqcup \overline{Events}, (I^P, r, t) \models occurred_{(i,t_o)}(u) \Leftrightarrow (I^{r'}, t, \models) fake_{(i,t_o)}(o)$

C. $Bad(r') = Bad(r) \cup \{(i, t_o)\}$

D. $\mathcal{A}(Failed(r')) = \mathcal{A}(Failed(r)) \cup \{i\}$

E. $\forall \eta \in \mathcal{A} \times \mathbb{N} \backslash \{(i, t_o)\}, \forall u \in \overline{Actions} \sqcup \overline{Events}$ *:*

$$\begin{cases} (I^P, r, t) \models \overline{occurred}_\eta(u) \Leftrightarrow (I^{r'}, t, \models) \overline{occurred}_\eta(u) \\ (I^P, r, t) \models fake_\eta(u) \Leftrightarrow (I^{r'}, t, \models) fake_\eta(u) \end{cases}$$

Proof. Let $r \in R^P, t \in \mathbb{N}, o \in \overline{Actions} \sqcup \overline{Events}$ and $(i, t_o) \in \mathcal{A} \times \mathbb{N}$ s.t $\begin{cases} (I^P, r, t) \models \overline{occurred}_{(i,t_o)}(o) \\ |\mathcal{A}(Failed(r)) \setminus \{i\} < f| \end{cases}$

Let us build such a $r' \in R^P$

Part A. Building rules for $\beta_i^m(r)$ and $\beta_{\epsilon_i}^m(r)$:

(a) *Init* : $\begin{cases} r'(0) & = r(0) \\ \beta_\epsilon^0(r') & = \beta_\epsilon^0(r) \cup fail(i) \end{cases}$

(b) *Agreement* for $(j, m) \neq (i, t_o - 1)$: $\begin{cases} \overline{\beta}_{\epsilon_j}^m(r') & = \overline{\beta}_{\epsilon_j}^m(r) \\ \beta_{f_j}^m(r') & = \beta_{f_j}^m(r) \\ \beta_{g_j}^m(r') & = \beta_{g_j}^m(r) \\ \beta_j^m(r') & = \beta_j^m(r) \end{cases}$

(c) *Fake* :

$$\begin{cases} \beta_{g_i}^{t_o-1}(r') & = \varnothing \\ \overline{\beta}_{\epsilon_i}^{t_o-1}(r') & = \varnothing \\ \beta_i^{t_o-1}(r') & = \varnothing \\ \beta_{b_i}^{t_o-1}(r') & = \beta_{b_i}^{t_o-1}(r) \cup \{fake(i, u) \mid \overline{\beta}_{\epsilon_i}^{t_o-1}(r)\} \cup \begin{cases} \text{If } \beta_{g_i}^{t_o-1}(r') = \varnothing \text{ then } \varnothing \\ \text{Else} \{fake(i, u) \mid \beta_i^{t_o-1}(r)\} \end{cases} \end{cases}$$

Part B. Let us prove A. :

Case I: $m = 0$. It holds with *Init*
Case II: $m > 0, (j, m) \neq (i, t_o)$.
$\begin{cases} r_j(m-1) =_{induction} r'_j(m-1) \\ \text{With } \textit{Agreement} : \text{ r, r' agree on (j,m-1).} \end{cases}$ So with Rq2.1.9 we have $r_j(m) = r'_j(m)$
Case III: $(j, m) = (i, t_o)$.
By induction we have $r'_i(t_o - 1) = r_i(t_o - 1)$.
Case 1: $\beta^{t_o-1}_{\epsilon_i}(r)\varnothing$. It can not be, because $(I^P, r, t) \models \overline{occurred}_{(i,t_o)}(o)$
Case 2: $\beta^{t_o-1}_{g_i}(r) = \varnothing$.

$$\begin{aligned} \overline{\beta}^{t_o-1}_{\epsilon_i}(r') \cup \sigma(\beta^{t_o-1}_{b_i}(r')) &=_{Fake} \sigma(\beta^{t_o-1}_{b_i}(r')) \\ &= \overline{\beta}^{t_o-1}_{\epsilon_i}(r) \cup \sigma(\beta^{t_o-1}_{b_i}(r)) \end{aligned}$$

Then by (1.2.22) $r'_i(t_o) = r_i(t_o)$
Case 3: $\beta^{t_o-1}_{g_i}(r) \neq \varnothing$.

$$\begin{aligned} \overline{\beta}^{t_o-1}_{\epsilon_i}(r') \cup \beta^{t_o-1}_{i}(r') \cup \sigma(\beta^{t_o-1}_{b_i}(r')) &=_{Fake} \sigma(\beta^{t_o-1}_{b_i}(r')) \\ &= \overline{\beta}^{t_o-1}_{\epsilon_i}(r) \cup \beta^{t_o-1}_{i}(r) \cup \sigma(\beta^{t_o-1}_{b_i}(r)) \end{aligned}$$

Then by (1.2.22) $r'_i(t_o) = r_i(t_o)$

Conclusion : $\forall (j, m) \in \mathcal{A} \times \mathbb{N}, r_j(m) = r'_j(m)$

Part C. Let us prove B. :
$fake(i, o) \in_{Fake} \beta^{t_o-1}_{b_i}(r)$ then $(I^{r'}, t, \models) fake_{(i,t_o)}(o)$i

(a) Let prove $(I^P, r, t) \models occurred_{(i,t_o)}(u) \Rightarrow (I^P, r', t) \models fake_{(i,t_o)}(u)$

Case I: $(I^P, r, t) \models fake_{(i,t_o)}(u)$.
$fake(i, u) \in \beta^{t_o-1}_{b_i}(r) \subset_{Fake} \beta^{t_o-1}_{b_i}(r')$
Case II: $(I^P, r, t) \models \overline{occurred}_{(i,t_o)}(u)$.
Case 1: $u \in \overline{Actions}$. ie $u \in \beta^{m-1}_{j}(r)$ and $\beta^{m-1}_{g_i}(r) \neq \varnothing$
$fake(i, u) \in_{Fake} \beta^{t_o-1}_{b_i}(r')$
Case 2: $u \in \overline{Events}$. ie $u \in \overline{\beta}^{m-1}_{\epsilon_j}(r)$
$fake(i, u) \in_{Fake} \beta^{t_o-1}_{b_i}(r')$

(b) Let prove $(I^P, r', t) \models fake_{(i,t_o)}(u) \Rightarrow (I^P, r, t) \models occurred_{(i,t_o)}(u)$
By *Fake* there is three cases
Case I: $u \in \beta^{t_o-1}_{b_i}(r)$.
$(I^P, r', t) \models fake_{(i,t_o)}(u)$
Case II: $u \in \beta^{t_o-1}_{i}(r)$ and $\beta^{t_o-1}_{g_i}(r) \neq \varnothing$.
$(I^P, r', t) \models \overline{occurred}_{(i,t_o)}(u)$
Case III: $u \in \overline{\beta}^{t_o-1}_{\epsilon_i}(r)$.
$(I^P, r', t) \models \overline{occurred}_{(i,t_o)}(u)$

Part D. Let us prove C. :

(a) Let prove $Bad\left(r\right) \subset Bad\left(r'\right)$:
Let $\left(j,m\right) \in Bad\left(r\right)$ ie $\begin{cases} \beta_{b_j}^{m-1}\left(r\right) \subset \beta_{b_j}^{m-1}\left(r'\right) \\ \beta_{b_j}^{m-1}\left(r\right) \neq \varnothing \end{cases}$
By *Fake* and *Agreement* we have $\beta_{b_j}^{m-1}\left(r\right) \subset \beta_{b_j}^{m-1}\left(r'\right)$
Therefore $\left(j,m\right) \in Bad\left(r'\right)$
Conclusion : $Bad\left(r\right) \subset Bad\left(r'\right)$

(b) Let prove $Bad\left(r'\right) \subset Bad\left(r\right) \cup \left\{\left(i,t_o\right)\right\}$:
Let $\beta = \left(j,m\right) \in Bad\left(r'\right)$,

Case I: $\left(j,m\right) \neq \left(i,t_o\right)$.
$\begin{cases} \beta_{b_j}^{m-1}\left(r'\right) =_{Agreement} \beta_{b_j}^{m-1}\left(r\right) \\ \beta_{b_j}^{m-1}\left(r'\right) \neq \varnothing \end{cases}$
Hence $\beta_{b_j}^{m-1}\left(r\right) \neq \varnothing$ then $\left(j,m\right) \in Bad\left(r'\right)$
Case II: $\left(j,m\right) = \left(i,t_o\right)$. With B. $\left(i,t_o\right) \in Bad\left(r'\right)$
Conclusion : $Bad\left(r'\right) \subset Bad\left(r\right) \cup \ \left\{\left(i,t_o\right)\right\}$

Conclusion : $Bad\left(r\right) \cup \left\{\left(i,t_o\right)\right\} = Bad\left(r'\right)$

Part E. Let us prove D. :
Same kind of proof than for C.

Part F. Let us prove E. :
Let $\left(j,m\right) \in \mathcal{A} \times \mathbb{N} \backslash \left\{\left(i,t_o\right)\right\}$

(a) Let prove $\left(I^P,r,t\right) \models \overline{occurred}_\beta(u) \Leftrightarrow \left(I^{r'},t,\models\right)\overline{occurred}_\beta(u)$
Case I: $\left(I^P,r,t\right) \models \overline{occurred}_\beta(u)$.

Case 1: $u \in \overline{Events}$.
$u \ \in \overline{\beta}_{\epsilon_j}^{m-1}\left(r\right) =_{Agreement} \ \overline{\beta}_{\epsilon_j}^{m-1}\left(r'\right)$.
Case 2: $u \in \ \overline{Actions}$.
$\begin{cases} u \ \in \beta_j^{m-1}\left(r\right) =_{Agreement} \beta_j^{m-1}\left(r'\right) \\ \beta_{g_j}^{m-1}\left(r\right) =_{Agreement} \beta_{g_j}^{m-1}\left(r'\right) \end{cases}$.
Therefore $\left(I^{r'},t,\models\right)\overline{occurred}_\beta(u)$
Case II: $\left(I^{r'},t,\models\right)\overline{occurred}_\beta(u)$. Same proof as Case I

(b) Let prove $\left(I^P,r,t\right) \models fake\left(\beta,u\right) \Leftrightarrow \left(I^{r'},t,\models\right)fake\left(\beta,u\right)$
Case I: $\left(I^P,r,t\right) \models fake\left(\beta,u\right)$. ie $fake\left(j,u\right) \in \beta_{b_j}^{m-1}\left(r\right)$
$fake\left(j,u\right) \in \beta_{b_j}^{m-1}\left(r\right) =_{Agreement} \beta_{b_j}^{m-1}\left(r'\right)$
Therefore $\left(I^{r'},t,\models\right)\overline{occurred}_\beta(u)$
Case II: $\left(I^{r'},t,\models\right)fake\left(\beta,u\right)$. Same proof as Case I

Part G. Let us check that there is no conflict between the previous rules and τ(Def 1.2.24) :
Let $\left(j,m\right) \in \mathcal{A} \times \mathbb{N}$

(a) The rules are reachable form the protocol
- $\alpha_\epsilon^m\left(r'\right) \in label\left(P_\epsilon\left(m\right),m\right) \cup \left\{\varnothing\right\}$:
We take $\alpha_{\epsilon_j}^m\left(r'\right) = \beta_{\epsilon_j}^m\left(r'\right) \ \alpha_\epsilon^m\left(r'\right) \subset GEvents$ because $r \in R^P$
- $\alpha_j^m\left(r'\right) \in label_j\left(P_j\left(r'_j\left(m\right)\right),m\right) \cup \left\{\varnothing\right\}$:
We take $\alpha_j^m\left(r'\right) = \beta_j^m\left(r'\right)$

Case I: $(j,m) = (i, t_o - 1)$.
$\beta_j^m(r') =_{Fake} \varnothing$ so it holds thanks to the *adversary*
Case II: $(j,m) \neq (i, t_o)$.

$$\begin{aligned}\beta_j^m(r') &=_{Agreement} \beta_j^m(r)\\ &\in label_j\left(P_j\left(r'_j(m)\right), m\right) \cup \{\varnothing\}\\ &\in_{A.} label_j\left(P_j\left(r_j(m)\right), m\right) \cup \{\varnothing\}\end{aligned}$$

(b) The rules pass through $filter_\epsilon$(Def 1.2.12) withthout any damage

- $r'(0) \in \mathscr{G}(0)$:
  It holds with *Init*
- $\beta_\epsilon^m(r') = \alpha_\epsilon^m(r')$:
  By construction

(c) The rules pass through *update*(Def 1.2.22) according to r :

Case I: $(j,m) \neq (i, t_o - 1)$.
By agreement it holds
Case II: $(j,m) = (i, t_o - 1)$.
$\begin{cases}\alpha_g^m(r') = \varnothing\\ \alpha_i^m(r') = \varnothing\end{cases}$ therefore it holds.

Part H. Let us check that $r' \in R^P[ba]$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by *Agreement* because $r \in R^P$

(b) $r \in FS$: ensured by *Agreement* because $r \in R^P$

(c) $r \in MB$:

$$\begin{aligned}\left|\mathcal{A}\left(Failed\left(r'\right)\right)\right| &=_{\text{with D.}} \left|\mathcal{A}\left(Failed\left(r\right)\right) \cup \{i\}\right|\\ &\leq \left|\mathcal{A}\left(Failed\left(r\right)\right) \setminus \{i\}\right| + 1\\ &\leq f\end{aligned}$$

Conclusion : $r' \in R^P$ verifies A. B. C. D. E.

□

Remark 3.1.5. $\forall(\beta,u) \in \mathcal{A} \times \mathbb{N} \times (\overline{Actions} \sqcup \overline{Events}), (I^P, r, t) \models occurred_\beta(u) \Leftrightarrow (I^{r'}, t, \models) occurred_\beta(u)$

Proof. With B. and E. it holds □

3.2 The ε-pede structure

Lemma 3.2.1. *Let* $r \in R^P, (i,j) \in \mathcal{A}^2, i \neq j, o \in \overline{Actions} \sqcup \overline{Events}, t \in \mathbb{N}, t' \geq t,$

$$(I^r, t', \models) K_j occurred_{(i,t)}(o) \Rightarrow (i, t-1) \rightarrow_r^L (j, t')$$

.

Proof. Let assume towards a contradiction that $(I^r, t', \models) K_j \, occurred_{(i,t)}(o)$ and $(i, t-1) \nrightarrow_r^L (j, t')$
Therefore $(i,t) \notin V^r_{(j,t')}$ with Lem1.6.7
Then we can build r' with Lem3.1.1 such that $\beta^t_{\epsilon_i}(r') = \varnothing$
Therefore $(I^r, t', \not\models) occurred_{(i,t)}(o)$ □

Definition 3.2.2. Let $o \in \overline{Actions} \sqcup \overline{Events}, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$

$$\overline{\Theta^r_\theta}(o) = \{\eta \in V^r_\theta | (I^P, r, t) \models \overline{occurred}_\eta(o)\} \tag{3.1}$$

$$\Theta^r_\theta(o) = \{\eta \in V^r_\eta | (I^P, r, t) \models occurred_\theta(o)\} \tag{3.2}$$

$$\Xi^r_\theta(o) = \{\xi : \text{path in } Past(r,\theta) | \exists \eta \in \Theta^r_\theta(o), \eta \mapsto^\xi \theta\} \tag{3.3}$$

N.B $(\beta, \theta) \in \xi$

Lemma 3.2.3. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, (i,t) \in \mathcal{A} \times \mathbb{N}$ *:*

$$\begin{cases} (I^P, r, t) \models K_i fail(i) \vee \overline{occurred}(o) \\ (I^P, r, t) \models \neg fail(i) \\ (I^P, r, t) \models \neg occurred_i(o) \\ \varepsilon = f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r)\right)| + 1 \end{cases} \Rightarrow (I^P, r, t) \models \varepsilon \, occurred(o)$$

Remark 3.2.4. We use $(I^P, r, t) \models K_i fail(i) \vee \overline{occurred}(o)$ as hypothesis rather than $(I^P, r, t) \models K_i \overline{occurred}(o)$ due to Lemma2.1.2

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$, $\begin{cases} (I^P, r, t) \models K_i fail(i) \vee \overline{occurred}(o) \\ (I^P, r, t) \models \neg fail(i) \\ (I^P, r, t) \models \neg occurred_i(o) \\ \varepsilon = f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r)\right)| + 1 \end{cases}$

For the sake of contradiction, let assume that $(I^P, r, t) \not\models \varepsilon \, occurred(o)$.
N.B : $\mathcal{A}\left(\overline{\Theta^r_\theta}(o)\right) <_{Hyp} \varepsilon$.

We will build, in three steps, an $r''' \in R^P$, $\begin{cases} r'''_i(t''') = r_i(t) \\ (I^{r'''}, t, \models) \overline{occurred}(o) \\ (I^{r'''}, t, \not\models) \overline{occurred}(o) \end{cases}$

Step 1. : By Lemma3.1.1 $\exists r' \in R^P$ such that :

(a) r and r' agree on V^r_θ

(b) $\forall (j,t') \in \mathcal{A} \times \mathbb{N}, t' \leq t, (j,t') \notin V^r_\theta \Rightarrow \beta^{t'}_{\epsilon_j}(r') = \varnothing$

(c) $Bad_{V^r_\theta}(r) = Bad_{V^{r'}_\theta}(r')$

(d) $\mathcal{A}\left(Failed_{V^{r'}_\theta}(r')\right) = \mathcal{A}\left(Bad_{V^{r'}_\theta}(r')\right)$

(e) $\forall \beta \in V^r_\theta, o \in \overline{Actions} \sqcup \overline{Events}$:

$$(I^P, r, t) \models \overline{occurred}_\beta(o) \Leftrightarrow (I^{r'}, t, \models) \overline{occurred}_\beta(o)$$
$$(I^P, r, t) \models fake(\beta, o) \Leftrightarrow (I^{r'}, t, \models) fake(\beta, o)$$

Let prove that $\forall (j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t, (I^{r'}, t, \models) occurred_{(j,m)}(o) \Leftrightarrow (j,m) \in \Theta^r_\theta(o)$:

Let $(j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t$

Case I: $(j,m) \in \Theta^r_\theta(o)$. with (d) it holds

Case II: $(I^{r'}, t, \models) occurred_{(j,m)}(o)$. by (b) $(j,m) \in V^r_\theta$ therefore $(j,m) \in \Theta^r_\theta(o)$

Let us prove that $|\mathcal{A}\left(\overline{\Theta_\theta^r}(o)\right)| \leq f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r')\right)|$:

$$\begin{aligned}|\mathcal{A}\left(\overline{\Theta_\theta^r}(o)\right)| &< \varepsilon \\ &< f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r)\right)| + 1 \\ &<_{(c)} f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r')\right)| + 1 \\ &\leq f - |\mathcal{A}\left(Bad_{V^r_{(i,t)}}(r')\right)|\end{aligned}$$

Step 2. : Then we can apply Lemma3.1.3 on r' therefore $\exists r'' \in R^P$ such that :

(1) r'' followed Lem3.1.1 A. B. C. E.

(2) $\forall j \ \in \mathcal{A}\left(\overline{\Theta_\theta^r}(o)\right), (j,0) \in Failed(r'',1)$

(3) $\mathcal{A}(Failed(r'')) = \mathcal{A}(Failed(r')) \cup \mathcal{A}\left(\overline{\Theta_\theta^r}(o)\right)$

Let us prove $\forall j \in \mathcal{A}(\Theta_\theta^r(o)), j \in \mathcal{A}(Failed(r'',t))$:
Let $(j,m) \in \Theta_\theta^r(o)$, therefore $(j,m) \in V_\theta^r$

Case I: $(j,m) \notin \overline{\Theta_\theta^r}(o)$. By def

$$\begin{aligned}(j,m) &\in_{\text{def of } \overline{\Theta_\theta^r}(o)} Bad(r,t) \\ &\in_{(c)} Bad(r',t) \\ &\in_{(1.C)} Bad(r'',t) \\ &\in Failed(r'',t)\end{aligned}$$

Let prove that $\Theta_\theta^{r''}(o) = \Theta_\theta^{r'}(o)$:
With (1.A 1.B and Corollary3.1.2.1) we have $Past(r'',\theta) = Past(r',\theta)$
Then with (1.D and 3.1.2) we have $\Theta_\theta^{r''}(o) = \Theta_\theta^{r'}(o)$

Step 3. : Eventually, we will apply Lemma3.1.4 $|\mathcal{A}\left(\overline{\Theta_\theta^{r''}}(o)\right)|$ times to r'' on each point of $\overline{\Theta_\theta^{r''}}(o)$ in order to build r''' :
Let us prove that the properties of Lemma3.1.4 holds by induction on $n =_{def} |\mathcal{A}\left(\overline{\Theta_\theta^{r''}}(o)\right)|$:

Case I: $n = 1$. By Lemma3.1.4

Case II: $n > 0$. Let r^{n-1} : r" after n-1 usage of Lemma3.1.4 over $O^{n-1}(O^{n-1} \subset \overline{\Theta_\theta^{r''}}(o)$ and $|O^{n-1}| = n-1)$.
By induction property:

A.$r'' \sim^A r^{n-1}$

B.$\forall (j,m) \in O^{n-1}, (I^{r^{n-1}}, t, \models) fake((j,m),o)$

C.$Bad\left(r^{n-1},t\right) = Bad(r'',t) \cup O^{n-1}$

D.$\mathcal{A}\left(Failed\left(r^{n-1},t\right)\right) = \mathcal{A}(Failed(r'',t)) \cup \mathcal{A}\left(O^{n-1}\right)$

E.$\forall (\beta,u) \in \left(\mathcal{A} \times \mathbb{N} \times (\overline{Actions} \sqcup \overline{Events})\right) \setminus \{((j,m),o) | (j,m) \in O^{n-1}\}$:

$$\begin{cases} (I^P, r, t) \models \overline{occurred}_\beta(u) \Leftrightarrow (I^{r'}, t, \models)\overline{occurred}_\beta(u) \\ (I^P, r, t) \models fake(\beta,u) \Leftrightarrow (I^{r'}, t, \models) fake(\beta,u) \end{cases}$$

Let $(j,m) \in \overline{\Theta_\theta^{r''}}(o) \setminus O^{n-1}$ then $O^n = \{(j,m)\} \cup O^{n-1}$. And r^n build by application of Lemma3.1.4 to r^{n-1} on ((j,m),o)

Let prove A.$r'' \sim^A r^{n-1} \sim^A r^n$ therefore $r'' \sim^A r^n$

Let prove B.We have : $\begin{cases} \forall (j,m) \in O^{n-1}, (I^{r^n}, t, \models) fake\left((j,m), o\right) \text{with Lemma3.1.4 E.} \\ (I^{r^n}, t, \models) fake\left((j,m), o\right) \end{cases}$

Therefore $\forall (j,m) \in O^{n-1}, (I^{r^{n-1}}, t, \models) fake\left((j,m), o\right)$

Let prove C.We have : $\begin{cases} Bad\left(r^{n-1}, t\right) = Bad\left(r'', t\right) \cup O^{n-1} \\ Bad\left(r^n, t\right) = Bad\left(r^n, t\right) \cup \{(j,m)\} \text{with Lemma3.1.4 C.} \end{cases}$

Let prove D.We have : $\begin{cases} \mathcal{A}\left(Failed\left(r^{n-1}, t\right)\right) = \mathcal{A}\left(Failed\left(r'', t\right)\right) \cup \mathcal{A}\left(O^{n-1}\right) \\ \mathcal{A}\left(Failed\left(r^n, t\right)\right) = \mathcal{A}\left(Failed\left(r^{n-1}, t\right)\right) \cup \{j\} \end{cases}$

Therefore $\mathcal{A}\left(Failed\left(r^n, t\right)\right) = \mathcal{A}\left(Failed\left(r'', t\right)\right) \cup \mathcal{A}\left(O^n\right)$

Let prove E.By the same kind of arguments it holds

Let prove that $\forall (j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t, (I^{r'''}, t, \models) occurred_{(j,m)}(o) \Leftrightarrow (j,m) \in \Theta_\theta^{r''}(o)$:

Let $(j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t$

Case I: $(j,m) \in \Theta_\theta^{r''}(o)$. it holds with E. and B.

Case II: $(I^{r'''}, t, \models) occurred_{(j,m)}(o)$. it holds with E. and B.

Let us prove $\forall (j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t, (I^{r'''}, t, \not\models) \overline{occurred}_{(j,m)}(o)$:

Lett $(j,m) \in \mathcal{A} \times \mathbb{N}, m \leq t, (I^{r'''}, t, \models) occurred_{(j,m)}(o)$

Therefore $(j,m) \in \Theta_\theta^{r''}(o)$

Case I: $(j,m) \in \overline{\Theta_\theta^{r''}}(o)$. by B. we have $(I^{r'''}, t, \models) fake_{(j,m)}(o)$

Case II: $(j,m) \in \Theta_\theta^{r''}(o)$. by E. we have $(I^{r'''}, t, \models) fake_{(j,m)}(o)$

Let us prove $r_i'''(t) = r_i(t)$:

$$\begin{aligned} r_i'''(t) &=_{A.} r_i''(t) \\ &=_{(1)} r_i'(t) \\ &=_{(a)} r_i(t) \end{aligned}$$

Let us prove $(I^P, r''', t) \models correct_i$:

$$\begin{aligned} & i \notin \mathcal{A}\left(Failed\left(r, t\right)\right) \\ & \notin_{(d)} \mathcal{A}\left(Failed\left(r', t\right)\right) \\ & \notin_{(3)} \mathcal{A}\left(Failed\left(r'', t\right)\right) \\ & \notin_{D. \text{ and } i \notin \overline{\Theta_\theta^r}(o)} \mathcal{A}\left(Failed\left(r''', t\right)\right) \end{aligned}$$

Therefore we have $\begin{cases} (I^P, r''', t) \models correct_i \\ (I^P, r, t) \models K_i fail\,(i) \vee \overline{occurred}(o) \\ (I^P, r''', t) \models correct_i \\ r_i'''(t) = r_i(t) \end{cases} \Rightarrow \begin{cases} (I^P, r''', t) \not\models \overline{occurred}(o) \\ (I^P, r''', t) \models \overline{occurred}(o) \end{cases}$

Contradiction.

Conclusion : $(I^P, r, t) \models \varepsilon occurred(o)$ holds

□

Definition 3.2.5. Let $o \in \overline{Actions} \sqcup \overline{Events}, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$

$$\Xi_\theta^r(o) = \left\{ \xi : \text{path in } Past(r,\theta) \mid \exists \eta \in \Theta_\theta^r(o), t' \leq t, \begin{cases} \eta \mapsto^\xi (i,t') \\ i \notin \mathcal{A}(\xi \backslash \{(i,t')\}) \end{cases} \right\} \tag{3.4}$$

$$E_2 = \{ i1 \to \cdots \to ik \mid \xi : (i1,t1) \to \cdots \to (ik,tk) \in \Xi_\theta^r(o) \} \tag{3.5}$$

$$Barrier_\theta^r(o) = \left\{ (j,t) \mid t \in \mathbb{N}, j \in \arg\min_{X \subset \mathcal{A} \backslash \{i\}} X separing \{i\} \text{ from } \mathcal{A}(\Theta_\theta^r(o)) \text{ in } E2 \right\} \tag{3.6}$$

N.B $(\beta, \theta) \in \xi$

Theorem 3.2.6 (Menger's Theorem 1972)**.** *Let G=(V,E) be a graph and $A, B \subset V$. Then the minimal number of vertices separing A from B in G is equal to the maximun number of (vertex-)disjoint A-B paths in G.*

Lemma 3.2.7. $|\mathcal{A}(Barrier_\theta^r(o))| \leq |\mathcal{A}(\Theta_\theta^r(o))|$
$\Xi_\theta^r(o)$ *is ε-independents from* $i \Rightarrow |\mathcal{A}(Barrier_\theta^r(o))| \geq \varepsilon =$ *is false*
$|\mathcal{A}(Barrier_\theta^r(o))| = \varepsilon \Rightarrow \Xi_\theta^r(o)$ *is ε-independents from* i

Proof.

- By def
- $\Xi_\theta^r(o)$ is ε-independents from i :
  Let $X \subset \Xi_\theta^r(o), \begin{cases} |X| = \varepsilon \\ X \text{ is independent for i} \end{cases}$
  Let $X' = \{ i1 \to \cdots \to ik \mid \xi : (i1,t1) \to \cdots \to (ik,tk) \in X \}$
  By construction X' is a set of $\mathcal{A}(\Theta_\theta^r(o)) - \{i\}$ (vertex-)disjoint path Therefore thanks to *Menger's Theorem*(3.2.6) we have $|Barrier_\theta^r(o)| \geq \varepsilon$
- a piori cette preuve n'est pas nécéssaire, cf = is not true

□

Definition 3.2.8 (Independent paths)**.** Let ξ, ξ' two paths, Ξ a set of paths, $\varepsilon \in \mathbb{N}$ and $i \in \mathcal{A}$:

1. ξ and ξ' are independent for i iff $\mathcal{A}(\xi) \cap \mathcal{A}(\xi') = \{i\}$
2. Ξ is independent for i iff $\forall (\xi, \xi') \in \Xi^2$, ξ and ξ' are independent for i
3. Ξ is ε-independent for i iff $\exists X \subset \Xi, \begin{cases} |X| = \varepsilon \\ X \text{ is independent for} \end{cases}$

Lemma 3.2.9. *Let* $o \in \overline{Agents} \cup \overline{Events}, r \in R^P$ *and* $\varepsilon = f - |\{j \in A(Bad(V_\theta^r)) | \forall m, (j,m) \notin \xi \in \Xi_\theta^r\}| + 1$

$$\begin{cases} (I^P, r, t) \models \neg fail(i) \wedge K_i(fail(i) \vee \overline{occurred}(o)) \\ (I^P, r, t) \not\models occurred_i(o) \end{cases} \Rightarrow |Barrier_\theta^r(o)| \geq \varepsilon$$

Proof. Let $r \in R^P, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}$,

Let us build $r' \in R^P$ such that $\begin{cases} r'_i(t) = r_i(t) \\ (I^{r'}, t, \not\models) \overline{occurred}(o) \\ (I^{r'}, t, \models) \neg fail_i \end{cases}$ and let $(j,m) \in \mathcal{A} \times \mathbb{N}$,

N.B : In the whole proof we will only deal about V_θ^r and ignore all the points outside thanks to Lem3.1.1.

Part A. $\underline{V_\theta^r = NotConcerned \sqcup DMZ \sqcup Barrier_\theta^r(o) \sqcup After :}$

$$NotConcerned = \left\{\eta \in V_\theta^r \mid V_\eta^r \cap \Theta_\theta^r(o) = \varnothing\right\} \setminus Barrier_\theta^r(o)$$

$$After = \left\{ \eta \in V_\theta^r \setminus Barrier_\theta^r(o) \mid \begin{cases} \forall \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta, \exists \lambda \in Barrier_\theta^r(o), \beta \mapsto^\xi \lambda \mapsto^\xi \eta \\ \exists \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta \end{cases} \right\}$$

$$DMZ = V_\theta^r \setminus (After \sqcup Barrier_\theta^r(o) \sqcup NotConcerned)$$

(a) First let us prove $V_\theta^r = NotConcerned \sqcup DMZ \sqcup Barrier_\theta^r(o) \sqcup After$ is a disjoint union

$$NotConcerned \cap After = \varnothing \tag{3.7}$$

Let $x \in NotConcerned$ therefore by (1.6.4) we have $\nexists \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta$
Hence $x \notin After$

(b) Nodes in $Barrier_\theta^r(o)$ are stable in time, ie if they are in $Barrier_\theta^r(o)$ then they where in from the begining and they will belong to this set for eternity.

$$\text{If } (j,m) \in Barrier_\theta^r(o) \text{ then } \forall m', (j,m') \in Barrier_\theta^r(o) \tag{3.8}$$

By definition of $Barrier_\theta^r(o)$

(c) An agent in $After$ is previously in $After$ or $NotConcerned$

$$\text{If } (j,m) \in After \text{ then } \forall m' \leq m, (j,m') \in After \sqcup NotConcerned \tag{3.9}$$

Let $(j,m) \in After$

Case I: $(j,m-1) \in NotConcerned$. Trivial

Case II: *otherwise*.
Let ξ a path : $\beta \in \Theta_\theta^r(o) \mapsto (j,m-1)$
Let $\xi^{ext} = \xi \to (j,m)$ therefore $\exists \lambda \in \xi^{ext}, \lambda \in Barrier_\theta^r(o)$.
Hence by (3.8) we know $\lambda \neq (j,m-1)$
Then $\forall \xi : \beta \in \Theta_\theta^r(o) \mapsto \eta, \exists \lambda \in Barrier_\theta^r(o), \beta \mapsto^\xi \lambda \mapsto^\xi \eta$
Therefore $(j,m-1) \in After$

(d) An agent in $NotConcerned$ is previously in $After$ or $NotConcerned$

$$\text{If } (j,m) \in NotConcerned \text{ then } \forall m' \leq m, (j,m') \in NotConcerned \tag{3.10}$$

Due to Lem1.6.6

(e) An agent in DMZ will be in DMZ for eternity

$$\text{If } (j,m) \in DMZ \text{ then } \forall (j,m') \in V_\theta^r, m' \geq m, (j,m') \in DMZ \tag{3.11}$$

Let $(j,m) \in DMZ \cup After$:
$(j,m+1) \notin_{(3.10)} NotConcerned$
$(j,m+1) \notin_{(3.8)} Barrier_\theta^r(o)$
$(j,m+1) \notin_{(3.9)} After$

(f) $\theta \in After$ by def

(g) $\Theta_\theta^r(o) \subset DMZ \cup Barrier_\theta^r(o)$

Part B. Building rules for $\beta_i^m(r)$ and $\beta_{\epsilon_i}^m(r)$:

(a) *Init* : $r'(0) = r(0)$

(b) *Conservation* for $(j,m) \in After \sqcup NotConcerned$ (ie $m \leq t$):
$\begin{cases} \beta_j^m(r') = \beta_j^m(r) \\ \beta_{\epsilon_j}^m(r') = \beta_{\epsilon_j}^m(r') \end{cases}$

(c) *Barrier* for $(j,m) \in Barrier_\theta^r(o), m \leq t$:
$$\begin{cases} \beta_{b_j}^m(r') & = \beta_{b_j}^m(r) \cup \{fake(j,u) \mid \begin{cases} \text{If } \beta_{g_j}^m(r) = \varnothing \text{ then } u \in \overline{\beta}_{\epsilon_j}^m(r)\} \\ \text{else } u \in \beta_j^m(r) \cup \overline{\beta}_{\epsilon_j}^m(r)\} \end{cases} \\ \overline{\beta}_{\epsilon_j}^m(r') \cup \beta_{g_j}^m(r') & = \varnothing \\ \beta_j^m(r') & = \varnothing \end{cases}$$

(d) *Freeze* for $(j,m) \in DMZ, m \leq t$:
$\begin{cases} \beta_{\epsilon_j}^m(r') = \varnothing \\ \beta_j^m(r') = \varnothing \end{cases}$

(e) *Extending* for $m > t$:

Extends the previous partial run r' by iterating the transition function τ.

Part C. Let us prove $\forall (j,m) \in NotConcerned \sqcup Barrier_\theta^r(o) \sqcup After, r'_j(m) = r'_j(m)$:

Let $(j,m) \in NotConcerned \sqcup Barrier_\theta^r(o) \sqcup After$, let us do it by induction on m.

Case I: $m = 0$. It holds with *Init*

Case II: $m > 0$.

Case 1: $(j,m) \in NotConcerned$. ie $(j,m-1) \in_{(3.10)} NotConcerned$
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \text{r,r' agree on (j,m-1) by } Conservation \end{cases}$
Then by Lem2.1.9 $r'_j(m) = r_j(m)$

Case 2: $(j,m) \in Barrier_\theta^r(o)$. ie $(j,m-1) \in_{(3.8)} Barrier_\theta^r(o)$

Case a: $\beta_{\epsilon_j}^m(r) \subset \{fail(j)\}$. ie $r_j(m) = r_j(m-1)$
$\beta_{\epsilon_j}^m(r') \subset_{Barrier} \{fail(j)\}$ ie

$$\begin{aligned} r'_j(m) &= r'_j(m-1) \\ &=_{induction} r_j(m-1) \\ &= r_j(m) \end{aligned}$$

Case b: $\beta_{\epsilon_j}^m(r) \not\subset \{fail(j)\}$ and $\beta_{g_j}^m(r) = \varnothing$.
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \sigma\left(\beta_{b_j}^m(r')\right) =_{Barrier} \sigma\left(\beta_{b_j}^m(r)\right) \cup \overline{\beta}_{\epsilon_j}^m(r) \end{cases}$
Then by the transition function(1.2.24) $r'_j(m) = r_j(m)$

Case c: $\beta_{\epsilon_j}^m(r) \not\subset \{fail(j)\}$ and $\beta_{g_j}^m(r) \neq \varnothing$.
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \sigma\left(\beta_{b_j}^m(r')\right) =_{Barrier} \sigma\left(\beta_{b_j}^m(r)\right) \cup \overline{\beta}_{\epsilon_j}^m(r) \cup \beta_g^r(j)m \end{cases}$
Then by the transition function(1.2.24) $r'_j(m) = r_j(m)$

Case 3: $(j,m) \in After$. ie $(j,m-1) \in_{(3.9)} After \sqcup NotConcerned$
$\begin{cases} r'_j(m-1) =_{induction} r_j(m-1) \\ \text{r,r' agree on (j,m-1) by } Conservation \end{cases}$
Then by Lem2.1.9 $r'_j(m) = r_j(m)$

Part D. Let us prove $\forall (j,m) \in NotConcerned \sqcup Barrier^r_\theta(o) \sqcup After, r'_j(m+1) = r'_j(m+1)$:
By C. we have $r'_j(m) = r'_j(m)$.
Therefore by the same proof than in C.II we have $r'_j(m+1) = r'_j(m+1)$

Part E. Let us prove $\forall (j,m) \in V^r_\theta, (I^{r'}, t, \not\models)\overline{occurred}_{(j,m)}(o)$:
Let $(j,m) \in V^r_\theta$, let do this by induction on m.

Case I: $m = 0$. It holds, nothing happen at time 0.
Case II: $m > 0$.
Case 1: $(j,m-1) \in DMZ$. ie $r'_j(m) =_{Freeze} r'_j(m-1)$
Therefore $\forall u \in \overline{Actions} \sqcup \overline{Events}, (I^{r'}, t, \not\models)\overline{occurred}_{(j,m)}(u)$
Case 2: $(j,m-1) \in Barrier^r_\theta(o)$.
By *Barrier* we have $\begin{cases} \overline{\beta}^m_{\epsilon_j}(r') = \varnothing \\ \beta^m_j(r') = \varnothing \end{cases}$
Therefore $\forall u \in \overline{Actions} \sqcup \overline{Events}, (I^{r'}, t, \not\models)\overline{occurred}_{(j,m)}(u)$
Case 3: $(j,m-1) \in NotConcerned \sqcup After$.
By *Conservation* we have $\begin{cases} \overline{\beta}^m_{\epsilon_j}(r') = \overline{\beta}^m_{\epsilon_j}(r) \\ \beta^m_j(r') = \beta^m_j(r) \end{cases}$
And by def of $NotConcerned$ (resp $After$) $(I^P, r, t) \not\models \overline{occurred}_{(j,m)}(u)$
Therefore $(I^{r'}, t, \not\models)\overline{occurred}_{(j,m)}(u)$

Part F. Let us prove than $(I^{r'}, t, \models)\neg fail_i$:
By hyp $(I^P, r, t) \models \neg fail_i$ and $\begin{cases} i \notin_{(3.11)} \mathcal{A}(DMZ) \\ i \notin_{def} \mathcal{A}(Barrier^r_\theta(o)) \end{cases}$.
Therefore with *Conservation* we have $(I^{r'}, t, \models)\neg fail_i$

Part G. Let us check that there is no conflict between the previous rules and τ(Def 1.2.24) :
In fact, we will check for $m \leq t$(*Extending* ensures this for $m > t$) that :

(a) The rules are reachable form the protocol
- $\exists X \in label(P_\epsilon(m), m) \cup \{\varnothing\}, \beta^m_\epsilon(r') = filter_\epsilon(r'(m-1), X, \beta^m_1(r'), \ldots, \beta^m_n(r'))$:
We will do this for each $\beta^{]}_{[}(j)nr'm$
Case I: $(j,m-1) \in DMZ$. ie $\beta^m_{\epsilon_j}(r') =_{Freeze} \varnothing$
Therefore it holds
Case II: $(j,m-1) \in Barrier^r_\theta(o)$. ie $\begin{cases} \beta^m_j(r') =_{Barrier} \varnothing \\ \beta^m_{g_j}(r') =_{Barrier} \varnothing \\ \overline{\beta}^m_{\epsilon_j}(r') =_{Barrier} \varnothing \end{cases}$
Therefore $\beta^m_{\epsilon_j}(r') =_{(1.2.12)} filter_\epsilon\left(r'(m-1), \beta^m_{\epsilon_j}(r'), \beta^m_1(r'), \ldots, \beta^m_n(r')\right)$
Case III: $(j,m-1) \in NotConcerned$. ie $\begin{cases} \beta^m_j(r') =_{Conservation} \beta^m_j(r) \\ \beta^m_{\epsilon_j}(r') =_{Conservation} \beta^m_{\epsilon_j}(r') \end{cases}$
Therefore we only have to prove than if $grecv(j,k,\mu,id) \in \overline{\beta}^m_{\epsilon_j}(r')$ then $\exists m' \leq$

$$m, \begin{cases} gsend(k,j,\mu,id) \in \beta_k^{m'}(r') \text{ and } \beta_{g_k}^{m'}(r') \neq \varnothing \\ \text{or} \\ fake(k, gsend(k,j,\mu,id)) \in \beta_{b_k}^{m'}(r') \end{cases}.$$

Let (k,m') such a point in r (exists because $r \in R^P$).
By def of $NotConcerned$ we have $(k,m') \in NotConcerned$ therefore with D. it holds.

Case IV: $(j, m-1) \in After$. ie $\begin{cases} \beta_j^m(r') =_{Conservation} \beta_j^m(r) \\ \beta_{\epsilon_j}^m(r') =_{Conservation} \beta_{\epsilon_j}^m(r') \end{cases}$

Therefore we only have to prove than if $grecv(j,k,\mu,id) \in \overline{\beta}_{\epsilon_j}^m(r')$ then $\exists m' \leq$

$$m, \begin{cases} gsend(k,j,\mu,id) \in \beta_k^{m'}(r') \text{ and } \beta_{g_k}^{m'}(r') \neq \varnothing \\ \text{or} \\ fake(k, gsend(k,j,\mu,id)) \in \beta_{b_k}^{m'}(r') \end{cases}.$$

Let (k,m') such a point in r (exists because $r \in R^P$).

Case 1: $(k,m') \in Barrier_\theta^r(o)$. With D. it holds
Case 2: $(k,m') \in After$. With D. it holds
Case 3: $(k,m') \in Concerned$. With D. it holds
Case 4: $(k,m') \in DMZ$. This case can not be due to the definition of $After$.

- $\beta_j^m(r') \in label_j\left(P_\epsilon(j) r'_j(m-1), m\right) \cup \{\varnothing\}$:

Case I: $(j, m-1) \in DMZ$. ie $\beta_j^m(r') =_{Freeze} \varnothing$
Therefore it holds
Case II: $(j, m-1) \in Barrier_\theta^r(o)$. ie $\beta_j^m(r') =_{Barrier} \varnothing$
Therefore it holds
Case III: $(j, m-1) \in NotConcerned \sqcup After$.

$$\begin{aligned} \beta_j^m(r') &=_{Conservation} \beta_j^m(r) \\ &\in_{r \in R} label_j(P_\epsilon(j) r_j(m-1), m) \cup \{\varnothing\} \\ &\in_{C.} label_j\left(P_\epsilon(j) r'_j(m-1), m\right) \cup \{\varnothing\} \end{aligned}$$

(b) The rules pass through *update*(Def 1.2.22) withthout any damage

- If $\alpha_g^m(r') = \varnothing$ then $\alpha_j^m(r') = \varnothing$:

Case I: $(j,m) \in Barrier_\theta^r(o) \sqcup DMZ$. ie $\beta_{g_j}^m(r') = \varnothing$
Therefore it holds
Case II: $(j,m) \in After \sqcup NotConcerned$.
It exactly follows r so we do not care.

Part H. Let us check that $r' \in R^{ba}$:

- $r'(0) \in \mathscr{G}(0)$:

$$\begin{aligned} r'(0) &=_{Init} r(0) \\ &\in_{r \in R} \mathscr{G}(0) \end{aligned}$$

Part I. Let us check that $r' \in R^P[ba]$:
Therefore we have to check that $r \in \psi^{ba}$.

(a) $r \in EDel$: ensured by Extending

(b) $r \in FS$: ensured by Extending

(c) $r \in MB$: we will check this for $m \leq t$(*Extending* ensures this for $m > t$)
$\mathcal{A}(Failed(r', m)) \subset \mathcal{A}(Failed(r, m)) \cup \mathcal{A}(Barrier^r_\theta(o))$. It is due to *Conservation* for $\mathcal{A}(Failed(r, m))$ and due to *Barrier* for $\mathcal{A}(Barrier^r_\theta(o))$
Thanks to Lem3.1.1 (cf. N.B at the begining of this proof) we have :

$$\mathcal{A}(Failed(r', m)) \subset \mathcal{A}\left(Bad_{V^r_\theta}(r, m)\right) \cup \mathcal{A}(Barrier^r_\theta(o))$$

Therefore

$$\begin{aligned} |\mathcal{A}(Failed(r', m))| &\leq |\mathcal{A}\left(Bad_{V^r_\theta}(r, m)\right) \setminus \mathcal{A}(Barrier^r_\theta(o))| + |\mathcal{A}(Barrier^r_\theta(o))| \\ &\leq f \end{aligned}$$

$$\underline{\text{Conclusion :}}\ r' \in R^P \text{ and } \begin{cases} r'_i(t) = r_i(t) \\ (I^{r'}, t, \models) \neg fail_i \\ (I^{r'}, t, \not\models) \overline{occurred}(o) \end{cases}$$

□

Lemma 3.2.10. *Let* $o \in \overline{Agents} \cup \overline{Events}, r \in R^P$,

$$\begin{cases} (I^{ba}, r, t) \not\models fail(i) \\ \varepsilon = f - |\{j \in A(Bad(V^r_\theta)) | \forall m, (j, m) \notin \xi \in \Xi^r_\theta\}| + 1 \\ \Xi^r_\theta \text{ is } \varepsilon \text{ agents disjoint without } i \end{cases} \Rightarrow (I^P, r, t) \models \overline{occurred}(o)$$

Proof. Ξ^r_θ is ε agents disjoint without i so there is $\{\xi_1, \ldots, \xi_\varepsilon\} \subset \Xi^r_\theta$
Therefore by definition of Ξ^r_θ, there is $\{a_1, ..., a_\varepsilon\} \subset A$ such that $\exists t_j < t, (I^P, r, t) \models occurred_{(a_j,t)}(o)$
Now, let prove that $\exists j \in [\![1; \varepsilon]\!]$, $(I^P, r, t) \models \overline{occurred_{(a_j,t_j)}}(o)$:
By definition of ε we have $|A(Bad(r, \theta)) \cap A(\Xi^r_\theta)| < \varepsilon$
Hence, let $a_j \in A(\Xi^r_\theta) \setminus A(Bad(r, \theta))$ and $(I^P, r, t) \models occurred_{(a_j,t)}(o)$ therefore $(I^P, r, t) \models \overline{occurred_{(a_j,t)}}(o)$
□

Definition 3.2.11 (ε-pede)**.** $(I^{ba}, r, t) \models \varepsilon$-pede(i) iff $\begin{cases} \varepsilon^r = f - |\{j \in A(Bad(V^r_\theta)) | \forall m, (j, m) \notin \xi \in \Xi^r_\theta\}| + 1 \\ \Xi^r_\theta \text{ is } \varepsilon^r \text{ agents disjoint without i} \end{cases}$

$(I^{ba}, r, t) \models K_i\varepsilon$-pede(i) iff $\forall (r', t') \in R^P \times \mathbb{N}, \begin{cases} \varepsilon^{r'} = f - |\{j \in A(Bad(V^{r'}_\theta)) | \forall m, (j, m) \notin \xi \in \Xi^{r'}_\theta\}| + 1 \\ \Xi^{r'}_\theta \text{ is } \varepsilon^{r'} \text{ agents disjoint without i} \end{cases}$

$(I^{ba}, r, t) \models K_i(f+1)$-pede(i) iff $\forall (r', t') \in R^P \times \mathbb{N}, \Xi^{r'}_\theta$ is $f+1$ agents disjoint without i
$Past(r, \theta = (i, t))$ is ε-pede iff

$$\forall (r', t') \in R^P \times \mathbb{N}, r_i(t) = r'_i(t'), \Xi^{r'}_{(i,t')} \text{ is } \varepsilon' \text{ agents disjoint without i}$$

Where $\varepsilon' = f - |\{j \in A(Bad(V^{r'}_{\{}(i, t')\}))| \forall m, (j, m) \notin \xi \in \Xi^{r'}_\theta\}| + 1$

Theorem 3.2.12 (Knowledge Gain Theorem)**.** *Let* $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, \theta = (i, t) \in \mathcal{A} \times \mathbb{N}, (I^P, r, t) \models \neg occurred_i(o)$,

$$(I^P, r, t) \models K_i\overline{occurred(o)} \text{ then } Past(r, \theta) \text{ is } \varepsilon\text{-pede}$$

$$(I^P, r, t) \models K_i(fail(i) \vee \overline{occurred}(o)) \text{ then } (I^P, r, t) \models K_i(fail(i) \vee \varepsilon - pede(i))$$

Remark 3.2.13. Generic remarque In fact when we deel with K_i we have always $fail(i) \vee \ldots$ because byzantine agents can fake part of their receives.

Proof. Let $(r,t) \in R^P \times \mathbb{N}$, by Lem1.5.10 we have $(I^P, r, t) \models K_i \neg occurred_i(o)$
Let $(r', t') \in R^P \times \mathbb{N}, r'_i(t') = r_i(t)$, therefore

Case I: $(I, r', t') \models fail(i)$. Trivial

Case II: $(I, r', t') \not\models fail(i)$. let prove $(I^{ba}, r', t') \models \varepsilon - pede(i)$
$\begin{cases} (I^{r'}, t', \models)_{r'_i(t')=r_i(t)} \neg fail(i) \wedge K_i(fail(i) \vee \overline{occurred}(o)) \\ (I^{r'}, t', \models)_{r'_i(t')=r_i(t)} K_i \neg occurred_i(o) \end{cases}$
So we can apply Lem3.2.9 to (r', t') hence we have $\Xi^{r'}_{\theta}$ is $\varepsilon^{r'}$agent disjoint without i.
Conclusion : $(I^{ba}, r', t') \models \varepsilon - pede(i)$.

$$\text{Conclusion : } (I^P, r, t) \models K_i(fail(i) \vee \varepsilon - pede(i))$$

□

Theorem 3.2.14 (ε-pede Theorem)**.** *Let* $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}, (I^P, r, t) \models \neg occurred_i(o)$,

$$(I^{ba}, r, t) \ \models K_i(fail(i) \vee \varepsilon - pede(i)) \ \textit{then} \ (I^P, r, t) \models K_i(fail_i \vee \overline{occurred}(o))$$

Proof. Let $(r,t) \in R^P \times \mathbb{N}$, by Lem1.5.10 we have $(I^P, r, t) \models K_i \neg occurred_i(o)$
Let $(r', t') \in R^P \times \mathbb{N}, r'_i(t') = r_i(t)$, therefore

Case I: $(I, r', t') \models fail(i)$. Trivial

Case II: $(I, r', t') \not\models \neg fail(i)$. let prove $(I, r', t') \models \overline{occurred}(o)$

$\begin{cases} (I^{ba}, r', t) \models \varepsilon - pede(i) \\ (I^{r'}, t', \models)_{r'_i(t')=r_i(t)} K_i \neg occurred_i(o) \end{cases}$
So we can apply Lem3.2.10 to (r', t') hence we have $(I^{r'}, t', \models)\overline{occurred}(o)$
Conclusion : $(I^{r'}, t', \models)\overline{occurred}(o)$

$$(I^{ba}, r, t) \ \models K_i(fail(i) \vee \varepsilon - pede(i)) \text{ then } (I^P, r, t) \models K_i(fail_i \vee \overline{occurred}(o))$$

□

Remark 3.2.15. Let $o \in \overline{Actions} \sqcup \overline{Events}, r \in R^P, \theta = (i,t) \in \mathcal{A} \times \mathbb{N}, (I^P, r, t) \models \neg occurred_i(o)$,

$$(I^{ba}, r, t) \ \models K_i(fail(i) \vee (f+1) - pede(i)) \text{ then } (I^P, r, t) \models K_i(fail_i \vee \overline{occurred}(o))$$

Proof.
Let prove that if $(I^{ba}, r, t) \ \models K_i(fail(i) \vee (f+1) - pede(i))$ then $(I^{ba}, r, t) \ \models K_i(fail(i) \vee \varepsilon - pede(i))$:

Case I: $(I, r', t') \models fail(i)$. Trivial

Case II: $(I, r', t') \not\models \neg fail(i)$. $(I^{ba}, r, t) \ \models (f+1) - pede(i)$ therefore $(I^{ba}, r, t) \ \models \varepsilon - pede(i)$

Eventually :
With Th3.2.14 we have $(I^P, r, t) \models K_i(fail_i \vee \overline{occurred}(o))$

□

Chapter 4

TODO

4.1 Deeper into Knowledge Gain

Title needed On s'interesse à ce qui doit être dans l'état local pour atteindre le savoir

Definition 4.1.1 (Minimal Set of intels)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, i \in A$,
$\overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o)$ is the minimal set of intels needed to gain knowledge of $\overline{occured}(o)$.

$$I \in \overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o) \text{ iff } \forall s, (I \in s \Leftrightarrow (\gamma^{ba}, s) \models K_i\overline{occured(o)})$$

$$(I,J) \in \overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o)^2, I \subset J \Rightarrow I = J$$

$\mathscr{I}_i^{(\gamma^{ba},P)}(o)$ is the minimal set of intels needed to gain knowledge of $occured(o)$.

$$I \in \mathscr{I}_i^{(\gamma^{ba},P)}(o) \text{ iff } \forall s, (I \in s \Leftrightarrow (\gamma^{ba}, s) \models K_i occured(o))$$

$$(I,J) \in \mathscr{I}_i^{(\gamma^{ba},P)}(o)^2, I \subset J \Rightarrow I = J$$

Remark 4.1.2. $\overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o) \subset \mathscr{I}_i^{(\gamma^{ba},P)}(o)$

Il faudrait que cet enesemble minimal soit calculable et après voir les complexités de calcul

Definition 4.1.3 (Trigger action)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, i \in A$,
$\overline{\mathscr{T}}_i^{(\gamma^{ba},P)}(o)$ triggering element of gaining knowledge of $\overline{occured}(o)$ in r at agent i.

$$T \in \overline{\mathscr{T}}_i^{(\gamma^{ba},P)}(o) \text{ iff } \exists (r,t) \in R(P,\gamma^{ba}) \times \mathbb{N}, \begin{cases} (I^[, b, a)]rt \models correct(i) \\ (I^[, b, a)]rt \models K_i\overline{occured}(o) \\ (I^{ba}, r, t-1) \not\models K_i\overline{occured(o)} \end{cases}, T = r_i(t) \backslash r_i(t-1)$$

$\mathscr{T}_i^{(\gamma^{ba},P)}(o)$ triggering element of gaining knowledge of $occured(o)$ in r at agent i.

$$T \in \mathscr{T}_i^{(\gamma^{ba},P)}(o) \text{ iff } \exists (r,t) \in R(P,\gamma^{ba}) \times \mathbb{N}, \begin{cases} (I^[, b, a)]rt \models correct(i) \\ (I^[, b, a)]rt \models K_i occured(o) \\ (I^{ba}, r, t-1) \not\models K_i occured(o) \end{cases}, T = r_i(t) \backslash r_i(t-1)$$

Remark 4.1.4. $\overline{\mathscr{T}}_i^{(\gamma^{ba},P)}(o) \subset \mathscr{T}_i^{(\gamma^{ba},P)}(o)$

Remark 4.1.5. Let $o \in \overline{Actions} \sqcup \overline{Events}, i \in A$,

$$I \in \overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o) \Rightarrow (\exists T \in \overline{\mathscr{T}}_i^{(\gamma^{ba},P)}(o), T \in I)$$

$$I \in \mathscr{I}_i^{(\gamma^{ba},P)}(o) \Rightarrow (\exists T \in \mathscr{T}_i^{(\gamma^{ba},P)}(o), T \in I)$$

Lemma 4.1.6. *Let $o \in \overline{Actions} \sqcup \overline{Events}, i \in A$,*

$$T \in \mathscr{T}_i^{(\gamma^{ba},P)}(o) \Rightarrow o \in T \vee T \cap \{recv()\} \neq \varnothing$$

Proof. Let $T \in \mathscr{T}_i^{(\gamma^{ba},P)}(o), \exists (r,t) \in R(P,\gamma^{ba}) \times \mathbb{N}, \begin{cases} (I^{[},b,a)]rt \models correct(i) \\ (I^{[},b,a)]rt \models K_i occured(o) \\ (I^{ba},r,t-1) \nvDash K_i occured(o) \end{cases}, r_i(t) = T :$

$r_i(t-1)$

$(I^{ba},r,t-1) \nvDash K_i occured(o)$ hence $\exists r' \in R(P,\gamma^{ba}, \begin{cases} (I^{ba},r',t-1) \nvDash occured(o) \\ r_i(t-1) = r'_i(t-1) \end{cases}$

Case I: $T = \varnothing$. Impossible

$T = \varnothing$ ie $r_i(t) = r_i(t-1)$ $\begin{matrix} (I^{ba},r,t-1) \nvDash K_i occured(o) \\ (I^{ba},r,t-1) \models K_i occured(o) \end{matrix}$ Contradiction

Case II: $T = \{a\}$.

Case 1: $a = o$. It holds

Case 2: $a \in \beta_i(t-1)$. we will extend r' to t, such that $\forall j \in A \backslash \{i\}, \beta'_{e_j}(t-1) = \varnothing$ and $\beta'_{e_i}(t-1) = \{go(i)\}, \beta'_i(t-1) = a$

then $\begin{matrix} r'_i(t) = r_i(t) \text{ with } \tau \\ (I^{ba},r',t) \nvDash occured(o), \text{ because } o \neq a \end{matrix}$ therefore $(I^{ba},r,t) \nvDash K_i occured(o)$ Contradiction

Case 3: $a = external(i,\mu)$. we will extend r' to t, such that $\forall j \in A \backslash \{i\}, \beta'_{e_j}(t-1) = \varnothing$ and $\beta'_{e_i}(t-1) = \{a\}$

then $\begin{matrix} r'_i(t) = r_i(t) \text{ with } \tau \\ (I^{ba},r',t) \nvDash occured(o), \text{ because } o \neq a \end{matrix}$ therefore $(I^{ba},r,t) \nvDash K_i occured(o)$ Contradiction

Case III: $|T| > 1$. $T =_{def} T_{recv} \cup T_{external} \cup T_i$

Case 1: $o \in T$. It holds

Case 2: $o \notin T \wedge T_{recv} = \varnothing$. we will extend r' to t, such that $\forall j \in A \backslash \{i\}, \beta'_{e_j}(t-1) = \varnothing$ and $\beta'_{e_i}(t-1) = \beta_{e_i}(t-1), \beta'_i(t-1) = \beta_i(t)$

then $\begin{matrix} r'_i(t) = r_i(t) \text{ with } \tau \\ (I^{ba},r',t) \nvDash occured(o), \text{ because } o \notin T \end{matrix}$ therefore $(I^{ba},r,t) \nvDash K_i occured(o)$ Contradiction

□

Lemma 4.1.7. *Let $o \in \overline{Actions} \sqcup \overline{Events}$,*

$$\exists P, P' s.t \overline{\mathscr{I}}_i^{(\gamma^{ba},P)}(o) \neq \overline{\mathscr{T}}_i^{(\gamma^{ba},P')}(o)$$

$$\exists P, P' s.t \mathscr{I}_i^{(\gamma^{ba},P)}(o) \neq \mathscr{T}_i^{(\gamma^{ba},P')}(o)$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}$, let take to trivial protocols :

1. P :
   (a) $Msgs = \{Hello\}$
   (b) $Int_{i \in A} = \varnothing$
   (c) $Ext_{i \in A} = \{Start\}$
   (d) If $External(i, Start) \in \beta_{e_i}(t)$ then $Send(i, A, Hello, \dots) \in \beta_i(t+1)$

2. P' :

   (a) $Msgs = \{Hello\}$

   (b) $Int_{i \in A} = \varnothing$

   (c) $External = \{Start\}$

   (d) $\forall t, \forall i, Send(i, A, Hello, \dots) \in \beta_i(t)$

$$\overline{\mathscr{I}}_i^{(\gamma^{ba}, P)}(o) = \{\{recv(i, j_1, Hello, \dots), \dots, recv(i, j_{f+1} Hello, \dots)\} | (j_1, \dots, j_{f+1}) \subset A\}$$

$$\overline{\mathscr{I}}_i^{(\gamma^{ba}, P')}(o) = \varnothing$$
$$\mathscr{I}_i^{(\gamma^{ba}, P)}(o) = \{\{recv(i, j, Hello, \dots)\} | j \in A\}$$
$$\mathscr{I}_i^{(\gamma^{ba}, P')}(o) = \varnothing$$

□

Remark 4.1.8. $\overline{\mathscr{I}}_i^{(\gamma^{ba}, P)}(o)$ and $\mathscr{I}_i^{(\gamma^{ba}, P)}(o)$ truly depend on P.

We want agent i to be able to compute at least $\overline{\mathscr{I}}_i^{(\gamma^{ba}, P)}(o)$, and we want to find some γ and P such that

$$(\gamma, s) \models K_i \overline{occured}(o) \Rightarrow \exists I \in \mathscr{I}_i^{(\gamma, P)}(o), I \in s$$

Chapter 5

The Protocol Agent Model

5.1 The Byzantine Asynchronous Protocol context

Definition 5.1.1 (γ^{ba-p})**.** $\gamma^{ba-p} \subset \gamma^{ba}$ such that :

1. One agent protocol P_A shared by all agents : $\forall i \ \in A, P_i = P_A$
2. $\forall i \in A, \forall s \ \in \ \Sigma_i, P_A \in s$

Remark 5.1.2. $\forall r \in R(P, \gamma^{ba-p}), (\gamma^{ba-p}, r, t) \models C_G P_A$

5.2 Deeper into $\mathscr{I}_i^{(\gamma^{ba-p}, P)}(o)$

Lemma 5.2.1. *Let* $P, r \in R(P, \gamma^{ba}), (i,t) \in V_\theta^r, (I^{ba}, r, t) \models correct_i$,

$$|\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq m \Rightarrow |A(V_\theta^r) \setminus \{i\}| \geq m$$

Proof. Let assume $|\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq m$,
Let $j \in \{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}, \exists \mu, \exists id, \exists t' < t, recv(i, j, \mu, id) \in \overline{\beta}_{\epsilon_i}^{t'}(r)$,
$(I^{ba}, r, t) \models correct_i$ therefore with τ(Def1.2.24 (3.)) we have : $\exists t'' < t', \begin{cases} send(j, i, \mu, id) \in \widehat{\beta_{e_j}}(t'') \\ \text{or} \\ fake(j, send(j, i, \mu, id)) \in \beta_{e_j}^b(t'') \end{cases}$
Then we have $(j, t') \rightarrow_r^{Send-recv/Failed-recv} (i, t'') \rightarrow_r^{Local} (i, t)$
Eventually with Lem1.6.7, $(j, t'') \in V_\theta^r$

Conclusion : $|A(V_\theta^r) \setminus \{i\}| \geq m$

□

5.3 Some classification

Definition 5.3.1 (Class $\mathscr{C}_{det}$)**.** $\mathscr{C}_{det}$ $\{P \in P(\Sigma, Int, Ext, Msg, Act),$ P deterministic$\}$

Remark 5.3.2.

Definition 5.3.3 (Class $\mathscr{C}_{m=k}$)**.** $\mathscr{C}_{m=k} = \{P \in P(\Sigma, Int, Ext, Msg, Act), |Msgs| = k\}$

Remark 5.3.4. When a message is sent to one agent it is sent to all in the same round, but their are not necessary received at the same time(due to the intrinsic nature of the Asynchronous context).

5.3.1 Without the spreading of messages

Definition 5.3.5 ($\mathscr{C}_{nrelay}$)**.** Let $o \in \overline{Actions} \cup \overline{Events}$,
$\mathscr{C}_{nrelay}(o) = \{P \in \mathscr{C}_{det} | \exists \mu \in Msgs, \forall i \in A, \forall s = E : s', m = send(i, \dots, \mu, \dots) \in P_i(s) \text{ iff } o \in E\}$

Remark 5.3.6. Let $o \in \overline{Actions} \cup \overline{Events}, P \in \mathscr{C}_{nrelay}(o), \exists \mu \in Msgs, \forall (r,t) \ \in R(P, \gamma^{ba-p}) \times \mathbb{N}$,
$\begin{cases} (I^{ba-p}, r, t) \models correct_i \\ send(i, \dots, \mu, \dots) \in r_i(t+1) \setminus r_i(t) \end{cases}$ then $(I^{ba-p}, r, t) \models \overline{occurred_{(i,t)}}(o)$

Lemma 5.3.7. *Let* $o \in \overline{Agents} \cup \overline{Events}, P \in \mathscr{C}_{nrelay}(o) \cap \mathscr{C}_{t=1}, (I^{ba-p}, r, t) \models correct_i$,

$$j \in A(V^r_{(i,t)}) \setminus \{i\} \Rightarrow (I^{ba}, r, t) \models \overline{occurred_j}(e) \vee fail_j$$

Proof. Let $o \in \overline{Agents} \cup \overline{Events}, P \in \mathscr{C}_{nrelay}(e) \cap \mathscr{C}_{t=1}, r \in R(P, \gamma^{ba}), \theta = (i,t) \in V^r$,

Case I: $A(V^r_\theta) = \{i\}$. Trivial
Case II: $|A(V^r_\theta)| > 1$. let $(j \neq i, t') \mapsto \theta \in Past(r, \theta)$
Then $((j, t' \leq t_j), (k \neq j, t_k \leq t)) \in (A \times \mathbb{N})^2, (j, t_j) \rightarrow^{E_{recv} \vee E_{frecv}} (k, t_k)$

Case 1: $(j, t_j) \rightarrow^{E_{recv}} (k, t_k)$.

$$\begin{cases} send(j, \dots) \in r_j(t_j + 1) \\ P \in \mathscr{C}_{nrelay}(o) \cap \mathscr{C}_{t=1} \end{cases} \Rightarrow o \in r_j(t_j)$$

Then $(I^{ba-p}, r, t) \models occurred_j(o)$
Case 2: $(j, t_j) \rightarrow^{E_{frecv}} (k, t_k)$. Then $(I^{ba-p}, r, t) \models fail_j$

□

Lemma 5.3.8. *Let* $o \in \overline{Agents} \cup \overline{Events}, i \in A, P \in \mathscr{C}_{nrelay}(e) \cap \mathscr{C}_{t=1}, r \in R(P, \gamma^{ba})$,
$(I^{ba-p}, r, t) \models \neg occurred_i(o), r_i(t)$ *is coherent,*

$$|\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq f + 1 \Leftrightarrow (I^{ba}, r, t) \models K_i(\overline{occurred}(o) \vee fail_i)$$

Proof.

1. $\underline{|\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq f + 1 \Rightarrow (I^{ba}, r, t) \models K_i(\overline{occurred}(e) \vee fail_i)}$
   Let $|\{j | recv(i, j, \dots) \in r_i(t)\}| \geq f + 1$, let $r' \in R(P, \gamma^{ba}), r'_i(t') = r_i(t)$:

   Case I: $(I, r', t') \models correct_i$. ie let prove $(I^{ba}, r', t') \models \overline{occurred_j}(e)$
   Therefore $\begin{cases} |\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq f + 1 \\ (I, r', t') \models correct_i \end{cases}$ hence with Lem5.2.1 $|A(V^{r'}_\theta) \setminus \{i\}| \geq f + 1$
   By Lem5.3.7 we have $\forall j \in A(V^{r'}_\theta) \setminus \{i\}, (I^{ba}, r', t') \models \overline{occurred_j}(e) \vee fail(j)$
   And by τ(Def1.2.24) we have $|A(Failed(V^{r'}_\theta))| \leq f$
   Conclusion : $\exists j \in A(V^r_\theta) \setminus \{i\}, (I^{ba}, r', t') \models \overline{occurred_j}(e)$
   Case II: $i \in A(Failed(r', t'))$. Trivial

   Conclusion :$(I^{ba}, r, t) \models K_i(\overline{occurred}(e) \vee fail_i)$

2. $\underline{|\{j | recv(i, j, \dots) \in r_i(t)\} \setminus \{i\}| \geq f + 1 \Leftarrow (I^{ba}, r, t) \models K_i(\overline{occurred}(e) \vee fail_i)}$
   By the shake of contradiction let $|\{j | recv(i, j, \dots) \in r_i(t)\}| < f$,
   $r_i(t)$ is coherent so let $(r', t') \in R(P, \gamma^{ba-p}) \times \mathbb{N}, \begin{cases} r'_i(t') = r_i(t) \\ Failed(r', t') = \varnothing \\ (I^{ba-p}, r', t') \models K_i(fail_i \vee \overline{occurred}(o)) \end{cases}$

Therefore, we can build r'' where : $\left\{ \begin{array}{l} recv(i,j,\dots) \in r_i''(t') \Rightarrow (I^{ba-p}, r', t') \models occurred_j(o) \wedge \neg\overline{occurred_j}(o) \\ r_i''(t') = r_i(t) \\ i \notin A(Failed(r'', t')) \end{array} \right.$

Hence : $\left\{ \begin{array}{l} (I^{ba-p}, r'', t') \models correct_i \\ (I^{ba-p}, r'', t') \not\models \overline{occurred}(o) \\ (I^{ba-p}, r'', t') \models K_i(\overline{occurred}(o) \vee fail_i) \end{array} \right.$ Contradiction.

Conclusion: $|\{j | recv(i,j,\dots) \in r_i(t)\}| \geq f+1$

□

Theorem 5.3.9. *Let* $o \in \overline{Agents} \cup \overline{Events}, i \in A, P \in \mathscr{C}_{nrelay}(e), r \in R(P, \gamma^{ba})$, $(I^{ba-p}, r, t) \models \neg occurred_i(o), r_i(t)$ *is coherent,* $\exists \mu \in Msgs$,

$$|\{j | recv(i,j,\mu,\dots) \in r_i(t)\} \backslash \{i\}| \geq f+1 \Leftrightarrow (I^{ba}, r, t) \models K_i(\overline{occurred}(o) \vee fail_i)$$

Proof. Let $\mu \in Msgs, \forall (r,t) \in R(P, \gamma^{ba-p}) \times \mathbb{N}, \left\{ \begin{array}{l} (I^{ba-p}, r, t) \models correct_i \\ send(i,\dots,\mu,\dots) \in r_i(t+1) \backslash r_i(t) \end{array} \right.$ then $(I^{ba-p}, r, t) \models \overline{occurred_{(i,t)}}(o)$,
μ exists due to Rq5.3.6.
Then we delete from r all the occurances of $Msgs \backslash \{\mu\}$ and we apply Lem5.3.8 □

5.3.2 With the spreading of messages

Definition 5.3.10 (Class $\mathscr{C}_{broadcast}$)**.** $\mathscr{C}_{broadcast} = \{ P \in \mathscr{C}_{det} | \forall s, (\exists j \in A, send(i,j,\mu,\dots) \in P_i(s)) \Rightarrow (\forall j \in A, send$

Remark 5.3.11 (Class $\mathscr{C}_{broadcast}$)**.** Let $P \in \mathscr{C}_{broadcast}, \forall (r,t) \in R(P, \gamma^{ba-p}) \times \mathbb{N}$,
$\left\{ \begin{array}{l} (I^{ba-p}, r, t) \models correct_i \\ \exists j \in A, send(i,j,\mu,\dots) \in P_i(r_i(t)) \end{array} \right.$ then $\forall j \in A, send(i,j,\mu,\dots) \in P_i(r_i(t))$

Definition 5.3.12 ($\mathscr{C}_{relay}(o)$)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2$,

$$\mathscr{C}_{relay}(o, \lambda, \mu) \left\{ P \in \mathscr{C}_{det} | \begin{array}{c} \forall i \in A, \forall r_i(t) = E : r_i(t-1), \\ \left\{ \begin{array}{l} o \in E \Leftrightarrow \exists j \in A, send(i,j,\mu,\dots) \in P_i(r_i(t)) \\ recv(i,\dots,\mu,\dots) \in E \Leftrightarrow \exists j \in A, send(i,j,\lambda,\dots) \in P_i(r_i(t)) \\ recv(i,\dots,\lambda,\dots) \in E \Leftrightarrow \exists j \in A, send(i,j,\lambda,\dots) \in P_i(r_i(t)) \end{array} \right. \end{array} \right\}$$

Remark 5.3.13. λ and μ can be equal.

Definition 5.3.14. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}_{relay}(o, \lambda, \mu), (r,t) \in R^{\lfloor} ba - p] \times \mathbb{N}$,

$$(I^{ba-p}, r, t) \models relay_{(i,t)}(o) \text{ iff } \left\{ \begin{array}{l} recv(\dots, \mu \vee \lambda, \dots) \in r_i(t) \backslash r_i(t-1) \\ send(\dots, \lambda, \dots) \in r_i(t+1) \backslash r_i(t) \end{array} \right.$$

Lemma 5.3.15. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r,t) \in R^{\lfloor} ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i$,

$$|\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\} \backslash \{i\}| \geq f+1 \Rightarrow \exists t' \geq t, |\{j | recv(i,j)\lambda \vee \mu \in r_i(t')\} \backslash \{i\}| \geq n-f-1$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r,t) \in R^{\lfloor} ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i, |\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\} \backslash \{i\}| \geq 2f+1$

Hence $\left| J(t) =_{def} \left\{ j \in A, \left\{ \begin{array}{l} \exists t' \leq t, send(j,i)\lambda \vee \mu \in r_j(t') \\ \forall t', (I^{ba}, r, t') \models correct_j \end{array} \right. \right\} \right| \geq 1$

1. $P \in \mathscr{C}_{broadcast}$ then $\forall j \in J(t), \exists t_j \leq t, \forall k \in \mathcal{A}, send(j,k,\lambda \vee \mu, \dots) \in r(t_j)$

2. With $EDel$ we will have $\forall k \in \mathcal{A}, \exists t_k \geq t, \forall j \in J(t), recv(k,j)\lambda \vee \mu \in r_k(t_k)$
3. Let $t_{max} = \max_{k \in \mathcal{A}}$, $P \in \mathscr{C}_{relay}(o,\lambda,\mu)$ then we have $|J(t_{max})| = n - f$
4. We can do again 1. and 2. therefore $\exists t'_{max} \geq t_{max}, \forall k \in \mathcal{A}, \forall j \in J(t_{max}), recv(k,j)\lambda \vee \mu \in r_k(t'_{max})$

Moreover $|J(t'_{max})\backslash\{i\}| \geq n - f - 1$

$$\underline{\text{Conclusion :}}\ |J(t'_{max})\backslash\{i\}| \geq n - f - 1$$

□

Definition 5.3.16 ($\mathscr{C}^c_{relay}(o)$)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2$,

$$\mathscr{C}^c_{relay}(o,\lambda,\mu) = \left\{ P \in \mathscr{C}_{relay}(o,\lambda,\mu) \middle| \begin{array}{l} \forall i \in A, \forall r_i(t) = E : r_i(t-1), \\ \left\{ \begin{array}{l} o \in E \Leftrightarrow \exists j \in A, send(i,j)\mu \in P_i(r_i(t)) \\ |\{j | recv(i,j)\mu \in E\}| \geq c \Leftrightarrow \exists j \in A, send(i,j)\lambda \in P_i(r_i(t)) \\ |\{j | recv(i,j)\lambda \in E\}| \geq c \Leftrightarrow \exists j \in A, send(i,j)\lambda \in P_i(r_i(t)) \end{array} \right. \end{array} \right\}$$

Remark 5.3.17. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2, P \in \mathscr{C}^c_{relay}(o,\lambda,\mu), (r,t) \in R(P,\gamma^{ba-p}) \times \mathbb{N}$,

$$(I^{ba-p}, r, t) \models relay_{(i,t)}(o) \text{ iff } \left\{ \begin{array}{l} |\{j \in A | recv(i,j)\mu \vee \lambda \in r_i(t) \setminus r_i(t-1)\}| \geq c \\ send(\dots,\dots)\lambda \in r_i(t+1) \setminus r_i(t) \end{array} \right.$$

Lemma 5.3.18. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2, P \in \mathscr{C}^c_{relay}(o,\lambda,\mu), r \in R^{\lfloor}ba - p], (I^{ba-p}, r, t) \models correct_i$,

$$\textit{If } (I^{ba}, r, t) \models relay_{(i,t)}(o) \textit{ then } (I^{ba}, r, t) \models \textit{ c-occurred(o)}$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2, P \in \mathscr{C}^c_{relay}(o,\lambda,\mu), r \in R^{\lfloor}ba - p], (I^{ba-p}, r, t) \models correct_i$,
and let $O =_{def} \{(i, t' \leq t) \in V^r_\theta | (I^{ba-p}, r, t) \models occurred_{(i,t')}(e)\}$

Let prove that : if $|A(O)| < c$ then $\nexists \beta = (i, t' \leq t), (I^{ba-p}, r, t) \models relay_\beta(e)$
For the shake of contradiction, let assume that a such $\beta = (i,t')$ exists with t' minimal
$P \in \mathscr{C}^c_{relay}(o,\lambda,\mu)$ therefore $J =_{def} |\{(j,t'') | recv(i,j)\mu \vee \lambda \in r_j(t'')\}| \geq c$
Moreover, $\forall (j,t'') \in J, (I^{ba-p}, r, t'') \models occurred_{(j,t'')}(e)$:

$$\text{Let } (j,t'') \in J, \left\{ \begin{array}{l} t'' \leq t' \\ (I^{ba-p}, r, t) \nvDash_{t' \text{ minimal}} relay_{(j,t''-1)}(e) \\ send(j,i,\lambda \vee \mu, \dots) \in r_j(t'') \setminus r_j(t''-1) \end{array} \right. \Rightarrow_{P \in \mathscr{C}^c_{relay}(o,\lambda,\mu)} (I^{ba-p}, r, t'') \models occurred_{(j,t'')}(o)$$

Hence $A(O) \geq c$ Contradiction.

Conclusion :If $(I^{ba}, r, t) \models relay_{(i,t)}(o)$ then $(I^{ba}, r, t) \models$ c-occurred(o) □

Lemma 5.3.19. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o,\lambda,\mu), (r,t) \in R^{\lfloor}ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i$

$$(I^{ba-p}, r, t) \models relay_{(i,t' \leq t)}(o) \Rightarrow (I^{ba-p}, r, t) \models \overline{occurred}(o)$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda,\mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o,\lambda,\mu), (r,t) \in R^{\lfloor}ba-p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i$,
First, let define t_r :

$$t_r = \min\{t', |\mathcal{A}\left(\{(j, t'' \leq t') \in V^r_\theta | (I^{ba-p}, r, t) \models occurred_{(j,t'')}(e)\}\right)| \geq c\}$$

Remarque : $t_r > 0$
$\forall (k,t''), t'' < t_r, (I^{ba-p}, r, t) \models \neg relay((k,t''))o$

Let prove it by induction over t'

Case I: $t' < t_r$. with Lem5.3.18 :
$(I^{ba-p}, r, t) \models \neg relay_{(i,t')}(o)$ therefore it holds.

Case II: $t' \geq t_r$. by induction $\forall j \in \mathcal{A}, \forall t'' < t', (I^{ba-p}, r, t) \models relay_{(j,t'')}(o) \wedge correct_j \Rightarrow (I^{ba}, r, t) \models \overline{occurred}(o)$
Let i s.t $(I^{ba}, r, t) \models \overline{relay_{(i,t')}}(e)$,
$P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu)$ therefore $|J = \{j | recv(i,j)\mu \vee \lambda \in r_i(t')\}| \geq f+1$
$\exists j \in J, (I^{ba}, r, t) \models correct_j$ so that

Case 1: $recv(i,j)\mu \in r_j(t')$. and $P \in \mathscr{C}_{relay}(o, \lambda, \mu)$:
Conclusion : $(I^{ba-p}, r, t) \models \overline{occurred}_j(o)$

Case 2: $recv(i,j)\lambda \in r_j(t')$. and $P \in \mathscr{C}_{relay}(o, \lambda, \mu)$:
Then $\exists t'' < t', (I^{ba-p}, r, t) \models relay_{(j,t'')}(o)$
Therefore by induction hypothesis $(I^{ba-p}, r, t) \models \overline{occurred}(o)$

Conclusion : $(I^{ba-p}, r, t) \models relay_{(i,t')}(o) \Rightarrow (I^{ba-p}, r, t) \models \overline{occurred}(o)$

□

Lemma 5.3.20. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu), (r,t) \in R[ba - p] \times \mathbb{N}$,

$$|\{ j | recv(i,j)\lambda \vee \mu \in r_i(t)\}| \geq f+1 \Rightarrow (I^{ba-p}, r, t) \models K_i \overline{occurred}(o) \vee fail_i$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu), (r,t) \in R[ba-p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i, |\{ j | recv(i,j)\lambda \vee \mu \in r_i(t)\}| \geq f+1$

Let $(r', t') \in R[ba - p] \times \mathbb{N}, r'_i(t') = r_i(t)$, let prove $(I^{ba-p}, r', t') \models \overline{occurred}(o) \vee fail_i$:

Case I: $(I^{ba-p}, r', t') \models fail_i$. It holds
Case II: $(I^{ba-p}, r', t') \models correct_i$. Let $|J = \{j | recv(i,j)\mu \vee \lambda \in r_i(t')\}|$
We have $\begin{cases} |\mathcal{A}(Failed(r', t'))| \leq f \text{ because of } \tau \\ |J| \geq f+1 \text{ because } P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \end{cases}$ so let $j \in J \setminus \mathcal{A}(Failed(r', t'))$
Therefore $(I^{ba-p}, r, t) \models correct_j$

Case 1: $recv(i,j)\mu \in r'_i(t')$. and $P \in \mathscr{C}_{relay}(o, \lambda, \mu)$:
Conclusion : $(I^{ba-p}, r', t') \models \overline{occurred}_j(o)$

Case 2: $recv(i,j)\lambda \in r'_i(t')$. and $P \in \mathscr{C}_{relay}(o, \lambda, \mu)$:
Then $\exists t'' < t', (I^{ba-p}, r, t) \models relay_{(j,t'')}(o)$
Therefore with Lem5.3.19 we have $(I^{ba-p}, r', t') \models \overline{occurred}(o)$

Conclusion : $(I^{ba-p}, r, t) \models K_i occurred(o) \vee fail_i$

□

Theorem 5.3.21. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, Msgs = \{\lambda, \mu\}, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu), (r,t) \in R[ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models \neg\overline{occurred}(o), r_i(t)$ *is coherent,*

$$|\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\}| \geq f+1 \Leftrightarrow (I^{ba-p}, r, t) \models K_i \overline{occurred}(o) \vee fail_i$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu), (r, t) \in R^{[}ba{-}p] \times \mathbb{N}, r_i(t)$ is coherent,

Let prove $|\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\}| \geq f+1 \Rightarrow (I^{ba-p}, r, t) \models K_i \overline{occurred}(e) \vee fail_i$:

With Lem5.3.20

Let prove$|\{j | recv((,i)j\lambda \vee \mu \in r_i(t)\}| \geq f+1 \Leftarrow (I^{ba-p}, r, t) \models K_i \overline{occurred}(o) \vee fail_i$:

$r_i(t)$ is coherent then let$(r', t') \in R^{[}ba - p], r'_i(t') = r_i(t), Failed(r', t') = \varnothing$
Then we can apply Th3.2.12 to r', therefore f+1-pede structure, and moreover $Msgs = \{\lambda, \mu\}$ therefore $|\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\}| \geq f+1$ □

Lemma 5.3.22. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{[}ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i,$

$$|\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\} \backslash \{i\}| \geq 2f+1 \Rightarrow (I^{ba-p}, r, t) \models \Diamond \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{[}ba - p] \times \mathbb{N}, (I^{ba-p}, r, t) \models correct_i, |\{j | recv(i,j)\lambda \vee \mu \in r_i(t)\} \backslash \{i\}| \geq 2f+1$

Hence $\left| J =_{def} \left\{ j \in A, \left\{ \begin{array}{l} \exists t' \leq t, send(j,i)\lambda \vee \mu \in r_j(t') \\ \forall t', (I^{ba}, r, t') \models correct_j \end{array} \right. \right\} \right| \geq f+1$

1. $P \in \mathscr{C}_{broadcast}$ then $\forall j \in J, \exists t_j \leq t, \forall k \in \mathcal{A}, send(j,k,\lambda \vee \mu, \dots) \in r(t_j)$
2. With $EDel$ we will have $\forall k \in A, \exists t_k, \forall j \in J, recv(k,j)\lambda \vee \mu \in r_k(t_k)$
3. Therefore $\exists t_{max}, \forall k \in A, |\{j | recv(k,j)\lambda \vee \mu \in r_k(t_{max})\} \cap J| \geq f+1$

Now let prove that for $j \in A, (I^{ba-p}, r, t_{max}) \models K_j \overline{occurred}(o) \vee fail_j$:

Case I: $j = i$. Done with Lem5.3.20

Case II: $j \in J$.

Case 1: $send(j, \dots)\mu \in r_j(t_{max})$.
$\left\{ \begin{array}{l} (I^{ba-p}, r, t_{max}) \models correct_j \\ P \in \mathscr{C}_{relay}(o, \lambda, \mu) \end{array} \right.$ therefore $(I^{ba-p}, r, t_{max}) \models \overline{occurred}_j(o)$

Case 2: $send(j, \dots)\lambda \in r_j(t_{max})$.
$\left\{ \begin{array}{l} (I^{ba-p}, r, t_{max}) \models correct_j \\ P \in \mathscr{C}_{relay}(o, \lambda, \mu) \end{array} \right.$ therefore $\exists t' \leq t_{max}, (I^{ba-p}, r, t_{max}) \models relay_{(j, t_{max})}(o)$
Then with Lem5.3.19 $(I^{ba-p}, r, t_{max}) \models occurred(o)$

Case III: $j \notin J \cup \{i\}$. therefore $|\{k | recv(j,k)\lambda \vee \mu \in r_j(t_{max})\} \backslash \{j\}| \geq f+1$

Case 1: $(I^{ba-p}, r, t_{max}) \models fail_j$. it is done

Case 2: $(I^{ba-p}, r, t_{max}) \models correct_j$.
With Lem5.3.20 we have $(I^{ba-p}, r, t_{max}) \models occurred(o)$

$$\text{Conclusion :} \ (I^{ba-p}, r, t) \models \Diamond \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$$

□

Lemma 5.3.23. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{\lfloor} ba - p] \times \mathbb{N},$

$$|\{j | recv(\dots, j, \lambda \vee \mu, \dots) \in r_i(t)\} \backslash \{i\}| \geq 2f+1 \Rightarrow (I^{ba}, r, t) \models K_i(fail_i \bigvee \Diamond \bigwedge_{j \in A} K_j(\overline{occurred}(e) \vee fail_j))$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, (\lambda, \mu) \in Msgs^2, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{\lfloor} ba - p] \times \mathbb{N},$
Let $(r', t') \in R^{\lfloor} ba - p] \times \mathbb{N}, r'_i(t') = r_i(t)$:

Case I: $(I^{ba-p}, r', t') \models correct_i$.
Therefore with Lem5.3.22 we have $(I^{ba-p}, r', t') \models \Diamond \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$

Case II: $(I^{ba-p}, r', t') \models fail_i$. It holds

$$\text{Conclusion :} \ (I^{ba-p}, r, t) \models K_i fail_i \bigvee \Diamond \bigwedge_{j \in A} K_j \overline{occurred}(o) \vee fail_j$$

□

Lemma 5.3.24. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, Msgs = \{\lambda, \mu\}, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{\lfloor} ba - p] \times \mathbb{N}, (I^{ba-p}, r, p) \models correct_i, r_i(t)$ *is coherent,*

$$\exists t' \geq t, |\{j | recv(i, j) \lambda \vee \mu \in r_i(t')\} \backslash \{i\}| \geq n - f \Leftarrow (I^{ba-p}, r, t) \models K_i fail_i \bigvee \Diamond \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, Msgs = \{\lambda, \mu\}, P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu) \cap \mathscr{C}_{broadcast}, (r, t) \in R^{\lfloor} ba - p] \times \mathbb{N}, (I^{ba-p}, r, p) \models correct_i, r_i(t)$ is coherent,
Therefore $(I^{ba-p}, r, t) \models \Diamond \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$ then $\exists t' \geq t, (I^{ba-p}, r, t') \models \bigwedge_{j \in \mathcal{A}} K_j \overline{occurred}(o) \vee fail_j$
Moreover with Th5.3.21 we have $\forall j \in \mathcal{A}, (I^{ba-p}, r, t') \models \overline{occurred}(o)$ or $|\{k | recv(j, k) \lambda \vee \mu \in r_k(t')\}| \geq f + 1$
With $P \in \mathscr{C}^{f+1}_{relay}(o, \lambda, \mu)$ and $|\mathcal{A}(Failed(r, t'))| \leq f$ we have :

$$|\{j | (I^{ba-p}, r, t') \models correct_j \ \text{ and } \ send(j, \dots) \lambda \vee \mu \in r_j(t' + 1)\}| \geq n - f$$

Eventually with $P \in \mathscr{C}_{broadcast}$ and $EDel$ we have $\exists t_{max}, \forall (j, k) \in \mathcal{A}^2, j \notin \mathcal{A}(Failed(r, t')), recv(k, j) \lambda \vee \mu \in r_k(t_{max})$
Hence $\forall k \in \mathcal{A}, |\{j | recv(k, j) \lambda \vee \mu \in r_k(t_{max})\} \backslash \{i\}| \geq n - f|$

$$\text{Conclusion :} \ |\{j | recv(i, j) \lambda \vee \mu \in r_i(t_{max})\} \backslash \{i\}| \geq n - f$$

□

Chapter 6

The Causal Cone Agent Model

6.1 The Byzantine Asynchronous Causal Cone Context

Definition 6.1.1 (Local states)**.** Let $i \in \mathcal{A}, \mathscr{L}_i^{cc} =_{def} (\mathscr{P}(A \times \mathbb{N}) \times \mathscr{P}((A \times \mathbb{N})^2)) \times \mathscr{L}_i$

Definition 6.1.2 (τ^{cc})**.** 1. The 1., 2., 3. of τ(Def1.2.24)

2. Let $i \in \mathcal{A}, r_i(t) = (cc, s) with s \in \mathscr{L}_i$:

$$\begin{aligned} r_i(t+1) &= \tau^{cc}(\beta_{e_i}(t), \beta_i^t(r), r_i(t)) \\ &= \big(Past(r,(i,t+1)), \tau(\beta_{e_i}(t), \beta_i^t(r), s)\big) \end{aligned}$$

Definition 6.1.3 (Causal Cone Agent Model(γ^{ba-cc}))**.** γ^{ba-cc} :

1. The power of agents of the Protocol History Model

Remark 6.1.4. Let $r \in R^{[}ba - cc], (i,t) \in \mathcal{A} \times \mathbb{N}, \forall t' \leq t, (I^{ba-cc}, r, t) \models K_i Past(r,(i,t'))$

Proof. Let $r \in R^{[}ba - cc], (i,t) \in \mathcal{A} \times \mathbb{N}$,
$\begin{cases} (I^{ba-cc}, r, t) \models K_i Past(r,(i,t)) \text{ by def of } \gamma^{ba-cc} \\ \forall t' \leq t, Past(r,(i,t')) \subset Past(r,(i,t)) \end{cases}$ therefore $\forall t' \leq t, (I^{ba-cc}, r, t) \models K_i Past(r,(i,t'))$ □

Lemma 6.1.5. $\gamma^{ba-cc} \not\subset \gamma^{ba}$

Proof. It is due to $\forall i \in \mathcal{A}, \mathscr{L}_i^{cc} \not\subset \mathscr{L}_i$ □

Definition 6.1.6 (ι)**.**

$$\begin{aligned} \iota : \mathscr{L}_i^{cc} &\to \mathscr{L}_i \\ (V, E, s) &\mapsto s \end{aligned}$$

6.2 Some classification

Theorem 6.2.1. *All the properties and the classes of The Protocol Agent Model hold for The Causal Cone Model*

Proof. Let $r \in R(P, \gamma^{ba-cc})$ we can extract $r' \in R(P, \gamma^{ba})$ with ι and apply the proofs on it. □

Definition 6.2.2. $d^+((i,t), G) = |\{k; k \neq i \text{ and } \exists t', ((i,t),(k,t')) \in E\}|$

Definition 6.2.3 ($\mathscr{C}_{d+=a}(o)$)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, a \in \mathbb{N}$,

$$\mathscr{C}_{d+=a}(o) = \left\{ P \in P(\Sigma, Int, Ext, Msg, Act) | \begin{array}{l} \forall r \in R^{[}ba - cc], \forall (i,t) \in \mathcal{A} \times \mathbb{N}, r_i(t) = \ldots, \ldots, (E : r_i(t-1)) \\ o \in E \Rightarrow |\{j; send(i,j) \in P_i(r_i(t))\} \backslash \{i\}| = a \end{array} \right\}$$

Lemma 6.2.4. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, a \in \mathbb{N}, P \in \mathscr{C}_{d+=a}(o), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, r_i(t) = (V,E,s)$,

$$|\{j | d^+((j,t'),(V,E)) = a\}| \geq f+1 \Rightarrow (I^{ba-cc}, r, t) \models K_i \overline{occurred}(o)$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, a \in \mathbb{N}, P \in \mathscr{C}_{d+=a}(o), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, r_i(t) = (V,E,s), |\{j | d^+((j,t'),(V,E)) = a\}| \geq f+1$
Let $(r',t') \in R^{[}ba-cc] \times \mathbb{N}, r'_i(t') = r_i(t)$:
With def of $\tau^{cc}, |\mathcal{A}(Failed(r',t'))| \leq f$ therefore let $j \notin \mathcal{A}(Failed(r',t')), t'' \leq t', d^+((j,t''),(V,E)) = a$
Moreover $P \in \mathscr{C}_{d+=a}(o)$ and $(I^{ba-cc}, r', t') \models correct_j$ then $o \in r'_j(t'')$

$$\begin{cases} t'' \leq t' \\ o \in r'_j(t'') \\ j \notin \mathcal{A}(Failed(r',t')) \end{cases} \text{then}_{Rq1.5.9} (I^{ba-cc}, r', t') \models \overline{occurred}_{(j,t'')}(o)$$

Conclusion :$(I^{ba-cc}, r', t') \models \overline{occurred}(o)$

$$\text{Conclusion : } (I^{ba-cc}, r, t) \models K_i \overline{occurred}(o)$$

□

Definition 6.2.5. $d^-((i,t),G) = |\{k; k \neq i \text{ and } \exists t', ((i,t),(k,t')) \in E\}|$

Definition 6.2.6 ($\mathscr{C}_{d-=a}(o)$)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, a \in \mathbb{N}$,

$$\mathscr{C}_{d-=a}(o) = \left\{ P \in P(\Sigma, Int, Ext, Msg, Act) | \begin{array}{l} \forall r \in R^{[}ba-cc], \forall (i,t) \in \mathcal{A} \times \mathbb{N}, r_i(t) = \ldots, \ldots, (E : r_i(t-1)) \\ o \in E \Rightarrow |\{j; recv(i,j) \in E\} \backslash \{i\}| = a \end{array} \right\}$$

Lemma 6.2.7. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, a \in \mathbb{N}, P \in \mathscr{C}_{d-=a}(o), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, r_i(t) = (V,E,s)$,

$$|\{(j,t') | d^-((j,t'),(V,E)) = a\}| \geq f+1 \Rightarrow (I^{ba-cc}, r, t) \models K_i \overline{occurred}(o)$$

Proof. Same proof than for Lem6.2.4 □

Definition 6.2.8 ($\mathscr{C}_{msg}(o,\lambda)$)**.** Let $o \in \overline{Actions} \sqcup \overline{Events}, \lambda \in Msgs$,

$$\mathscr{C}_{msg}(o,\lambda) = \left\{ P \in P(\Sigma, Int, Ext, Msg, Act) | \begin{array}{l} \forall (i,j) \in \mathcal{A}^2, \forall t \in \mathbb{N}, \\ \text{If } send(i,j)\lambda \in P(r_i(t)) \text{ then } \exists (k,t) \in Past(r,(i,t)), o \in r_k(t'') \end{array} \right\}$$

Remark 6.2.9.
1. $\mathscr{C}_{nrelay}(o) \subset \mathscr{C}_{msg}(o,\lambda)$
2. $\mathscr{C}_{relay}(o,\lambda,\mu) \subset \mathscr{C}_{msg}(o,\lambda) \cup \mathscr{C}_{msg}(o,\mu)$

Lemma 6.2.10. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, \lambda \in Msgs, P \in \mathscr{C}_{msg}(o,\lambda), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, (I^{ba-cc}, r, t) \models correct_i$,

$$\exists \{j_1; \ldots; j_{f+1}\} \subset \mathcal{A}, \forall k \neq l, \begin{cases} \exists t_l \leq t, recv(i, j_l)\lambda \in r_i(t_l) \\ \exists t_k \leq t, recv(i, j_k)\lambda \in r_i(t_k) \\ \mathcal{A}\left(V^r_{(j_l,t_l)}\right) \cap \mathcal{A}\left(V^r_{(j_k,t_k)}\right) = \varnothing \end{cases} \Longrightarrow (I^{ba-cc}, r, t) \models \overline{occurred}(o)$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, \lambda \in Msgs, P \in \mathscr{C}_{msg}(o,\lambda), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, (I^{ba-cc}, r, t) \models correct_i, \{j_1; \ldots; j_{f+1}\} \subset \mathcal{A}, \forall k \neq l, \begin{cases} \exists t_l \leq t, recv(i, j_l)\lambda \in r_i(t_l) \\ \exists t_k \leq t, recv(i, j_k)\lambda \in r_i(t_k) \\ \mathcal{A}\left(V^r_{(j_l,t_l)}\right) \cap \mathcal{A}\left(V^r_{(j_k,t_k)}\right) = \varnothing \end{cases}$

With $\begin{cases} |\mathcal{A}(Failed(r,t))| \leq f \\ \forall k \neq l, \mathcal{A}\left(V^r_{(j_l,t_l)}\right) \cap \mathcal{A}\left(V^r_{(j_k,t_k)}\right) = \varnothing \end{cases}$ we have $\exists m \in [\![1; f+1]\!], \mathcal{A}(Past(r,(j_m,t_m))) \cap$

$\mathcal{A}(Failed(r,t)) = \varnothing$

Therefore by definition of $\mathscr{C}_{msg}(o,\lambda)$ we have $\exists (j_o,t_o) \in Past(r,(j_m,t_m)), \begin{cases} o \in r_{j_o}(t_o) \\ o \notin \mathcal{A}Failed(r,t) \end{cases}$.

Eventually we have with Rq1.5.9 $(I^{ba-cc},r,t) \models \overline{occurred}_{(i_o,t_o)}(o)$.

$$\underline{\text{Conclusion :}}\ (I^{ba-cc},r,t) \models \overline{occurred}(o)$$

□

Lemma 6.2.11. *Let* $o \in \overline{Actions} \sqcup \overline{Events}, \lambda \in Msgs, P \in \mathscr{C}_{msg}(o,\lambda), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}$,

$$\exists \{j_1;\ldots;j_{f+1}\} \subset \mathcal{A}, \forall k \neq l, \begin{cases} \exists t_l \leq t, recv(i,j_l)\lambda \in r_i(t_l) \\ \exists t_k \leq t, recv(i,j_k)\lambda \in r_i(t_k) \\ \mathcal{A}\left(V^r_{(j_l,t_l)}\right) \cap \mathcal{A}\left(V^r_{(j_k,t_k)}\right) = \varnothing \end{cases} \implies (I^{ba-cc},r,t) \models K_i fail_i \vee \overline{occurred}(o)$$

Proof. Let $o \in \overline{Actions} \sqcup \overline{Events}, \lambda \in Msgs, P \in \mathscr{C}_{msg}(o,\lambda), r \in R^{[}ba-cc], (i,t) \in \mathcal{A} \times \mathbb{N}, (I^{ba-cc},r,t) \models correct_i, \{j_1;\ldots;j_{f+1}\} \subset \mathcal{A}, \forall k \neq l, \begin{cases} \exists t_l \leq t, recv(i,j_l)\lambda \in r_i(t_l) \\ \exists t_k \leq t, recv(i,j_k)\lambda \in r_i(t_k) \\ \mathcal{A}\left(V^r_{(j_l,t_l)}\right) \cap \mathcal{A}\left(V^r_{(j_k,t_k)}\right) = \varnothing \end{cases}$

$\underline{\text{Let } (r',t') \in R^{[}ba-cc] \times \mathbb{N}, r'_i(t') = r_i(t) :}$

Case I: $(I^{ba-cc},r',t') \models fail_i$. It holds

Case II: $(I^{ba-cc},r',t') \models correct_i$. We apply Lem6.2.10
So $(I^{ba-cc},r',t') \models \overline{occurred}(o)$

$$\underline{\text{Conclusion :}}(I^{ba-cc},r',t') \models K_i fail_i \vee \overline{occurred}(o)$$

□

Remark 6.2.12. The reverse is false

Chapter 7

The Full History Agent Model

7.1 The Full History Context

Definition 7.1.1 (Messages)**.** $\Omega^{fh}(Msgs) = \{send(i,j)(s,\mu)id | (i,j) \in \mathcal{A}^2, \mu \in Msgs, s \in \mathscr{L}_j\} \cup \{recv(i,j)(s,\mu)id | (i,j) \in \mathcal{A}^2, \mu \in Msgs, s \in \mathscr{L}_j\}$

$\forall (r,t) \in R^{\lfloor ba-fh \rfloor} \times \mathbb{N}, \forall (i,j) \in \mathcal{A}^2, \forall \mu \in Msgs, \begin{cases} (I^{ba-fh}, r, t) \models correct_i \\ send(i,j)(s,\mu) \in r_i(t) \backslash r_i(t-1) \end{cases} \Rightarrow s = r_i(t-1)$

Definition 7.1.2 (γ^{ba-fh})**.** Power of Protocol Agent

Remark 7.1.3. N.B : Byzantine agent can alterate the spreading of states

Lemma 7.1.4. $\gamma^{ba-fh} \subset \gamma^{ba-pa}$

Proof. By def □

Lemma 7.1.5. *1.* $\gamma^{ba-fh} \not\subset \gamma^{ba-cc}$

2. $\gamma^{ba-cc} \not\subset \gamma^{ba-fh}$

Proof. Due to the nature of local states □

7.1.1 Properties of the model

Theorem 7.1.6. *All the properties of The Protocol Agent Model hold for The Full History Agent Model*

Proof. Because $\gamma^{ba-fh} \subset \gamma^{ba-p}$ □

Nécessite un timestamp des messages

Definition 7.1.7 (Extracted Causal Cone(ECC))**.** Let $r \in R(P, \gamma^{ba-fh}), (i,t) \in V^r, Past^{ecc}(r,(i,t))$ is the Extracted Causal Cone from $r_i(t)$. It is build with :

1. If $recv(i,j,(r_j(t''),\dots),\dots) \in r_i(t' \leq t) \backslash r_i(t'-1)$ then $(j,t''),(i,t')) \in Past^{ecc}(r,(i,t))$
2. $recv(i,j,(r_j(t''),\dots),\dots) \in r_i(t' \leq t)$ then $Past^{ecc}(r,(j,t'')) \subset Past^{ecc}(r,(j,t))$

Lemma 7.1.8. *Causal Cone Model* $\not\subset$ *Full History Model*

Proof. Let $A = \{1,2,3\}, f = 1,$ P the Hello protocol, $\forall s, \forall i \in A, P_i(s) \subset \{send(i,j,(s,Hello)) | j \in A \backslash \{i\}\}$
Let r such that :

- $\beta_1(0) = \{send(1, 2, (r_1(0), Hello), 0)\}$
- $\widehat{\beta_{e_2}}(0) = \{recv(2, 1, (r_1(0), Hello), 0)\}$
- $\beta^b_{e_2}(1) = \{fake(2, send(2, 3, (r_2(0), Hello), 1))\}$
- $\widehat{\beta_{e_3}}(1) = \{recv(3, 2, (r_2(0), Hello), 1)\}$
- All the other $\beta \cdots = \varnothing$

Some properties about r :

$A(Failed(r, 1)) = \{2\}$

$Past(r, (3, 2)) = (\{(1, 0), (2, 1), (3, 0), (3, 1), (3, 2)\}, \{((3, 0), (3, 1)), ((3, 1), (3, 2)), ((1, 0), (2, 1)), ((2, 1), (3, 2))\})$

$Past^{ecc}(r, (3, 2)) = (\{(2, 1), (3, 0), (3, 1), (3, 2)\}, \{((3, 0), (3, 1)), ((3, 1), (3, 2)), ((2, 1), (3, 2))\})$

Therefore $Pastr, (3, 2)) \not\subset Past^{ecc}(r, (3, 2))$ hence Causal Cone Model $\not\subset$ Full History Model $\square$

Chapter 8

Conclusion

In the future work, we plan first to extend the framework in order to cover heterogeneous agents i.e agent that will not be process the same way by the environment. A perfect example of such heterogeneous processing is the *timely-f-source* model where agents are synchronous and communication are reliable, moreover there is a non-faulty agent(called *source*) that has time-bound communication and multicast with at least f distinct agents. In order to do this heterogeneous processing, and not to fix the specific group of agents(i.e having a different one in different runs) we need to give so seed to the environment protocol to do this we can define an initial state for the environment. And also to cover different thresholds of failure i.e mix benign faults (ex : crash failure) with Byzantine one. Secondly, we plan to study a possible hierarchy of the extensions, in order to provide transparent property conservation from an extension to another one. One additional way is to study the apriori-knowledge needed by an agent in order to gain knowledge.

Bibliography

[1] Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.

[2] 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 (JACM)*, 61(2):13, 2014.

[3] Reinhard Diestel. *Graph Theory*. Springer, 2000.

[4] Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156–186, 1990.

[5] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[6] Patrik Fimml. Knowledge in distributed systems with byzantine failures. Master's thesis, Faculty of Informatics at the TU Wien, 2017. In preparation.

[7] Joseph Y Halpern, Yoram Moses, and Orli Waarts. A characterization of eventual byzantine agreement. *SIAM Journal on Computing*, 31(3):838–865, 2001.

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

[9] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[10] Ruben Michel. A categorical approach to distributed systems expressibility and knowledge. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 129–143. ACM, 1989.

[11] Yoram Moses and Mark R Tuttle. Programming simultaneous actions using common knowledge: Preliminary version. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 208–221. IEEE, 1986.

Index