# Derivation of Heard-Of Predicates From Elementary Behavioral Patterns

Adam Shimi, Aurélie Hurault, and Philippe Queinnec

IRIT – Université de Toulouse, 2 rue Camichel, F-31000 Toulouse, France
{firstname}.{lastname}@irit.fr

**Abstract.** There are many models of distributed computing, and no unifying mathematical framework for considering them all. One way to sidestep this issue is to start with simple communication and fault models, and use them as building blocks to derive the complex models studied in the field. We thus define operations like union, succession or repetition, which makes it easier to build complex models from simple ones while retaining expressivity.
To formalize this approach, we abstract away the complex models and operations in the Heard-Of model. This model relies on (possibly asynchronous) rounds; sequence of digraphs, one for each round, capture which messages sent at a given round are received before the receiver goes to the next round. A set of sequences, called a heard-of predicate, defines the legal communication behaviors – that is to say, a model of communication. Because the proposed operations behave well with this transformation of operational models into heard-of predicates, we can derive bounds, characterizations, and implementations of the heard-of predicates for the constructions.

**Keywords:** Message-passing · Asynchronous Rounds · Failures · Heard-Of Model

## 1 Introduction

### 1.1 Motivation

Let us start with a round-based distributed algorithm; such an algorithm is quite common in the literature, especially in fault-tolerant settings. We want to formally verify this algorithm using the methods of our choice: proof-assistant, model-checking, inductive invariants, abstract interpretation... But how are we supposed to model the context in which the algorithm will run? Even a passing glance at the distributed computing literature shows a plethora of models defined in the mixture of english and mathematics.

Thankfully, there are formalisms for abstracting round-based models of distributed computing. One of these is the Heard-Of model of Charron-Bost and Schiper [4]; it boils down the communication model to a description of all accepted combinations of received messages. Formally, this is done by considering communications graphs, one for each round, and taking the sets of infinite sequences of graphs that are allowed by the model. Such a set is called a heard-of predicate, and captures a communication model.

An angle of attack for verification is therefore to find the heard-of predicate corresponding to a real-world environment, and use the techniques from the literature to verify an algorithm for this heard-of predicate. But which heard-of predicate should be used? What is the "right" predicate for a given environment? For some cases, the predicates are given in Charron-Bost and Schiper [4]; but this does not solve the general case.

Actually, the answer is quite subtle. This follows from a fundamental part of the Heard-Of model: communication-closedness [7]. This means that for $p$ to use a message from $q$ at round $r$, $p$ must receive it before or during its own round $r$. And thus, knowing whether $p$ receives the message from $q$ at the right round or not depends on how $p$ waits for messages. That is, it depends on the specifics of how rounds are implemented on top of it.

Once again, the literature offers a solution: Shimi et al. [12] propose to first find a delivered predicate – a description of which messages will eventually be delivered, without caring about rounds –, and then to derive the heard-of predicate from it. This derivation explicitly studies strategies, the aforementioned rules for how processes waits for messages before changing round.

But this brings us back to square one: now we are looking for the delivered predicate corresponding to a real-world model, instead of the heard-of predicate. Basic delivered predicates for elementary failures are easy to find, but delivered predicates corresponding to combinations of failures are often not intuitive.

In this paper, we propose a solution to this problem: building a complex delivered predicate from simpler ones we already know. For example, consider a system where one process can crash and may recover later, and another process can definitively crash. The delivered predicate for at most one crash is $PDel_1^{crash}$, and the predicate where all the messages are delivered is $PDel^{total}$. Intuitively, a process that can crash and necessarily recover is described by the behavior of $PDel_1^{crash}$ followed by the behavior of $PDel^{total}$. We call this the succession of these predicates, and write it $PDel_1^{recover} \triangleq PDel_1^{crash} \rightsquigarrow PDel^{total}$. In our system, the crashed process may never recover: hence we have either the behavior of $PDel_1^{recover}$ or the behavior of $PDel_1^{crash}$. This amounts to a union (or a disjunction); we write it $PDel_1^{canrecover} \triangleq PDel_1^{recover} \cup PDel_1^{crash}$. Finally, we consider a potential irremediable crash, additionally to the previous predicate. Thus we want the behavior of $PDel_1^{crash}$ and the behavior of $PDel_1^{canrecover}$. We call it the combination (or conjunction) of these predicates, and write it $PDel_1^{crash} \bigotimes PDel_1^{canrecover}$ The complete system is thus described by $PDel_1^{crash} \bigotimes ((PDel_1^{crash} \rightsquigarrow PDel^{total}) \cup PDel_1^{crash})$. In the following, we will also introduce an operator $\omega$ to express repetition. For example, a system where, repeatedly, a process can crash and recover is $(PDel_1^{crash} \rightsquigarrow PDel^{total})^{\omega}$.

Lastly, the analysis of the resulting delivered predicate can be bypassed: its heard-of predicate arises from our operations applied to the heard-of predicates of the elementary building blocks.

### 1.2 Related Work

The heard-of model was proposed by Charron-Bost and Schiper [4] as a combination of the ideas of two previous work. First, the concept of a fault model where the only information is which message arrives, from Santoro and Widmayer [11]; and second, the idea of abstracting failures in a round per round fashion, from Gafni [8]. Replacing the operational fault detectors of Gafni with the fault model of Santoro and Widmayer gives the heard-of model.

This model was put to use in many ways. Obviously computability and complexity results were proven: new algorithms for consensus in the original paper by Charron-Bost and Schiper [4]; characterizations for consensus solvability by Coulouma et al. [5] and Nowak et al. [10]; a characterization for approximate consensus solvability by Charron-Bost et al. [3]; a study of $k$ set-agreement by Biely et al. [1]; and more.

The clean mathematical abstraction of the heard-of model also works well with formal verification. The rounds provide structure, and the reasoning can be less operational than in many distributed computing abstractions. We thus have a proof assistant verification of consensus algorithms in Charron-Bost et al. [2]; cutoff bounds for the model checking of consensus algorithms by Marić et al. [9]; a DSL to write code following the structure of the heard-of model and verify it with inductive invariants by Drăgoi et al. [6]; and more.

### 1.3 Contributions

The contributions of the paper are:

- A definition of operations on delivered predicates and strategies, as well as examples using them in Section 2.
- The study of oblivious strategies, the strategies only looking at messages for the current round, in Section 3. We provide a technique to extract a strategy dominating the oblivious strategies of the built predicate from the strategies of the initial predicates; exact computations of the generated heard-of predicates; and a sufficient condition on the building blocks for the result of operations to be dominated by an oblivious strategy.
- The study of conservative strategies, the strategies looking at everything but messages from future rounds, in Section 4. We provide a technique to extract a strategy dominating the conservative strategies of the build predicate from the strategies of the initial predicates; upper bounds on the generated heard-of predicates; and a sufficient condition on the building blocks for the result of operations to be dominated by a conservative strategy.

Due to size constraints, many of the complete proofs are not in the paper itself, and can be found in the full paper [13].

## 2 Operations and Examples

### 2.1 Basic concepts

We start by providing basic definitions and intuitions. The model we consider proceed by rounds, where processes send messages tagged with a round number, wait for some messages with this round number, and then compute the next state and increment the round number. $\mathbb{N}^*$ denotes the non-zero naturals.

**Definition 1 (Collections and Predicates).** *Let $\Pi$ a set of processes. An element of $(\mathbb{N}^* \times \Pi) \mapsto \mathcal{P}(\Pi)$ is either a **Delivered collection** $c$ or a **Heard-Of collection** $h$ for $\Pi$, depending on the context. $c_{tot}$ is the total collection such that $\forall r > 0, \forall p \in \Pi : c_{tot}(r, p) = \Pi$.*

*An element of $\mathcal{P}((\mathbb{N}^* \times \Pi) \mapsto \mathcal{P}(\Pi))$ is either a **Delivered predicate** $PDel$ or a **Heard-Of predicate** $PHO$ for $\Pi$. $\mathcal{P}_{tot} = \{c_{tot}\}$ is the total delivered predicate.*

For a heard-of collection $h$, $h(r, p)$ are the senders of messages for round $r$ that $p$ has received at or before its round $r$, and thus has known while at round $r$. For a delivered collection $c$, $c(r, p)$ are the senders of messages for round $r$ that $p$ has received, at any point in time. Some of these messages may have arrived early, before $p$ was at $r$, or too late, after $p$ has left round $r$. $c$ gives an operational point of view (which messages arrive), and $h$ gives a logical point of view (which messages are used).

*Remark 1.* We also regularly use the "graph-sequence" notation for a collection $c$. Let $Graphs_\Pi$ be the set of graphs whose nodes are the elements of $\Pi$. A collection $gr$ is an element of $(Graphs_\Pi)^\omega$. We say that $c$ and $gr$ represent the same collection when $\forall r > 0, \forall p \in \Pi : c(r, p) = In_{gr[r]}(p)$, where $In(p)$ is the incoming vertices of $p$. We will usually not define two collections but use one collection as both kind of objects; the actual type being used in a particular expression can be deduced from the operations on the collection. For example $c[r]$ makes sense for a sequence of graphs, while $c(r, p)$ makes sense for a function.

In an execution, the local state of a process is the pair of its current round and all the received messages up to this point. We disregard any local variable, since our focus is on which messages to wait for. A message is represented by a pair $\langle round, sender \rangle$. For a state $q$, and a round $r > 0$, $q(r)$ is the set of peers from which the process has received a message for round $r$.

**Definition 2 (Local State).** *Let $Q = \mathbb{N}^* \times \mathcal{P}(\mathbb{N}^* \times \Pi)$. Then $q \in Q$ is a **local state**.*

*For $q = \langle r, mes \rangle$, we write $q.round$ for $r$, $q.mes$ for $mes$ and $\forall i > 0 : q(i) \triangleq \{k \in \Pi \mid \langle i, k \rangle \in q.mes\}$.*

We then define strategies, which constrain the behavior of processes. A strategy is a set of states from which a process is allowed to change round. It captures rules like "wait for at least $F$ messages from the current round", or "wait for these specific messages". Strategies give a mean to constrain executions.

**Definition 3 (Strategy).** *$f \in \mathcal{P}(Q)$ is a **strategy**.*

### 2.2 Definition of Operations

We can now define operations on predicates and their corresponding strategies. The intuition behind these operations is the following:

- The union of two delivered predicates is equivalent to an OR on the two communication behaviors. For example, the union of the delivered predicate for one crash at round $r$ and of the one for one crash at round $r+1$ gives a predicate where there is either a crash at round $r$ or a crash at round $r+1$.
- The combination of two behaviors takes every pair of collections, one from each predicate, and computes the intersection of the graphs at each round. Meaning, it adds the loss of messages from both, to get both behaviors at once. For example, combining $PDel_1^{crash}$ with itself gives $PDel_2^{crash}$, the predicate with at most two crashes. Although combination intersects graphs round by round in a local fashion, it actually combines two collections globally, and thus can combine several global predicates like hearing from a given number of process during the whole execution.
- For succession, the system starts with one behavior, then switch to another. The definition is such that the first behavior might never happen, but the second one must appear.
- Repetition is the next logical step after succession: instead of following one behavior with another, the same behavior is repeated again and again. For example, taking the repetition of at most one crash results in a potential infinite number of crash-and-restart, with the constraint of having at most one crashed process at any time.

**Definition 4 (Operations on predicates).** *Let $P_1, P_2$ be two delivered or heard-of predicates.*

- *The **union** of $P_1$ and $P_2$ is $P_1 \cup P_2$.*
- *The **combination** $P_1 \bigotimes P_2 \triangleq \{c_1 \bigotimes c_2 \mid c_1 \in P_1, c_2 \in P_2\}$, where for $c_1$ and $c_2$ two collections, $\forall r > 0, \forall p \in \Pi : (c_1 \bigotimes c_2)(r,p) = c_1(r,p) \cap c_2(r,p)$.*
- *The **succession** $P_1 \leadsto P_2 \triangleq \bigcup_{c_1 \in P_1, c_2 \in P_2} c_1 \leadsto c_2$,*
  *with $c_1 \leadsto c_2 \triangleq \{c \mid \exists r \geq 0 : c = c_1[1,r].c_2\}$.*
- *The **repetition** of $P_1$, $(P_1)^\omega \triangleq \{c \mid \exists (c_i)_{i \in \mathbb{N}^*}, \exists (r_i)_{i \in \mathbb{N}^*} : r_1 = 0 \wedge \forall i \in \mathbb{N}^* : (c_i \in P_1 \wedge r_i < r_{i+1} \wedge c[r_i + 1, r_{i+1}] = c_i[1, r_{i+1} - r_i])\}$.*

For all operations on predicates, we provide an analogous one for strategies. We show later that strategies for the delivered predicates, when combined by the analogous operation, retain important properties on the result of the operation on the predicates.

**Definition 5 (Operations on strategies).** *Let $f_1, f_2$ be two strategies.*

- *Their **union** $f_1 \cup f_2 \triangleq$ the strategy such that $\forall q$ a local state: $(f_1 \cup f_2)(q) \triangleq f_1(q) \vee f_2(q)$.*

- *Their* ***combination*** $f_1 \bigotimes f_2 \triangleq \{q_1 \bigotimes q_2 \mid q_1 \in f_1 \wedge q_2 \in f_2 \wedge q_1.round = q_2.round\}$*, where for* $q_1$ *and* $q_2$ *at the same round* $r$*,* $q_1 \bigotimes q_2 \triangleq \langle r\{\langle r', k\rangle \mid r' > 0 \wedge k \in q_1(r') \cap q_2(r')\}\rangle$
- *Their* ***succession*** $f_1 \rightsquigarrow f_2 \triangleq f_1 \cup f_2 \cup \{q_1 \rightsquigarrow q_2 \mid q_1 \in f_1 \wedge q_2 \in f_2\}$ *where*
$$q_1 \rightsquigarrow q_2 \triangleq \left\langle \begin{array}{l} q_1.round + q_2.round, \\ \left\{ \langle r, k\rangle \mid r > 0 \wedge \begin{pmatrix} k \in q_1(r) & \textit{if } r \leq q_1.round \\ k \in q_2(r - q_1.round) & \textit{if } r > q_1.round \end{pmatrix} \right\} \end{array} \right\rangle$$
- *The* ***repetition*** *of* $f_1$*,* $f_1^{\omega} \triangleq \{q_1 \rightsquigarrow q_2 \rightsquigarrow ... \rightsquigarrow q_k \mid k \geq 1 \wedge q_1, q_2, ..., q_k \in f_1\}$*.*

The goal is to derive new strategies for the resulting model by applying operations on strategies for the starting models. This allows, in some cases, to bypass strategies, and deduce the Heard-Of predicate for a given Delivered predicate from the Heard-Of predicates of its building blocks.

### 2.3 Executions and Domination

Before manipulating predicates and strategies, we need to define what is an execution: a specific ordering of events corresponding to a delivered collection. An execution is an infinite sequence of either delivery of messages ($deliver(r, p, q)$), change to the next round ($next_j$), or a deadlock ($stop$). Message sending is implicit after every change of round. An execution must satisfy three rules: no message is delivered before it is sent, no message is delivered twice, and once there is a $stop$, the rest of the sequence can only be $stop$.

**Definition 6 (Execution).** *Let* $\Pi$ *be a set of* $n$ *processes. Let the set of transitions* $T = \{next_j \mid j \in \Pi\} \cup \{deliver(r, k, j) \mid r \in \mathbb{N}^* \wedge k, j \in \Pi\} \cup \{stop\}$*.* $next_j$ *is the transition for* $j$ *changing round,* $deliver(r, k, j)$ *is the transition for the delivery to* $j$ *of the message sent by* $k$ *in round* $r$*, stop models a deadlock. Then,* $t \in T^{\omega}$ *is an* ***execution*** $\triangleq$

- ***(Delivery after sending)***
  $\forall i \in \mathbb{N} : t[i] = deliver(r, k, j) \implies \mathbf{card}(\{l \in [0, i[ \mid t[l] = next_k\}) \geq r - 1$
- ***(Unique delivery)***
  $\forall \langle r, k, j\rangle \in (\mathbb{N}^* \times \Pi \times \Pi) : \mathbf{card}(\{i \in \mathbb{N} \mid t[i] = deliver(r, k, j)\}) \leq 1$
- ***(Once stopped, forever stopped)***
  $\forall i \in \mathbb{N} : t[i] = stop \implies \forall j \geq i : t[j] = stop$

*Let* $c$ *be a delivered collection. Then,* $execs(c)$*, the* ***executions of*** $c \triangleq$
$$\left\{ t \textit{ an execution} \;\middle|\; \begin{array}{l} \forall \langle r, k, j\rangle \in \mathbb{N}^* \times \Pi \times \Pi : \\ \quad (k \in c(r, j) \wedge \mathbf{card}(\{i \in \mathbb{N} \mid t[i] = next_k\}) \geq r - 1) \\ \quad \Longleftrightarrow \\ \quad (\exists i \in \mathbb{N} : t[i] = deliver(r, k, j)) \end{array} \right\}$$
*For a delivered predicate PDel,* $execs(PDel) \triangleq \{execs(c) \mid c \in PDel\}$*.*

*Let* $t$ *be an execution,* $p \in \Pi$ *and* $i \in \mathbb{N}$*. The state of* $p$ *in* $t$ *after* $i$ *transitions is* $q_p^t[i] \triangleq \langle \mathbf{card}(\{l < i \mid t[l] = next_p\}) + 1, \{\langle r, k\rangle \mid \exists l < i : t[l] = deliver(r, k, p)\}\rangle$

Notice that such executions do not allow process to "jump" from say round 5 to round 9 without passing by the rounds in-between. The reason is that the Heard-Of model does not give processes access to the decision to change rounds: processes specify only which messages to send depending on the state, and what is the next state depending on the current state and the received messages.

Also, the only information considered here is the round number and the received messages. This definition of execution disregards the message contents and the internal states of processes, as they are irrelevant to the implementation of Heard-Of predicates.

Recall that strategies constrain when processes can change round. Thus, the executions that conform to a strategy change rounds only when allowed by it, and do it infinitely often if possible.

**Definition 7 (Executions of a Strategy).** *Let $f$ be a strategy and $t$ an execution. $t$ is an* ***execution of*** *$f \triangleq t$ satisfies:*

- ***(All nexts allowed)*** $\forall i \in \mathbb{N}, \forall p \in \Pi : (t[i] = next_p \implies q^t_p[i] \in f)$
- ***(Fairness)*** $\forall p \in \Pi : \mathbf{card}(\{i \in \mathbb{N} \mid t[i] = next_p\}) < \aleph_0 \implies \mathbf{card}(\{i \in \mathbb{N} \mid q^t_p[i] \notin f\}) = \aleph_0$

*For a delivered predicate PDel, $execs_f(PDel) \triangleq \{t \in execs(PDel) \mid t$ is an execution of $f\}$.*

The fairness property can approximately be expressed in LTL as $\forall p \in \Pi : \Diamond\Box(q^t_p \in f) \Rightarrow \Box\Diamond next_p$. Note however that executions are here defined as sequences of transitions, whereas LTL models are sequences of states.

An important part of this definition considers executions where processes cannot necessarily change round after each delivery. That is, in the case of "waiting for at most $F$ messages", an execution where more messages are delivered than $F$ at some round is still an execution of the strategy. This hypothesis captures the asynchrony of processes, which are not always scheduled right after deliveries. It is compensated by a weak fairness assumption: if a strategy forever allows the change of round, it must eventually happen.

Going back to strategies, not all of them are equally valuable. In general, strategies that block forever at some round are less useful than strategies that don't – they forbid termination in some cases. The validity of a strategy captures the absence of such an infinite wait.

**Definition 8 (Validity).**
*An execution $t$ is* ***valid*** $\triangleq \forall p \in \Pi : \mathbf{card}(\{i \in \mathbb{N} \mid t[i] = next_p\}) = \aleph_0$.

*Let PDel a delivered predicate and $f$ a strategy. $f$ is a* ***valid strategy*** *for PDel $\triangleq \forall t \in execs_f(PDel) : t$ is a valid execution.*

Because in a valid execution no process is ever blocked at a given round, there are infinitely many rounds. Hence, the messages delivered before the changes of round uniquely define a heard-of collection.

**Definition 9 (Heard-Of Collection of Executions and Heard-Of Predicate of Strategies).** *Let $t$ be a valid execution. $h_t$ is the* ***heard-of collection of*** *$t \triangleq$*

$$\forall r \in \mathbb{N}^*, \forall p \in \Pi : h_t(r,p) = \left\{ k \in \Pi \;\middle|\; \exists i \in \mathbb{N} : \begin{pmatrix} q_p^t[i].round = r \\ \land\ t[i] = next_p \\ \land\ \langle r,k \rangle \in q_p^t[i].mes \end{pmatrix} \right\}$$

*Let $PDel$ be a delivered predicate, and $f$ be a valid strategy for $PDel$. We write $PHO_f(PDel)$ for the heard-of predicate composed of the collections of the executions of $f$ on PDel: $PHO_f(PDel) \triangleq \{h_t \mid t \in execs_f(PDel)\}$.*

Lastly, the heard-of predicate of most interest is the strongest one that can be generated by a valid strategy on the delivered predicate. Here strongest means the one that implies all the other heard-of predicates that can be generated on the same delivered predicate. The intuition boils down to two ideas:

- The strongest predicate implies all the heard-of predicates generated on the same $PDel$, and thus it characterizes them completely.
- When seeing predicates as sets, implication is the reverse inclusion. Hence the strongest predicate is the one included in all the others. Less collections means more constrained communication, which means a more powerful model.

This notion of strongest predicate is formalized through an order on strategies and their heard-of predicates.

**Definition 10 (Domination).** *Let PDel be a delivered predicate and let $f$ and $f'$ be two valid strategies for PDel. $f$* ***dominates*** *$f'$ for PDel, written $f' \prec_{PDel} f$, $\triangleq PHO_{f'}(PDel) \supseteq PHO_f(PDel)$.*

*A greatest element for $\prec_{PDel}$ is called a* ***dominating strategy*** *for PDel. Given such a strategy $f$, the* ***dominating predicate*** *for PDel is $PHO_f(PDel)$.*

### 2.4 Examples

We now show the variety of models that can be constructed from basic building blocks. Our basic blocks are the model $PDel^{total}$ with only the collection $c_{total}$ where all the messages are delivered, and the model $PDel^{crash}_{1,r}$ with at most one crash that can happen at round $r$.

**Definition 11 (At most 1 crash at round $r$).** $\mathcal{P}^{crash}_{1,r} \triangleq$

$$\left\{ c \text{ a delivered collection} \;\middle|\; \exists \Sigma \subseteq \Pi : \begin{array}{c} |\Sigma| \geq n-1 \\ \land\ \forall j \in \Pi \begin{pmatrix} \forall r' \in [1,r[ : & c(r',j) = \Pi \\ \land & c(r,j) \supseteq \Sigma \\ \land\ \forall r' \geq r : & c(r',j) = \Sigma \end{pmatrix} \end{array} \right\}.$$

From this family of predicates, various predicates can be built. Table 1 show some of them, as well as the Heard-Of predicates computed for these predicates based on the results from Section 3.3 and Section 3.4. For example the predicate

| Description | Expression | HO | Proof |
|---|---|---|---|
| At most 1 crash | $\mathcal{P}_1^{crash} = \bigcup_{i=1}^{\infty} \mathcal{P}_{1,i}^{crash}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-1\})$ | [12] |
| At most $F$ crashes | $\mathcal{P}_F^{crash} = \bigotimes_{j=1}^{F} \mathcal{P}_1^{crash}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | [12] |
| At most 1 crash, which will restart | $\mathcal{P}_1^{recover} = \mathcal{P}_1^{crash} \rightsquigarrow \mathcal{P}^{total}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-1\})$ | Thm 4 |
| At most $F$ crashes, which will restart | $\mathcal{P}_F^{recover} = \bigotimes_{j=1}^{F} \mathcal{P}_1^{recover}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | Thm 4 |
| At most 1 crash, which can restart | $\mathcal{P}_1^{canrecover}$ $= \mathcal{P}_1^{recover} \cup \mathcal{P}_1^{crash}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-1\})$ | Thm 4 |
| At most $F$ crashes, which can restart | $\mathcal{P}_F^{canrecover}$ $= \bigotimes_{j=1}^{F} \mathcal{P}_1^{canrecover}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | Thm 4 |
| No bound on crashes and restart, with only 1 crash at a time | $\mathcal{P}_1^{recovery} = (\mathcal{P}_1^{crash})^{\omega}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-1\})$ | Thm 4 |
| No bound on crashes and restart, with max $F$ crashes at a time | $\mathcal{P}_F^{recovery} = \bigotimes_{j=1}^{F} \mathcal{P}_1^{recovery}$ | $HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | Thm 4 |
| At most 1 crash, after round $r$ | $\mathcal{P}_{1,\geq r}^{crash} = \bigcup_{i=r}^{\infty} \mathcal{P}_{1,i}^{crash}$ | $\subseteq HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-1\})$ | Thm 10 |
| At most $F$ crashes, after round $r$ | $\mathcal{P}_{F,\geq r}^{crash} = \bigcup_{i=r}^{\infty} \mathcal{P}_{F,i}^{crash}$ | $\subseteq HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | Thm 10 |
| At most $F$ crashes with no more than one per round | $\mathcal{P}_F^{crash\neq}$ $= \bigcup_{i_1 \neq i_2 \dots \neq i_F} \bigotimes_{j=1}^{F} \mathcal{P}_{1,i_j}^{crash}$ | $\subseteq HOProd(\{T \subseteq \Pi \mid \lvert T\rvert \geq n-F\})$ | Thm 10 |

**Table 1.** A list of delivered predicate built using our operations, and their corresponding heard-of predicate. The *HOProduct* operator is defined in Definition 16.

with at most one crash $\mathcal{P}_1^{crash}$ If a crash happens, it happens at one specific round $r$. We can thus build $\mathcal{P}_1^{crash}$ from a disjunction for all values of $r$ of the predicate with at most one crash at round $r$; that is, by the union of $\mathcal{P}_{1,r}^{crash}$ for all $r$.

### 2.5 Families of strategies

Strategies as defined above are predicates on states. This makes them incredibly expressive; on the other hand, this expressivity creates difficulty in reasoning about them. To address this problem, we define families of strategies. Intuitively, strategies in a same family depend on a specific part of the state – for example the messages of the current round. Equality of these parts of the state defines an equivalence relation; the strategies of a family are strategies on the equivalence classes of this relation.

**Definition 12 (Families of strategies).** *Let* $\approx: Q \times Q \rightarrow bool$. *The family of strategies defined by* $\approx$, $family(\approx) \triangleq \{f$ *a strategy* $\mid \forall q_1, q_2 \in \Pi : q_1 \approx q_2 \implies (q_1 \in f \iff q_2 \in f)\}$

## 3 Oblivious Strategies

The simplest non-trivial strategies use only information from the messages of the current round. These strategies that do not remember messages from previous rounds, do not use messages in advance from future rounds, and do not use the round number itself. These strategies are called oblivious. They are simple, the Heard-Of predicates they implement are relatively easy to compute, and they require little computing power and memory to implement. Moreover, many examples above are dominated by such a strategy. Of course, there is a price to pay: oblivious strategies tend to be coarser than general ones.

### 3.1 Minimal Oblivious Strategy

An oblivious strategy is defined by the different subsets of $\Pi$ from which it has to receive a message before allowing a change of round.

**Definition 13 (Oblivious Strategy).** *Let obliv be the function such that* $\forall q \in Q : obliv(q) = \{k \in \Pi \mid \langle q.round, k\rangle \in q.mes\}$*. Let* $\approx_{obliv}$ *the equivalence relation defined by* $q_1 \approx_{obliv} q_2 \triangleq obliv(q_1) = obliv(q_2)$*. The family of oblivious strategies is* $family(\approx_{obliv})$*. For* $f$ *an oblivious strategy, let* $Nexts_f \triangleq \{obliv(q) \mid q \in f\}$*. It uniquely defines* $f$*.*

We will focus on a specific strategy, that dominates the oblivious strategies for a predicate. This follows from the fact that it waits less than any other valid oblivious strategy for this predicate.

**Definition 14 (Minimal Oblivious Strategy).** *Let* $PDel$ *be a delivered predicate. The* ***minimal oblivious strategy*** *for* $PDel$ *is* $f_{min} \triangleq$
$\{q \mid \exists c \in PDel, \exists p \in \Pi, \exists r > 0 : obliv(q) = c(r, p)\}$*.*

**Lemma 1 (Domination of Minimal Oblivious Strategy).** *Let* $PDel$ *be a* $PDel$ *and* $f_{min}$ *be its minimal oblivious strategy. Then* $f_{min}$ *is a dominating oblivious strategy for* $PDel$*.*

*Proof (Proof idea).* $f_{min}$ is valid, because for every possible set of received messages in a collection of $PDel$, it accepts the corresponding oblivious state by definition of minimal oblivious strategy. It is dominating among oblivious strategies because any other valid oblivious strategy must allow the change of round when $f_{min}$ does it: it contains $f_{min}$. If an oblivious strategy does not contain $f_{min}$, then there is a collection of $PDel$ in which at a given round, a certain process might receive exactly the messages for the oblivious state accepted by $f_{min}$ and not by $f$. This entails that $f$ is not valid.

### 3.2 Operations Maintain Minimal Oblivious Strategy

As teased above, minimal oblivious strategies behave nicely under the proposed operations. That is, they give minimal oblivious strategies of resulting delivered predicates. One specificity of minimal oblivious strategies is that there is no need for the succession operation on strategies, nor for the repetition. An oblivious strategy has no knowledge about anything but the messages of the current round, and not even its round number, so it is impossible to distinguish a union from a succession, or a repetition from the initial predicate itself.

**Theorem 1 (Minimal Oblivious Strategy for Union and Succession).** *Let $PDel_1, PDel_2$ be two delivered predicates, $f_1$ and $f_2$ the minimal oblivious strategies for, respectively, $PDel_1$ and $PDel_2$. Then $f_1 \cup f_2$ is the minimal oblivious strategy for $PDel_1 \cup PDel_2$ and $PDel_1 \rightsquigarrow PDel_2$.*

*Proof (Proof idea).* Structurally, all proofs in this section consist in showing equality between the strategies resulting from the operations and the minimal oblivious strategy for the delivered predicate.

For a union, the messages that can be received at each round are the messages that can be received at each round in the first predicate or in the second. This is also true for succession. Given that $f_1$ and $f_2$ are the minimal oblivious strategies of $PDel_1$ and $PDel_2$, they accept exactly the states with one of these sets of current messages. And thus $f_1 \cup f_2$ is the minimal oblivious strategy for $PDel_1 \cup PDel_2$ and $PDel_1 \rightsquigarrow PDel_2$.

**Theorem 2 (Minimal Oblivious Strategy for Repetition).** *Let $PDel$ be a delivered predicate, and $f$ be its minimal oblivious strategy. Then $f$ is the minimal oblivious strategy for $PDel^{\omega}$.*

*Proof (Proof idea).* The intuition is the same as for union and succession. Since repetition involves only one PDel, the sets of received messages do not change and $f$ is the minimal oblivious strategy.

For combination, a special symmetry hypothesis is needed.

**Definition 15 (Totally Symmetric PDel).** *Let $PDel$ be a delivered predicate. $PDel$ is **totally symmetric** $\triangleq \forall c \in PDel, \forall r > 0, \forall p \in \Pi, \forall r' > 0, \forall q \in \Pi, \exists c' \in PDel : c(r,p) = c'(r',q)$*

Combination is different because combining collections is done round by round. As oblivious strategies do not depend on the round, the combination of oblivious strategies creates the same combination of received messages for each round. We thus need these combinations to be independent of the round – to be possible at each round – to reconcile those two elements.

**Theorem 3 (Minimal Oblivious Strategy for Combination).** *Let $PDel_1, PDel_2$ be two totally symmetric delivered predicates, $f_1$ and $f_2$ the minimal oblivious strategies for, respectively, $PDel_1$ and $PDel_2$. Then $f_1 \bigotimes f_2$ is the minimal oblivious strategy for $PDel_1 \bigotimes PDel_2$.*

*Proof (Proof idea).* The oblivious states of $PDel_1 \bigotimes PDel_2$ are the combination of an oblivious state of $PDel_1$ and of one of $PDel_2$ at the same round, for the same process. Thanks to total symmetry, this translates into the intersection of any oblivious state of $PDel_1$ with any oblivious state of $PDel_2$. Since $f_1$ and $f_2$ are the minimal oblivious strategy, they both accept exactly the oblivious states of $PDel_1$ and $PDel_2$ respectively. Thus, $f_1 \bigotimes f_2$ accept all combinations of oblivious states of $PDel_1$ and $PDel_2$, and thus is the minimal oblivious strategy of $PDel_1 \bigotimes PDel_2$.

### 3.3 Computing Heard-Of Predicates

The computation of the heard-of predicate generated by an oblivious strategy is easy thanks to a characteristic of this HO: it is a product of sets of possible messages.

**Definition 16 (Heard-Of Product).** *Let $S \subseteq \mathcal{P}(\Pi)$. The* ***heard-of product generated by S****, $HOProd(S) \triangleq \{h \mid \forall p \in \Pi, \forall r > 0 : h(r,p) \in S\ \}$.*

**Lemma 2 (Heard-Of Predicate of an Oblivious Strategy).** *Let $PDel$ be a delivered predicate containing $c_{tot}$ and let $f$ be a valid oblivious strategy for $PDel$. Then $PHO_f(PDel) = HOProd(Nexts_f)$.*

*Proof.* Proved in [12, Theorem 20, Section 4.1].

Thanks to this characterization, the heard-of predicate generated by the minimal strategies for the operations is computed in terms of the heard-of predicate generated by the original minimal strategies.

**Theorem 4 (Heard-Of Predicate of Minimal Oblivious Strategies).** *Let $PDel, PDel_1, PDel_2$ be delivered predicates containing $c_{tot}$. Let $f, f_1, f_2$ be their respective minimal oblivious strategies. Then:*

- $PHO_{f_1 \cup f_2}(PDel_1 \cup PDel_2) = PHO_{f_1 \cup f_2}(PDel_1 \rightsquigarrow PDel_2)$ $= HOProd(Nexts_{f_1} \cup Nexts_{f_2})$.
- *If $PDel_1$ or $PDel_2$ are totally symmetric, $PHO_{f_1 \bigotimes f_2}(PDel_1 \bigotimes PDel_2) =$ $HOProd(\{n_1 \cap n_2 \mid n_1 \in Nexts_{f_1} \wedge n_2 \in Nexts_{f_2}\})$.*
- $PHO_f(PDel^\omega) = PHO_f(PDel)$.

*Proof (Proof idea).* We apply Lemma 2. The containment of $c_{tot}$ was shown in the proof of Theorem 5. As for the equality of the oblivious states, it follows from the intuition in the proofs of the minimal oblivious strategy in the previous section.

### 3.4 Domination by an Oblivious Strategy

From the previous sections, we can compute the Heard-Of predicate of the dominating oblivious strategies for our examples. We first need to give the minimal oblivious strategy for our building blocks $PDel_1^{crash}$ and $PDel^{total}$.

**Definition 17 (Waiting for $n-F$ messages).** *The strategy to wait for $n-F$ messages is: $f^{n,F} \triangleq \{q \in Q \mid |obliv(q)| \geq n - F\}$*

For all $F < n$, $f^{n,F}$ is the minimal oblivious strategy for $PDel_F^{crash}$ (shown by Shimi et al. [12, Thm. 17]). For $PDel^{total}$, since every process receives all the messages all the time, the strategy waits for all the messages ($f^{n,0}$).

Using these strategies, we deduce the heard-of predicates of dominating oblivious strategies for our examples.

- For $PDel_1^{recover} \triangleq PDel_1^{crash} \rightsquigarrow PDel^{total}$, the minimal oblivious strategy $f_1^{recover} = f^{n,1} \cup f^{n,0} = f^{n,1}$. This entails that $PHO_{f_1^{recover}} = HOProd(\{T \subseteq \Pi \mid |T| \geq n-1\})$.
- For $PDel_1^{canrecover} \triangleq PDel_1^{recover} \cup PDel_1^{crash}$, the minimal oblivious strategy $f_1^{canrecover} = f_1^{recover} \cup f^{n,1} = f^{n,1}$. This entails that $PHO_{f_1^{canrecover}} = HOProd(\{T \subseteq \Pi \mid |T| \geq n-1\})$.
- For $PDel_1^{crash} \bigotimes PDel_1^{canrecover}$ the minimal oblivious strategy $f = f^{n,1} \bigotimes f_1^{canrecover} = f^{n,1} \bigotimes f^{n,1} = f^{n,2}$. This entails that $PHO_f = HOProd(\{T \subseteq \Pi \mid |T| \geq n-2\})$.

The computed predicate is the predicate of the dominating *oblivious* strategy. But the dominating strategy might not be oblivious, and this predicate might be too weak. The following result shows that $PDel_1^{crash}$ and $PDel^{total}$ satisfy conditions that imply their domination by an oblivious strategy. Since these conditions are invariant by our operations, all PDel constructed with these building blocks are dominated by an oblivious strategy.

**Theorem 5 (Domination by Oblivious for Operations).**
*Let $PDel, PDel_1, PDel_2$ be delivered predicates that satisfy:*

- ***(Total collection)*** *They contains the total collection $c_{tot}$,*
- ***(Symmetry up to a round)*** *$\forall c$ a collection in the predicate, $\forall p \in \Pi, \forall r > 0, \forall r' > 0, \exists c'$ a collection in the predicate: $c'[1, r'-1] = c_{tot}[1, r'-1] \wedge \forall q \in \Pi : c'(r', q) = c(r, p)$*

*Then $PDel_1 \cup PDel_2$, $PDel_1 \bigotimes PDel_2$, $PDel_1 \rightsquigarrow PDel_2$, $PDel^\omega$ satisfy the same two conditions and are dominated by oblivious strategies.*

Both $\mathcal{P}_1^{crash}$ from Table 1 and $\mathcal{P}^{total} = \{c_{tot}\}$ satisfy this condition. So do all the first 8 examples from Table 1, since they are built from these two.

## 4 Conservative Strategies

We now broaden our family of considered strategies, by allowing them to consider past and present rounds, as well as the round number itself. This is a generalization of oblivious strategies, that tradeoff simplicity for expressivity, while retaining a nice structure. Even better, we show that both our building blocks and all the predicates built from them are dominated by such a strategy. For the examples then, no expressivity is lost.

### 4.1 Minimal Conservative Strategy

**Definition 18 (Conservative Strategy).** *Let cons be the function such that* $\forall q \in Q,\ cons(q) \triangleq \langle q.round, \{\langle r, k\rangle \in q.mes \mid r \leq q.round\}\rangle$*. Let* $\approx_{cons}$ *the equivalence relation defined by* $q_1 \approx_{cons} q_2 \triangleq cons(q_1) = cons(q_2)$*. The family of conservative strategies is* $family(\approx_{cons})$*. We write* $Nexts_f^R \triangleq \{cons(q) \mid q \in f\}$ *for the set of conservative states in* $f$*. This uniquely defines* $f$*.*

In analogy with the case of oblivious strategies, we can define a minimal conservative strategy of $PDel$, and it is a strategy dominating all conservative strategies for this delivered predicate.

**Definition 19 (Minimal Conservative Strategy).** *Let* $PDel$ *be a delivered predicate. The* ***minimal conservative strategy*** *for* $PDel$ *is* $f_{min} \triangleq$ *the conservative strategy such that* $f = \{q \in Q \mid \exists c \in PDel, \exists p \in \Pi, \forall r \leq q.round : q(r) = c(r, p)\}$*.*

**Lemma 3 (Domination of Minimal Conservative Strategy).** *Let* $PDel$ *be a delivered predicate and* $f_{min}$ *be its minimal conservative strategy. Then* $f_{min}$ *dominates the conservative strategies for* $PDel$*.*

*Proof (Proof idea).* Analogous to the case of minimal oblivious strategies: it is valid because it allows to change round for each possible conservative state (the round and the messages received for this round and before) of collections in $PDel$. And since any other valid conservative strategy $f$ must accept these states (or it would block forever in some execution of a collection of $PDel$), we have that $f$ contains $f_{min}$ and thus that $f_{min}$ dominates $f$.

### 4.2 Operations Maintain Minimal Conservative Strategies

Like oblivious strategies, minimal conservative strategies give minimal conservative strategies of resulting delivered predicates.

**Theorem 6 (Minimal Conservative Strategy for Union).**
*Let* $PDel_1, PDel_2$ *be two delivered predicates,* $f_1$ *and* $f_2$ *the minimal conservative strategies for, respectively,* $PDel_1$ *and* $PDel_2$*. Then* $f_1 \cup f_2$ *is the minimal conservative strategy for* $PDel_1 \cup PDel_2$*.*

*Proof (Proof idea).* A prefix of a collection in $PDel_1 \cup PDel_2$ comes from either $PDel_1$ or $PDel_2$, and thus is accepted by $f_1$ or $f_2$. And any state accepted by $f_1 \cup f_2$ corresponds to some prefix of $PDel_1$ or $PDel_2$.

For the other three operations, slightly more structure is needed on the predicates. More precisely, they have to be independent of the processes. Any prefix of a process $p$ in a collection of the predicate is also the prefix of any other process $q$ in a possibly different collection of the same PDel. Hence, the behaviors (fault, crashes, loss) are not targeting specific processes. This restriction fits the intuition behind many common fault models.

**Definition 20 (Symmetric PDel).** *Let $PDel$ be a delivered predicate. $PDel$ is* ***symmetric*** $\triangleq \forall c \in PDel, \forall p \in \Pi, \forall r > 0, \forall q \in \Pi, \exists c' \in PDel, \forall r' \leq r : c'(r', q) = c(r', p)$

**Theorem 7 (Minimal Conservative Strategy for Combination).** *Let $PDel_1, PDel_2$ be two symmetric delivered predicates, $f_1$ and $f_2$ the minimal conservative strategies for, respectively, $PDel_1$ and $PDel_2$. Then $f_1 \bigotimes f_2$ is the minimal conservative strategy for $PDel_1 \bigotimes PDel_2$.*

*Proof (Proof idea).* Since $f_1$ and $f_2$ are the minimal conservative strategies of $PDel_1$ and $PDel_2$, $Nexts^R f_1$ is the set of the conservative states of prefixes of $PDel_1$ and $Nexts^R_{f_2}$ is the set of the conservative states of prefixes of $PDel_2$. Also, the states accepted by $f_1 \bigotimes f_2$ are the combination of the states accepted by $f_1$ and the states accepted by $f_2$. And the prefixes of $PDel_1 \bigotimes PDel_2$ are the prefixes of $PDel_1$ combined with the prefixes of $PDel_2$ **for the same process**. Thanks to symmetry, we can take a prefix of $PDel_2$ and any process, and find a collection such that the process has that prefix. Therefore the combined prefixes for the same process are the same as the combined prefixes of $PDel_1$ and $PDel_2$. Thus, $Nexts^R_{f_1 \bigotimes f_2}$ is the set of conservative states of prefixes of $PDel_1 \bigotimes PDel_2$, and $f_1 \bigotimes f_2$ is its minimal conservative strategy.

**Theorem 8 (Minimal Conservative Strategy for Succession).** *Let $PDel_1, PDel_2$ be two symmetric delivered predicates, $f_1$ and $f_2$ the minimal conservative strategies for, respectively, $PDel_1$ and $PDel_2$. Then $f_1 \rightsquigarrow f_2$ is the minimal conservative strategy for $PDel_1 \rightsquigarrow PDel_2$.*

*Proof (Proof idea).* Since $f_1$ and $f_2$ are the minimal conservative strategies of $PDel_1$ and $PDel_2$, $Nexts^R f_1$ is the set of the conservative states of prefixes of $PDel_1$ and $Nexts^R_{f_2}$ is the set of the conservative states of prefixes of $PDel_2$. Also, the states accepted by $f_1 \rightsquigarrow f_2$ are the succession of the states accepted by $f_1$ and the states accepted by $f_2$. And the prefixes of $PDel_1 \rightsquigarrow PDel_2$ are the successions of prefixes of $PDel_1$ and prefixes of $PDel_2$ **for the same process**. But thanks to symmetry, we can take a prefix of $PDel_2$ and any process, and find a collection such that the process has that prefix.

Therefore the succession of prefixes for the same process are the same as the succession of prefixes of $PDel_1$ and $PDel_2$. Thus, $Nexts^R_{f_1 \rightsquigarrow f_2}$ is the set of conservative states of prefixes of $PDel_1 \rightsquigarrow PDel_2$, and is therefore its minimal conservative strategy.

**Theorem 9 (Minimal Conservative Strategy for Repetition).** *Let $PDel$ be a symmetric delivered predicate, and $f$ be its minimal conservative strategy. Then $f^\omega$ is the minimal conservative strategy for $PDel^\omega$.*

*Proof (Proof idea).* The idea is the same as in the succession.

### 4.3 Computing Heard-Of Predicates

Here we split from the analogy with oblivious strategies: the heard-of predicate of conservative strategies is hard to compute, as it dependss in intricate ways on the delivered predicate itself.

Yet it is still possible to compute interesting information on this HO: upper bounds. These are overapproximations of the actual HO, but they can serve for formal verification of LTL properties. Indeed, the executions of an algorithm for the actual HO are contained in the executions of the algorithm for any overapproximation of the HO, and LTL properties must be true for all executions of the algorithm. So proving the property on an overapproximation also proves it on the actual HO.

**Theorem 10 (Upper Bounds on HO of Minimal Conservative Strategies).** *Let $PDel, PDel_1, PDel_2$ be delivered predicates containing $c_{tot}$.*
*Let $f^{cons}, f_1^{cons}, f_2^{cons}$ be their respective minimal conservative strategies,*
*and $f^{obliv}, f_1^{obliv}, f_2^{obliv}$ be their respective minimal oblivious strategies. Then:*

- $PHO_{f_1^{cons} \cup f_2^{cons}}(PDel_1 \cup PDel_2) \subseteq HOProd(Nexts_{f_1^{obliv}} \cup Nexts_{f_2^{obliv}})$.
- $PHO_{f_1^{cons} \rightsquigarrow f_2^{cons}}(PDel_1 \rightsquigarrow PDel_2) \subseteq HOProd(Nexts_{f_1^{obliv}} \cup Nexts_{f_2^{obliv}})$.
- $PHO_{f_1^{cons} \bigotimes f_2^{cons}}(PDel_1 \bigotimes PDel_2) \subseteq HOProd(\{n_1 \cap n_2 \mid n_1 \in Nexts_{f_1^{obliv}} \wedge n_2 \in Nexts_{f_2^{obliv}}\})$.
- $PHO_{(f^{cons})^\omega}(PDel^\omega) \subseteq HOProd(Nexts_{f^{obliv}})$.

*Proof (Proof idea).* These bounds follow from the fact that an oblivious strategy, is a conservative strategy, and thus the minimal conservative strategy dominates the minimal oblivious strategy.

## 5 Conclusion

To summarize, we propose operations on delivered predicates that allow the construction of complex predicates from simpler ones. The corresponding operations on strategies behave nicely regarding dominating strategies, for the conservative and oblivious strategies. This entails bounds and characterizations of the dominating heard-of predicate for the constructions.

What needs to be done next comes in two kinds: first, the logical continuation is to look for constraints on delivered predicates for which we can compute the dominating heard-of predicate of conservative strategies. More ambitiously, we will study strategies looking in the future, i.e. strategies that can take into account messages from processes that have already reached a strictly higher round than the recipient. These strategies are useful for inherently asymmetric delivered predicates. For example, message loss is asymmetric, in the sense that we cannot force processes to receive the same set of messages.

*Funding* This work was supported by project PARDI ANR-16-CE25-0006.

## References

1. Martin Biely, Peter Robinson, Manfred Schmid, Ulrich Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018. `doi:10.1016/j.tcs.2018.02.019`.

2. Bernadette Charron-Bost, Henri Debrat, and Stephan Merz. Formal verification of consensus algorithms tolerating malicious faults. In *Stabilization, Safety, and Security of Distributed Systems*, pages 120–134, 2011. `doi:10.1007/978-3-642-24550-3_11`.
3. Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Automata, Languages, and Programming*, pages 528–539, 2015. `doi:10.1007/978-3-662-47666-6_42`.
4. Bernadette Charron-Bost and André Schiper. The heard-of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, April 2009. `doi:10.1007/s00446-009-0084-6`.
5. Étienne Coulouma, Emmanuel Godard, and Joseph Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theoretical Computer Science*, 584:80–90, 2015. `doi:10.1016/j.tcs.2015.01.024`.
6. Cezara Drăgoi, Thomas A. Henzinger, and Damien Zufferey. Psync: A partially synchronous language for fault-tolerant distributed algorithms. *SIGPLAN Not.*, 51(1):400–415, January 2016. `doi:10.1145/2914770.2837650`.
7. Tzilla Elrad and Nissim Francez. Decomposition of distributed programs into communication-closed layers. *Science of Computer Programming*, 2(3):155–173, 1982. `doi:10.1016/0167-6423(83)90013-8`.
8. Eli Gafni. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In *17th ACM Symposium on Principles of Distributed Computing*, PODC '98, pages 143–152, 1998. `doi:10.1145/277697.277724`.
9. Ognjen Marić, Christoph Sprenger, and David Basin. Cutoff bounds for consensus algorithms. In *Computer Aided Verification*, pages 217–237, 2017. `doi:10.1007/978-3-319-63390-9_12`.
10. Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, 2019. `doi:10.1145/3293611.3331624`.
11. Nicola Santoro and Peter Widmayer. Time is not a healer. In *6th Symposium on Theoretical Aspects of Computer Science STACS 89*, pages 304–313, 1989. `doi:10.1007/BFb0028994`.
12. Adam Shimi, Aurélie Hurault, and Philippe Quéinnec. Characterizing asynchronous message-passing models through rounds. In *22nd Int'l Conf. on Principles of Distributed Systems (OPODIS 2018)*, pages 18:1–18:17, 2018. `doi:10.4230/LIPIcs.OPODIS.2018.18`.
13. Adam Shimi, Aurélie Hurault, and Philippe Queinnec. Derivation of heard-of predicates from elementary behavioral patterns, 2020. `arXiv:2004.10619`.