Unreliable Failure Detectors via Operational Semantics*

Uwe Nestmann and Rachele Fuzzati

School of Computer and Communication Sciences
EPFL-I&C, 1015 Lausanne, Switzerland

Abstract. The concept of *unreliable failure detectors for reliable distributed systems* was introduced by Chandra and Toueg as a fine-grained means to add weak forms of synchrony into asynchronous systems. Various kinds of such failure detectors have been identified as each being the weakest to solve some specific distributed programming problem. In this paper, we provide a fresh look at failure detectors from the point of view of programming languages, more precisely using the formal tool of operational semantics. Inspired by this, we propose a new failure detector model that we consider easier to understand, easier to work with and more natural. Using operational semantics, we prove formally that representations of failure detectors in the new model are equivalent to their original representations within the model used by Chandra and Toueg.

1 Executive Summary

Background In the field of Distributed Algorithms, a widely-used computation model is based on *asynchronous* communication between a fixed number n of connected processes, where *no timing assumptions* can be made. Moreover, processes are subject to *crash-failure*: once crashed, they do not recover. The concept of *unreliable failure detectors* was introduced by Chandra and Toueg [CT96] as a fine-grained means to add weak forms of synchrony into asynchronous systems. Various kinds of such failure detectors have been identified as each being the weakest to solve some specific distributed programming problem [CHT96].

The two communities of Distributed Algorithms and Programming Languages do not always speak the same "language". In fact, it is often not easy to understand each other's terminology, concepts, and hidden assumptions. Thus, in this paper, we provide a fresh look at the concept of failure detectors from the point of view of programming languages, using the formal tool of operational semantics. This paper complements previous work [NFM03] in which we used an operational semantics for a distributed process calculus to formally prove that a particular algorithm (also presented in [CT96]) solves the Distributed Consensus problem. Readers who are interested in proofs about algorithms *within* our new model (rather than proofs *about* it) are thus referred to our previous paper.

* Appears in Proceedings of ASIAN'03 (© Springer-Verlag). Supported by the Swiss National Science Foundation, grant No. 21-67715.02, the Hasler Foundation, grant No. DICS 1825, an EPFL start-up grant, and the EU FET-GC project PEPITO.

$$(\textsc{env})\ \frac{\text{“ failure detection events happens in the environment ”}}{\Gamma\ \longrightarrow \Gamma'}$$

$$(\textsc{tau})\ \frac{\Gamma\ \longrightarrow \Gamma' \qquad N \xrightarrow{\tau@i} N' \qquad \text{“ } i \text{ not crashed in } \Gamma \text{ ”}}{\Gamma \vdash N\ \longrightarrow \Gamma' \vdash N'}$$

$$(\textsc{suspect})\ \frac{\Gamma\ \longrightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad \text{“ } i \text{ not crashed in } \Gamma \text{ ”} \qquad \boxed{\text{“ } j \text{ may be suspected by } i \text{ in } \Gamma \text{ ”}}}{\Gamma \vdash N\ \longrightarrow \Gamma' \vdash N'}$$

Table 1. Uniform “Abstract” Operational Semantics Scheme

The work of Chandra and Toueg emphasized the axiomatic treatment of qualitative properties rather than quantitative ones. Like them, also our current focus is on issues of correctness, not of performance. Moreover, Chandra and Toueg did not primarily aim at providing concrete design support for an implementation of failure detectors. Like them, also we rather seek mathematically useful and convincing semantic abstractions of failure detectors.

Vehicle of Discussion In Table 1, we propose a *uniform scheme* to describe the operational semantics of process networks in the context of failure detectors. For convenience, we abstract from the way how the steps $N\ \longrightarrow N'$ of process networks are generated (from the code that implements the respective algorithm), so we do not provide rules for this. It is sufficient for our purposes to observe that a process i in a network carries out essentially two kinds of transitions $N\ \longrightarrow N'$, distinguished by whether it requires the suspicion of some process j by process i, or not. Formally, we use labels $\text{suspect}_j@i$ and $\tau@i$ to indicate these two kinds.

In summary, Table 1 presents a two-layered operational semantics scheme. One layer, in addition to the transitions $N\ \longrightarrow N'$ of process networks, also describes the transitions $\Gamma\ \longrightarrow \Gamma'$ of the network's environment, keeping track of crashes and providing failure detection, as indicated by rule (ENV). Another layer, with the rules (TAU) and (SUSPECT), deals with the compatibility of network and environment transitions, conveniently focusing on the environment conditions for the two kinds of transitions of process networks. For example, the boxed condition exploits the failure detector information that in our scheme is to be provided via the environment component Γ.

A *system run* in this uniform scheme is an infinite sequence of transitions

$$\Gamma_0 \vdash N_0\ \longrightarrow \Gamma_1 \vdash N_1 \cdots\ \longrightarrow \Gamma_t \vdash N_t\ \longrightarrow \cdots$$

that we often simply abbreviate as $(\Gamma_t \vdash N_t)_{t\in\mathbb{N}_0}$. We also use the projections onto the respective *environment runs* $(\Gamma_t)_{t\in\mathbb{N}_0}$ and *network runs* $(N_t)_{t\in\mathbb{N}_0}$, which exist by definition of the rules (TAU) and (SUSPECT) of Table 1.

Overview We start the main part of the paper by an introduction (§2) to the various kinds of failure detectors proposed by Chandra and Toueg, including Ω which appeared in [CHT96]. Already in this introduction, we rephrase their previous work using the scheme of Table 1. In addition, we use this exercise to come up with a well-motivated proposal for a new model and way to represent failure detectors (§3). We formalize our proposal according to the scheme of Table 1 and redefine all previously introduced FDs within the new model (§4). Exploiting the common scheme and the formality of the framework of operational semantics, we formally prove that our redefinitions are "equivalent" to the original definitions by a mutual simulation of all possible system runs that are derivable in either case (§5) and draw some conclusions from having done so (§6).

Contribution In summary, this paper contains an original presentation, using operational semantics, of existing work by Chandra and Toueg, which is targeted at an audience in process calculi and programming language semantics. The paper also provides, as its main contribution, a new model to represent failure detectors that tries to eliminate a number of drawbacks of the original model used by Chandra and Toueg. Many other failure detectors have been studied in the literature; for the current paper, we restrict our attention to the ones introduced in [CT96, CHT96]. The technical contribution is a formal comparison of the representations of these classical failure detectors in the new model with their representations on the old model, which was greatly simplified by having both models fit the scheme of Table 1. To conclude, we argue that our new model for FDs is easier to understand, easier to work with, and more natural than the one used by Chandra and Toueg (see the justification in §6).

Related Work We are not aware of any related or competing approaches.

Acknowledgments We very much thank André Schiper and Sam Toueg for enlightening discussions about failure detectors and, more generally, distributed algorithms, but they may not necessarily agree with all conclusions that we drew from our work. Many thanks also to Pawel Wojciechowski and the anonymous referees for their comments on a draft of this paper.

2 A Fresh Look at the Model of Chandra and Toueg

Recall that we are addressing asynchronous message-passing distributed system with no bounds on message delays and a fixed number n of processes. Let $\mathbb{P} := \{1 \ldots, n\}$ denote the set of process names. All processes are supposed to run the very same algorithm. Processes may crash; when they do so, they do not recover. Systems evolve in time. $\mathbb{T}$ denotes some discrete time domain; for simplicity, we may just assume that $\mathbb{T} = \mathbb{N}_0$. At any time, the *state* of a system is fully determined by the states of the individual processes while running the algorithm, together with the messages currently present in the global message buffer. We do not formalize global states, but treat them abstractly throughout the paper.

2.1 Schedules

A *schedule*, of the form $(\mathbb{T}, I, S)$, is essentially a sequence S of global steps in time $\mathbb{T}$, while running some algorithm starting within the initial global state I, where the message buffer is empty. Sometimes, we refer to just S as being the schedule. A *step* is usually produced by any one of the n processes according to the algorithms' instructions: in atomic fashion, a process receives some messages (possibly none) from the message buffer, *possibly* checks whether it is allowed to suspect another process, and sends out new messages (possibly none) to the message buffer, while changing its state. Often, it is left rather informal and imprecise how steps are actually defined, and there are a number of variations for this. In both papers [CT96, CHT96], it is assumed that *schedules are infinite*.

A Simple Operational Semantics View To avoid the details of generating global steps from an algorithm, and to abstract away from the details of message-passing, we model schedules simply as infinite sequences of *labeled transitions*

$$N \xrightarrow{\mu@i} N'$$

that denote steps between abstract global states (ranged over by N) by performing an action μ due to the activity of some process i. The label μ depends on whether i needs to (be able to) suspect another process j, or not. If it does so, we indicate this by a transition arrow labeled with $\mu := \text{suspect}_j$, otherwise we simply use the label $\mu := \tau$, which is commonly used to indicate that "some" not further specified activity takes place by process i.

In this paper, we are not at all interested in how schedules themselves are generated. An example of this can be found in our earlier paper [NFM03], where we used a reasonably standard process calculus to do this.

2.2 Unreliable Failure Detectors

According to Chandra and Toueg [CT96], a failure detector (FD) is a separate module attached locally to a process; each process i has its own private FD_i. At any moment in time, each FD outputs a list of (names of) processes that it currently suspects to have crashed. FDs are unreliable: they may

- make mistakes,
- disagree among them, and
- change their mind indefinitely often.

Process i interacts with *its* FD_i explicitly by only being allowed to suspect another process j at any given time t, if FD_i's output list contains j at this time.

Example 1 ("Application"). When a process needs to go on by the help of another process—e.g., via reception of a message—it may typically specify to "either wait for this process, or suspect it and continue otherwise". However, it is only allowed to choose the second option if its FD permits it at the very moment that the process looks at the FDs output, which it may have to do infinitely often if the FD insists on not permitting the required suspicion.

More formally, the concept of process crashes is modeled by means of *failure patterns* $F : \mathbb{T} \to 2^{\mathbb{P}}$ that describe monotonically when in a run crashes happen. For example, $F(42) = \{3, 7\}$ means that processes 3 and 7 have crashed during the time interval $[0, 42]$. Similarly, the concept of failure detection is modeled by so-called *failure detector histories* $H : \mathbb{T} \times \mathbb{P} \to 2^{\mathbb{P}}$. For example, $H(42, 5) = \{6, 7\}$ means that at time 42 processes 6 and 7 are suspected by the FD of process 5. Given the previous example F, this means that process 7 is correctly suspected, while process 6 is erroneously suspected. Mathematically, a FD is a function[1]

$$\mathcal{D} : (\mathbb{T} \to 2^{\mathbb{P}}) \to 2^{(\mathbb{T} \times \mathbb{P} \to 2^{\mathbb{P}})}$$

that maps failure patterns F to sets of failure detector histories. Such sets may be specified by additional properties, as exemplified in Section 2.3. From now on, whenever we mention some F and H in the same context, then $H \in \mathcal{D}(F)$ is silently assumed; we may write $H_{\mathcal{D}}$ to indicate the referred FD.

Finally, *system runs* R are quintuples $(F, H, \mathbb{T}, I, S)$. Subject to the shared time domain $\mathbb{T}$, the schedule S of a run is required to be "compatible" with the failure pattern F and detector history H: (1) a process cannot take a step after it has crashed (according to F); (2) when a process takes a step and queries its failure detector module, it gets the current value output by its local failure detector module (according to H).

A process is *correct in a given run*, if it does not crash in this run. It may, though, crash in other runs. Let $\text{crashed}(R) := \text{crashed}(F) := \bigcup_{t \in \mathbb{T}} F(t)$ denote the processes that *have crashed* in run R according to its failure pattern F. Consequently, $\text{correct}(R) := \text{correct}(F) := \mathbb{P} \backslash \text{crashed}(F)$. Usually, one considers only runs in which $\text{correct}(F) \neq \emptyset$, i.e., in every run at least one process survives. Sometimes, as for the Consensus algorithm that we studied in [NFM03], it is even required that less than $n/2$ processes may crash. Abstractly, we use $\text{maxfail}(n)$ to denote the maximal number of crashes permitted in a run.

A Simple Operational Semantics View To make the model fit our universal scheme of Table 1, we need to recast the information contained in failure patterns F and failure detector histories H in an evolutionary manner as environment transitions. Both F and H are totally defined over the whole time domain $\mathbb{T}$. Thus, we may simply use transitions $(t, F, H) \to (t{+}1, F, H)$, in which time t just passes, while we leave F and H unchanged. Rule ($\mathbb{T}$-ENV) of Table 2 serves us to generate such transitions formally.

System configurations are of the form $\Gamma \vdash N$, where Γ is an element of the domain $\mathbb{T} \times (\mathbb{T} \to 2^{\mathbb{P}}) \times (\mathbb{T} \times \mathbb{P} \to 2^{\mathbb{P}})$, and N represents some state of the algorithm. The rules ($\mathbb{T}$-TAU) and ($\mathbb{T}$-SUSPECT) in Table 2 formally describe the conditions for a transition of an algorithm in state N to produce system transitions depending on the current information about crashes and failure detectors

[1] The term *failure detector* is overloaded to denote the single devices that are attached to processes, as well as the mathematical object that governs the output that any of these single devices may yield during runs.

$$(\mathbb{T}\text{-ENV})\ \frac{\square}{(t,F,H) \rightarrow (t{+}1,F,H)}$$

$$(\mathbb{T}\text{-TAU})\ \frac{(t,F,H) = \Gamma \rightarrow \Gamma' \qquad N \xrightarrow{\tau@i} N' \qquad i \notin F(t)}{\Gamma \vdash N \ \rightarrow\ \Gamma' \vdash N'}$$

$$(\mathbb{T}\text{-SUSPECT})\ \frac{(t,F,H) = \Gamma \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad i \notin F(t) \qquad \boxed{j \in H(i,t)}}{\Gamma \vdash N \ \rightarrow\ \Gamma' \vdash N'}$$

Table 2. Operational Semantics for the Failure Detectors of [CT96]

at any time $t \in \mathbb{T}$. Note that in both cases the process i who is responsible for the transition must not (yet) have crashed at the time t when the transition is supposed to happen: $i \notin F(t)$. Note further that if it is required to suspect some process j to perform the transition, then the respective failure detector of process i must *currently permit* to do so: $j \in H(i,t)$. It is easily possible to generalize this representation to the case where, to carry out a single transition, process i would need to suspect more than one other process; for simplicity, we only consider a single suspicion.

System runs can now be fit together dynamically as infinite sequences of system transitions that are derivable by operational semantics rules.

Definition 1. *A $\mathbb{T}(\mathcal{D})$-run is an infinite sequence $((t,F,H) \vdash N_t)_{t\in\mathbb{T}}$ generated by* ($\mathbb{T}$-ENV)*,* ($\mathbb{T}$-TAU)*, and* ($\mathbb{T}$-SUSPECT)*, for some F, H with $H \in \mathcal{D}(F)$.*

2.3 "Sufficiently Reliable" Failure Detectors

Probably the main novelty of Chandra and Toueg's paper [CT96] was the definition and study of a number of FDs $\mathcal{D}$ that only differ in their degree of reliability, as expressed by a combination of safety and liveness properties. These are formulated in terms of permitted and enforced suspicions according to the respective failures reported in F and the failure detection recorded in $H \in \mathcal{D}(F)$:

completeness addresses *crashed processes that must be suspected* by (the FDs of) "complete" processes.
accuracy addresses *correct processes that must not be suspected* by (the FDs of) "accurate" processes.

Note that these definitions refer to suspicions allowed by the output of the individual FDs at *any* time (according to H). By "complete" and "accurate" processes, we indicate that there is some flexibility in the definition of the set of processes that the property shall be imposed on. Note that H is, in principle, a total function. Therefore, at any moment, it provides FD output for every process—crashed or not—so there are at least three obvious possibilities to define the meaning of "complete" and "accurate" processes:

1. *all* processes ($\in \mathbb{P}$)
2. only processes that are still *alive* at time t ($\in \mathbb{P} \setminus F(t)$)
3. only *correct* processes ($\in \text{correct}(F) = \mathbb{P} \setminus \text{crashed}(F)$)

Obviously, it does not make much sense to speculate, as in solution 1, about the output of FDs of crashed processes, because the respective process would never ever again contact its FD.[2] It seems much more natural to select solution 2, because it precisely considers just those processes that are alive. If completeness or accuracy shall hold only eventually, then solution 3 becomes interesting: since every incorrect process that is still alive at some moment will crash later on, the property would just hold at a later time. In infinite runs, and for properties with an eventual character, solutions 2 and 3 become "equivalent" [CT96].

Eight Candidates Various instantiations of completeness and accuracy have been proposed. We recall the eight FDs of Chandra and Toueg, defined by all possible combinations of the following variations of completeness and accuracy. Note that their defining properties are quantified over *all possible runs*. For every given run, the components F and $H \in \mathcal{D}(F)$ are fixed, as well as the derived notions of which processes are considered to be correct in this run.

strong completeness *Eventually*, every process that crashes is permanently suspected by *every* correct process.

$$\forall F, H : \exists \hat{t} : \forall p \in \text{crashed}(F) : \forall q \in \text{correct}(F) : \forall t \geq \hat{t} : p \in H(q, t)$$

weak completeness *Eventually*, every process that crashes is permanently suspected by *some* correct process.

$$\forall F, H : \exists \hat{t} : \forall p \in \text{crashed}(F) : \exists q \in \text{correct}(F) : \forall t \geq \hat{t} : p \in H(q, t)$$

Combined with the strong/weak versions of completeness, the following notions of accuracy induce eight variants of FDs, with their denotations listed in brackets.

strong accuracy ($\mathcal{P}/\mathcal{Q}$) No process is suspected before it crashes.

$$\forall F, H : \forall t : \forall p, q \in \mathbb{P} \setminus F(t) : p \notin H(q, t)$$

Note that the "accuracy set" is the alive processes.

weak accuracy ($\mathcal{S}/\mathcal{W}$) Some correct process is never suspected.

$$\forall F, H : \exists p \in \text{correct}(F) : \forall t : \forall q \in \mathbb{P} \setminus F(t) : p \notin H(q, t)$$

Note that also here the "accuracy set" is the alive processes.

[2] It might be more appropriate to define H as partial function where $H(i, t)$ is only defined if $i \notin F(t)$. One might also conclude that the (F, H)-based model is too rich.

eventual strong accuracy ($\Diamond\mathcal{P}/\Diamond\mathcal{Q}$) There is a time after which *correct* processes are *not suspected* by any correct process.

$$\forall F, H : \exists \hat{t} : \forall t \geq \hat{t} : \forall p \in \text{correct}(F) : \forall q \in \text{correct}(F) : p \notin H(q, t)$$

eventual weak accuracy ($\Diamond\mathcal{S}/\Diamond\mathcal{W}$) There is a time after which *some correct* processes is *never suspected* by any correct process.

$$\forall F, H : \exists \hat{t} : \exists p \in \text{correct}(F) : \forall t \geq \hat{t} : \forall q \in \text{correct}(F) : p \notin H(q, t)$$

Note that, except under strong and weak accuracy, (the FDs of) processes that crash (in a given run) may behave completely unconstrained (in this run) before they have crashed. Although the formulation of the above FDs might appear a bit ad-hoc (see Gärtner [Gär01] for a gentle and systematic overview), some of them are known to provide the weakest FDs required to solve certain well-known distributed programming problems: $\Diamond\mathcal{W}$ solves Consensus, where less than $n/2$ processes may crash; $\mathcal{S}$ solves Consensus, where less than n processes may crash; $\mathcal{P}$ solves the Byzantine General's Problem [CT96].

Another important contribution found in [CT96] is the concept of reducibility between FDs. Essentially, it studies reduction algorithms $T_{\mathcal{D}\to\mathcal{D}'}$ that transform the outputs of $\mathcal{D}$ into outputs of $\mathcal{D}'$. As a consequence, any problem that can be solved using $\mathcal{D}'$ can also be solved using $\mathcal{D}$, written $\mathcal{D} \succeq \mathcal{D}'$. If such a relation holds in both directions, then we write $\mathcal{D} \cong \mathcal{D}'$. Interestingly, FDs with either strong or weak completeness are not that much different with respect to their ability to solve problems: $\mathcal{P} \cong \mathcal{Q}$, $\mathcal{S} \cong \mathcal{W}$, $\Diamond\mathcal{P} \cong \Diamond\mathcal{Q}$, $\Diamond\mathcal{S} \cong \Diamond\mathcal{W}$.

2.4 Another Prominent Candidate: Ω

In this subsection, we try to explain that completeness is required in the (F, H)-based model used by Chandra and Toueg only because this model is unrealistically rich, which we might regard as a deficiency of the model.

Without the completeness property, the (F, H)-based model allows a FD to be *incomplete*, i.e., to indefinitely not suspect a crashed process. We may conceive this as unrealistic if we, for instance, assume that FDs work with timeouts. Given that any crashed process can only have sent a finite number of messages before having crashed, any FD will be reporting suspicion of a crashed process at the latest after "timeout" times "number of sent messages" units of time.

It is instructive to replay the argument in the dual model of "presence detectors", where the outputs of FDs are inversed, i.e., where H tells which processes are currently "trusted" by the FD. Intuitively, this dual model feels more direct since the trust in a process to be alive is often based on "feeling its heartbeat". Of course, since it is just a mathematically dual, also this model is too rich: an incomplete FD may be expressed by always listing a crashed process as being trusted.[3] However, it is easy to constrain this model as to avoid incomplete FDs.

[3] In this model, completeness more intuitively specifies the natural requirement that "you cannot feel the heartbeat of a crashed process infinitely long". Strong com-

$$(\Omega\text{-ENV}) = (\mathbb{T}\text{-ENV}) \qquad (\Omega\text{-TAU}) = (\mathbb{T}\text{-TAU})$$

$$(\Omega\text{-SUSPECT})\ \frac{(t,F,H) = \Gamma \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j @ i} N' \qquad i \notin F(t) \qquad \boxed{j \neq H(i,t)}}{\Gamma \vdash N \quad \rightarrow \quad \Gamma' \vdash N'}$$

Table 3. Operational Semantics for the "Presence" Detector Ω of [CHT96]

Interestingly, the detector Ω, as introduced in another paper by Chandra and Toueg, jointly with Hadzilacos [CHT96], represents one particular model variant of presence detectors that is "sufficiently poor" to render incomplete FDs impossible. With Ω, every FD at any moment in time outputs *only a single process* that is believed to be "correct"[4] or trusted; $H : \mathbb{P} \times \mathbb{T} \rightarrow \mathbb{P}$.

(Ω) Eventually, all correct processes always trust the same correct process.

$$\exists \hat{t} \in \mathbb{T} : \exists q \in \text{correct}(F) : \forall p \in \text{correct}(F) : \forall t > \hat{t} : H(p,t) = q$$

Observe why it is no longer possible to indefinitely enforce trust on crashed processes: since the output of H contains only a single process, the associated process *can* always suspect any of the $n-1$ other processes. The property above eventually stabilizes to a single correct process, then permanently allowing the suspicion of the remaining other processes.

It is straightforward to provide an operational semantics view for Ω by adapting the previous configurations to the new type of H. The rule (Ω-SUSPECT) in Table 3 shows the duality of presence detectors versus failure detectors: the condition on a suspected j is inversed.

Definition 2. *A $\mathbb{T}(\Omega)$-run is an infinite sequence $((t,F,H) \vdash N_t)_{t \in \mathbb{T}}$ generated by (Ω-ENV), (Ω-TAU), and (Ω-SUSPECT), for some F, H with $H \in \Omega(F)$.*

The FD Ω was introduced only as an auxiliary concept in the proof that $\Diamond\mathcal{W}$ is the weakest FD solving Consensus, which works because $\Omega \cong \Diamond\mathcal{W}$.

3 Proposal of a New Model of FDs

There are a number of observations on the (F,H)-based specification of FDs that motivate us to do it differently.

from static to dynamic The use of (F,H) as "predicting" the failures of processes and their detection by others in a run appears to be counter-intuitive from the point of view of programming language semantics where events of a computation are to happen dynamically and possibly non-deterministically.

pleteness makes sense in precisely this respect. The intuition of weak completeness is less clear: why should the heartbeat argument apply to only one process?

[4] Original called this way in [CHT96], but it is different from the notion of a correct process which is precisely about a full run, and not just one moment in it.

from failure detection to presence detection In Subsection 2.4, we mentioned the problem of completeness, which is inherent in the (F, H)-model, and how Ω does a good job in avoiding the problem in a poorer dual model. We propose to use similar ideas also for FDs that are not equivalent to Ω.

intuition mismatch The use of a *total* function H to model outputs of FD modules counters the intuition that a process crash should imply the crash of its attached FD at the same time. This could be repaired by making H a partial function that is only defined when the respective process has not yet crashed. Below, we go even further and replace the H completely.

We do not see the need to record in H an abundance of unreliable information. The quest to model a minimal amount of information has two consequences.

from unstable to stable The use of unreliable FD-outputs listed in H very carefully models a lot of unstable (and therefore questionable!) information, namely information that changes nondeterministically over time. However, in order to characterize the above six FD properties, only (eventually) stable information is used. We therefore propose to model only this kind of stable information—once it has become stable—and to freely allow any suspicions for which there is no stable information yet, as in the "poor" output of Ω.

from local to global Chandra and Toueg intended to give an abstract model of FDs, but we feel that by attaching individual modules to every process it is still too concrete, i.e., close to implementation aspects. Furthermore, the above completeness and accuracy properties are all defined globally on the set of all FD modules, not on individual ones. Thus, we rather propose to model FDs as a single global entity and have all processes share access to it. As a side-effect, the freedom (="imprecision") in the formulation of FD properties with respect to the proper choice of the "accuracy set" disappears.

Summing up, we seek to model the environment of process networks dynamically as a global device that exclusively stores stable information. But, apart from crashes that occur irrevocably, which information should this precisely be?

Looking again at the previous (F, H)-based FDs, the principle behind the more complicated notions of accuracy seems to be that of "justified trust". Correct processes—those that, according to F, were *immortal* in the given run—are *trusted forever* (according to H) in the given run, either eventually or already from the very beginning. If, in some dynamic operational semantics scenario, we want to model the moment when such a process becomes trusted, we must ensure this process not to crash afterwards—it must become immortal at this very moment. Then, we call such a process *trusted-immortal*.

4 An Operational Semantics View of the New Model

As motivated in the previous section, we propose a new model—defined by its operational semantics—that can be used to represent all of the FDs of [CT96] solely based on stable/reliable information that is not fixed before a run starts,

$$(\mathbb{D}\text{-ENV})\ \frac{(\mathsf{TI} \cup TI) \cap C = \emptyset \qquad (\mathsf{C} \cup C) \cap TI = \emptyset \qquad |\mathsf{C} \cup C| \leq \text{maxfail}(n)}{(\mathsf{TI}, \mathsf{C}) \quad \rightarrow \quad (\mathsf{TI} \uplus TI, \mathsf{C} \uplus C)}$$

$$(\mathbb{D}\text{-TAU})\ \frac{(\mathsf{TI}, \mathsf{C}) = \Gamma \ \rightarrow \Gamma' \qquad N \xrightarrow{\tau@i} N' \qquad i \notin \mathsf{C}}{\Gamma \vdash N \ \rightarrow \Gamma' \vdash N'}$$

$$(\mathcal{X}\text{-SUSPECT})\ \frac{(\mathsf{TI}, \mathsf{C}) = \Gamma \ \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad i \notin \mathsf{C} \qquad \boxed{\text{condition}_{\mathcal{X}}(\Gamma, j)}}{\Gamma \vdash N \ \rightarrow \Gamma' \vdash N'}$$

Table 4. Operational Semantics Scheme with Reliable Information

but is dynamically appearing along its way. It turns out that two kinds of information suffice: (1) which processes have crashed, and (2) which processes have become *trusted-immortal*. Both kinds of information may occur at any moment in time, and they remain irrevocable in any continuation of the current run.

We use the symbol $\mathbb{D}$ to recall the softer more *d*ynamic character as opposed to time $\mathbb{T}$ just passing for some predefined crash and detection schedule.

Modeling Stable Reliable Information Rule ($\mathbb{D}$-ENV) in Table 4 precisely models the nondeterministic appearance of crashed and trusted-immortal processes in full generality. Environments $\Gamma = (\mathsf{TI}, \mathsf{C}) \in 2^{\mathbb{P}} \times 2^{\mathbb{P}}$ record sets TI of trusted-immortal processes and sets C of crashed processes. In a single step, an environment may be increased by further trusted-immortal processes ($\in TI$) and further crashed processes ($\in C$). The two empty-intersection conditions on TI and C assure a simple sanity property: processes shall not be crashed and trusted-immortal at the same time. Note that we also assume the operator $\uplus$ to imply the empty intersection of its operands: processes may crash or become trusted-immortal only once. The condition concerning maxfail(n) is obvious: we should not have more processes crash than permitted. The sets TI and C may both be empty, which implies that the environment may also do idle steps; this is necessary for runs whose number of steps is greater than the number of processes, like in the infinite runs that we are looking at.

Rule ($\mathbb{D}$-TAU) straightforwardly permits actions $\tau@i$ if $i \notin \mathsf{C}$. Rule ($\mathcal{X}$-SUSPECT) requires in addition that the suspected process j is permitted to be suspected by Γ, depends on the FD accuracy that we intend to model.

Failure Detection In our model, trusted-immortal processes are intended to be never again suspected by *any* other process. In Table 5, we specify different incarnations for the rule ($\mathcal{X}$-SUSPECT) that are targeted at the the various notions of accuracy that our FDs are intended to satisfy.

Strong Accuracy (as in $\mathcal{P}/\mathcal{Q}$) can be expressed very simply and directly in our environment model, because it does not explicitly talk about *correct* processes.

$$(\mathcal{P}/\mathcal{Q}\text{-SUSPECT})\ \frac{(\mathsf{TI},\mathsf{C}) = \Gamma\ \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad i \notin \mathsf{C} \qquad \boxed{j \in \mathsf{C}}}{\Gamma \vdash N\ \rightarrow \Gamma' \vdash N'}$$

$$(\mathcal{S}/\mathcal{W}\text{-SUSPECT})\ \frac{(\mathsf{TI},\mathsf{C}) = \Gamma\ \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad i \notin \mathsf{C} \qquad \boxed{j \notin \mathsf{TI} \neq \emptyset}}{\Gamma \vdash N\ \rightarrow \Gamma' \vdash N'}$$

$$(\Diamond\text{-SUSPECT})\ \frac{(\mathsf{TI},\mathsf{C}) = \Gamma\ \rightarrow \Gamma' \qquad N \xrightarrow{\text{suspect}_j@i} N' \qquad i \notin \mathsf{C} \qquad \boxed{j \notin \mathsf{TI}}}{\Gamma \vdash N\ \rightarrow \Gamma' \vdash N'}$$

Table 5. Operational Semantics with Reliable Detectors

Rule ($\mathcal{P}/\mathcal{Q}$-SUSPECT) says precisely that "no site is suspected before it has crashed" by requiring that any suspected process j must be part of the set C. Note that the component TI is not used at all; if we were interested in just strong accuracy, it would suffice to record information about crashed processes.

Definition 3. *A $\mathbb{D}(\mathcal{P}/\mathcal{Q})$-run is an infinite sequence $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ generated by* ($\mathbb{D}$-ENV)*,* ($\mathbb{D}$-TAU)*, and* ($\mathcal{P}/\mathcal{Q}$-SUSPECT)*.*

Weak Accuracy (as in $\mathcal{S}/\mathcal{W}$) builds on rule ($\mathcal{S}/\mathcal{W}$-SUSPECT). In order to get that "some correct process is never suspected", the idea is that some process must become trusted-immortal before *any* suspicion in the system may take place. A process i may always suspect process j unless the failure detector tells otherwise, i.e., unless it imposes to trust j by $j \notin \mathsf{TI}$. Note that if we allowed suspicions before the "election" of at least one trusted-immortal, then even a process becoming trusted-immortal later on might have been suspected before.

Definition 4. *A $\mathbb{D}(\mathcal{S}/\mathcal{W})$-run is an infinite sequence $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ generated by* ($\mathbb{D}$-ENV)*,* ($\mathbb{D}$-TAU)*, and* ($\mathcal{S}/\mathcal{W}$-SUSPECT)*.*

The other versions of "eventual accuracy" cannot be expressed solely by operational semantics rules; additional liveness properties are required.

Eventual Weak Accuracy (as in $\Diamond\mathcal{S}/\Diamond\mathcal{W}$) builds on rule ($\Diamond$-SUSPECT), which is a slightly more liberal variant of ($\mathcal{S}/\mathcal{W}$-SUSPECT): suspicions may take place without some process having become trusted-immortal. However, we need to add to the condition on runs that *eventually* at least one process indeed turns out to be trusted-immortal such it cannot be suspected *afterwards*. The detector Ω of [CHT96] is very close to this very intuition, as well (confirming $\Omega \cong \Diamond\mathcal{W}$).

Definition 5. *A $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$-run is an infinite sequence $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ generated by* ($\mathbb{D}$-ENV)*,* ($\mathbb{D}$-TAU)*, and* ($\Diamond$-SUSPECT)*, where there is a reachable state $(\mathsf{TI}_{\hat{t}}, \mathsf{C}_{\hat{t}}) \vdash N_{\hat{t}}$ with $\mathsf{TI}_{\hat{t}} \neq \emptyset$.*

Eventual Strong Accuracy (as in $\Diamond\mathcal{P}/\Diamond\mathcal{Q}$) is a nuance trickier: like its weak counterpart, it builds directly on rule ($\Diamond$-SUSPECT) and adds to it a condition on runs, but now a more restrictive one: in any run, there must be a state $\Gamma = (\mathsf{TI}, \mathsf{C})$ with $\mathsf{TI} \cup \mathsf{C} = \mathbb{P}$. In such a state, all decisions about correctness and crashes have been taken. This witnesses that $\Diamond\mathcal{P}$ is called *eventually perfect* [CT96]: in fact, the condition $j \notin \mathsf{TI}$ becomes equivalent to the perfect condition $j \in \mathsf{C}$ of $\mathcal{P}/\mathcal{Q}$.

Definition 6. *A $\mathbb{D}(\Diamond\mathcal{P}/\Diamond\mathcal{Q})$-run is an infinite sequence $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ generated by* ($\mathbb{D}$-ENV)*,* ($\mathbb{D}$-TAU)*, and* ($\Diamond$-SUSPECT)*, where there is a reachable state $(\mathsf{TI}_{\hat{t}}, \mathsf{C}_{\hat{t}}) \vdash N_{\hat{t}}$ with $\mathsf{TI}_{\hat{t}} \cup \mathsf{C}_{\hat{t}} = \mathbb{P}$.*

Note that we did not explicitly mention completeness properties in our redefinitions. In fact, as inspired by Ω (see Subsection 2.4), they are built-in implicitly. With the rules ($\mathcal{S}/\mathcal{W}$-SUSPECT) and ($\Diamond$-SUSPECT), the suspicion of a crashed process is *always* allowed. With rule ($\mathcal{P}/\mathcal{Q}$-SUSPECT), the suspicion of a crashed process is allowed *immediately after it crashed*. Note that completeness is thus provided in the strongest possible manner, strictly implying strong completeness. It does not only hold eventually, but "as soon as possible", subject only to accuracy constraints. This built-in strength is a consequence of the principle of our model to store only stable information to govern the possibility of suspicions, leaving complete freedom to suspect in all those cases where the fate of the suspected process has not yet been decided on in the current run.

Having *re*defined runs that are allowed for particular FDs, we must of course also argue that they correspond to the original counterparts of Chandra and Toueg. This we do formally in the following section.

5 Validation of the New Model

We compare our $\mathbb{D}$-representations of FDs with the $\mathbb{T}$-representations proposed in [CT96] extensionally through mutual "inclusion" of their sets of runs. Essentially, we are looking for a mutual simulation of $\mathbb{T}$-runs and $\mathbb{D}$-runs sharing the same network run (by projecting onto the N-component). To this aim, it will be crucial to formally relate the respective notions of environment.

Definition 7. $(\mathsf{TI}, \mathsf{C})$ corresponds to (t, F, H)*, written* $(\mathsf{TI}, \mathsf{C}) \sim (t, F, H)$*, if*

1. $\mathsf{C} = F(t)$*, and*
2. $j \in \mathsf{TI}$ *implies that* $\forall i \in \mathbb{A}_t : \forall t' \geq t : j \notin H(i, t')$.

where $\mathbb{A}_t := F(t)$ or $\mathbb{A}_t := \mathrm{correct}(F)$ depending on the respective "accuracy set" (see Section 2) of the version of accuracy that we are considering.

It is also convenient to use case 2 in the opposite direction for the case of $t' = t$: If $(\mathsf{TI}, \mathsf{C}) \sim (t, F, H)$, then $\forall i \in \mathbb{A}_t : \forall j : j \in H(i, t)$ implies that $j \notin \mathsf{TI}$.

Example 2. For all F, H: $(\emptyset, F(0)) \sim (0, F, H)$. This correspondence holds, because $\mathsf{C} = F(0)$ is defined to satisfy case 1, and $\mathsf{TI} = \emptyset$ trivially implies case 2.

We present the main theorems in a generic manner. Let $\mathcal{D}^{\mathrm{s}} \in \{ \mathcal{P}, \mathcal{S}, \Diamond\mathcal{P}, \Diamond\mathcal{S} \}$ and $\mathcal{D}^{\mathrm{w}} \in \{ \mathcal{Q}, \mathcal{W}, \Diamond\mathcal{Q}, \Diamond\mathcal{W} \}$ denote the FDs with respect to strong and weak completeness. We use $\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}}$ to conveniently denote the respective variants.

First, we offer a rather obvious observation.

Lemma 1. *Let* $\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}} \in \{ \mathcal{P}/\mathcal{Q},\ \mathcal{S}/\mathcal{W},\ \Diamond\mathcal{P}/\Diamond\mathcal{Q},\ \Diamond\mathcal{S}/\Diamond\mathcal{W} \}$.
Every $\mathbb{T}(\mathcal{D}^{\mathrm{s}})$*-run is also a* $\mathbb{T}(\mathcal{D}^{\mathrm{w}})$*-run.*

Proof. Trivial, because strong completeness implies weak completeness. □

Now, we show the main mutual simulation theorem. It also underlines the fact that the kind of completeness does not really matter much, which confirms the respective result of [CHT96], that the eight FDs collapse into just four.

Theorem 1. *Let* $\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}} \in \{ \mathcal{P}/\mathcal{Q},\ \mathcal{S}/\mathcal{W},\ \Diamond\mathcal{P}/\Diamond\mathcal{Q},\ \Diamond\mathcal{S}/\Diamond\mathcal{W} \}$.

1. *If* $((t, F, H) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\mathcal{D}^{\mathrm{w}})$*-run,*
   then there is $(\Gamma_t)_{t\in\mathbb{T}}$ *such that* $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{D}(\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}})$*-run.*
2. *If* $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{D}(\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}})$*-run,*
   then there are F, H *such that* $((t, F, H) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\mathcal{D}^{\mathrm{s}})$*-run.*

Note that part 1 requires only a $\mathbb{T}(\mathcal{D}^{\mathrm{w}})$-run, while part 2 provides a $\mathbb{T}(\mathcal{D}^{\mathrm{s}})$-run, which is due to the strength of the built-in completeness of the $\mathbb{D}$-model.

Proof. See Appendix A.

We also show that Ω is "equivalent" to $\Diamond\mathcal{S}/\Diamond\mathcal{W}$.

Theorem 2.

1. *If* $((t, F, H) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\Omega)$*-run,*
   then there is $(\Gamma_t)_{t\in\mathbb{T}}$ *such that* $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$*-run.*
2. *If* $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$*-run,*
   then there are F, H *such that* $((t, F, H) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\Omega)$*-run.*

This theorem allows us to denote $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$-runs as $\mathbb{D}(\Omega)$-runs, and justifies the model that we used when proving a Consensus algorithm correct in [NFM03].

We could prove Theorem 2 "directly" just like we did in the proof of Theorem 1. However, we can also profit from the work of Chandra and Toueg, whose results translate into our setting as in Proposition 1 below.

There are algorithmic FD-transformations $T_{\Omega\to\Diamond\mathcal{W}}$ and $T_{\Diamond\mathcal{S}\to\Omega}$ such that

- for all $F, H : H \in \Omega(F)$ implies $T_{\Omega\to\Diamond\mathcal{W}}(H) \in \Diamond\mathcal{W}(F)$, and
- for all $F, H : H \in \Diamond\mathcal{S}(F)$ implies $T_{\Diamond\mathcal{S}\to\Omega}(H) \in \Omega(F)$.

Proposition 1 ([CHT96]). *Let* R *denote* $((t, F, H) \vdash N_t)_{t\in\mathbb{T}}$.

1. *If* R *is a* $\mathbb{T}(\Omega)$*-run, then* $((t, F, T_{\Omega\to\Diamond\mathcal{W}}(H)) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\Diamond\mathcal{W})$*-run.*
2. *If* R *is a* $\mathbb{T}(\Diamond\mathcal{S})$*-run, then* $((t, F, T_{\Diamond\mathcal{S}\to\Omega}(H)) \vdash N_t)_{t\in\mathbb{T}}$ *is a* $\mathbb{T}(\Omega)$*-run.*

Proof (of Theorem 2).

1. By Proposition 1(1) and Theorem 1(1).
2. By Theorem 1(2) and Proposition 1(2).

□

6 Conclusions

We propose a new model of FDs that we consider easier to understand, easier to work with, and more natural than the model used by Chandra and Toueg.

- It is arguably easier to understand, because the environment information that it provides to check the conditions of the rules of Table 5 is designed to be minimal—shared, global, and reliably stable—and to support just those moments when suspicions are effectively needed with a maximal flexibility. This is in contrast to the (F, H)-based model with individual FD modules that at any moment in time produce unreliable output (sets of currently suspected processes) that their master process may not be interested in for a long time; they might even have crashed already.

  It is certainly easier to understand from the computational point of view due to the dynamic modeling of events concerning crashes and their detection.
- It is easier to work with, because it is generally more light-weight in that only stable information is considered. Moreover, our model is simpler since the strongest possible completeness property is built-in, so we do not have to explicitly care about it when, e.g., looking for the weakest FD solving a distributed computing problem in our model. Also, to exploit any of the accuracy properties in proofs, it suffices to check rather simple syntactic conditions in states of a given run. Starting in the initial state, a finite search suffices, profiting from the built-in monotonicity of the stable information.
- It is more natural, for two reasons: (1) It avoids the need to impose additional completeness properties by allowing dynamic nondeterminism on suspicions until they possibly become forbidden forever. (2) It avoids the problem of selecting the "accuracy set" of eventually reliable individual FDs, where the (F, H)-based model leaves the choice to FDs of correct, alive, or all processes. In our model, FD modules are not modeled individually as belonging each to individual processes, but failure detection is modeled by using a global shared entity. In a dynamic operational scenario as ours, the only reasonable choice for the counterpart of the "accuracy set" is the alive processes.

In this paper, we concentrated on the FDs presented in [CT96, CHT96], but we see no obstacle in applying our principles of Section 3 to other (F, H)-based FDs.

References

[CHT96] T. D. Chandra, V. Hadzilacos and S. Toueg. The Weakest Failure Detector for Solving Consensus. *Journal of the ACM*, 43(4):685–722, 1996.

[CT96] T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, 43(2):225–267, 1996.

[Gär01] F. C. Gärtner. A Gentle Introduction to Failure Detectors and Related Problems. Technical Report TUD-BS-2001-01, TU Darmstadt, Apr. 2001.

[NFM03] U. Nestmann, R. Fuzzati and M. Merro. Modeling Consensus in a Process Calculus. In R. Amadio and D. Lugiez, eds, *Proceedings of CONCUR 2003*, volume 2761 of *LNCS*, pages 399–414. Springer, Aug. 2003.

A Proofs of Theorem 1

Proof. For each case of $\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}}$, the proof follows the same pattern.

1. Let $((t,F,H) \vdash N_t)_{t\in\mathbb{T}}$ be a $\mathbb{T}(\mathcal{D}^{\mathrm{w}})$-run. Any step

$$(t,F,H) \vdash N_t \;\rightarrow (t{+}1,F,H) \vdash N_{t+1} \tag{1}$$

must now be simulated by a *derivable* step

$$\Gamma_t \vdash N_t \;\rightarrow \Gamma_{t+1} \vdash N_{t+1} \tag{2}$$

for some $(\Gamma_t)_{t\in\mathbb{T}}$. In order to *verify* the derivability of transition (2), the semantics tells us to check the two possibilities of action $\tau@i$ and $\mathrm{suspect}_j@i$ carried out by N_t. Naturally, knowing the derivability of transition (1), we can deduce some knowledge about the output of F and H at time t.
We construct Γ_t using this knowledge; formally, we establish $\Gamma_t \sim (t,F,H)$. To this aim, we prove that the correspondence is preserved under appropriately defined environment transitions, indicated by $\Gamma_t :\!\rightarrow \Gamma_{t+1}$.

Lemma 2. *If* $\Gamma_t \sim (t,F,H)$ *and* $\boxed{\Gamma_t :\!\rightarrow \Gamma_{t+1}}$*, then* $\Gamma_{t+1} \sim (t{+}1,F,H)$.

Proof (of Lemma 2 by construction of $\Gamma_t :\!\rightarrow \Gamma_{t+1}$*).*
In fact, it is never a problem to have $\Gamma_{t+1} \sim (t{+}1,F,H)$ satisfy its first condition $\mathsf{C}(t{+}1) = F(t{+}1)$ by simply setting $\mathsf{C}(t{+}1) := F(t{+}1)$. Note that since F is steadily increasing, then the condition $\mathsf{C}\uplus C$ of ($\mathbb{D}$-ENV) is satisfied. In order to have $\Gamma_{t+1} \sim (t{+}1,F,H)$ satisfy its second condition, we must be very cautious *when* we add elements to TI_t to become TI_{t+1}.

case $\mathcal{D}^{\mathrm{w}}{=}\mathcal{Q}$ **:** Never add elements to TI_{t+1}. Then any transition $\Gamma_t \;\rightarrow \Gamma_{t+1}$ is derivable, and Lemma 2 holds immediately.

case $\mathcal{D}^{\mathrm{w}}{=}\mathcal{W}$ **:** Here, we may assume *weak accuracy*: some correct process p is never suspected. We simply set $\mathsf{TI}_0 := \{p\}$ to get the desired effect, and afterwards never again change the TI_t component.
Note that with $\Gamma_0 := (\{p\}, F(0))$, we have $\Gamma_0 \sim (0,F,H)$ precisely due to the weak accuracy assumption of the $\mathbb{T}(\mathcal{W})$-run.
Note that then, as a consequence, Lemma 2 immediately holds for all transitions (of all $t > 0$). Of course, it is also needed in these transition to check that p will not accidentally be chosen to crash; this is guaranteed, because of weak accuracy requiring a *correct* process, which is thus $\notin \mathrm{crashed}(F)$ and will therefore never enter any C_t.

case $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{W}$ **:** Here, we may assume *eventual weak accuracy*: eventually, after time $\hat{t}$, some correct process p will never again be suspected, so we set $\mathsf{TI}_0 = \cdots = \mathsf{TI}_{\hat{t}-1} = \emptyset$ and $\mathsf{TI}_{\hat{t}} := \{p\}$.
The critical transition for Lemma 2 to hold is $\Gamma_{\hat{t}-1} \;\rightarrow \Gamma_{\hat{t}}$. Here, the eventual weak accuracy property makes $\Gamma_{\hat{t}} \sim (\hat{t},F,H)$ satisfy condition 2.

case $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{Q}$ **:** Here, we may assume *eventual strong accuracy*: eventually, after time $\hat{t}$, no correct process p will ever again be suspected, so we set $\mathsf{TI}_0 = \cdots = \mathsf{TI}_{\hat{t}-1} = \emptyset$ and $\mathsf{TI}_{\hat{t}} := \mathrm{correct}(F)$.
The argument for Lemma 2 to hold is a replay of the previous case. □

The "constructive preservation" property of Lemma 2 provides us with the required assumptions to simulate subsequent steps and, thus, allows us to iteratively simulate the whole infinite run, starting in all cases but one at $\Gamma_0 = (\emptyset, F(0))$. Let us first look at individual simulation steps.

Lemma 3. *If $\Gamma_t \sim (t, F, H)$ and transition* (1)*, then transition* (2)*.*

Proof (of Lemma 3). Check the conditions to derive transitions (1), either due to ($\mathbb{T}$-TAU) or due to ($\mathbb{T}$-SUSPECT), and then observe that the correspondence of environments also enables to derive the respective transition (2). If transition (1) required $i \notin F(t)$, which it does in both ($\mathbb{T}$-TAU) and ($\mathbb{T}$-SUSPECT), then the first condition on $\sim$ provides us with the required $i \notin \mathsf{C}_{\Gamma_t}$ in ($\mathbb{D}$-TAU) and in any of the ($\mathcal{X}$-SUSPECT). We may then focus on the more interesting boxed condition of the rules ($\mathcal{X}$-SUSPECT).

case $\mathcal{D}^{\mathrm{w}}{=}\mathcal{Q}$: The enabling conditions for ($\mathcal{P}/\mathcal{Q}$-SUSPECT) only depend on the respective C and hold trivially.

case $\mathcal{D}^{\mathrm{w}}{=}\mathcal{W}$: Recall that the second condition on $\sim$ tells us that $j \in H(i,t)$ implies $j \notin \mathsf{TI}_t$. Since, by definition of $\Gamma_t :\!\rightarrow \Gamma_{t+1}$ for the case $\mathcal{W}$, also $\mathsf{TI}_t \neq \emptyset$ holds for all $t > 0$. Together, this ($j \notin \mathsf{TI}_t \neq \emptyset$) implies that suspicion steps can always be simulated using ($\mathcal{S}/\mathcal{W}$-SUSPECT).

cases $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{W}$ and $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{Q}$: Again, $j \in H(i,t)$ implies $j \notin \mathsf{TI}_t$. This is already sufficient to simulate suspicion steps using ($\Diamond$-SUSPECT). □

The basic requirement on $\mathbb{D}(\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}})$-runs (matching the Definitions 3–6) is to consist of sequences of derivable transitions. This holds in all cases by the infinite iteration of Lemma 3. However, the $\Diamond$-runs (i.e., the $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$-runs and $\mathbb{D}(\Diamond\mathcal{P}/\Diamond\mathcal{Q})$-runs) require an additional condition.

case $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{W}$: The resulting run is a $\mathbb{D}(\Diamond\mathcal{S}/\Diamond\mathcal{W})$-run, because there is $\hat{t}$ in which we set $\mathsf{TI}_{\hat{t}} := \{p\}$.

case $\mathcal{D}^{\mathrm{w}}{=}\Diamond\mathcal{Q}$: The resulting run is a $\mathbb{D}(\Diamond\mathcal{P}/\Diamond\mathcal{Q})$-run, because there is $\hat{t}$ in which we set $\mathsf{TI}_{\hat{t}} := \mathrm{correct}(F)$. However, this is not necessarily yet the moment needed to establish a $\mathbb{D}(\Diamond\mathcal{P}/\Diamond\mathcal{Q})$-run. In order to find this moment $\hat{t}'$, we only have to wait until all the processes in $\mathrm{crashed}(F)$ have actually crashed. By our definition of C_t, the required property of Definition 5 that $\mathsf{TI}_{\hat{t}'} \cup \mathsf{C}_{\hat{t}'} = \mathbb{P}$ becomes valid.

2. The structure of this proof is similar to the previous one.
Let $R := (\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$ be a $\mathbb{D}(\mathcal{D}^{\mathrm{s}}/\mathcal{D}^{\mathrm{w}})$-run with $\Gamma_t = (\mathsf{TI}_t, \mathsf{C}_t)$ for $t \in \mathbb{T}$.
Before, we constructed a sequence of Γ_t for some fixed F, H such that $\Gamma_t \sim (t, F, H)$ for all $t \geq 0$. Here, we construct the functions F_R, H_R with $H_R \in \mathcal{D}^{\mathrm{s}}(F_R)$ by means of the information found in the various Γ_t. While it is obvious that F_R should very closely follow the information recorded in C, there is a lot of freedom in the choice of H_R—allowing suspicions or not—because it is supposed to contain a lot of information that is never checked in the projection $(N_t)_{t\in\mathbb{T}}$ of R. We are going to choose H_R to permit the maximum amount of suspicions. As a consequence, this choice gives us the strongest possible completeness property essentially for free.

Independent of the FD, we may transform any given $\mathbb{D}$-run into a $\mathbb{T}$-run by constructing F_R in a uniform manner:

$$\forall t \in \mathbb{T} : F_R(t) \stackrel{\text{def}}{=} \mathsf{C}_t$$

For the construction of H_R, we need to distinguish among the cases of $\mathcal{D}^{\text{s}}$ between the perfect FD ($\mathcal{P}$) and the imperfect FDs ($\mathcal{S}, \Diamond\mathcal{P}, \Diamond\mathcal{S}$).
For the perfect FD, we set:

$$\forall t \in \mathbb{T} : \forall i \in \mathbb{P} : H_R(i,t) \stackrel{\text{def}}{=} \mathsf{C}_t$$

For the imperfect FDs, we need some auxiliary functions.
Let R denote the run $(\Gamma_t \vdash N_t)_{t\in\mathbb{T}}$.
Let $\text{ti}(R) := \bigcup_{t\in\mathbb{T}} \mathsf{TI}_t$ denote the set of trusted-immortal processes of R.
If $j \in \text{ti}(R)$, then let t_j be (uniquely!) defined by $j \notin \mathsf{TI}_{t_j-1}$ and $j \notin \mathsf{TI}_{t_j}$, thus denoting the moment in which j becomes trusted-immortal.
To define H_R, we start by allowing suspicions of every process by every other process at any time. (We may, of course, leave out useless permissions to self-suspect, but this does not matter for the result.)

$$\forall t \in \mathbb{T} : \forall i \in \mathbb{P} : H_R(i,t) \stackrel{\text{def}}{=} \mathbb{P}$$

From these sets, we subtract (in imperative programming style, with $\forall$ denoting `forall` loops) a number of processes, depending on the time at which trusted-immortal processes have become so.

$$\forall j \in \text{ti}(R) : \forall i \in \mathbb{P} : \forall t \geq t_j : H_R(i,t) \stackrel{\text{def}}{=} H_R(i,t) \setminus \{j\}$$

By construction, we immediately get that $\Gamma_t \sim (t, F_R, H_R)$.
The correspondence is also preserved by the Γ-transitions of R.

Lemma 4. *If $\Gamma_t \sim (t, F_R, H_R)$ and $\Gamma_t \longrightarrow \Gamma_{t+1}$, then $\Gamma_{t+1} \sim (t{+}1, F_R, H_R)$.*

As before, we need a simulation lemma; now, it addresses the other direction.

Lemma 5. *If $\Gamma_t \sim (t, F_R, H_R)$ and transition (2), then transition (1).*

It holds for symmetric reasons, because it is also exploiting the same correspondence properties of the constructed pair F_R, H_R.

The only difference (in fact, a simplification) in the final iteration of the simulation is in the case of the $\mathbb{D}(\Diamond\mathcal{P}/\Diamond\mathcal{Q})$-run. Here, the moment $\hat{t}$ that according to Definition 6 shows $\mathsf{TI}_{\hat{t}} \cup \mathsf{C}_{\hat{t}} = \mathbb{P}$ is *precisely* the earliest moment that provides weak accuracy for the $\mathbb{T}(\Diamond\mathcal{P})$-run.

Now, the remaining argument is to show that the components (F_R, H_R) of a $\mathbb{T}(\mathcal{D}^{\text{s}})$-run also satisfy *strong completeness*. In fact, by construction, we have the following completeness property for the case of imperfect FDs:

in every run,
every crashed process is always suspected by every process.

This obviously strictly implies strong completeness. The case of perfect FDs is similar, just replacing the words *always suspected* by the words *suspected right after it crashed*, which also strictly implies strong completeness. □