The complexity of verifying population protocols

Javier Esparza[1] · Stefan Jaax[1] · Mikhail Raskin[1] · Chana Weil-Kennedy[1]

Abstract

Population protocols (Angluin et al. in PODC, 2004) are a model of distributed computation in which indistinguishable, finite-state agents interact in pairs to decide if their initial configuration, i.e., the initial number of agents in each state, satisfies a given property. In a seminal paper Angluin et al. classified population protocols according to their communication mechanism, and conducted an exhaustive study of the expressive power of each class, that is, of the properties they can decide (Angluin et al. in Distrib Comput 20(4):279–304, 2007). In this paper we study the correctness problem for population protocols, i.e., whether a given protocol decides a given property. A previous paper (Esparza et al. in Acta Inform 54(2):191–215, 2017) has shown that the problem is decidable for the main population protocol model, but at least as hard as the reachability problem for Petri nets, which has recently been proved to have non-elementary complexity. Motivated by this result, we study the computational complexity of the correctness problem for all other classes introduced by Angluin et al., some of which are less powerful than the main model. Our main results show that for the class of observation models the complexity of the problem is much lower, ranging from Π_2^p to PSPACE.

Keywords Reachability analysis · Parameterized verification · Population protocols · Distributed computing

1 Introduction

Population protocols are a theoretical model for the study of ad hoc networks of tiny computing devices without any infrastructure [5,6]. The model postulates a "soup" of indistinguishable, finite-state agents that behave identically. Agents repeatedly interact in pairs, changing their states according to a joint transition function. A global fairness condition ensures that every global configuration that is reachable infinitely often is also reached infinitely often. The purpose of a population protocol is to allow agents to collectively compute some information about their initial configuration, defined as the function that assigns to each local state the number of agents that initially occupy it. For example, assume that initially each agent picks a boolean value by choosing, say, q_0 or q_1 as its initial state. The (many)

majority protocols described in the literature allow the agents to eventually reach a stable consensus on the value chosen by a majority of the agents. More formally, let x_0 and x_1 denote the initial numbers of agents in states q_0 and q_1; majority protocols compute the predicate $\varphi(x_0, x_1) \colon \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ given by $\varphi(x_0, x_1) = (x_1 \geq x_0)$. Throughout the paper, we use the term "predicate" as an abbreviation for "function from $\mathbb{N}^k$ to $\{0, 1\}$ for some k".

The expressive power of population protocols (that is, which predicates they can compute), and their efficiency (how fast they can compute them) have been both extensively studied (see e.g. [2–4,28]). In a seminal paper [7], Angluin et al. showed that population protocols can compute exactly the predicates definable in Presburger arithmetic. In the same publication, they observed that while the two-way communication discipline of the standard population protocol model is adequate for natural computing applications, where agents represent molecules or cells that communicate by means of physical encounters, it is less so when agents represent electronic devices, where communication usually takes place by asynchronous message-passing, and information flows only from the sender to the receiver. For this reason, they also conducted a thorough investigation of the expressive power of the population protocol model when two-way communica-

✉ Javier Esparza
    esparza@in.tum.de

[1] Technical University of Munich, Munich, Germany

tion is replaced by one-way communication. They classified one-way communication models into *transmission* models, where the sender is allowed to change its state as a result of sending a message, and *observation* models, where it is not. Intuitively, in observation models the receiver observes the state of the sender, who may not even be aware that it is being observed. Further, they distinguished between *immediate delivery* models, where a send event and its corresponding receive event occur simultaneously, *delayed delivery* models, where delivery may take time, but receivers are always willing to receive any message, and *queued delivery* models, where delivery may take time, and receivers may choose to postpone incoming messages until they have sent a message themselves. This results in five one-way models: immediate and delayed observation, immediate and delayed transmission, and queued transmission. Angluin et al. showed that no one-way model is more expressive than the two-way model, and some of them are strictly less expressive. In fact, they characterized the expressive power of each model in terms of natural classes of Presburger predicates.

In this paper we investigate the correctness problem for population protocols, that is, the problem of deciding if a given protocol computes a given Presburger predicate. For each possible input, deciding if the protocol reaches a consensus only requires to inspect one of these finite transition systems, and can be done automatically using a model checker. This approach has been followed in [19,21,45,49], but it only proves the correctness of a protocol for a finite number of (typically small) inputs. The question whether the protocol reaches the right consensus for *all* inputs remained open until 2015, when Esparza et al. showed that the problem is decidable [32]. However, in the same paper they proved that the correctness problem is at least as hard as the reachability problem for Petri nets. This problem, which was known to be EXPSPACE-hard since the 1970s [42], has recently been shown to be TOWER-hard [24], where TOWER is the union of the classes of problems solvable in k-EXPTIME for every $k \geq 0$. Motivated by this high complexity of the two-way model, we examine the complexity of the problem for the one-way models studied in [7]. We show that, very satisfactorily, for observation models the complexity decreases dramatically. In our two main positive results, we prove that correctness is Π_2^p-complete for the delayed observation model, and PSPACE-complete for the immediate observation model, when predicates are specified as quantifier-free formulas of Presburger arithmetic[1]. Surprisingly, we show that this is also the complexity of checking that the protocol is correct for one single given input. So, loosely speaking, in observation models checking correctness for one input and

for all infinitely many possible inputs has the same complexity.

In the second part of the paper we present negative results on the transmission models: In all of them, correctness is at least as hard as the reachability problem for Petri nets, and thus TOWER-hard. Further, for the delayed delivery and queued delivery models the single input case is already TOWER-hard, while for the immediate transmission model the single-input problem is PSPACE-complete. On the positive side, we show that the decidability proof of [32] can be easily extended to the immediate and delayed transmission models, but not to the queued transmission model. In fact, for the queued transmission model we leave the decidability of the correctness problem as an open question. However, we also show that this question is less relevant for queued models than for the others. Indeed, in this model the fairness condition of [7] bears no immediate relation to the probabilistic interpretation of population protocols used in the literature in order to study their efficiency. Table 1 summarizes the results and shows their places in the paper.

The paper is organized as follows. Section 2 recalls the protocol models introduced in [7]. Section 3 presents our lower bounds for observation models. Sections 4, 5 and 6, the most involved part of the paper, prove the results leading to the upper bounds for observation models. Section 7 contains the decidability and TOWER-hardness results for transmission-based models. Section 8 gives a brief overview of the most closely related models and approaches that we are aware of.

Previous versions of some of the results of this paper were published in [33] and [34].

2 Protocol models

After some preliminaries (Sect. 2.1), we recall the definitions of the models introduced by Angluin et al. in [7] (Sects. 2.2 to 2.4), formalize the correctness problem (Sect. 2.5), and rephrase it in two different ways as a reachability problem (Sect. 2.6).

2.1 Multisets and populations

A *multiset* on a finite set E is a mapping $C : E \rightarrow \mathbb{N}$, i.e. $C(e)$ denotes the number of occurrences of an element $e \in E$ in C. Operations on $\mathbb{N}$ are extended to multisets by defining them componentwise on each element of E. We define in this way the sum $C_1 + C_2$, comparison $C_1 \leq C_2$, or maximum $\max\{C_1, C_2\}$ of two multisets C_1, C_2. Subtraction, denoted $C_1 - C_2$, is allowed only if $C_1 \geq C_2$. We let $|C| \overset{\text{def}}{=} \sum_{e \in E} C(e)$ denote the total number of occurrences of elements in C, also called the *size* of C. We sometimes write

---

[1] Since Presburger arithmetic admits quantifier elimination, the quantifier-free fragment is as expressive as the full language, if one adds divisibility predicates with constant divisor.

Table 1 Decidability and complexity results

Communication Model			Single-input corr.	All-inputs corr.
One-way	Observation	Immediate	PSPACE-complete	PSPACE-complete
		Delayed	Π_2^p-complete	Π_2^p-complete
	Transmission	Immediate	PSPACE-complete	TOWER-hard, decidable
		Delayed	TOWER-hard, decidable	TOWER-hard, decidable
		Queued	TOWER-hard	TOWER-hard
Two-way	Transmission	Immediate	PSPACE-complete [32]	TOWER-hard, decidable [32]

multisets using set-like notation. For example, both $\langle a, 2 \cdot b \rangle$ and $\langle a, b, b \rangle$ denote the multiset C such that $C(a) = 1$, $C(b) = 2$ and $C(e) = 0$ for every $e \in E \setminus \{a, b\}$. Sometimes we use yet another representation, by assuming a total order $e_1 \prec e_2 \prec \cdots \prec e_n$ on E, and representing a multiset C by the vector $(C(e_1), \ldots, C(e_n)) \in \mathbb{N}^n$.

A *population* P is a multiset on a finite set E with at least two elements, i.e. $P(E) \geq 2$. The set of all populations on E is denoted $\text{Pop}(E)$.

2.2 A unified model

We recall the unified framework for protocols introduced by Angluin et al. in [7], which allows us to give a generic definition of the predicate computed by a protocol.

Definition 2.1 A *generalized protocol* is a quintuple $\mathscr{P} = (\text{Conf}, \Sigma, \text{Step}, I, O)$ where

- *Conf* is a countable set of *configurations*.
- Σ is a finite *alphabet* of *input symbols*. The elements of $\text{Pop}(\Sigma)$ are called *inputs*.
- *Step* $\subseteq \text{Conf} \times \text{Conf}$ is a reflexive *step relation*, capturing when a first configuration can reach another one in one step.
- $I : \text{Pop}(\Sigma) \to \text{Conf}$ is an *input function* that assigns to every input an initial configuration.
- $O : \text{Conf} \to \{0, 1\}$ is a partial *output function* that assigns an output to each configuration on which it is defined.

We write $C \to C'$ and $C \xrightarrow{*} C'$ to denote $(C, C') \in \text{Step}$ and $(C, C') \in \text{Step}^*$, respectively. We say C' is *reachable* from C if $C \xrightarrow{*} C'$. An *execution* of $\mathscr{P}$ is a (finite or infinite) sequence of configurations $C_0, C_1, \ldots$ such that $C_j \to C_{j+1}$ for every $j \geq 0$. Observe that, since we assume that the step relation is reflexive, all maximal executions (i.e., all executions that cannot be extended) are infinite.

An execution $C_0, C_1, \ldots$ is *fair* if for every $C \in \text{Conf}$ the following property holds: If there exist infinitely many indices $i \geq 0$ such that $C_i \xrightarrow{*} C$, then there exist infinitely many indices $j \geq 0$ such that $C_j = C$. In words, in

fair sequences every configuration which can be reached infinitely often is reached infinitely often.

A fair execution $C_0, C_1, \ldots$ *converges to* $b \in \{0, 1\}$ if there exists an index $m \geq 0$ such that for all $j \geq m$ the output function is defined on C_j and $O(C_j) = b$. A protocol outputs $b \in \{0, 1\}$ for input $a \in \text{Pop}(\Sigma)$ if every fair execution starting at $I(a)$ converges to b. A protocol *computes* a predicate $\varphi : \text{Pop}(\Sigma) \to \{0, 1\}$ if it outputs $\varphi(a)$ for every input $a \in \text{Pop}(\Sigma)$.

The *correctness problem* for a class of protocols consists of deciding for a given protocol $\mathscr{P}$ in the class, and a given predicate $\varphi : \text{Pop}(\Sigma) \to \{0, 1\}$, where Σ is the alphabet of $\mathscr{P}$, whether $\mathscr{P}$ computes φ. The goal of this paper is to determine the decidability and complexity of the correctness problem for the classes of protocols introduced by Angluin et al. in [7].

In the rest of the section we formally define the six protocol classes studied by Angluin et al., and summarize the results of [7] that characterize the predicates they can compute. Angluin et al. distinguish between models in which agents interact directly with each other, with zero-delay, and models in which agents interact through messages with possibly non-zero transit time. We describe them in Sects. 2.3 and 2.4, respectively.

2.3 Immediate delivery models

In immediate interaction models, a configuration only needs to specify the current state of each agent. In delayed models, the configuration must also specify which messages are in transit. Angluin et al. study three immediate delivery models.

Standard Population Protocols (PP). Population protocols describe the evolution of a population of finite-state agents. Agents are indistinguishable, and interaction is two-way. When two agents meet, they exchange full information about their current states, and update their states in reaction to this information.

Definition 2.2 A *standard population protocol* is a quintuple $\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$ where Q is a finite set of states, $\delta : Q^2 \to Q^2$ is the transition function, Σ is a finite set of

input symbols, $\iota\colon \Sigma \to Q$ is the initial state mapping, and $o\colon Q \to \{0, 1\}$ is the state output mapping.

Observe that δ is a total function, and so we assume that every pair of agents can interact, although the result of the interaction can be that the agents do not change their states. Every standard population protocol determines a protocol in the sense of Definition 2.1 as follows, where $C, C' \in \mathrm{Pop}(Q)$, $D \in \mathrm{Pop}(\Sigma)$, and $b \in \{0, 1\}$:

- the configurations are the populations over Q, that is, $Conf = \mathrm{Pop}(Q)$;
- $(C, C') \in Step$ if there exist states $q_1, q_2, q_3, q_4 \in Q$ such that $\delta(q_1, q_2) = (q_3, q_4)$, $C \geq \langle q_1, q_2 \rangle$, and $C' = C - \langle q_1, q_2 \rangle + \langle q_3, q_4 \rangle$. The inequality cannot be omitted because some of the states can coincide.
- $I(D) = \sum_{\sigma \in \Sigma} D(\sigma)\iota(\sigma)$; in other words, if the input D contains k copies of $\sigma \in \Sigma$, then the configuration $I(D)$ places k agents in the state $\iota(\sigma)$;
- $O(C) = b$ if $o(q) = b$ for all $q \in Q$ such that $C(q) > 0$; in other words, $O(C) = b$ if in the configuration C all agents are in states with output b. We often call a configuration C satisfying this property a *b-consensus*.

The two other models with immediate delivery are one-way. They are defined as subclasses of the standard population protocol model.

Immediate Transmission Protocols (IT). In these protocols, at each step an agent (the sender) sends its state to another agent (the receiver). Communication is immediate, that is, sending and receiving happen in one atomic step. The new state of the receiver depends on both its old state and the old state of the sender, but the new state of the sender depends only on its own old state, and not on the old state of the receiver. Formally:

Definition 2.3 A standard population protocol $\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$ is an *immediate transmission protocol* if there exist two functions $\delta_1\colon Q \to Q$, $\delta_2\colon Q^2 \to Q$ satisfying $\delta(q_1, q_2) = (\delta_1(q_1), \delta_2(q_1, q_2))$ for every $q_1, q_2 \in Q$.

Immediate Observation Protocol (IO). In these protocols, the state of a first agent can be observed by a second agent, which updates its state using this information. Unlike in the immediate transmission model, the first agent does not update its state (intuitively, it may not even know that it has been observed). Formally:

Definition 2.4 A standard population protocol $\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$ is an *immediate observation protocol* if there exists a function $\delta_2\colon Q^2 \to Q$ satisfying $\delta(q_1, q_2) = (q_1, \delta_2(q_1, q_2))$ for every $q_1, q_2 \in Q$.

Notation. We sometimes write $q_1, q_2 \to q_3, q_4$ for $\delta(q_1, q_2) = (q_3, q_4)$. In the case of IO protocols we sometimes write $q_2 \xrightarrow{q_1} q_4$ for $\delta(q_1, q_2) = (q_1, q_4)$, and say that the agent moves from q_2 to q_4 by observing q_1.

2.4 Delayed delivery models

In delayed delivery models agents communicate by sending and receiving messages. The set of messages that can be sent (and received) is finite. Messages are sent to and received from one single pool of messages; in particular, the sender does not choose the recipient of the message. The pool can contain an unbounded number of copies of a message. Agents update their state after sending or receiving a message. Angluin et al. define the following three delayed delivery models.

Queued Transmission Protocols (QT). The set of messages an agent is willing to receive depends on its current state. In particular, in some states the agent may not be willing to receive any message.

Definition 2.5 A *queued transmission protocol* is a septuple $\mathscr{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$ where Q is a finite set of states, M is a finite set of messages, $\delta_s\colon Q \to M \times Q$ is the partial send function, $\delta_r\colon Q \times M \rightharpoonup Q$ is the partial receive function, Σ is a finite set of input symbols, $\iota\colon \Sigma \to Q$ is the initial state mapping, and $o\colon Q \to \{0, 1\}$ is the state output mapping.

Every queued transmission protocol determines a protocol in the sense of Definition 2.1 as follows, where $C, C' \in \mathrm{Pop}(Q)$, $D \in \mathrm{Pop}(\Sigma)$, and $b \in \{0, 1\}$:

- the configurations are the populations over $Q \cup M$, that is, $Conf = \mathrm{Pop}(Q \cup M)$;
- $(C, C') \in Step$ if there exist states q_1, q_2 and a message m such that
  - $\delta_s(q_1) = (m, q_2)$, $C \geq \langle q_1 \rangle$, and $C' = C - \langle q_1 \rangle + \langle m, q_2 \rangle$; or
  - $\delta_r(q_1, m) = q_2$, $C \geq \langle q_1, m \rangle$, and $C = C' - \langle q_1, m \rangle + \langle q_2 \rangle$.
- $I(D) = \sum_{\sigma \in \Sigma} C(\sigma)\iota(\sigma)$; notice that since ι does not map symbols of Σ to M, the configuration $I(D)$ has no messages;
- $O(C) = b$ if $o(q) = b$ for all $q \in Q$ such that $C(q) > 0$.

Delayed Transmission Protocols (DT). DT protocols are the subclass of QT protocols in which, loosely speaking, agents can never refuse to receive a message. This is modeled by requiring the receive transition function to be total.

Definition 2.6 A queued transmission protocol $\mathscr{P}$ is a *delayed transmission protocol* if its receive function δ_r is a total function.

Delayed Observation Protocols (DO). Intuitively, DO protocols are the subclass of DT-protocols in which "sender" and "receiver" actually means "observee" and "observer". This is modeled by forbidding the sender to change its state when it sends a message (since the observee many not even know it is being observed).

Definition 2.7 Let $\mathscr{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$ be a queued transmission protocol. $\mathscr{P}$ is a *delayed observation protocol* if δ_r is a total function and for every $q \in Q$ the send funtion δ_s satisfies $\delta_s(q) = (m, q)$ for some $m \in M$.

Notation. We write $q_1 \xrightarrow{m+} q_2$ when $\delta_s(q_1) = (m, q_2)$, and $q_1 \xrightarrow{m-} q_2$ when $\delta_r(q_1, m) = q_2$, denoting that the message m is added to or removed from the pool of messages. In the case of DO protocols, we sometimes write simply $q_1 \xrightarrow{m+}$.

The following fact follows immediately from the definitions, but is very important.

Fact. In immediate delivery protocols (PP, IT, IO), if $C \xrightarrow{*} C'$ then $|C| = |C'|$. Indeed, in these models configurations are elements of $\mathrm{Pop}(Q)$, and so the size of a configuration is the total number of agents, which does not change when transitions occur. In particular, for every configuration C the number of configurations reachable from C is finite.

In delayed delivery protocols (QT, DT, DO), configurations are elements of $\mathrm{Pop}(Q \cup M)$, and so the size of a configurations is the number of agents *plus* the number of messages sent but not yet received. Since transitions can increase or decrease the number of messages, the number of configurations reachable from a given configuration can be infinite.

Table 2 summarizes the different transition functions and restrictions of the models.

2.5 Expressive power and correctness problem

Let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a finite alphabet. We introduce the class of predicates $\varphi \colon \mathrm{Pop}(\Sigma) \to \{0, 1\}$ definable in Presburger arithmetic, the first-order theory of addition.

A population $P \in \mathrm{Pop}(\Sigma)$ is completely characterized by the number k_i of occurrences of each input symbol σ_i in P, and so we can identify P with the vector $(k_1, \ldots, k_n)$. A predicate $\varphi \colon \mathrm{Pop}(\Sigma) \to \{0, 1\}$ is a *threshold* predicate if there are coefficients $a_1, \ldots, a_n, b \in \mathbb{Z}$ such that $\varphi(k_1, \ldots, k_n) = 1$ iff $\sum_{i=1}^n a_i \cdot k_i < b$. The class of Presburger predicates is the closure of the threshold predicates under boolean operations and existential quantification. By the well-known result that Presburger arithmetic has a quantifier elimination procedure, a predicate is Presburger iff it is a boolean combination of threshold and *modulo* predicates, defined as the predicates of the form $\sum_{i=1}^n a_i \cdot k_i \equiv_c b$ (see e.g. [22]). Abusing language, we call a boolean combination of threshold and modulo terms a *quantifier-free Presburger predicate*.

In [7], Angluin et al. characterize the predicates computable by the six models of protocols we have introduced. Remarkably, all the classes compute only Presburger predicates. More precisely:

- DO computes the boolean combinations of predicates of the form $x \geq 1$, where x is an input symbol. This is the class of predicates that depend only on the presence or absence of each input symbol.
- IO computes the boolean combinations of predicates of the form $x \geq c$, where x is an input symbol and $c \in \mathbb{N}$.
- IT and DT compute the Presburger predicates that are similar to a boolean combination of modulo predicates for sufficiently large inputs; for the exact definition of similarity we refer the reader to [7].
- PP and QT compute exactly the Presburger predicates.

The results of [7] are important in order to define the correctness problem. The inputs to the problem are a protocol

Table 2 Transition functions and restrictions of the five models

Standard Population Protocol (PP)	$\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$	$\delta \colon Q^2 \to Q^2$
Immediate Transmission (IT)	$\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$	$\exists \delta_1, \delta_2$ such that $\forall q_1, q_2 \in Q, \delta(q_1, q_2) = (\delta_1(q_1), \delta_2(q_1, q_2))$
	New sender state does not depend on receiver state	
Immediate Observation (IO)	$\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$	$\exists \delta_2$ such that $\forall q_1, q_2 \in Q, \delta(q_1, q_2) = (q_1, \delta_2(q_1, q_2))$
	Sender state does not change	
Queued Transmission (QT)	$\mathscr{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$	$\delta_s \colon Q \to M \times Q, \delta_r \colon Q \times M \rightharpoonup Q$
	An agent can send a message, or receive one. The set of messages it can receive depends on its state.	
Delayed Transmission (DT)	$\mathscr{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$	$\delta_s \colon Q \to M \times Q, \delta_r \colon Q \times M \to Q$
	Each agent can receive every message	
Delayed Observation (DO)	$\mathscr{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$	$\delta_s \colon Q \to M \times Q, \delta_r \colon Q \times M \to Q \; \forall q \in Q, \delta_s(q) = (m, q)$ for some $m \in M$
	Each agent can receive every message and the sender state does not change	

and a predicate. The protocol is represented by giving its sets of places, transitions, etc. However, we still need a finite representation for Presburger predicates. There are three possible candidates: full Presburger arithmetic, quantifier-free Presburger arithmetic, and semilinear sets. Semilinear sets are difficult to parse by humans, and no paper on population protocols uses them to describe predicates. Full Presburger arithmetic is very succinct, but its satisfiability problem lies between 2-NEXP and 2-EXPSPACE [10,35,37]. Since the satisfiability problem can be easily reduced to the correctness problem, choosing full Presburger arithmetic would "mask" the complexity of the correctness problem in the size of the protocol for several protocol classes. This leaves quantifier-free Presburger arithmetic, which also has several advantages of its own. First, standard predicates studied in the literature (like majority, threshold, or remainder predicates) are naturally expressed without quantifiers. Second, there is a synthesis algorithm for population protocols that takes a quantifier-free Presburger predicate as input and outputs a population protocol (not necessarily efficient or succinct) that computes it [5,6]; a recent, more involved algorithm even outputs a protocol with polynomially many states in the size of the predicate [13]. Third, the satisfiability problem for quantifier-free Presburger predicates is "only" NP-complete, and, as we shall see, the complexity in the size of the protocol will always be higher for all protocol classes.

Taking these considerations into account, we formally define the correctness problem as follows:

Correctness problem

Given: A protocol $\mathscr{P}$ over an alphabet Σ, belonging to one of the six classes PP, DO, IO, DT, IT, QT; a quantifier-free Presburger predicate φ over Σ.
Decide: Does $\mathscr{P}$ compute the predicate represented by φ?

We also study the correctness problem over a single input. We refer to it as the single-instance correctness problem and define it in the following way:

Single-instance correctness problem

Given: A protocol $\mathscr{P}$ over an alphabet Σ and with initial state mapping ι, belonging to one of the six classes PP, DO, IO, DT, IT, QT; an input $D \in Pop(\Sigma)$, and a boolean b.
Decide: Do all fair executions of $\mathscr{P}$ starting at $I(D)$ converge to b?

2.6 Correctness as a reachability problem

In the coming sections we will obtain matching upper and lower bounds on the complexity of the correctness problem for different protocol classes. The upper bounds are obtained by reducing the correctness problem to two different reach-

ability problems. The reductions require the protocols to be *well behaved*. We first define well-behaved protocols, and then present the two reductions.

Well-behaved protocols. Let $\mathscr{P}$ be a generalized protocol. A configuration C of $\mathscr{P}$ is a *bottom configuration* if $C \xrightarrow{*} C'$ implies $C' \xrightarrow{*} C$ for every configuration C'. In other words, C is a bottom configuration if it belongs to a bottom strongly connected component (SCC) of the configuration graph of the protocol.

Definition 2.8 A generalized protocol is *well-behaved* if every fair execution contains a bottom configuration.

We show that all our protocols are well behaved, with the exception of queued-transmission protocols. Essentially, this is the reason why the decidability of the correctness problem for QT is still open.

Lemma 2.9 *Standard population protocols (PP) and delayed-transmission protocols (DT) are well behaved.*

Proof In standard population protocols, if $C \xrightarrow{*} C'$ then $|C| = |C'|$. It follows that for every configuration $C \in Conf$ the set of configurations reachable from C is finite. So every fair execution eventually visits a bottom configuration.

In delayed-transmision protocols, the size of a configuration is equal to the number of agents plus the number of messages in transit. So there is no bound on the size of the configurations reachable from a given configuration C, and in particular the set of configurations reachable from C can be infinite. However, since agents can always receive any message, for every configuration C there is at least one reachable configuration Z without any message in transit. Since the number of such configurations with a given number of agents is finite, for every fair execution $\pi = C_0, C_1, \ldots$ there is a configuration Z without messages in transit such that $C_i = Z$ for infinitely many i. By fairness, every configuration C' reachable from Z also appears infinitely often in π, and so every configuration C' reachable from Z verifies $C' \xrightarrow{*} Z$. So Z is a bottom configuration.   □

Since IT and IO are subclasses of PP and DO is a subclass of DT, the proof is valid for IT, IO, and DT as well. The following example shows that queued-transmission protocols are not necessarily well-behaved.

Example 2.10 Consider a queued-transmission protocol in which an agent in state q can send a message m, staying in q. Assume further that no agent can ever receive a message m (because, for example, there are no receiving transitions for it). Then any execution in which the agent in state q sends the message m infinitely often and never receives any messages is fair: Indeed, after k steps the system can only reach

configurations with at least k messages, and so no configuration is reachable from infinitely many configurations in the execution. Since this fair execution does not visit any bottom configuration, the protocol is not well-behaved. Moreover, if q is the only state of the protocol, there are no bottom configurations at all.

Characterizing correctness of well-behaved protocols. We start with a useful lemma valid for arbitrary protocols.

Lemma 2.11 ([7]) *Every finite execution of a generalized protocol can be extended to a fair execution.*

Proof Let *Conf* be the set of configurations of the protocol, and let π be a finite execution. Fix an infinite sequence $\rho = C_0, C_1, \ldots$ of configurations such that every configuration of *Conf* appears infinitely often in ρ. Define the infinite execution $\pi_0 \pi_1 \pi_2 \ldots$ and the infinite subsequence $C_{i_0}, C_{i_1}, C_{i_2} \ldots$ of ρ inductively as follows. For $i = 0$, let $\pi_0 := \pi$ and $C_{i_0} := C_0$. For every $j \geq 0$, let $\pi_0 \ldots \pi_j \pi_{j+1}$ be any execution leading to the first configuration of ρ after C_{i_j} that is reachable from the last configuration of $\pi_0 \ldots \pi_j$. It is easy to see that $\pi_0 \pi_1 \pi_2 \ldots$ is fair. □

Now we introduce some notations. Let $\mathscr{P} = (Conf, \Sigma, Step, I, O)$ be a generalized protocol, and let φ be a predicate.

– The sets of predecessors and successors of a set $\mathscr{M}$ of configurations of $\mathscr{P}$ are defined as follows:

$$pre^*(\mathscr{M}) \overset{\text{def}}{=} \{C' \in Conf \mid \exists C \in \mathscr{M} . C' \overset{*}{\to} C\}$$
$$post^*(\mathscr{M}) \overset{\text{def}}{=} \{C \in Conf \mid \exists C' \in \mathscr{M} . C' \overset{*}{\to} C\}$$

– For every $b \in \{0, 1\}$, we define $Con_b \overset{\text{def}}{=} O^{-1}(b)$, the set of configurations with output b. We call Con_b the set of b-consensus configurations.
– For every $b \in \{0, 1\}$, we let St_b denote the set of configurations C such that every configuration reachable from C (including C itself) has output b. *St* stands for *stable output*. It follows easily from the definitions of pre^* and $post^*$ that

$$St_b = \overline{pre^* \left(\overline{Con_b} \right)},$$

where $\overline{\mathscr{M}} \overset{\text{def}}{=} Conf \setminus \mathscr{M}$ for every set of configurations $\mathscr{M} \subseteq Conf$. Indeed, the equation states that a configuration belongs to St_b iff it cannot reach any configuration with output $1 - b$, or with no output.
– For every $b \in \{0, 1\}$, we define $I_b \overset{\text{def}}{=} \{I(D) \mid D \in Pop(\Sigma) \land \varphi(D) = b\}$. In other words, I_b is the set of initial configurations for which $\mathscr{P}$ should output b in order to compute φ.

Proposition 2.12 *Let $\mathscr{P} = (Conf, \Sigma, Step, I, O)$ be a well-behaved generalized protocol and let φ be a predicate. $\mathscr{P}$ computes φ iff*

$$post^*(I_b) \subseteq pre^*(St_b)$$

holds for every $b \in \{0, 1\}$.

Proof Assume that $post^*(I_b) \subseteq pre^*(St_b)$ holds for $b \in \{0, 1\}$. Let $\pi = C_0, C_1, \ldots$ be a fair execution with $C_0 \in I_b$ for some $b \in \{0, 1\}$. We show that π converges to b. Protocol $\mathscr{P}$ is well-behaved, so π contains a bottom configuration C of a bottom SCC $B \subseteq \mathscr{B}$. By assumption, we know that St_b is reachable from C, so there exists $C' \in St_b$ such that $C \overset{*}{\to} C'$. This entails $C' \in B$. Since for all $D \in St_b$, if $D \overset{*}{\to} D'$ then $D' \in St_b$, we obtain that $B \subseteq St_b$. Every configuration of St_b is a b-consensus so π converges to b.

Assume that $\mathscr{P}$ computes φ, i.e. that every fair execution starting in I_b converges to b for $b \in \{0, 1\}$. Let us show that $post^*(I_b) \subseteq pre^*(St_b)$ holds. Consider $C \in post^*(I_b)$. There exists $C_0 \in I_b$ such that $C_0 \overset{*}{\to} C$ and, by Lemma 2.11, this finite execution can be extended to a fair infinite execution π. Since $\mathscr{P}$ is well-behaved, the execution contains a bottom configuration C' of a bottom SCC $B \subseteq \mathscr{B}$. If $B \subseteq St_b$ then $C \in pre^*(St_b)$ and our proof is done. Suppose this is not the case, i.e. $B \cap \overline{St_b} \neq \emptyset$. This means that there is a configuration $\hat{C} \notin Con_b$ that is in B. It is thus reachable from any configuration of π and so by fairness it is reached infinitely often. Thus π does not converge to b, contradicting the correctness assumption. □

A second characterization. Proposition 2.12 is useful when it is possible to compute adequate finite representations of the sets $post^*(I_b)$ and $pre^*(St_b)$. We will later see that this is the case for IO and DO protocols. Unfortunately, such finite representations have not yet been found for PP or for transmission protocols. For this reason, our results for these classes will be based on a second characterization.

Let $\mathscr{P} = (Conf, \Sigma, Step, I, O)$ be a well-behaved generalized protocol, and let $\mathscr{B}$ denote the set of bottom configurations of $\mathscr{P}$. Further, for every $b \in \{0, 1\}$, let $\mathscr{B}_b$ denote the set of configurations $C \in \mathscr{B}$ such that every configuration C' reachable from C satisfies $O(C') = b$. Equivalently, $\mathscr{B}_b \overset{\text{def}}{=} \mathscr{B} \cap St_b$.

Observe that every fair execution of a well-behaved protocol eventually gets trapped in a bottom strongly-connected component of the configuration graph and, by fairness, visits all its configurations infinitely often. Further, if any configuration of the SCC belongs to $\mathscr{B}_b$, then all of them belong to $\mathscr{B}_b$. This occurs independently of whether the SCC contains finitely or infinitely many configurations.

Proposition 2.13 *Let $\mathscr{P}$ be a well-behaved generalized protocol and let φ be a predicate. $\mathscr{P}$ computes φ iff for every $b \in \{0, 1\}$ the set $\mathscr{B} \setminus \mathscr{B}_b$ is not reachable from I_b.*

Proof Assume that $\mathscr{B} \setminus \mathscr{B}_b$ is reachable from $\varphi^{-1}(b)$ for some $b \in \{0, 1\}$. Then there exists an input $a \in \text{Pop}(\Sigma)$ and an execution $C_0, C_1, \dots, C_i$ such that $\varphi(a) = b$, $I(a) = C_0$, and $C_i \in \mathscr{B} \setminus \mathscr{B}_b$. By Lemma 2.11 the execution can be extended to a fair execution $C_0, C_1, \dots$. Since $C_{i+k} \xrightarrow{*} C_i$ for every $k \geq 0$, the execution visits C_i and all its successors infinitely often. Since $C_i \notin \mathscr{B}_b$, the execution does not converge to b. So $\mathscr{P}$ does not compute φ.

Assume that $\mathscr{P}$ does not compute φ. Then there exists an input $a \in \text{Pop}(\Sigma)$, a boolean $b \in \{0, 1\}$, and a fair execution $\pi = C_0, C_1, \dots$ such that $\varphi(a) = b$ and $I(a) = C_0$, but π does not converge to b. Since $\mathscr{P}$ is well-behaved, π contains a configuration $C_i \in \mathscr{B}$. Since π does not converge to b, there is $j > i$ such that $O(C_j)$ is undefined, or defined but different from b. Since C_j belongs to the same SCC as C_i, we have $C_i \notin \mathscr{B}_b$. □

3 Lower bounds for observation models

We prove that the correctness problem is PSPACE-hard for IO protocols and Π_2^p-hard for DO protocols, and that these results also hold for the single-instance problem.

3.1 Correctness of IO protocols is PSPACE-**hard**

We prove that the single-instance correctness and correctness problems for IO protocols are PSPACE-hard by reduction from the acceptance problem for bounded-tape Turing machines. We show that the standard simulation of bounded-tape Turing machines by 1-safe Petri nets, as described for example in [20,29], can be modified to produce an IO protocol. This can be done for IO protocols but not for DO protocols: the simulation of the Turing machine relies on the fact that a transition will only occur in an IO protocol if an agent observes another agent in a certain state *at the present moment*.

We fix a deterministic Turing machine M with set of control states Q, alphabet Σ containing the empty symbol $\sqcup$, and partial transition function $\delta \colon Q \times \Sigma \to Q \times \Sigma \times D$ ($D = \{-1, +1\}$). Let K denote an upper bound on the number of tape cells visited by the computation of M on empty tape. We assume that K is provided with M in unary encoding.

The *implementation* of M is the IO protocol $\mathscr{P}_M$ described below. Strictly speaking, $\mathscr{P}_M$ is not a complete protocol, only two sets of states and transitions. The rest of the protocol, which is slightly different for the single-instance correctness and the correctness problems, is described in the proofs.

States of $\mathscr{P}_M$. The protocol $\mathscr{P}_M$ contains two sets of *cell states* and *head states* modeling the state of the tape cells and the head, respectively. The cell states are:

– $off[\sigma, n]$ for each $\sigma \in \Sigma$ and $1 \leq n \leq K$. An agent in $off[\sigma, n]$ denotes that cell n contains symbol σ, and the cell is "off", i.e., the head is not on it.
– $on[\sigma, n]$ for each $\sigma \in \Sigma$ and $1 \leq n \leq K$, with analogous intended meaning.

The head states are:

– $at[q, n]$ for each $q \in Q$ and $1 \leq n \leq K$. An agent in $at[q, n]$ denotes that the head is in control state q and at cell n.
– $move[q, \sigma, n, d]$ for each $q \in Q$, $\sigma \in \Sigma$, $1 \leq n \leq K$ and every $d \in D$ such that $1 \leq n + d \leq K$. An agent in $move[q, \sigma, n, d]$ denotes that head is in control state q, has left cell n after writing symbol σ on it, and is currently moving in the direction given by d.

Finally, the protocol also contains two special states *observer* and *success*. Intuitively, $\mathscr{P}_M$ uses them to detect that M has accepted.

Transitions of $\mathscr{P}_M$. Intuitively, the implementation of M contains a set of *cell transitions* in which a cell observes the head and changes its state, a set of *head transitions* in which the head observes a cell. Each of these sets contains transitions of two types. The set of cell transitions contains:

– Type **1a**: A transition $off[\sigma, n] \xrightarrow{at[q,n]} on[\sigma, n]$ for every state $q \in Q$, symbol $\sigma \in \Sigma$, and cell $1 \leq n \leq K$.
  The n-th cell, currently *off*, observes that the head is on it, and switches itself *on*.
– Type **1b**: A transition $on[\sigma, n] \xrightarrow{move[q,\sigma',n,d]} off[\sigma', n]$ for every $q \in Q$, $\sigma \in \Sigma$, and $1 \leq n \leq K$ such that $1 \leq n + d \leq K$.
  The n-th cell, currently *on*, observes that the head has left after writing σ', and switches itself *off* (accepting the character the head intended to write).

The set of head transitions contains:

– Type **2a**: A transition

$$at[q, n] \xrightarrow{on[\sigma,n]} move[\delta_Q(q, \sigma), \delta_\Sigma(q, \sigma), n, \delta_D(q, \sigma)]$$

for every $q \in Q$, $\sigma \in \Sigma$, and $1 \leq n \leq K$ such that $1 \leq n + \delta_D(q, \sigma) \leq K$.
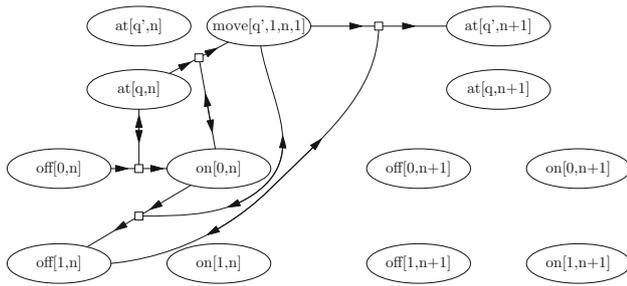
Fig. 1 Some of the states and transitions involved in modelling a Turing machine

The head, currently on cell n, observes that the cell is *on*, writes the new symbol on it, and leaves.

- Type **2b**: A transition $move[q, \sigma, n, d] \xrightarrow{off[\sigma, n]} at[q, n + d]$ for every $q \in Q$, $\sigma \in \Sigma$, and $1 \le n \le K$ such that $1 \le n + d \le K$.

  The head, currently moving, observes that the old cell has turned *off*, and places itself on the new cell.

Figure 1 graphically represents some of the states and transitions of $\mathscr{P}_M$; the double arcs indicates the states being observed. We define the configuration of $\mathscr{P}_M$ that corresponds to a given configuration of the Turing machine.

Definition 3.1 Given a configuration c of M with control state q, tape content $\sigma_1 \sigma_2 \cdots \sigma_K$, and head on cell $n \le K$, let C_c be the configuration that puts one agent in $off[\sigma_i, i]$ for each $1 \le i \le K$, one agent in $at[q, n]$, and no agents elsewhere.

Theorem 3.2 below formalizes the relation between the Turing machine M and its implementation $\mathscr{P}_M$.

Theorem 3.2 *For every two configurations c, c' of M that write at most K cells: $c \to c'$ iff $C_c \xrightarrow{t_1 t_2 t_3 t_4} C_{c'}$ in $\mathscr{P}_M$ for some transitions t_1, t_2, t_3, t_4 of types **1a**, **2a**, **1b**, **2b**, respectively.*

Proof By Lemma A.3, for all c there is either zero or one possibility for the sequence t_1, t_2, t_3, t_4 starting in C_c. It is easy to see from the definition of steps configuration $move[\cdot, \cdot, \cdot, \cdot]$ states that if such a sequence exists, it results in c' such that $c \to c'$. If such a sequence doesn't exist, the failure must occur when trying to populate a $move[\cdot, \cdot, \cdot, \cdot]$ state. In that case the configuration c must be blocked, either by the transition being undefined or by going out of bounds. □

Now we can finally prove the PSPACE lower bound.

Theorem 3.3 *The single-instance correctness and correctness problems for IO protocols are PSPACE-hard.*

Proof By reduction from the following problem: Given a polynomially space-bounded deterministic Turing machine

M with two distinguished states q_{acc}, q_{rej}, such that the computation of M on empty tape ends when the head enters for the first time q_{acc} or q_{rej} (and one of the two occurs), decide whether M accepts, i.e., whether the computation on empty tape reaches q_{acc}. The problem is known to be PSPACE-hard.

Single-instance correctness. We construct a protocol $\mathscr{P}$ and an input D_0 such that M accepts on empty tape iff all fair executions of $\mathscr{P}$ starting at the configuration $I(D_0)$ converge to 1.

Definition of $\mathscr{P}$. Let $\mathscr{P}_M$ be the IO protocol implementation of M. We add two states to $\mathscr{P}_M$, called *observer* and *success*. We also add transitions allowing an agent in state *observer* to move to *success* by observing any agent in a state of the form $at[q_{acc}, i]$, as well as transitions allowing an agent in *success* to "attract" agents in all other states to *success*:

(i) *observer* $\xrightarrow{at[q_{acc}, i]}$ *success* for every $1 \le i \le K$, and

(ii) $q \xrightarrow{success}$ *success* for every $q \ne success$.

Further, we set the output function to 1 for the state *success*, and to 0 for all other states. Finally, we choose the alphabet of input symbols of $\mathscr{P}$ as $\{1, 2, \ldots, K+2\}$, and define the input function as follows: $\iota(i) = off[\text{\_}, i]$ for every $1 \le i \le K$; $\iota(K + 1) = at[q_0, 0]$; and $\iota(K + 2) = observer$.

Definition of D_0. We choose D_0 as the input satisfying $D_0(i) = 1$ for every input symbol of $\mathscr{P}$. It follows that $I(D_0)$ is the configuration of $\mathscr{P}$ corresponding to the initial configuration of M on empty tape. By Theorem 3.2, the fair executions of $\mathscr{P}$ from $I(D_0)$ simulate the execution of M on empty tape.

Correctness of the reduction. If M accepts, then, since $\mathscr{P}$ simulates the execution of M on empty tape, every fair execution of $\mathscr{P}$ starting at $I(D_0)$ eventually puts an agent in a state of the form $at[q_{acc}, i]$. This agent stays there until the agent in state *observer* eventually moves to *success* (transitions of (i)), after which all agents are eventually attracted to *success* (transitions of (ii)). So all fair computations of $\mathscr{P}$ starting at $I(D_0)$ converge to 1. If M rejects, then no computation of $\mathscr{P}$ starting at $I(D_0)$ (fair or not) ever puts an agent in *success*. Since all other states have output 0, all fair computations of $\mathscr{P}$ starting at $I(D_0)$ converge to 0.

Correctness. Notice that the hardness proof for single-instance correctness establishes PSPACE-hardness already for restricted instances $(\mathscr{P}, D)$ satisfying $D(q) \in \{0, 1\}$ for every state q. Call this restricted variant the 0/1-single-instance correctness problem for IO. We claim that the 0/1-single-instance correctness problem for IO is polynomial-time reducible to the correctness problem for IO. By PSPACE-hardness of the 0/1-single-instance correctness

problem for IO, the claim entails PSPACE-hardness for the latter.

Let us now show the claim. Given an IO protocol $\mathscr{P}$ and some configuration D for the $0/1$-instance-correctness problem, we provide a polynomial-time construction of an IO protocol $\mathscr{P}'$ such that $\mathscr{P}'$ computes the constant predicate $\varphi(\mathbf{x}) = 0$ if and only if every fair run of $\mathscr{P}$ starting in D stabilizes to 0. It is well known that, given two protocols $\mathscr{P}_1$ and $\mathscr{P}_2$ with n_1 and n_2 states and computing two predicates φ_1 and φ_2, it is possible to construct a third protocol computing $\varphi_1 \wedge \varphi_2$, often called the *synchronous product*, whose states are pair of states of $\mathscr{P}_1$ and $\mathscr{P}_2$, and has therefore $O(n_1 \cdot n_2)$ states (see e.g. [6]). We define $\mathscr{P}'$ as the synchronous product of $\mathscr{P}$ with a protocol $\mathscr{P}_D$ that computes whether the input is equal to D. The output function of $\mathscr{P}'$ maps the product state (q_1, q_2) to 1 if and only if both q_1 and q_2 map to output 1 in their respective protocols. Thus, a fair run of $\mathscr{P}'$ stabilizes to 1 if and only if the input configuration equals D and $\mathscr{P}$ stabilizes to 1 for input D, which is precisely the case if $(\mathscr{P}, D)$ is a positive instance for the $0/1$-single-instance problem.

It remains to show that $\mathscr{P}_D$ is polynomial-time constructible. Such a protocol is well-known, but we repeat the definition. Let $D = (d_1, \ldots, d_m)$ with $d_i \in \{0, 1\}$, and let $i_1 \leq i_2 \leq \ldots \leq i_k$ be the maximal sequence of indices satisfying $d_{i_j} = 1$ for every j. Since every population has at least two agents, we have $k \geq 2$. We first construct an IO protocol $\mathscr{P}_\psi$ that computes the predicate $\psi = d_{i_1} \geq 1 \wedge d_{i_2} \geq 1 \wedge \ldots \wedge d_{i_k} \geq 1$, using $m + k - 1$ states: The states of $\mathscr{P}_\psi$ are $Q_{\mathscr{P}} \uplus \{2, \ldots, k\}$ where $Q_{\mathscr{P}}$ is the set of states of $\mathscr{P}$. The input mapping of $\mathscr{P}_\psi$ is identical to the input mapping of $\mathscr{P}$. Let q_{i_j} denote the state that corresponds to the entry d_{i_j} in D. The transitions of $\mathscr{P}_\psi$ are given by

$$q_{i_2} \xrightarrow{q_{i_1}} 2$$

$$q_{i_j} \xrightarrow{j-1} j \qquad \text{for every } 1 < j \leq k,$$

$$q \xrightarrow{k} k \qquad \text{for every state } q.$$

All states except k shall map to output 0. It is readily seen that $\mathscr{P}_\psi$ computes ψ. Further notice that the predicate $\mathbf{x} = D$ is equivalent to $\psi \wedge |\mathbf{x}| \leq k$. Moreover, it is well-known that the right conjunct $|\mathbf{x}| \leq k$ is computable with k states in an immediate observation protocol (see e.g. [6]), and thus we can define $\mathscr{P}_D$ as the synchronous product of the protocol $\mathscr{P}_\psi$ with the protocol that computes $|\mathbf{x}| \leq k$, using $\mathsf{poly}(k)$ states. This completes the proof. $\qquad \square$

3.2 Correctness of DO protocols is Π_2^p-hard

We show that the single-instance correctness and the correctness problems are Π_2^p-hard for DO protocols, where $\Pi_2^p = \mathsf{coNP}^{\mathsf{coNP}}$ is one of the two classes at the second level of the polynomial hierarchy [48]. Consider the natural complete problem for Σ_2^p: Given a boolean circuit Γ with inputs $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_m)$, is there a valuation of $\mathbf{x}$ such that for every valuation of $\mathbf{y}$ the circuit outputs 1? We call the inputs of $\mathbf{x}$ and $\mathbf{y}$ *existential* and *universal*, respectively. Given Γ with inputs $\mathbf{x}$ and $\mathbf{y}$, we construct in polynomial time a DO protocol $\mathscr{P}_\Gamma$ with input symbols $\{x_1, \ldots, x_n\}$ that computes the false predicate, i.e., the predicate answering 0 for all inputs, iff Γ does *not* satisfy the property above. This shows that the correctness problem for DO protocols is Π_2^p-hard. A little modification of the proof shows that single-instance correctness is also Π_2^p-hard.

The section is divided in several parts. We first introduce basic notations about boolean circuits. Then we sketch a construction that, given a boolean circuit Γ, returns a *circuit evaluation protocol* $\widehat{\mathscr{P}_\Gamma}$ that nondeterministically chooses values for the input nodes, and simulates an execution of Γ on these inputs. In a third step we add some states and transitions to $\widehat{\mathscr{P}_\Gamma}$ to produce the final DO protocol $\mathscr{P}_\Gamma$. The fourth and final step proves the correctness of the reduction.

Boolean circuits. A boolean circuit Γ is a directed acyclic graph. The nodes of Γ are either *input nodes*, which have no incoming edges, or *gates*, which have at least one incoming edge. A gate with k incoming edges is labeled by a boolean operation of arity k. We assume that k is bounded by some constant. This assumption is innocuous since it is well known that every boolean function can be implemented using a combination of gates of constant arity. The nodes with outgoing edges leading to a a gate g are called the *arguments* of g. There is a distinguished *output gate* g_o without outgoing edges. We assume that every node is connected to the output gate by at least one path.

A circuit configuration assigns to each input node a boolean value, 0 or 1, and to each gate a value, 0, 1, or $\square$, where $\square$ denotes that the value has not yet been computed and so it is still unknown. A configuration is initial if it assigns $\square$ to all gates. The step relation between circuit configurations is defined as usual: a gate can change its value to the result of applying the boolean operation to the arguments; if at least one of the arguments has value $\square$, then by definition the result of the boolean operation is also $\square$.

The protocol $\widehat{\mathscr{P}_\Gamma}$. Given a circuit Γ with output node g_o, we define the circuit evaluation protocol $\widehat{\mathscr{P}_\Gamma} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$. As mentioned above, $\widehat{\mathscr{P}_\Gamma}$ nondeterministically chooses input values for Γ, and simulates an execution on them.

States. The set Q of states contains all tuples $(n, v_n, \mathrm{arg}, v_o)$, where:

– n is a node of Γ (either an input node or a gate);
– $v_n \in \{0, 1, \square\}$ represents the current opinion of the agent about the value of n;
– arg $\in \{0, 1, \square\}^k$, where k is the number of arguments of n, represents the current opinion of the agent about the values of the arguments of n (if n is an input node then arg is the empty tuple);
– $v_o \in \{0, 1, \square\}$ represents the current opinion of the agent about the value of the output gate g_o.

Alphabet, input and output functions. The set Σ of input symbols is the set of nodes of Γ. The initial state mapping ι maps each node n to the state $\iota(n) := (n, \square, (\square, \ldots, \square), \square)$, i.e., to the state with node n, and with all values still unknown. The output function is defined by

$$o(n, v_n, \text{arg}, v_o) := \text{if } v_o \neq \square \text{ then } v_o \text{ else } 0 .$$

Intuitively, agents have opinion 1 if they think the circuit outputs 1, and 0 if they think the circuit outputs 0 or has not yet produced an output.

Messages. The set M of messages contains all pairs (n, v), where n is a node, and $v \in \{0, 1, \square\}$ is a value.

Transitions. An agent in state $(n, v_n, \text{arg}, v_o)$ can

– Send the message (n, v_n), i.e., an agent can send its node and its current opinion on the value of the node.
– Receive a message (m, v_m), after which the agent updates its state as follows:

   (1) If n is an input node and $v_n = \square$, then if $m = n$ the agent moves to state $(n, 0, \text{arg}, v_o)$, i.e., updates its value to 0, and if $m = g_o$ it moves to state $(n, 1, \text{arg}, v_o)$, i.e., updates its value to 1. Intuitively, this is an artificial but simple way of ensuring that each input node nondeterministically chooses a value, 0, or 1, depending on whether it first receives a message from itself, or from the output node. [2]
   (2) If n is a gate and m is an argument of n, then the agent moves to $(n, v_n', \text{arg}', v_o)$, where arg$'$ is the result of updating the value of m in arg to v_m, and v_n' is the result of applying the boolean operation of the gate to arg.
   (3) If n is any node, $m = g_o$, and $v_m \neq \square$, then the agent moves to $(n, 0, \text{arg}, v_m)$, i.e., it updates its opinion of the output of the circuit to v_m.

---

[2] Alternatively, one could include an explicit rule for this non-deterministic behavior. We choose to model it this way to preserve the deterministic definition of the DO model introduced by Angluin et al. in [7].

Notice that if an agent is initially in state $\iota(n)$, then it remains forever in states having n as node. So it makes sense to speak of *the* node of an agent.

Let us examine the behaviour of $\widehat{\mathscr{P}_\Gamma}$ from the initial configuration C_0 that puts exactly one agent in state $\iota(n)$ for every node n. The executions of $\widehat{\mathscr{P}_\Gamma}$ from C_0 exactly simulate the executions of the circuit. Indeed, the transitions of (1) ensure that each input agent (i.e., every agent whose node is an input node) eventually chooses a value, 0 or 1. The transitions of (2) simulate the computations of the gates. Finally, the transitions of (3) ensure that every node eventually updates its opinion of the value of g_o to the value computed by Γ for the chosen input. The following lemma, proved in the "Appendix B", formalizes this.

Lemma 3.4 *Let Γ be a circuit and let $\widehat{\mathscr{P}_\Gamma}$ be its evaluation protocol. Let C_0 be the initial configuration that puts exactly one agent in state $\iota(n)$ for every node n. A fair execution starting at C_0 eventually reaches a configuration C where each input agent is in a state with value 0 or 1, and these values do not change afterwards. The tail of the execution starting at C converges to a stable consensus equal to the output of Γ on these assigned inputs.*

Observe, however, that $\widehat{\mathscr{P}_\Gamma}$ also has initial configurations whose executions may not simulate any execution of Γ. For example, this is the case of an initial configuration that puts two agents in state $\iota(n)$ for some node n, and the executions in which one of these agents chooses input 0 for n, and the other input 1. It is also the case of an initial configuration that puts zero agents in state $\iota(n)$ for some node n. Observe further that $\widehat{\mathscr{P}_\Gamma}$ can only select values for the inputs, and simulate an execution of Γ. We need a protocol that selects values for the existential inputs, and can then repeatedly simulate the circuit for different values of the universal inputs. These two problems are solved by appropriately extending $\widehat{\mathscr{P}_\Gamma}$ with new states and transitions.

The protocol $\mathscr{P}_\Gamma$. We add a new state and some transitions to $\widehat{\mathscr{P}_\Gamma}$ in order to obtain the final protocol $\widehat{\mathscr{P}_\Gamma}$.

– Add a new *failure* state $\perp$ with $o(\perp) = 0$ to the set of states Q, and a new message $m_\perp$ to the set of messages M.
– Add the following send and receive transitions:

   (4) An agent in state $\perp$ can send the message $m_\perp$.
   (5) An agent in state $\perp$ that receives any message (including $m_\perp$) stays in state $\perp$; an agent (in any state, including $\perp$) that receives $m_\perp$ moves to state $\perp$.

   (In particular, if some agent ever reaches state $\perp$, then all agents eventually reach state $\perp$ and stay there, and so the protocol converges to 0.)

(6) If an agent in state $(n, v_n, \text{arg}, v_o)$, where n is an existential input node and $v_n \neq \square$, receives a message (n, v_n') such that $v_n \neq v_n' \neq \square$, then the agent moves to state $\bot$.

(Intuitively, if an agent discovers that another agent has chosen a different value for the same existential input, then the agent moves to $\bot$, and so, by the observation above, the protocol converges to 0.)

(7) If an agent in state $(n, v_n, \text{arg}, v_o)$, where n is a universal input node and $v_n \neq \square$, receives a message $(g_o, 1)$, then the agent moves to state $(n, 1 - v_n, \text{arg}, v_o)$.

(Intuitively, this allows the protocol to flip the values of any universal inputs whenever the output gate takes value 1.)

Proof of the reduction. We claim that $\mathscr{P}_\Gamma$ does not compute the false predicate (i.e., the predicate that answers 0 for every input) iff $\exists \mathbf{x} \forall \mathbf{y} \, \Gamma(\mathbf{x}, \mathbf{y}) = 1$, that is, if there is a valuation of the existential inputs of Γ such that, for every valuation of the universal inputs, Γ returns 1. Let us sketch the proof of the claim. We consider two cases:

$\exists \mathbf{x} \forall \mathbf{y} \, \Gamma(\mathbf{x}, \mathbf{y}) = 1$ *is true.* Let C_0 be the initial configuration that puts exactly one agent in state $\iota(n)$ for every node n. We show that not every fair execution from C_0 converges to 0, and so that $\mathscr{P}_\Gamma$ does not compute the 0 predicate.

Let $\mathbf{x}_0$ be a valuation of $\mathbf{x}$ such that $\forall \mathbf{y} \, \Gamma(\mathbf{x}_0, \mathbf{y}) = 1$. The execution proceeds as follows: first, the agents for the inputs of $\mathbf{x}$ receive messages, sent either by themselves or by the output node, that make them choose the values of $\mathbf{x}_0$. An inspection of the transitions of $\mathscr{P}_\Gamma$ shows that these values cannot change anymore. Let C be the configuration reached after the agents have received the messages. Since $\Gamma(\mathbf{x}_0, \mathbf{y}) = 1$ holds for every $\mathbf{y}$, by Lemma 3.4 every configuration C' reachable from C can reach a consensus of 1. Indeed, it suffices to first let the agents receive all messages of C' (which does not change the values of the existential inputs), then let the agents for $\mathbf{y}$ that still have value $\square$ pick a boolean value (nondeterministically), and then let all agents simulate the circuit. Since $\Gamma(\mathbf{x}_0, \mathbf{y}) = 1$ holds for every $\mathbf{y}$, after the simulation the node for g_o has value 1. Finally, we let all agents move to states satisfying $v_o = 1$.

$\exists \mathbf{x} \forall \mathbf{y} \, \Gamma(\mathbf{x}, \mathbf{y}) = 1$ *is false.* This case requires a finer analysis. We have to show that $\mathscr{P}_\Gamma$ computes the false predicate, i.e., that every fair execution from every initial configuration converges to 0. By fairness, it suffices to show that for every initial configuration C_0 and for every configuration C reachable from C_0, it is possible to reach from C a stable consensus of 0.

Thanks to the $\bot$ state, which is introduced for this purpose, configurations C in which two agents for the same existen-

tial input node choose inconsistent values eventually reach the configuration with all agents in state $\bot$, which is a stable consensus of 0. Thanks to the assumption that every node is connected to the output gate by at least one path, configurations C in which there are no agents for some node cannot reach any configuration in which some agent populates a state with $v_o = 1$, and so C itself is a stable consensus of 0. So, loosely speaking, configurations in which the agents pick more than one value, or can pick no value at all, for some existential input eventually reach a stable consensus of 0.

Consider the case in which, for every node n, the configuration C has at least one agent in a state with node n. By fairness, C eventually reaches a configuration C' at which each agent for an existential input has chosen a boolean value, and we can assume that all agents for the same input choose the same value. This fixes a valuation $\mathbf{x}_0$ of the existential inputs. Recall that this valuation cannot change any more, since the protocol has no transitions for that. By assumption, there is $\mathbf{y}_0$ such that $\Gamma(\mathbf{x}_0, \mathbf{y}_0) = 0$. We sketch how to reach a stable consensus of 0 from C'. First, let the agents consume all messages of C', and let C'' be the resulting configuration. If C'' cannot reach any configuration with circuit output 1, then the configuration reached after informing each agent about the value of g_o is a stable consensus of 0, and we are done. Otherwise, starting from such a configuration with output 1, let the agents send and receive the appropriate messages so that all agents for $\mathbf{y}$ choose the values of $\mathbf{y}_0$. After that, let the agent for g_o consume all remaining messages, if any, and let the protocol simulate Γ on $\mathbf{x}_0, \mathbf{y}_0$. Notice that the simulation can be carried out even if there are multiple agents for the same gate g. Indeed, in this case, for every argument g' of g, we let at least one of the agents corresponding to g' send the message with the correct value for g' to all the agents for n. Since $\Gamma(\mathbf{x}_0, \mathbf{y}_0) = 0$ by assumption, the agents for g_o eventually update their value to 0, and eventually all agents change their opinion about the output of the circuit to 0. Let C''' be the configuration so reached. We claim that C''' is a stable consensus of 0. Indeed, the state of a gate cannot change without a change in the argument values or the output gate g_o. Therefore it is enough to prove that the input values cannot change. Since no transition can change $\mathbf{x}_0$, this can only happen by changing the values $\mathbf{y}_0$ of the universal inputs. But these values can only change by the transitions of (7), which require the agent to receive a message $(g_o, 1)$. This is not possible because the current value of g_o is 0, and the claim is proved.

This concludes the reduction to the correctness problem for DO protocols. We can easily transform it into a reduction to the single-instance correctness problem. Indeed, it suffices to observe that the executions of the circuit Γ correspond to the fair executions of $\mathscr{P}_\Gamma$ from the unique initial configuration C_0 with exactly one agent in state $\iota(n)$ for every node n.

So $\mathscr{P}_\Gamma$ computes 0 from C_0 iff $\exists \mathbf{x} \forall \mathbf{y} \, \Gamma(\mathbf{x}, \mathbf{y}) = 1$, and we are done. So we have:

Theorem 3.5 *The single-instance correctness and correctness problems for DO protocols are Π_2^p-hard.*

4 Reachability in observation models: the pruning and shortening theorems

In the next three sections we prove that the correctness problem is PSPACE-complete for IO protocols and Π_2^p-complete for DO protocols. These are the most involved results of this paper. They can only be obtained after a detailed study of the reachability problem of IO and DO protocols, which we believe to be of independent interest. The roadmap for the three sections is as follows.

Section 4. Section 4.1 introduces *message-free delayed-observation protocols* (MFDO), an auxiliary model very close to DO protocols, but technically more convenient. As its name indicates, agents of MFDO protocols do not communicate by messages. Instead, they directly observe the current *or past* states of other agents. As a consequence, a configuration of an MFDO protocol is completely determined by the states of its agents, which has technical advantages. At the same time, MFDO and DO protocols are very close, in the following sense. We call a configuration of a DO protocol a *zero-message* configuration if all messages sent by the agents have already been received. Given a DO protocol $\mathscr{P}$ we can construct an MFDO protocol $\widehat{\mathscr{P}}$, with the same set of states, such that for any two zero-message configurations Z, Z' of $\mathscr{P}$, we have $Z \xrightarrow{*} Z'$ in $\mathscr{P}$ iff $Z \xrightarrow{*} Z'$ in $\widehat{\mathscr{P}}$. (Observe that, since $\mathscr{P}$ and $\widehat{\mathscr{P}}$ have the same set of states, a zero-message configuration of $\mathscr{P}$ is also a configuration of $\widehat{\mathscr{P}}$.) So, any question about the reachability relation between zero-message configurations of $\mathscr{P}$ can be "transferred" to $\widehat{\mathscr{P}}$, and answered there.

The rest of the section is devoted to the Pruning and Shortening Theorems. Say that a configuration C is coverable from C' if there exists a configuration C'' such that $C' \xrightarrow{*} C'' \geq C$. The Pruning Theorems state that if a configuration C of a protocol with n states is coverable from C', then it is also coverable from a "small" configuration $D \leq C'$, where small means $|D| \leq |C| + f(n)$ for a low-degree polynomial f. The Shortening Theorem states that every execution $C \xrightarrow{*} C'$ can be "shortened" to an execution $C \xrightarrow{\xi} C'$, where $\xi = t_1^{k_1} t_2^{k_2} \ldots t_m^{k_m}$ and $m \leq f(n)$ for some low-degree polynomial f that depends only on n, not on C or C'. Intuitively, if we assume that the k_i occurrences of t_i are executed synchronously in one step, then the execution only takes m steps.

Section 5. This section applies the Pruning and Shortening Theorems to the reachability problem between *counting sets* of configurations. Intuitively, a counting set of configurations is a union of *cubes*, and a cube is the set of all configurations C lying between a lower bound configuration L and an upper bound configuration U with possibly infinite components. Observe that counting sets may be infinite, but always have a finite representation. The reachability problem for counting sets asks, given two counting sets $\mathscr{C}$ and $\mathscr{C}'$, whether some configuration of $\mathscr{C}'$ is reachable from some configuration of $\mathscr{C}'$. The section proves two very powerful Closure Theorems for IO and DO. The Closure Theorems state that for every counting set $\mathscr{C}$, the set $post^*(\mathscr{C})$ of all configurations reachable from $\mathscr{C}$ is also a counting set; further, the same holds for the set $pre^*(\mathscr{C})$ of all configurations from which $\mathscr{C}$ can be reached. So, loosely speaking, counting sets are closed under reachability. Furthermore, the section shows that if $\mathscr{C}$ has a representation with "small" cubes, in a sense to be determined, then so do $pre^*(\mathscr{C})$ and $post^*(\mathscr{C})$.

Section 6. This section applies the Pruning, Shortening, and Closure Theorems to prove the PSPACE and Π_2^p upper bounds for the correctness problems of IO and DO protocols, respectively. The section shows that this is also the complexity of the single-instance correctness problems.

Notation. Throughout these sections, the last three components of the tuples describing protocols (input symbol set Σ, initial set mapping ι, and output mapping o) play no role. Therefore we represent a DO protocol by the simplified tuple $(Q, M, \delta_s, \delta_r)$, and an IO protocol as just a pair (Q, δ).

Section 4.2 proves the Pruning Theorems for IO and MFDO protocols. Section 4.3 proves the Shortening Theorem for MFDO protocols. Finally, making use of the tight connection between MFDO and DO protocols, Sect. 4.4 proves the Pruning and Shortening Theorems for DO protocols.

4.1 An auxiliary model: message-free delayed-observation protocols

Immediate observation and delayed observation protocols present similarities. Essentially, in an immediate observation protocol an agent updates its state when it observes that another agent is *currently* in a certain state q, while in a delayed observation protocol the agent observes that another agent *was* in a certain state q, provided that agent *emitted a message when it was in q*. In a message-free delayed observation protocol we assume that a sufficient amount of such messages is always emitted by default; this allows us to dispense with the message, and directly postulate that an agent can observe whether another agent went through a given state in the past. So the model is message-free, and, since

agents can observe events that happened in the past, we call it "message-free delayed observation".

Definition 4.1 A message-free delayed observation (MFDO) protocol is a pair $\mathscr{P} = (Q, \delta)$, where Q is a set of states and $\delta : Q^2 \to Q$ is a transition function. Considering δ as a set of transitions, we write $q \xrightarrow{o} q'$ for $((q, o), q') \in \delta$. The set of finite executions of $\mathscr{P}$ is the set of finite sequences of configurations defined inductively as follows. Every configuration C_0 is a finite execution. A finite execution $C_0, C_1, \ldots, C_i$ enables a transition $q \xrightarrow{o} q'$ if $C_i(q) \geq 1$ and there exists $j \leq i$ such that $C_j(o) \geq 1$. (We say the agent of C_i at state q *observes* that there *was* an agent in state o at C_j.) If C_i enables $q \xrightarrow{o} q'$, then $C_0, C_1, \ldots, C_i, C_{i+1}$ is also a finite execution of $\mathscr{P}$, where $C_{i+1} = C_i - \{q\} + \{q'\}$. An infinite sequence of configurations is an execution of $\mathscr{P}$ if all its finite prefixes are finite executions.

We assign to every DO protocol an MFDO protocol.

Definition 4.2 Let $\mathscr{P}_{DO} = (Q, M, \delta_r, \delta_s)$ be a DO protocol. The *MFDO protocol corresponding to* $\mathscr{P}_{DO}$ is $\mathscr{P}_{MFDO} = (Q, \delta)$, where δ is the set of transitions $q \xrightarrow{o} q'$ such that $q' = \delta_r(q, m)$ for some message $m \in M$, and o is a state satisfying $\delta_s(o) = (m, o)$.

Notice that if Q has multiple states $o_1, \ldots, o_k$ such that $\delta_s(o_i) = (m, o_i)$ for every $1 \leq i \leq k$, then $\mathscr{P}_{MFDO}$ contains a transition $q \xrightarrow{o_i} q'$ for every $1 \leq i \leq k$.

Example 4.3 Consider the DO protocol $\mathscr{P}_{DO} = (Q, M, \delta_r, \delta_s)$ where $Q = M = \{a, b, ab\}$ and $\Sigma = \{a, b\}$. The send transitions are given by $\delta_s(q) = (q, q)$ for all $q \in Q$, i.e., every state can send a message with its own identity to itself, denoted $q \xrightarrow{q+} q$. The receive transitions are $\delta_r(a, b) = ab$ and $\delta_r(b, a) = ab$, denoted $a \xrightarrow{b-} ab$ and $b \xrightarrow{a-} ab$.

The corresponding MFDO protocol is $\mathscr{P}_{MFDO} = (Q, \delta)$, where δ contains the transitions $a \xrightarrow{b} ab$ and $b \xrightarrow{a} ab$.

Notice that an agent of a DO protocol can "choose" not to send a message when it goes through a state, and thus not enable a future transition that consumes such a message. This does not happen in MFDO protocols. In particular, if a configuration C of an MFDO protocol enables a transition $q \xrightarrow{o} q$, then the transition remains enabled forever, and in particular C^ω is an execution. This is not the case for a transition $q \xrightarrow{o-} q'$ of a DO protocol, because each occurrence of the transition consumes one message, and eventually there are no messages left.

Despite this difference, a DO protocol and its corresponding MFDO protocol are equivalent with respect to reachability questions in the following sense. Observe that a configuration of $\mathscr{P}_{DO}$ with zero messages is also a configuration of $\mathscr{P}_{MFDO}$. From now on, given a DO protocol,

we denote by $\mathscr{Z}$ the set of its *zero-message configurations*. For every $Z \in \mathscr{Z}$, we overload the notation Z by also using it to denote the configuration of the corresponding MFDO protocol which is the restriction of Z to a multiset over Q. The following lemma shows that for any two configurations Z and Z' with zero messages, Z' is reachable from Z in $\mathscr{P}_{DO}$ iff it is reachable in $\mathscr{P}_{MFDO}$.

Lemma 4.4 *Let* $\mathscr{P}_{DO} = (Q, M, \delta_s, \delta_r)$ *be a DO protocol, and let* $\mathscr{P}_{MFDO} = (Q, \delta)$ *be its corresponding MFDO protocol. Let* $Z, Z' \in \mathscr{Z}$ *be two zero-message configurations. Then* $Z \xrightarrow{*} Z'$ *in* $\mathscr{P}_{DO}$ *if and only if* $Z \xrightarrow{*} Z'$ *in* $\mathscr{P}_{MFDO}$.

Proof **DO to MFDO.** Let $Z \xrightarrow{\xi} Z'$ be an execution of $\mathscr{P}_{DO}$ with $Z, Z' \in \mathscr{Z}$. Let $\xi = t_1 t_2 \cdots t_n$, and let $C_0, C_1, C_2, \ldots, C_n$ be the configurations describing the number of agents in each state along ξ. In particular, $C_0 = Z$ and $C_n = Z'$. Define the sequence τ as follows. For every transition t_i:

– If t_i is a send transition (i.e., if $t_i = q \xrightarrow{m+} q$ for some q and m), then delete t_i.
  Observe that, since the occurrence of t_i does not change the state of any agent, we have $C_i = C_{i+1}$, and so in particular $C_i \xrightarrow{\epsilon} C_{i+1}$ in $\mathscr{P}_{MFDO}$.
– If t_i is a receive transition, i.e., if $t_i = q \xrightarrow{m-} q'$ for some q, q', and m, then replace it by the transition $q \xrightarrow{o} q'$, where o is any state satisfying $t_j = o \xrightarrow{m+} o$ for some index $j \leq i$.
  Observe that the transition t_j must exist, because every message received has been sent. Further, since both t_i and $q \xrightarrow{o} q'$ move an agent from q to q', we have $C_i \xrightarrow{u_i} C_{i+1}$ in $\mathscr{P}_{MFDO}$ for $u_i = q \xrightarrow{o} q'$.

The result follows from the fact that in both cases we have $C_i \xrightarrow{*} C_{i+1}$ in $\mathscr{P}_{MFDO}$.

MFDO to DO. Let $Z \xrightarrow{\tau} Z'$ be an execution of $\mathscr{P}_{MFDO}$, and let $\tau = u_1 u_2 \cdots u_n$, where $u_i = q_i \xrightarrow{o_i} q_{i+1}$. We define the sequence ξ, such that $Z' \xrightarrow{\xi} Z$, in two steps as follows.

1. First replace every transition u_i by $q_i \xrightarrow{m-} q_{i+1}$ for a message $m \in M$ such that $\delta_s(o_i) = (m, o_i)$. Transition $q_i \xrightarrow{m-} q_{i+1}$ exists in $\mathscr{P}_{DO}$ by construction of $\mathscr{P}_{MFDO}$.
2. For each message $m \in M$ in ξ, denote by q_m the state such that $\delta_s(q_m) = (m, q_m)$ and let $\#(m, \xi)$ denote the number of times m is consumed along ξ. If there are multiple states with such property, we choose the state that occurs earliest in the original execution. Add $\#(m, \xi)$ iterations of $q_m \xrightarrow{m+} q_m$ at the first configuration along ξ in which state q_m is populated. This ensures that the messages that the agents need to move from q_i to q_{i+1} are always available

to be received and that all the messages will be consumed at the end of the execution.

Thus ξ is enabled and goes from Z' to Z. $\qquad\square$

4.2 Pruning theorems for IO and MFDO protocols

The Pruning Theorems for IO and MFDO protocols are proved in the same way. Given an execution $C'' \xrightarrow{\xi} C' \geq C$, we examine the trajectories of the different agents during the execution of ξ. For this, we assign trajectories to the agents in an arbitrary way, but consistent with the configurations reached during the execution. For example, consider a protocol with states q_1, q_2, q, q_1', q_2' in which two agents, initially in states q_1 and q_2, first move to q, after which one of them moves to q_1' and the other to q_2'. Since the two agents are indistinguishable, we can choose to assume that their trajectories were q_1, q, q_1' and q_2, q, q_2', or that they were q_1, q, q_2' and q_2, q, q_1'. After "splitting" the execution into a multiset of trajectories, one for each agent, we "prune" the multiset, keeping only those trajectories that are "necessary" to cover C. This yields a smaller multiset, which we then "transform back" into an execution.

4.2.1 Pruning theorem for IO protocols

Definition 4.5 A *trajectory* of an IO protocol $\mathscr{P} = (Q, \delta)$ is a sequence $\tau = q_1 \ldots q_n$ of states. We let $\tau(i)$ denote the i-th state of τ. The *i-th step* of τ is the pair $\tau(i)\tau(i+1)$ of adjacent states.

A *history* is a multiset of trajectories of the same length. The length of a history is the common length of its trajectories. Given a history H of length n and index $1 \leq i \leq n$, the *i-th configuration of H*, denoted C_H^i, is defined as follows: for every state p, $C_H^i(q)$ is the number of trajectories $\tau \in H$ such that $\tau(i) = q$. The configurations C_H^1 and C_H^n are called the *initial* and *final* configurations of H.

Example 4.6 Let $\mathscr{P} = (Q, \delta)$ be the IO protocol with $Q = \{q_1, q_2, q_3\}$ and $\delta = \{t_1, t_2, t_3, t_4\}$, where

$$t_1 = q_1 \xrightarrow{q_1} q_2 \quad t_3 = q_1 \xrightarrow{q_3} q_3$$
$$t_2 = q_2 \xrightarrow{q_2} q_3 \quad t_4 = q_2 \xrightarrow{q_3} q_3$$

We use this protocol as running example throuout the section. Histories of $\mathscr{P}$ can be graphically represented. Figure 2 shows a history H of length 7. It consists of five trajectories: one trajectory from q_3 to q_3 passing only through q_3, and four trajectories from q_1 to q_3 which follow different state sequences. The first configuration of H is $C_H^1 = (4, 0, 1)$ and the seventh and last configuration is $C_H^7 = (0, 0, 5)$.
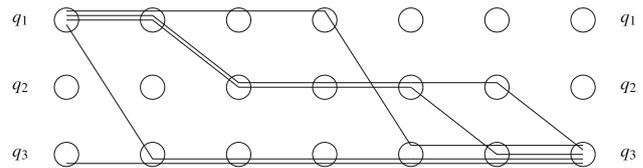
Fig. 2 Realizable history in IO protocol with three states

Definition 4.7 A history H of length $n \geq 1$ is *realizable* in an IO protocol $\mathscr{P}$ if there exist transitions $t_1, \ldots, t_{n-1}$ of $\mathscr{P}$ and numbers $k_1, \ldots, k_{n-1} \geq 0$ such that

$$C_H^1 \xrightarrow{t_1^{k_1}} C_H^2 \cdots C_H^{n-1} \xrightarrow{t_{n-1}^{k_{n-1}}} C_H^n \, ,$$

where for every transition t we define $C \xrightarrow{t^0} C'$ iff $C = C'$.

Remark 4.8 Notice that histories of length 1 are always realizable. Observe also that there may be more than one realizable history corresponding to a firing sequence, because the firing sequence does not keep track of which agent visits which states, while the history does.

Example 4.9 The history H of Fig. 2 is realizable in $\mathscr{P}$. Indeed, we have $C_H^1 \xrightarrow{t_3 \, t_1^2 \, t_3 \, t_2 \, t_4} C_H^7$.

We introduce well structured histories. Intuitively, they are the histories in which at every step all agents that move execute the same transition, and so there are states q, q' such that all the agents move from q to q'.

Definition 4.10 A step $\tau(i)\tau(i+1)$ of a trajectory τ is *horizontal* if $\tau(i) = \tau(i+1)$, and *non-horizontal* otherwise.

A history H of length n is *well structured* if for every $1 \leq i \leq n - 1$ one of the two following conditions hold:

(i) For every trajectory $\tau \in H$, the i-th step of τ is horizontal.
(ii) For every two trajectories $\tau_1, \tau_2 \in H$, if the i-th steps of τ_1 and τ_2 are non-horizontal, then they are equal.

Example 4.11 The history of Fig. 2 is well structured. The third step of all five trajectories is horizontal. The second step is horizontal for three trajectories, and non-horizontal for the other two; the two non-horizontal steps are equal, namely $q_1 q_2$.

Characterizing histories. We show that the set of executions of an IO protocol is completely determined by its well-structured and realizable histories. The proof is purely technical, and can be found in the "Appendix C".

Lemma 4.12 *Let $\mathscr{P}$ be an IO protocol. For every configuration C, C' the following holds: $C \xrightarrow{*} C'$ iff there exists a*

well-structured and realizable history in $\mathscr{P}$ with C and C' as initial and final configurations.

We now proceed to give a syntactic characterization of the well-structured and realizable histories.

Definition 4.13 A history H is *compatible* with an IO protocol $\mathscr{P}$ if for every trajectory τ of H and for every non-horizontal step $\tau(i)\tau(i+1)$ of τ, the protocol $\mathscr{P}$ contains a transition $\tau(i) \xrightarrow{o} \tau(i+1)$ for a state o such that H contains a trajectory τ' with $\tau'(i) = \tau'(i+1) = o$.

Intuitively, a history is compatible with a protocol if for every non-horizontal step from, say, q to q', the protocol has a transition of the form $q \xrightarrow{o} q'$ for some observed state o. Since the transition can only happen if an agent in q observes o, there must be another agent in state o (the one with trajectory τ').

Example 4.14 The history of Fig. 2 is compatible with the IO protocol of Example 4.6. Consider for example the trajectory $\tau = q_1 q_1 q_2 q_2 q_2 q_3 q_3$. It has two non-horizontal steps, namely $\tau(2)\tau(3) = q_1 q_2$ and $\tau(5)\tau(6) = q_2 q_3$. The corresponding transitions are $q_1 \xrightarrow{q_1} q_2$ and $q_2 \xrightarrow{q_2} q_3$.

Lemma 4.15 *Let $\mathscr{P}$ be an IO protocol. A well-structured history is realizable in $\mathscr{P}$ iff it is compatible with $\mathscr{P}$.*

Pruning. We introduce *bunches of trajectories*, and present a lemma about pruning bunches. Then, we prove the Pruning Theorem for IO protocols.

Definition 4.16 A *bunch* is a multiset of trajectories of the same length and with the same initial and final states.

Example 4.17 The history of Fig. 2 consists of a trajectory from q_3 to q_3 (which can be considered a bunch of size 1), and a bunch of four trajectories with initial state q_1 and final state q_3.

We show that every well-structured and realizable history containing a bunch of more than $|Q|$ trajectories can be "pruned", meaning that the bunch can be replaced by a smaller one, while keeping the history well-structured and realizable.

Lemma 4.18 *Let $\mathscr{P} = (Q, \delta)$ be an IO protocol. Let H be a well-structured and realizable history of $\mathscr{P}$ containing a bunch $B \subseteq H$ of size larger than $|Q|$. There exists a nonempty bunch B' of size at most $|Q|$, of the same length and with the same initial and final states as B, such that the history $H' \stackrel{def}{=} H - B + B'$ (where $+$ and $-$ denote multiset addition and multiset subtraction, respectively) is also well-structured and realizable.*

Proof Let Q_B be a set of all states visited by at least one trajectory in the bunch B. For every $q \in Q_B$, let $f(q)$ and $l(q)$ be the earliest and the latest moment in time at which q is visited by any of the trajectories (the first and last occurrences can belong to different trajectories).

For every $q \in Q_B$, let $\tau_q = \tau_{q,1}\tau_{q,2}\tau_{q,3}$, where $\tau_{q,1}$ is a prefix of length $f(q) - 1$ of some trajectory of B with q at the moment $f(q)$; $\tau_{q,2} = q^{l(q)-f(q)}$; and $\tau_{q,3}$ is a suffix of some trajectory of B with the state q at the moment $l(q)$, starting at the moment $l(q)$. The prefix and the suffix exist by the definition of $f(q)$ and $l(q)$.

Let $B' = \{\tau_q \mid q \in Q_B\}$, and let $H' = H - B + B'$. We prove that H' is well structured and compatible with $\mathscr{P}$. By Lemma 4.15, this proves that H' is well structured and realizable in $\mathscr{P}$.

Let us first show that H' is well structured. Notice that every trajectory of B' is the concatenation of a prefix of a trajectory of B, a sequence of horizontal steps, and a suffix of another trajectory of B. Hence, if B' contains a trajectory whose i-th step is non-horizontal, then the same holds for B. It follows:

– If the i-th step of H satisfies condition (i) of Definition 4.10, then so does the i-th step of H'.
– If the i-th step of H satisfies condition (ii), then all its non-horizontal i-th steps are equal. So all non-horizontal i-th steps of H' are also equal, which implies that the i-th step of H' also satisfies condition (ii).

Let us now show that H' is compatible with $\mathscr{P}$. Let τ' be a trajectory of H', and let $\tau'(i)\tau'(i+1)$ be a non-horizontal step of τ'. We show that $\mathscr{P}$ has a transition $\tau'(i) \xrightarrow{o'} \tau'(i+1)$, where the state o' satisfies that some trajectory $\tau'' \in H'$ satisfies $\tau''(i) = \tau''(i+1) = o'$.

Since $\tau'(i)\tau'(i+1)$ is a non-horizontal step, by the argument above H contains a trajectory τ such that $\tau(i)\tau(i+1) = \tau'(i)\tau'(i+1)$. Further, H is realizable in $\mathscr{P}$ by assumption, and so by Lemma 4.15 H is compatible with $\mathscr{P}$. So $\mathscr{P}$ has a transition $\tau(i) \xrightarrow{o} \tau(i+1)$, and H has a trajectory $\tilde{\tau}$ such that $\tilde{\tau}(i) = \tilde{\tau}(i+1) = o$. Choose $o' := o$. Since $\tau(i)\tau(i+1) = \tau'(i)\tau'(i+1)$, we have that $\tau'(i) \xrightarrow{o'} \tau'(i+1)$ is a transition of $\mathscr{P}$. It remains to show that some trajectory $\tau'' \in H'$ satisfies $\tau''(i) = \tau''(i+1) = o'$. Consider two cases:

- $\tilde{\tau} \notin B$. Then $\tilde{\tau} \in H'$. Since $\tilde{\tau}(i) = \tilde{\tau}(i+1) = o$, we can choose $\tau'' := \tilde{\tau}$.
- $\tilde{\tau} \in B$. Then, since $\tilde{\tau}(i) = \tilde{\tau}(i+1) = o$, we have $o \in Q_B$. So $f(o) \leq i < i + 1 \leq l(o)$. By the definition of B', the history H' contains a trajectory τ_o for the state o, which stays at state o from time $f(o)$ to time $l(o)$. So we have $\tau_o(i)\tau_o(i+1) = o$, and we can choose $\tau'' := \tau_o$.

$\square$
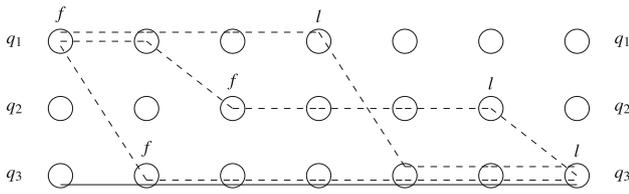
Fig. 3 History H of Fig. 2 after pruning

Example 4.19 Consider the well-structured and realizable history of Fig. 2. It leads from configuration $(4, 0, 1)$ to $(0, 0, 5)$. The bunch B from q_1 to q_3 is of size four, and so bigger than $|Q| = 3$. The set Q_B of states visited by trajectories of B is equal to Q.

Figure 3 shows for every state $q \in Q_B$ the first and last moments $f(q)$ and $l(q)$. Lemma 4.18 shows that we can replace B in H by the smaller bunch B' consisting of the trajectories $\tau_{q_1}, \tau_{q_2}, \tau_{q_3}$, drawn in dashed lines in Fig. 3. Notice that the non-horizontal 5-th step in H does not appear in the new well-structured and realizable history $H' = H - B + B'$. The history H' satisfies $C^1_{H'} = (3, 0, 1) \xrightarrow{t_3 t_1 t_3 t_4} (0, 0, 4) = C^7_{H'}$.

Using Lemma 4.18 we can now prove the Pruning Theorem for IO protocols:

Theorem 4.20 (IO Pruning) *Let $\mathscr{P} = (Q, \delta)$ be an IO protocol, let L' and L be multisets of states of $\mathscr{P}$, and let $C' \xrightarrow{*} C$ be an execution of $\mathscr{P}$ such that $L' \leq C'$ and $C \geq L$. There exist configurations D' and D such that*

$$
\begin{array}{ccc}
C' & \xrightarrow{*} & C \\
\geq & & \geq \\
D' & \xrightarrow{*} & D \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|D'| = |D| \leq |L| + |L'| + |Q|^3$.

Remark 4.21 We will often use the theorem when L' or L is empty, which is why we call them multisets of states instead of configurations.

Proof Let $L' \leq C' \xrightarrow{*} C \geq L$. By Lemma 4.12, there is a well-structured realizable history H with C' and C as initial and final configurations, respectively. Let $H_L \subseteq H$ be an arbitrary sub(multi)set of H with the multiset of final states L, and $H_{L'}$ be a sub(multi)set of H with multiset of initial states L'. Define H_0 as their union (maximum) $max(H_L, H_{L'})$, and let $H' = H - H_0$. Further, for every $p, p' \in Q$, let $H'_{p, p'}$ be the bunch of all trajectories of H' with p and p' as initial and final states, respectively. We have

$$
H' = \sum_{p, p' \in Q} H'_{p, p'}
$$

So H' is the union of $|Q|^2$ (possibly empty) bunches. Applying Lemma 4.18 to each bunch of H' with more than $|Q|$ trajectories yields a new history

$$
H'' = \sum_{p, p' \in p} H''_{p, p'}
$$

where the sum represents multiset addition, such that $|H''_{p, p'}| \leq |Q|$ for every $p, p' \in Q$, and such that the history $H'' + H_0$ is well structured and realizable.

Let D' and D be the initial and final configurations of $H'' + H_0$. We show that D' and D satisfy the required properties:

- $D' \xrightarrow{*} D$, because $H'' + H_0$ is well structured and realizable.
- $D' \geq L'$ and $D \geq L$, because $H_0 \leq H'' + H_0$.
- $|D'| \leq |L'| + |L| + |Q|^3$ because $|H'' + H_0| = \sum_{p', p} |H''_{p, p'}| + |H_0| \leq |Q|^2 \cdot |Q| + |H_{L'}| + |H_L| = |L'| + |L| + |Q|^3$.

This concludes the proof. □

Remark 4.22 A slight modification of our construction allows one to prove Theorem 4.20 (but not Lemma 4.18) with $2|Q|^2$ overhead instead of $|Q|^3$. We provide more details in the "Appendix C". However, since some results of Sect. 5 explicitly rely on Lemma 4.18, we prove Theorem 4.20 as a consequence of Lemma 4.18 for simplicity.

4.2.2 Pruning theorem for MFDO protocols

The proof of the Pruning Theorem for MFDO protocols is similar to the one for IO protocols. It follows the same sequence of steps, but with some differences.

Trajectories and histories of MFDO protocols are defined as for DO protocols. Well-structured and realizable histories also have the same definition, and Lemma 4.12 holds, with the same proof. Let us see an example:

Example 4.23 Recall the MFDO protocol $\mathscr{P}_{MFDO} = (Q, \delta)$ of Example 4.3, with $Q = \{a, b, ab\}$ and $\delta = \{t_1, t_2\}$, where $t_1 = a \xrightarrow{b} ab$ and $t_2 = b \xrightarrow{a} ab$. Figure 4 shows a graphical representation of a history H of $\mathscr{P}_{MFDO}$. It consists of five trajectories: one trajectory from a to ab, and four trajectories from b to ab, following different state sequences. The first configuration of H is $C^1_H = (1, 4, 0)$, and the fourth and last configuration is $C^4_H = (0, 0, 5)$. The history is well structured and realizable. In particular, we have

$$
C^1_H \xrightarrow{t_2 t_1 t^3_2} C^4_H .
$$
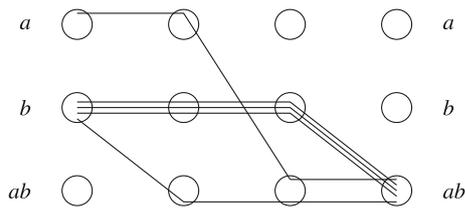
Fig. 4 Realizable history in $\mathscr{P}_{MFDO}$ of Example 4.3

For MFDO-protocols we also need the notion of the sets of states visited along a history.

Definition 4.24 Let H be a history of an MFDO protocol of length n. The set of states *visited in the first i steps of H* is $\mathscr{S}_H^i := \{\tau(j) \mid \tau \in H, j \leq i\}$. The set of states *visited by H* is denoted $\mathscr{S}_H$ and defined by $\mathscr{S}_H := \mathscr{S}_H^n$.

Example 4.25 Let H be the history of Fig. 4. We have $\mathscr{S}_H^1 = \{a, b\}$, $\mathscr{S}_H^i = \{a, b, ab\}$ for $i = 2, 3, 4$, and $\mathscr{S}_H = \{a, b, ab\}$.

Characterizing Histories. As for IO protocols, we introduce a notion of compatibility.

Definition 4.26 A history H is *compatible* with an MFDO protocol $\mathscr{P}$ if for every trajectory τ of H and for every non-horizontal step $\tau(i)\tau(i+1)$ of τ, the protocol $\mathscr{P}$ contains a transition $\tau(i) \overset{o}{\to} \tau(i+1)$ such that $o \in \mathscr{S}_H^i$, i.e., such that o has been visited by time i.

Remark 4.27 Notice the difference with IO protocols. In the IO case, compatibility requires that some agent visits o exactly at time i, a requirement captured by the condition $\tau'(i) = \tau'(i+1) = o$. In the MFDO case, compatibility requires that some agent visits state o at time i *or earlier*, captured by the condition $o \in \mathscr{S}_H^i$.

Lemma 4.28 *Let $\mathscr{P}$ be an MFDO protocol. A well-structured history is realizable in $\mathscr{P}$ iff it is compatible with $\mathscr{P}$.*

Example 4.29 The history H of Fig. 4 is well structured, realizable, and compatible with the MFDO protocol of Example 4.23.

Pruning. We prove that the construction of the Pruning Theorem for IO protocols yields the same results for MFDO protocols.

Theorem 4.30 *[(MFDO Pruning) Let $\mathscr{P} = (Q, \delta)$ be an MFDO protocol, let L' and L be multisets of states of $\mathscr{P}$, and let $C' \overset{*}{\to} C$ be an execution of $\mathscr{P}$ such that $L' \leq C'$ and $C \geq L$. There exist configurations D' and D such that*

$$
\begin{array}{ccc}
C' & \overset{*}{\longrightarrow} & C \\
\geq & & \geq \\
D' & \overset{*}{\longrightarrow} & D \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|D'| = |D| \leq |L| + |L'| + |Q|^3$.

Proof Let H be a well-structured and realizable history for the execution $L' \leq C' \overset{*}{\to} C \geq L$. Let H' be the result of pruning H using the construction of theorem 4.20. We already know that H' is well-structured and covers L' and L by its initial and final configuration. Let us show that it is compatible with $\mathscr{P}$. By the definition of compatibility (Definition 4.26), and since $H' \subseteq H$, it suffices to show that $\mathscr{S}_H^i = \mathscr{S}_{H'}^i$ holds for every i. But this follows from the fact that, by the definition of H', each state is *first* visited in H' at the same moment that it is *first* visited in H. □

Remark 4.31 For MFDO protocols we can also obtain a linear bound. Intuitively, the reason is that in order to construct the smaller history H' from H we no longer need to concatenate prefixes and suffixes of trajectories of H, but just pick an adequate subset of them. We provide more details in the "Appendix C". One can apply the improved bound to the results of Sect. 5, but some technical special cases arise in the proofs, therefore we use theorem 4.30 for simplicity and uniformity.

4.3 Shortening theorem for MFDO protocols

We introduce a new measure of the length of executions, the *aggregated length* of an execution.

Definition 4.32 Let $\mathscr{P} = (Q, \delta)$ be an MFDO protocol, and let ξ be a nonempty sequence of transitions of $\mathscr{P}$. Let $(k_1, \ldots, k_m)$ be the unique tuple of positive natural numbers such that $\xi = t_1^{k_1} t_2^{k_2} \ldots t_m^{k_m}$ and $t_i \neq t_{i+1}$ for every $i = 1, \ldots, m - 1$. We say that ξ has *aggregated length m*, and let $|\xi|_a$ denote the aggregated length of ξ.

The Shortening Theorem states that we can replace "long" executions of an MFDO protocol with shorter executions in terms of aggregated length.

Theorem 4.33 (MFDO Shortening) *Let $\mathscr{P} = (Q, \delta)$ be an MFDO protocol, and let $C \overset{*}{\to} C'$ be an execution of $\mathscr{P}$. There exists a sequence ξ such that $C \overset{\xi}{\to} C'$ and $|\xi|_a \leq |Q|^4$.*

Proof Let H be a well-structured and realizable history for the execution $C \overset{*}{\to} C'$, and let n be the length of H. We have $\mathscr{S}_H^1 \subseteq \mathscr{S}_H^2 \subseteq \cdots \subseteq \mathscr{S}_H^n$. Since H is well structured, for

every $1 \leq i \leq n-1$ either $\mathscr{S}_H^i = \mathscr{S}_H^{i+1}$, or $\mathscr{S}_H^{i+1}$ contains exactly one more state than $\mathscr{S}_H^i$.

Let $T_0 = 1$, let $T_1, T_2, \ldots, T_{k-1}$ be the time moments immediately before the set of visited states increases, that is, the set of indices satisfying $\mathscr{S}_H^{T_i} \subset \mathscr{S}_H^{T_i+1}$, and let $T_k = n$. Observe that $k \leq |Q|$.

For every $0 \leq j \leq k$, let H_j be the initial segment of H of length T_j. We prove by induction over j that there is a well-formed and realizable history H_j' satisfying the following conditions:

(i) $\mathscr{S}_{H_j} = \mathscr{S}_{H_j'}$, that is, H_j and H_j' visit the same states;
(ii) there exists a bijection b between the trajectories of H and H_j' such that the T_j-th state of τ and the last state of $b(\tau)$ coincide; and
(iii) H_j' has length at most $j(|Q|(|Q|-1)^2 + 1)$.

The theorem then follows from the fact that, since H_k' has length at most $|Q|(|Q|(|Q|-1)^2 + 1) < |Q|^4$ and is realizable, it can be realized by an execution of aggregated length at most $|Q|^4$.

The base case of the induction is $j = 0$. Since $T_0 = 1$, we can set $H_0' \overset{\text{def}}{=} H_0$. For the induction step, assume we have already constructed H_j' satisfying conditions (i)-(iii). We construct H_{j+1}' by extending each trajectory of H_j'. We illustrate how to perform the extensions on the example of Fig. 5.

Example 4.34 Figure 5 shows at the top the fragment of a history H between times T_j and $T_{j+1} = T_j + 6$. The history H consists of three trajectories τ_1, τ_2, τ_3. We assume that $\mathscr{S}_H^{T_j} = \{q_1, q_2, q_4, q_5\}$, i.e., up to time T_j the three trajectories have visited all states but q_3. We then have $\mathscr{S}_H^{T_{j+1}} = \{q_1, \ldots, q_5\}$.

Let τ be an arbitrary trajectory of H, and for every $0 \leq i \leq j$ let τ_i be the prefix of τ of length T_i. By condition (ii), there exists a bijection b that assigns to τ a trajectory $\tau_j' \overset{\text{def}}{=} b(\tau)$ of H_j'. Further, τ_j and τ_j' have the same initial and final states. We describe an algorithm that extends the history H_j' to H_{j+1}' with the same final configuration as H_{j+1}.

The algorithm initializes a variable $\tilde{\tau} := \tau_j'$ for each trajectory $\tau \in H$. In a first step, the algorithm sets $\tilde{\tau} := \tau_j' \tau(T_j + 1)$. In our example, the three trajectories of H_j' are extended as shown in the bottom part of Fig. 5.

Let $H_j'^+$ be the history obtained after applying this first step. It is easy to see that, since H_j and H_j' satisfy conditions (i)-(iii), so do $H_j'^+$ and the prefix of H of length $T_j + 1$.

The algorithm now proceeds to execute a loop. Let $\mathscr{B}[q, q', j]$ be the bunch of trajectories $\tau \in H$ such that $\tau(T_j + 1) = q$ and $\tau(T_{j+1}) = q'$, and let E be an arbitrary but fixed enumeration of the pairs (q, q') of states such
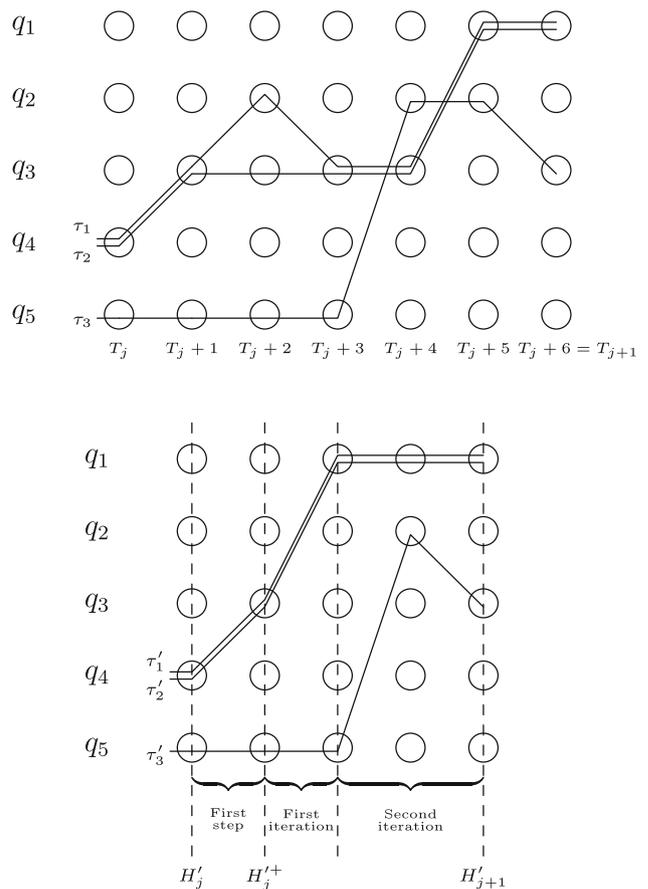
Fig. 5 Illustration of the proof of Theorem 4.33

that $\mathscr{B}[q, q', j]$ is nonempty. The algorithm loops through every pair $(q, q') \in E$, extending each $\tilde{\tau}$ in a way to be described later. After the loop, the algorithm sets τ_{j+1}' to the final value of $\tilde{\tau}$. Observe that each variable $\tilde{\tau}$ gets extended $|Q|(|Q|-1)$ times.

Example 4.35 The history at the top of Fig. 5 has two nonempty bunches, namely $\mathscr{B}[q_3, q_1, j] = \{\tau_1, \tau_2\}$, and $\mathscr{B}[q_5, q_3, j] = \{\tau_3\}$. In what follows we assume that $E = (q_3, q_1)(q_5, q_3)$.

Before describing the body of the loop for a given pair (q, q') of states, we need to state and prove a claim.

Claim. For every $(q, q') \in E$ there exists a sequence $sh(q, q')$ (where sh stands for "short") leading from q to q' and satisfying the following properties:

- each state in $sh(q, q')$ is in $\mathscr{S}_H^{T_j}$;
- each step in $sh(q, q')$ corresponds to a protocol transition observing some state in $\mathscr{S}_H^{T_j}$;
- $sh(q, q')$ has length $|Q|$.

To prove the claim, observe first that, by the definition of E, there exists at least one trajectory $\tau \in \mathscr{B}[q, q', j]$. Pick any

such trajectory. The steps of τ between times T_j and T_{j+1} form a path in the oriented graph of transitions of the protocol enabled by the set $\mathscr{S}_H^{T_j}$ of visited states. Let $sh(q, q')$ be the result of removing all cycles from this path. By construction only states from $\mathscr{S}_H^{T_j}$ are used, and only transitions enabled by observing states from the same set are performed. Clearly, we have $|sh(q, q')| \leq |Q|$.

Example 4.36 In Fig. 5, the segment of τ_1 between times $T_j + 1$ and $T_j + 6 = T_{j+1}$ is the sequence $q_3 q_2 q_3 q_3 q_1 q_1$ of states. The trajectory $sh(q_3, q_1)$ obtained from τ_1 by "cutting out the cycles" is $q_3 q_1$.

For each pair $(q, q') \in E$, the algorithm picks an arbitrary trajectory of $\mathscr{B}[q, q', j]$, constructs the shortened trajectory $sh(q, q')$, and for every trajectory $\tau \in H$ it extends the current trajectory $\tilde{\tau}$ as follows:

(1) If $\tau \in \mathscr{B}[q, q', j]$, then the algorithm extends $\tilde{\tau}$ with $sh(q, q')$ (more precisely, with the result of dropping the first state in $sh(q, q')$).
(2) Otherwise, the algorithm extends $\tilde{\tau}$ by replicating its final state $|sh(q, q')| - 1$ times. In other words, it extends $\tilde{\tau}$ with $|sh(q, q')| - 1$ horizontal steps.

Observe that after each iteration of the loop all trajectories have the same length. The histories consisting of all the trajectories after the same iteration are well-formed (all added non-horizontal steps are copies of the same one) and realizable (because of the second condition in the claim). In particular, after the last iteration of the algorithm, we obtain a wellformed and realizable history.

Example 4.37 Recall that $E = (q_3, q_1)(q_5, q_3)$. Assume that for $(q, q') := (q_3, q_1)$ the algorithm picks τ_1 (it could also pick τ_2). The algorithm sets $sh(q_3, q_1) := q_3 q_1$, and in the first iteration of the loop it extends $\tilde{\tau}_1$ and $\tilde{\tau}_2$ with q_1, and $\tilde{\tau}_3$ with q_5 (see the bottom of Fig. 5).

For $(q, q') := (q_5, q_3)$ the algorithm necessarily picks τ_3 and sets $sh(q_5, q_3) := q_5 q_2 q_3$. In the second iteration of the loop $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are extended with horizontal steps $q_1 q_1$, and $\tilde{\tau}_3$ with $q_5 q_2 q_3$.

Let us show that the realizable history H'_{j+1} constructed by the algorithm satisfies properties (i)-(iii). Property (i) follows directly from the fact that the algorithm only extends the trajectories of H'_j with steps taken from the trajectories of H_{j+1}. For property (ii), we observe that for every pair of states (q, q'), the bunches $\mathscr{B}[q, q', j+1]$ and $\mathscr{B}'[q, q', j+1]$ (defined as $\mathscr{B}[q, q', j + 1]$, but for the history H'_{j+1}) have the same size. So the bijection can be obtained as the union of bijections between these bunches. Finally, let us prove property (iii). Since the sequences $sh(q, q')$ have length at most $|Q|$, they consist of at most $|Q| - 1$ steps. Since

$|E| \leq |Q|(|Q| - 1)$, during the loop every trajectory gets extended at most $|Q|(|Q| - 1)$ times. So the trajectories of H'_{j+1} have at most $|Q|(|Q| - 1)^2 + 1$ more steps than the trajectories of H'_j, and at most $(j + 1)(|Q|(|Q| - 1)^2 + 1)$ steps. Since H'_{j+1} is well structured, its aggregated length is bounded by the number of steps of its trajectories, and we are done. □

Remark 4.38 An optimised version of the construction allows to obtain a *quadratic* bound for the aggregated length of the history after shortening. We provide a rough outline in the "Appendix C" in case the reader is interested in carrying out such optimisation.

4.4 Pruning and shortening theorems for DO protocols

In Sect. 4.1 we showed that reachability in MFDO and zero-message reachability in DO are essentially equivalent notions. Using this correspondence, we derive Pruning and Shortening Theorems for DO protocols from the corresponding results for MFDO protocols. Recall that we denote by $\mathscr{Z}$ the set of zero-message configurations of a DO protocol, and that a configuration $Z \in \mathscr{Z}$ can be seen both as a DO configuration and (by abuse of notation) as an MFDO configuration.

Corollary 4.39 (DO Pruning) *let $Z, Z' \in \mathscr{Z}$ be zero-message configurations of $\mathscr{P}$, let L' and L be multisets of states of $\mathscr{P}$, and let $Z' \xrightarrow{*} Z$ be an execution of $\mathscr{P}$ such that $L' \leq Z'$ and $Z \geq L$. There exist zero-message configurations Y' and Y such that*

$$
\begin{array}{ccc}
Z' & \xrightarrow{\ *\ } & Z \\
\geq & & \geq \\
Y' & \xrightarrow{\ *\ } & Y \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|Y'| = |Y| \leq |L| + |L'| + |Q|^3$.

Proof By Lemma 4.4, if $Z'' \xrightarrow{*} Z' \geq Z$ in DO protocol $\mathscr{P}$, then $Z'' \xrightarrow{*} Z' \geq Z$ in the corresponding MFDO protocol (see Definition 4.2). By applying Theorem 4.30 to $Z'' \xrightarrow{*} Z' \geq Z$ in the MFDO protocol, there exist Y'' and Y' such that

$$
\begin{array}{ccc}
Z' & \xrightarrow{\ *\ } & Z \\
\geq & & \geq \\
Y' & \xrightarrow{\ *\ } & Y \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|Y| \leq |L| + |L'| + |Q|^3$. By Lemma 4.4, $Y' \xrightarrow{*} Y$ is also valid in our DO protocol with $Y', Y \in \mathscr{Z}$. □

Corollary 4.40 (DO Shortening) *Let $\mathscr{P} = (Q, M, \delta_s, \delta_r)$ be a DO protocol, let Z and Z' be zero-message configurations of $\mathscr{P}$, and let $Z \xrightarrow{*} Z'$ be an execution of $\mathscr{P}$. There exists a sequence ξ such that $Z \xrightarrow{\xi} Z'$ and $|\xi|_a \leq |Q|^4 + |Q|$.*

Proof By Lemma 4.4, if $Z \xrightarrow{*} Z'$ in DO protocol $\mathscr{P}$, then $Z \xrightarrow{*} Z'$ in the corresponding MFDO protocol. By applying Theorem 4.33 to $Z \xrightarrow{\xi} Z'$, there exists ξ such that $Z \xrightarrow{\xi} Z'$ in the corresponding MFDO protocol and $|\xi|_a \leq |Q|^4$.

Following the construction of a DO sequence from an MFDO sequence described in the proof of Lemma 4.4, we show that we can construct a sequence ξ' in $\mathscr{P}$ such that $Z \xrightarrow{\xi'} Z'$ and $|\xi'|_a \leq |\xi|_a + |M|$. In the first step of the construction, we replace each transition of ξ by a corresponding receive transition in $\mathscr{P}$. Then for each message $m \in M$ that appears in these receive transitions, we add a sequence of identical send transitions $q_m \xrightarrow{m+} q_m$ the first time that state q_m that can send m is reached. Thus the constructed DO sequence ξ' has an aggregated length of at most $|\xi|_a + |M|$, and since $|\xi|_a \leq |Q|^4$ and $|M| \leq |Q|$ we get our result. □

5 Set reachability in observation models: the closure theorems

We introduce counting sets, a class of possibly infinite sets of configurations with a finite representation in terms of so-called counting constraints. We then prove the Closure Theorems for IO and MFDO, stating that the sets of predecessors and successors of a counting set are also counting sets. Further, we show that if the original counting set has a representation with "small" cubes, then the sets of predecessors and successors also have succinct representations.

Counting constraints and counting sets. Let $\mathscr{P}$ be an IO or MFDO protocol with set of states Q. A set $\mathscr{C}$ of configurations of $\mathscr{P}$ is a *cube* if there exist mappings $L: Q \to \mathbb{N}$ and $U: Q \to \mathbb{N} \cup \{\infty\}$ such that $C \in \mathscr{C}$ iff $L \leq C \leq U$. (Observe that the components of U may be equal to ∞, and that both L and U are unique.) We call L and U the *lower bound* and *upper bound* of $\mathscr{C}$, respectively, and call the pair (L, U) the *representation* of $\mathscr{C}$. Given two mappings $L: Q \to \mathbb{N}$ and $U: Q \to \mathbb{N} \cup \{\infty\}$, the cube represented by (L, U) is denoted $[\![L, U]\!]$.

A *counting constraint* is a finite set $\Gamma = \{(L_1, U_1), \ldots, (L_n, U_n)\}$ of representations of cubes. We say that Γ *represents* the set $[\![\Gamma]\!] \stackrel{\text{def}}{=} [\![L_1, U_1]\!] \cup \cdots \cup [\![L_n, U_n]\!]$. A set $\mathscr{S}$ is a *counting set* if $\mathscr{S} = [\![\Gamma]\!]$ for some counting constraint Γ.

Observe that, while a cube has a unique representation, the same counting set may be represented by more than one counting constraint. For example, consider a protocol with just one state. The counting constraints $\{(1, 3), (2, 4)\}$, $\{(1, 2), (3, 4)\}$, and $\{(1, 4)\}$ define the same counting set, namely the cube $[\![1, 4]\!]$.

Measures of counting constraints. We introduce two measures of the size of a counting constraint. Let $\mathscr{C}$ be a cube with representation (L, U). The *l-norm* of $\mathscr{C}$, denoted $\|\mathscr{C}\|_l$, is the sum of the components of L. The *u-norm* of $\mathscr{C}$, denoted $\|\mathscr{C}\|_u$, is the sum of the components of U that are not equal to ∞, if there are any, and 0 otherwise.

The *l-norm* and *u-norm* of a counting constraint $\Gamma = \{\mathscr{C}_1, \ldots, \mathscr{C}_m\}$ are defined by

$$\|\Gamma\|_l \stackrel{\text{def}}{=} \max_{i \in [1,m]} \{\|\mathscr{C}_i\|_l\} \text{ and } \|\Gamma\|_u \stackrel{\text{def}}{=} \max_{i \in [1,m]} \{\|\mathscr{C}_i\|_u\}.$$

The *l-norm* (respectively *u-norm*) of a counting set $\mathscr{S}$ is the smallest *l-norm* (respectively *u-norm*) of a counting constraint representing $\mathscr{S}$, that is

$$\|\mathscr{S}\|_l \stackrel{\text{def}}{=} \min_{\mathscr{S} = [\![\Gamma]\!]} \{\|\Gamma\|_l\} \text{ and } \|\mathscr{S}\|_u \stackrel{\text{def}}{=} \min_{\mathscr{S} = [\![\Gamma]\!]} \{\|\Gamma\|_u\}.$$

Example 5.1 Cube $\mathscr{C}$ with representation $(1, 4)$ has *l-norm* 1 and *u-norm* 4. The counting constraint $\Gamma = \{(2, 4), (3, 5)\}$ has *l-norm* 3 and *u-norm* 5.

The following proposition, whose proof is given in the "Appendix D", shows that a Boolean combination of counting sets is still a counting set and bounds the size of the counting constraints representing such combinations.

Proposition 5.2 ([33], Proposition 5) *Let Γ_1, Γ_2 be counting constraints.*

- *There exists a counting constraint Γ with $[\![\Gamma]\!] = [\![\Gamma_1]\!] \cup [\![\Gamma_2]\!]$ such that $\|\Gamma\|_u \leq \max\{\|\Gamma_1\|_u, \|\Gamma_2\|_u\}$ and $\|\Gamma\|_l \leq \max\{\|\Gamma_1\|_l, \|\Gamma_2\|_l\}$.*
- *There exists a counting constraint Γ with $[\![\Gamma]\!] = [\![\Gamma_1]\!] \cap [\![\Gamma_2]\!]$ such that $\|\Gamma\|_u \leq \|\Gamma_1\|_u + \|\Gamma_2\|_u$ and $\|\Gamma\|_l \leq \|\Gamma_1\|_l + \|\Gamma_2\|_l$.*
- *There exists a counting constraint Γ with $[\![\Gamma]\!] = \mathbb{N}^n \setminus [\![\Gamma_1]\!]$ such that $\|\Gamma\|_u \leq n\|\Gamma_1\|_l$ and $\|\Gamma\|_l \leq n\|\Gamma_1\|_u + n$.*

Loosely speaking, Proposition 5.2 shows that applying boolean operations to counting sets does not increase much the size of its representation. Now we prove the Closure Theorem, showing that this is also the case for the operations of computing the set of successors or predecessors of a counting set.

Closure Theorem for IO protocols. The Closure Theorem for IO protocols is an easy consequence of the following lemma:

Lemma 5.3 *Let $\mathscr{P}$ be an IO protocol with state set Q and let $\mathscr{C} \subseteq Pop(Q)$ be a cube. For all $C' \in pre^*(\mathscr{C})$, there exists a cube $\mathscr{C}'$ such that*

1. *$C' \in \mathscr{C}' \subseteq pre^*(\mathscr{C})$, and*
2. *$\|\mathscr{C}'\|_l \leq \|\mathscr{C}\|_l + |Q|^3$ and $\|\mathscr{C}'\|_u \leq \|\mathscr{C}\|_u$.*

Proof Let L, U be mappings such that $\mathscr{C} = [\![L, U]\!]$. Let C' be a configuration of $pre^*(\mathscr{C})$. There exists a configuration $C \in \mathscr{C}$ such that $C' \longrightarrow C$, and $C \geq L$. By the Pruning Theorem there exist configurations D' and D such that

$$
\begin{array}{ccc}
C' & \xrightarrow{\;*\;} & C \geq L \\
\geq & & \geq \\
D' & \xrightarrow{\;*\;} & D \geq L
\end{array}
$$

and $|D'| \leq |L| + |Q|^3$. Since $C \in \mathscr{C}$, we have $U \geq C \geq D \geq L$. So $D \in \mathscr{C}$, and therefore $D' \in pre^*(\mathscr{C})$.

We find a cube $\mathscr{C}'$ satisfying conditions (1) and (2). For this, we choose appropriate lower and upper bounds L', U', and set $\mathscr{C}' = [\![L', U']\!]$. First, we set $L' \overset{\text{def}}{=} D'$. For the definition of U', we use the tools of the Pruning Theorem section, in which the movements of the agents are de-anonymized into trajectories. Let $H_{C'}$ be a well-structured realizable history of $\mathscr{P}$ leading from C' to C, and let q be a state of Q. We define $U'(q)$ as follows:

(i) If some trajectory of $H_{C'}$ starting at q leads to a state r such that $U(r) = \infty$, then set $U'(q) \overset{\text{def}}{=} \infty$.
(ii) If every trajectory of $H_{C'}$ starting at q leads to states r such that $U(r) < \infty$, then set $U'(q) = C'(q)$.

We prove that $\mathscr{C}' \overset{\text{def}}{=} [\![L', U']\!]$ satisfies the conditions of the lemma.

Property 1: $C' \in \mathscr{C}' \subseteq pre^*(\mathscr{C})$.

Since $\mathscr{C}' \overset{\text{def}}{=} [\![L', U']\!]$, we first prove $L' \leq C' \leq U'$. The inequality $L' \leq C'$ follows from $C' \geq D'$ (see the diagram above) and $L' \overset{\text{def}}{=} D'$. Let us now show that $C'(q) \leq U'(q)$ holds for every state q. If $U'(q) = \infty$ there is nothing to show. If $U'(q)$ is finite, i.e., if *Case 2* above holds, then $U'(q) = C'(q)$, and we are done.

It remains to prove $[\![L', U']\!] \subseteq pre^*(\mathscr{C})$, which requires more effort. We show that for every configuration $R' \in [\![L', U']\!]$ there exists a history $H_{R'}$ leading from R' to a configuration $R \in \mathscr{C}$, i.e., to a configuration R satisfying $L \leq R \leq U$. Since $R' \in [\![L', U']\!]$ and $L' \overset{\text{def}}{=} D'$, we have $R' \geq D'$. So we construct $H_{R'}$ by adding trajectories to $H_{D'}$: Since $H_{D'}$ leads to D, this guarantees that $H_{R'}$ leads to a configuration R such that $R \geq D \geq L$ (see Fig. 6). Further,

$$
\begin{array}{ccccc}
U' \geq & C' & \xrightarrow{\quad\quad\quad H'_C \quad\quad\quad} & C & \leq U \\
 & \geq & & \geq & \\
 & R' & \xrightarrow{\quad\quad\quad H'_R \quad\quad\quad} & R & \\
 & \geq & & \geq & \\
L' \leq & D' & \xrightarrow{\quad\quad\quad H'_D \quad\quad\quad} & D & \geq L
\end{array}
$$
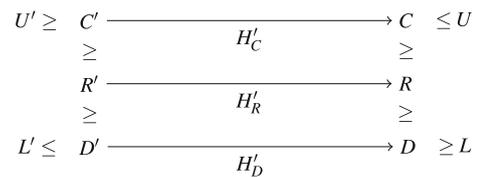
Fig. 6 Construction of the proof of Lemma 5.3

to ensure that $H_{R'}$ starts at R', for every $q \in Q$ we add to $H_{D'}$ exactly $(R'(q) - D'(q))$ trajectories starting at q. It remains to choose these trajectories in such a way that $R \leq U$ holds.

We add trajectories so that $R(q) \leq C(q)$ holds, which, since $C(q) \leq U(q)$ (see Fig. 6), ensures $R(q) \leq U(q)$.

We add trajectories to $H_{D'}$ by replication, i.e., we only add copies of trajectories already present in $H_{D'}$. Recall that for every state $q \in Q$ we have to add $(R'(q) - D'(q))$ trajectories starting at q. We decide which trajectories to add according to two cases, very similar to the cases (i) and (ii) above:

(i′) $H_{D'}$ contains a trajectory τ leading from q to a state r such that $U(r) = \infty$.
In this case we add $(R'(q) - D'(q))$ copies of τ.
(ii′) Every trajectory of $H_{D'}$ leading from q to some state r satisfies $U(r) < \infty$.
In this case, by the definition of U' (see (ii) above), we have $U'(q) = C'(q)$. Since $R' \leq U'$ by hypothesis, we get $D'(q) \leq R'(q) \leq C'(q)$, and so $(R'(q) - D'(q)) \leq (C'(q) - D'(q))$, i.e., we need to add at most $C'(q) - D'(q)$ trajectories.

For each state $r \in Q$, let $n_{C'}[q, r]$ and $n_{D'}[q, r]$ be the sizes of the bunches of trajectories of $H_{C'}$ and $H_{D'}$ leading from q to r, respectively. By this definition, and the definition of the pruning operation, we have

(a) $C'(q) - D'(q) = \sum_{r \in Q} (n_{C'}[q, r] - n_{D'}[q, r])$.
(b) For every $r \in Q$: $n_{C'}[q, r] \geq n_{D'}[q, r]$, and
(c) For every $r \in Q$: $n_{C'}[q, r] \geq 1$ implies $n_{D'}[q, r] \geq 1$.

We add trajectories as follows: we loop through the states r such that $n_{C'}[q, r] \geq 1$. We take any trajectory of $H_{D'}$ leading from q to r (which exists by (c)), and replicate it $n_{C'}[q, r] - n_{D'}[q, r]$ times or less, until the quota of $R'(q) - D'(q)$ trajectories has been reached. The quota is eventually reached by (a).

We claim that this procedure produces a history $H_{R'}$ such that $n_{R'}[q, r] \leq n_{C'}[q, r]$ for every $q, r \in Q$ such that $U(r) < \infty$. Indeed, fix r such that $U(r) < \infty$. If q satisfies (i′), then no trajectory from q to r is replicated, i.e., $n_{R'}[q, r] = n_{C'}[q, r]$. If q satisfies (ii′), then $n_{R'}[q, r] \leq n_{C'}[q, r]$. By the claim, $R(r) \leq C(r)$ for every state r such

that $U(r) < \infty$. Since $C \leq U$, we have $R \leq U$, and we are done.

Property 2: $\|\mathscr{C}'\|_l \leq \|\mathscr{C}\|_l + |Q|^3$ and $\|\mathscr{C}'\|_u \leq \|\mathscr{C}\|_u$.

For the l-norm, recall that $L' \stackrel{\text{def}}{=} D'$. Since $H_{D'}$ leads from D' to D, we have $|L'| = |D'| = |D|$. By the Pruning Theorem

$$\|(L', U')\|_l \leq |L| + |Q|^3 = \|(L, U)\|_l + |Q|^3 .$$

For the u-norm, notice that by (i) and (ii), every trajectory of $H_{C'}$ starting at a state q satisfying $U'(q) < \infty$ leads to a state r satsfying $U(r) < \infty$. Using this observation, we get:

$$
\begin{aligned}
&\|(L', U')\|_u \\
&= \sum_{q | U'(q) < \infty} U'(q) \\
&= \sum_{q \in Q | U'(q) < \infty} C'(q) && \text{(Def. of } U') \\
&= \sum_{q \in Q | U'(q) < \infty} \sum_{r \in Q} n_{C'}[q, r] && \text{(Def. of } n_{C'}[q, r]) \\
&\leq \sum_{q \in Q} \sum_{r \in Q | U(r) < \infty} n_{C'}[q, r] && \text{(Observation)} \\
&= \sum_{r \in Q | U(r) < \infty} \sum_{q \in Q} n_{C'}[q, r] && \text{(Algebra)} \\
&= \sum_{r \in Q | U(r) < \infty} C(r) && (H_{C'} \text{ leads to } C) \\
&\leq \sum_{r \in Q | U(r) < \infty} U(r) && (C \leq U) \\
&= \|(L, U)\|_u
\end{aligned}
$$

$\square$

Theorem 5.4 (IO Closure) *Let $\mathscr{P}$ be an IO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set of configurations of $\mathscr{P}$ represented by a counting constraint Γ. Then $pre^*(\mathscr{S})$ is also a counting set, and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre^*(\mathscr{S})$ and*

$$\|\Gamma'\|_u \leq \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \leq \|\Gamma\|_l + |Q|^3$$

The same holds for $post^*$.

Proof By the definition of a counting set, there exist cubes $\mathscr{C}_1, \ldots, \mathscr{C}_k$ such that $\mathscr{S} = \bigcup_{i=1}^{k} \mathscr{C}_i$, and therefore $pre^*(\mathscr{S}) = \bigcup_{i=1}^{k} pre^*(\mathscr{C}_i)$ By Lemma 5.3, for every configuration $C' \in pre^*(\mathscr{S})$ there is a cube $\mathscr{C}'$ such that $C' \in \mathscr{C}'$, $\mathscr{C}' \subseteq pre^*(\mathscr{S})$, and $\|\mathscr{C}'\|_l \leq \|\mathscr{C}_i\|_l + |Q|^3$, and $\|\mathscr{C}'\|_u \leq \|\mathscr{C}_i\|_u$ for some $1 \leq i \leq k$. So $pre^*(\mathscr{S}) = \bigcup_{C' \in pre^*(\mathscr{S})} \mathscr{C}'$. Since there are only finitely many cubes $\mathscr{C}'$ with a given bound on their lower and upper norms, $pre^*(\mathscr{S}) = \bigcup_{i=1}^{k'} \mathscr{C}'_i$ for some k', and so a counting set.

Let Γ and Γ' be the counting constraint defined as the set of the representations of $\{\mathscr{C}_1, \ldots, \mathscr{C}_k\}$ and $\{\mathscr{C}'_1, \ldots, \mathscr{C}'_{k'}\}$,

respectively. By the definition of the norm of a counting constraint, we have $\|\mathscr{C}'_i\|_l \leq \|\Gamma\|_l + |Q|^3$ and $\|\mathscr{C}'_i\|_u \leq \|\Gamma\|_u$ for every $1 \leq i \leq k'$. So $\|\Gamma'\|_u \leq \|\Gamma\|_u$ and $\|\Gamma'\|_l \leq \|\Gamma\|_l + |Q|^3$.

The result for $post^*(\mathscr{S})$ can be proven in the exact same way, as the pruning theorem is symmetric. $\square$

Closure Theorem for MFDO protocols. The Closure Theorem for MFDO protocols can be proved in the same way as for IO protocols.

Lemma 5.5 *Let $\mathscr{C}$ be a cube of an MFDO protocol $\mathscr{P}$ of with state set Q. For all $C' \in pre^*(\mathscr{C})$, there exists a cube $\mathscr{C}'$ such that*

1. $C' \in \mathscr{C}' \subseteq pre^*(\mathscr{C})$, and
2. $\|\mathscr{C}'\|_l \leq \|\mathscr{C}\|_l + |Q|^3$ and $\|\mathscr{C}'\|_u \leq \|\mathscr{C}\|_u$.

Theorem 5.6 (MFDO Closure) *Let $\mathscr{P}$ be an MFDO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set defined by a counting constraint Γ. Then $pre^*(\mathscr{S})$ is also a counting set and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre^*(\mathscr{S})$, and*

$$\|\Gamma'\|_u \leq \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \leq \|\Gamma\|_l + |Q|^3$$

The same holds for $post^*$.

The Closure Theorem for MFDO protocols yields a Closure Theorem for DO protocols. In DO protocols, counting constraints are still defined as bounds associated to elements of Q, and thus they define counting sets which are sets of zero-message configurations. To express the following result we need operators on zero-message configurations.

Zero-message predecessors and successors. Let $\mathscr{P}$ be a DO protocol, and let $\mathscr{Z}$ be the set of its zero-message configurations. For every set $\mathscr{M} \subseteq \mathscr{Z}$, we respectively define the set of zero-message predecessors and the set of zero-message successors as

$$
\begin{aligned}
pre_z^*(\mathscr{M}) &= pre^*(\mathscr{M}) \cap \mathscr{Z} \\
post_z^*(\mathscr{M}) &= post^*(\mathscr{M}) \cap \mathscr{Z}.
\end{aligned}
$$

Corollary 5.7 (DO Closure) *Let $\mathscr{P}$ be a DO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set of zero-message configurations defined by a counting constraint Γ. Then $pre_z^*(\mathscr{S})$ is also a counting set and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre_z^*(\mathscr{S})$, and*

$$\|\Gamma'\|_u \leq \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \leq \|\Gamma\|_l + |Q|^3$$

The same holds for $post_z^*$.

6 Upper bounds for observation models

We use the Pruning, Shortening, and Closure Theorems proved in the past sections to prove that the correctness problem for IO protocols is in PSPACE, and that the correctness problem for DO protocols is in Π_2^p. These upper bounds match the lower bounds proved in Theorem 3.3 and Theorem 3.5.

For the following results, we need the predicates φ we consider to be describable by counting constraints. A predicate $\varphi: \mathbb{N}^k \to \{0, 1\}$ is *describable by counting constraint* if there is a counting constraint Γ such that $\varphi(\mathbf{v}) = 1$ iff $\mathbf{v}$ satisfies Γ. If φ is a predicate over $\text{Pop}(\Sigma)$ that is describable by counting constraint, k is the dimension of the symbol alphabet Σ, and populations $D \in \text{Pop}(\Sigma)$ are seen as vectors $\mathbf{v} \in \mathbb{N}^k$. Fortunately, as mentioned in Sect. 2.5, Angluin et al. show in [7] that IO protocols compute exactly the predicates representable by counting constraints, and DO protocols compute a subset of these.

Lemma 6.1 *Let $\mathscr{P}$ be an IO or DO protocol with Q its set of states, and let φ be a predicate describable by a counting constraint Γ. Then $I_b|_Q$ and $Con_b|_Q$, the restrictions of I_b and Con_b to their components over Q, are describable by counting constraints for $b \in \{0, 1\}$. Moreover, the norms of these counting constraints are bounded in the norms of the counting constraint associated to φ and in $n = |Q|$:*

$$\|I_0|_Q\|_l \leq n\|\Gamma\|_u + n \qquad \|I_0|_Q\|_u \leq n\|\Gamma\|_l$$
$$\|I_1|_Q\|_l = \|\Gamma\|_l \qquad \|I_1|_Q\|_u = \|\Gamma\|_u$$
$$\|C_0|_Q\|_l = \|C_1|_Q\|_l = 0 \qquad \|C_0|_Q\|_u = \|C_1|_Q\|_u = 0$$

Proof Let $\mathscr{P}$ be an IO or DO protocol over an alphabet Σ with initial state mapping ι, and Q its set of states. Predicate φ is a predicate describable by counting constraint Γ which is over $\text{Pop}(\Sigma)$, i.e. the bounds of the cubes of Γ are mappings Σ to $\mathbb{N}$. We extend this to a counting constraint over agent configurations of $\mathscr{P}$ by having the bounds of the cubes be mappings from Q to $\mathbb{N}$: states of $\iota(\Sigma)$ map to $\mathbb{N}$ as before, and states to which no input symbols are mapped by ι have upper and lower bounds equal to 0. Without loss of generality we assume that each symbol of Σ is mapped to one state, i.e. ι is injective. Notice that the norms of this extension of Γ are still equal to $\|\Gamma\|_l$ and $\|\Gamma\|_u$. We abusively also note this extension Γ.

Recall that $I_b = I(\varphi^{-1}(b))$ in the generalized protocol notation. In the IO or DO notation,

$$I_b|_Q = \{\iota(D) | \exists D \in \text{Pop}(\Sigma) . \varphi(D) = b\}$$

where $\iota(D)$ is the agent configuration $\sum_{\sigma \in \Sigma} D(\sigma)\iota(\sigma)$. Then $I_b|_Q$ is describable by the counting constraint Γ for

$b = 1$ and by the counting constraint corresponding to $1 - \varphi$ for $b = 0$. The bounds on the norm of $I_0|_Q$ are a consequence of Proposition 5.2.

The set $Con_b|_Q$ is given by the cube of upper bound equal to 0 on all states q with output $1 - b$ and ∞ otherwise, and the lower bound equal to 0 everywhere. This cube is of upper and lower norm 0. $\qquad\square$

Remark 6.2 Initial configurations are zero-message in all protocol models, so $I_b|_Q$ is exactly I_b. For $\mathscr{P}$ an IO protocol, $Con_b|_Q$ is exactly Con_b for $b \in \{0, 1\}$.

6.1 Correctness of IO protocols is in PSPACE

Since IO protocols are well-behaved protocols (by Lemma 2.9), we can apply the reformulation of correctness as a reachability problem of Proposition 2.12. An IO protocol $\mathscr{P}$ is correct for a predicate φ if and only if

$$post^*(I_b) \subseteq pre^*(St_b) \tag{1}$$

for $b \in \{0, 1\}$. By Theorem 5.4, Proposition 5.2 and Lemma 6.1 above, St_b is a counting set of norms $\|St_b\|_l \leq n \in O(n)$, $\|St_b\|_u \leq n^3 + n^2 \in O(n^3)$, with $n \stackrel{\text{def}}{=} |Q|$.

Thus Equation (1) formulates the problem of correctness of an IO protocol as a predicate with boolean and reachability operators over counting sets. We use the results of Sect. 5 to show that we only need to examine "small" configurations to verify such predicates, thus yielding a PSPACE algorithm for checking correctness. We start by giving a lemma for general predicates with boolean and reachability operators over counting sets, then apply it to the predicate for correctness.

Lemma 6.3 *Let $\mathscr{S}_1$ and $\mathscr{S}_2$ be two functions that take as arguments an IO protocol $\mathscr{P}$ and a counting constraint X, and return counting sets $\mathscr{S}_1(\mathscr{P}, X)$ and $\mathscr{S}_2(\mathscr{P}, X)$ respectively.*

Assume that $\mathscr{S}_1(\mathscr{P}, X)$ and $\mathscr{S}_2(\mathscr{P}, X)$ have norms at most exponential in the size of the $(\mathscr{P}, X)$, as well as PSPACE-decidable membership (given input $(C, \mathscr{P}, X)$, decide whether $C \in \mathscr{S}_i(\mathscr{P}, X)$).

Then the same is true about the counting sets $\mathscr{S}_1(\mathscr{P}, X) \cap \mathscr{S}_2(\mathscr{P}, X)$, $\mathscr{S}_1(\mathscr{P}, X) \cup \mathscr{S}_2(\mathscr{P}, X)$, $\overline{\mathscr{S}_1(\mathscr{P}, X)}$, $pre^*(\mathscr{S}_1(\mathscr{P}, X))$, and $post^*(\mathscr{S}_1(\mathscr{P}, X))$. Furthermore, given $\mathscr{P}$ and X, the emptiness problem for these sets is in PSPACE.

Proof The exponential bounds for the norms follow immediately from Proposition 5.2 and Theorem 5.4. The membership complexity for union, intersection and complement is easy to see. Without loss of generality it suffices to prove that membership in $post^*(\mathscr{S}_1(\mathscr{P}, X))$ is in PSPACE.

By Savitch's Theorem NPSPACE=PSPACE, so we provide a nondeterministic algorithm. Given $(C, \mathscr{P}, X)$, we want to decide whether $C \in post^*(\mathscr{S}_1(\mathscr{P}, X))$. The algorithm first

guesses a configuration $C_0 \in \mathscr{S}_1(\mathscr{P}, X)$ of the same size as C, verifies that C_0 belongs to $\mathscr{S}_1(\mathscr{P}, X)$, and then guesses an execution starting at C_0, step by step, checking after each step if the reached configuration is C. Notice that all intermediate configurations of such an execution have the same size as C. At any moment in time the algorithm only stores three configurations, the current one, the next configuration in the execution, and the input one.

We can now observe that the emptiness problem is in PSPACE for any counting set with exponentially bounded norm and PSPACE-decidable membership. We again use Savitch's Theorem. If the counting set is nonempty, it has an element of size equal to the l-norm of the set. Such an element can be described in polynomial space. Therefore we can guess it and verify the set membership. □

Theorem 6.4 *The correctness problem for IO protocols is in* PSPACE.

Proof Let $\mathscr{P} = (Q, \delta, \Sigma, \iota, o)$ be an IO protocol, and φ a predicate over $\mathrm{Pop}(\Sigma)$. According to Proposition 2.12, $\mathscr{P}$ computes φ if and only if

$$post^*(I_b) \cap \overline{pre^*(St_b)} = \emptyset. \tag{2}$$

for $b \in \{0, 1\}$. By Lemma 6.1, I_b and Con_b are counting sets of polynomial norm.

By repeated application of Lemma 6.3, we observe that membership in $post^*(I_b)$, $pre^*(St_b)$, $\overline{pre^*(St_b)}$, and finally $post^*(I_b) \cap \overline{pre^*(St_b)}$ is in PSPACE; furthermore, emptiness of $post^*(I_b) \cap \overline{pre^*(St_b)}$ is in PSPACE as a problem with input $\mathscr{P}$ and φ. □

6.2 Correctness of DO protocols is in Π_2^p

We show that both the single-instance correctness and the correctness problem for DO protocols are in Π_2^p.

Throughout the section we use the symbol Z, possibly with accents or subscripts, to denote zero-message configurations. As before we denote the set of zero-message configurations by $\mathscr{Z}$.

We start with a characterization of non-correctness of a protocol for a given input.

Lemma 6.5 *Let $\mathscr{P}$ be a DO protocol with input alphabet Σ, let φ be a predicate over $\mathrm{Pop}(\Sigma)$, and let $D \in \mathrm{Pop}(\Sigma)$ be an input to $\mathscr{P}$. $\mathscr{P}$ does not compute $\varphi(D)$ on input D iff there exist zero-message configurations Z, Z_{nc} such that*

(i) $I(D) \xrightarrow{*} Z \xrightarrow{*} Z_{nc}$;
(ii) Z_{nc} is not a $\varphi(D)$-consensus; and
(iii) for every Z' reachable from Z there exists C such that $Z' \xrightarrow{*} C$ and $C|_Q = Z$.

Proof ($\Leftarrow$) Assume that there exist Z, Z_{nc} satisfying (i)-(iii). We show that no configuration reachable from Z is a stable $\varphi(D)$-consensus, which implies that $\mathscr{P}$ does not compute $\varphi(D)$. Let $\tilde{C}$ be an arbitrary configuration reachable from Z. By consuming all messages of $\tilde{C}$, the protocol can move from $\tilde{C}$ to some zero-message configuration $\tilde{Z}$ and, by (iii), to a configuration C such that $C|_Q = Z$. By (i), there exists a transition sequence ξ such that $Z \xrightarrow{\xi} Z_{nc}$. Since $C|_Q = Z$, we have $C \xrightarrow{\xi} C_{nc}$ for some configuration C_{nc} such that $C_{nc}|_Q = Z_{nc}$ (the sequence just "ignores" the messages of C). Summarizing, we have

$$Z \xrightarrow{*} \tilde{C} \xrightarrow{*} \tilde{Z} \xrightarrow{*} C \xrightarrow{*} C_{nc}$$

and so in particular $\tilde{C} \xrightarrow{*} C_{nc}$. By (ii) and $C_{nc}|_Q = Z_{nc}$, the configuration C_{nc} is not a $\varphi(D)$-consensus, and so $\tilde{C}$ is not a stable $\varphi(D)$-consensus.

($\Rightarrow$) Assume that $\mathscr{P}$ does not compute $\varphi(D)$ on input D. Let B be a bottom configuration reachable from $I(D)$ with no stable consensus reachable from it. Let Z be an arbitrary zero-message configuration reachable from B. By the assumption that B cannot reach a stable consensus, there is a configuration $Z \xrightarrow{*} C_{nc}$ which contains an agent with the output $1 - \varphi(D)$. Recall that we always have at least two agents, because configurations of our protocol models are defined as the populations over Q or $Q \cup M$, and populations are multisets with at least two elements. Given a configuration $C \in \overline{Con_{\varphi(D)}}$, we can keep one agent of C "aside" that has output $1 - \varphi(D)$ and let the other agents of C consume all the messages. This method applied to $C = C_{nc}$ yields a zero-message configuration Z_{nc} such that $Z \xrightarrow{*} C_{nc} \xrightarrow{*} Z_{nc}$ which is not a $\varphi(D)$-consensus. This proves properties (i) and (ii). To prove the property (iii) observe that B was a bottom configuration and therefore for every $Z \xrightarrow{*} Z'$ we have $B \xrightarrow{*} Z \xrightarrow{*} Z'$ and therefore $Z' \xrightarrow{*} B \xrightarrow{*} Z$. We can now define $C = Z$. □

Theorem 6.6 *Single-instance correctness of DO protocols is in Π_2^p.*

Proof Let $\mathscr{P} = (Q, M, \delta_r, \delta_s, \Sigma, \iota, o)$ be a DO protocol, let φ a predicate over $\mathrm{Pop}(\Sigma)$, and let $D \in \mathrm{Pop}(\Sigma)$ be an input to $\mathscr{P}$. We show that the problem of checking whether $\mathscr{P}$ with input D computes $\varphi(D)$ lies in Π_2^p.

It suffices to show that the problem of checking the existence of Z and Z_{nc} satisfying conditions (i)-(iii) of Lemma 6.5 is in Σ_2^p. By the Shortening Theorem for DO protocols (Corollary 4.40), we can guess two configurations Z and Z_{nc} satisfying (i) and (ii) in polynomial time, by nondeterministically traversing a computation of polynomial length (recall that all configurations reachable from $I(D)$ have the same size as $I(D)$), and checking in linear time that Z_{nc} is not a

$\varphi(D)$-consensus. The rest of the proof shows that checking (iii) is in co-NP. We proceed in three steps:

– We define the *saturation* of a zero-message configuration.
– We replace condition (iii) by an equivalent condition (iv) on the saturation of Z' (see Claim 2 below)
– We show that checking (iv) is in co-NP.

Saturation. Let Z be an arbitrary zero-message configuration, and let $|Z|$ be the number of agents of Z. For every state $q \in Q$ such that $Z(q) > 0$, let one of the agents in q send $|Z||Q| + |Q|^2$ messages $\delta_s(q)$. As long as there are reachable states q that have not yet sent $|Z||Q| + |Q|^2$ messages, let an agent go to q by a shortest path (which is of length at most $|Q| - 1$, see proof of Theorem 4.33) and let the agent send $|Z||Q| + |Q|^2$ messages $\delta_s(q)$. The resulting configuration $S(Z)$, called the *saturation* of Z, has the following properties:

(a) $Z \xrightarrow{*} S(Z)$.
By definition.
(b) For every message-type m, either $S(Z)$ has no messages of type m, or it has at least $|Z||Q|$ of them.
Indeed at most $|Q|$ messages are consumed in the addition of a new message-type, as a shortest path has length at most $|Q| - 1$ with at most one message consumed per step. And at most $|Q|$ message-types can be added (as each state sends only one type of message), and therefore each message-type has at most $|Q|^2$ messages consumed in $Z \xrightarrow{*} S(Z)$.
(c) For every configuration C, if $S(Z) \xrightarrow{*} C$ then $S(Z) \xrightarrow{\xi} C'$ for some configuration C' such that $C'|_Q = C|_Q$, and some sequence ξ that does not send any messages. Indeed, no new message types can be added to $S(Z)$ because otherwise we would have added them during the saturation step. There are enough messages of each type for $|Z|$ agents to move to new states by less than $|Q|$ steps (along the shortest paths), so no new messages are needed to reach C.

From condition (iii) to condition (iv). We claim:

Claim 1 Let Z be a zero-message configuration. Condition (iii) of Lemma 6.5 is equivalent to:

(iv) for every Z' reachable from Z there exists C such that $S(Z') \xrightarrow{*} C$ and $C|_Q = Z$.

To show that (iv) implies (iii), let Z' be reachable from Z. By (iv), there exists C such that $S(Z') \xrightarrow{*} C$ and $C|_Q = Z$. Since $Z' \xrightarrow{*} S(Z')$, we have $Z' \xrightarrow{*} C$. So (iii) holds. To prove that (iii) implies (iv), let Z' be reachable from Z. Since $Z' \xrightarrow{*} S(Z')$, we have $Z \xrightarrow{*} S(Z')$. By (iii), there exists C such that $S(Z') \xrightarrow{*} C$ and $C|_Q = Z$, and we are done.

Checking (iv) is in co- NP. Condition (iv) states that every Z' reachable from Z satisfies $P(Z, Z')$, where

$$P(Z, Z') \stackrel{\text{def}}{=} \exists C. \ S(Z') \xrightarrow{*} C \ \wedge \ C|_Q = Z \ .$$

We prove that the negation of (iv), i.e., the existence of Z' reachable from Z satisfying $\neg P(Z, Z')$, is in NP. By the Shortening Theorem (Corollary 4.40), Z' can be guessed in polynomial time. So it suffices to prove the second and final claim:

Claim 2 For every zero-message configuration Z', we can check in deterministic polynomial time whether $P(Z, Z')$ holds.

By property (c) of the saturation $S(Z')$ of Z', checking $P(Z, Z')$ reduces to deciding if there is a history of length $|Q| - 1$ whose trajectories transfer the agents from their states in $S(Z')$ to their states in Z, while sending no messages, and consuming only messages in $S(Z')$. We reduce this question to an integer max-flow problem, which can be solved in polynomial time by e.g. Edmonds-Karp algorithm. Consider the following directed graph $G_{Z,Z'}$ with capacities:

– The nodes of $G_{Z,Z'}$ are $|Q|$ copies of Q, written $q^{(1)}, q^{(2)}, \ldots q^{(|Q|)}$ for each $q \in Q$, plus a source node s, and a target node t.
– $G_{Z,Z'}$ has edges from s to each $q^{(1)}$ with capacity $S(Z')(q)$, and from $q^{(|Q|)}$ to t with capacity $Z(q)$.
– For each $i = 1, \ldots, |Q| - 1$, $G_{Z,Z'}$ has an edge from $q^{(i)}$ to $q'^{(i+1)}$ whenever the protocol has a receive transition from q to q' that consumes a message of $S(Z')$, or when $q = q'$. These edges have infinite capacity.

A flow value in this graph cannot exceed $\sum_{q \in Q} S(Z')(q) = |Z|$. Integer flows of value $|Z|$ naturally correspond to histories of length $|Q| - 1$ leading from $S(Z')$ to a configuration C such that $C|_Q = Z$, and vice versa. The flow through an edge $(q^{(i)}, q'^{(i+1)})$ gives the number of trajectories τ of H such that $\tau(i)\tau(i + 1) = q \ q'$. So we have: $P(Z, Z')$ holds iff the maximum integer flow of $G_{Z,Z'}$ is equal to $|Z|$. □

We formulate a new characterization of DO correctness, which considers only the reachability of zero-message configurations.

Proposition 6.7 *A DO protocol $\mathscr{P}$ is correct for a predicate φ iff the following holds for $b \in \{0, 1\}$:*

$$post_z^*(I_b) \subseteq pre_z^*(St_b^Z)$$

where St_b^Z is the set of zero-message configurations Z such that every zero-message configuration reachable from Z has output b.

Proof Notice that $post_z^*(I_b)$ is well-defined because DO initial configurations are always zero-message. By definition, St_b^Z is the set of zero-message configurations described by $\overline{pre_z^*\left(\overline{Con_b} \cap \mathscr{Z}\right)} \cap \mathscr{Z}$. We prove the following claim
Claim. The set equality $St_b^Z = St_b \cap \mathscr{Z}$ holds.

This can be rewritten as

$$\overline{pre_z^*\left(\overline{Con_b} \cap \mathscr{Z}\right)} \cap \mathscr{Z} = \overline{pre^*(\overline{Con_b})} \cap \mathscr{Z}.$$

Consider a configuration Z in $\overline{pre_z^*\left(\overline{Con_b} \cap \mathscr{Z}\right)} \cap \mathscr{Z}$. We assume that $Z \notin \overline{pre^*(\overline{Con_b})} \cap \mathscr{Z}$, i.e. $Z \notin \overline{pre^*(\overline{Con_b})}$, and derive a contradiction. Since $Z \notin \overline{pre^*(\overline{Con_b})}$, from Z we can reach a configuration of $\overline{Con_b}$. We show that we can also reach a configuration of $\overline{Con_b} \cap \mathscr{Z}$. We again use that a configuration contains at least two agents. Given a configuration $C \in \overline{Con_b}$, we can keep one agent of C "aside" that has output $1-b$ and let the other agents of C consume all the messages. This is possible because δ_r is a total function, and thus every $C \in \overline{Con_b}$ can reach a configuration of $\overline{Con_b} \cap \mathscr{Z}$, thus amounting to a contradiction for Z.

Conversely, let $Z \in \overline{pre^*(\overline{Con_b})} \cap \mathscr{Z}$, and assume $Z \notin \overline{pre_z^*\left(\overline{Con_b} \cap \mathscr{Z}\right)}$. Then Z can reach a configuration of $\overline{Con_b} \cap \mathscr{Z}$, which is also a configuration of $\overline{Con_b}$. This is a contradiction, and so the claim is proved.

Recall the characterization of correctness in Proposition 2.12 which states that a DO protocol $\mathscr{P}$ is correct for a predicate φ if and only if

$$post^*(I_b) \subseteq pre^*(St_b) \tag{3}$$

for $b \in \{0, 1\}$. We use the claim above to show that $post_z^*(I_b) \subseteq pre_z^*(St_b^Z)$ holds if and only if (3) holds.

Suppose (3) holds. Let Z be a configuration of $post_z^*(I_b)$. Since $post_z^*(I_b) \subseteq post^*(I_b)$, there exists some $C \in St_b$ such that $Z \xrightarrow{*} C$. Because δ_r is a total function, we can let the agents of C consume all the messages so that $C \xrightarrow{*} Z'$ for some zero-message configuration Z'. All configurations reachable from St_b are still in St_b so $Z' \in St_b \cap \mathscr{Z} = St_b^Z$ by the claim, and we are done.

Suppose $post_z^*(I_b) \subseteq pre_z^*(St_b^Z)$ holds. Let C be a configuration of $post^*(I_b)$. As before we let the agents of C consume all its messages so that $C \xrightarrow{*} Z$ for some zero-message configuration Z that is thus in $post_z^*(I_b)$. By assumption, there exists some $Z' \in St_b^Z$ such that $Z \xrightarrow{*} Z'$. Since $St_b^Z = St_b \cap Z \subseteq St_b$, we are done. □

Theorem 6.8 *The correctness problem for DO protocols is in Π_2^p.*

Proof We prove that the non-correctness problem for DO protocols is in Σ_2^p. Let $\mathscr{P}$ be a DO protocol and let φ be a predicate. By definition, $\mathscr{P}$ is not correct if there exists an input $D \in Pop(\Sigma)$ such that $\mathscr{P}$ does not compute $\varphi(D)$ on input D. (Observe that, by the definition of DO protocols, the initial configuration $I(D)$ is a zero-message configuration.) We start with a claim:
Claim. If such an input D exists, then it can be chosen *of polynomial size* in $\mathscr{P}$ and φ.

By Proposition 6.7, $\mathscr{P}$ computes φ if and only if

$$post_z^*(I_b) \cap \overline{pre_z^*(St_b^Z)} = \emptyset. \tag{4}$$

We show that if (4) does not hold, then $post_z^*(I_b) \cap \overline{pre_z^*(St_b^Z)}$ contains a configuration, say Z, with a polynomial number of agents in $\mathscr{P}$ and φ. Since transitions do not change the number of agents of a configuration, there exist an input D such that $I(D) \xrightarrow{*} Z$ and $|D| = |I(D)| = |Z|$, proving the claim.

By Lemma 6.1, $Con_b|_Q$ and $I_b|_Q$ are counting sets with norms of linear size in the size of $\mathscr{P}$ and φ. Sets $Con_b|_Q$ and $I_b|_Q$ are the projections onto $\mathbb{N}^Q$ of the sets $Con_b \cap \mathscr{Z}$ and I_b, respectively. Thus, by Proposition 5.2 and Corollary 5.7, the set $post_z^*(I_b) \cap \overline{pre_z^*(St_b^Z)}$ is represented by a counting constraint Γ whose l-norm is polynomial in $\mathscr{P}$ and φ. More precisely, we have

$$\|\Gamma\|_l \leq |Q|^4 + |Q|^3 + |Q|^3 \in O(|Q|^4).$$

So if (4) does not hold, then the set $post_z^*(I_b) \cap \overline{pre_z^*(St_b^Z)}$ contains a a zero-message configuration with $\|\Gamma\|_l$ agents, and the claim is proved.

By Lemma 6.5 and the claim, $\mathscr{P}$ does not compute φ iff there exist an input D of polynomial size in $\mathscr{P}$ and φ, such that there exist zero-message configurations Z, Z_{nc} satisfying conditions (i)-(iii) of the lemma.

By Theorem 6.6, checking the existence of Z and Z_{nc} for a given input D lies in Σ_2^p. Since the input D and the boolean $b \in \{0, 1\}$ can be guessed in polynomial time in $\mathscr{P}$ and φ, checking that $\mathscr{P}$ does not compute φ also lies in Σ_2^p. □

7 Hardness and decidability of correctness for transmission-based models

7.1 Correctness of transmission-based models is TOWER-hard

In this section we establish lower bounds for the complexity of the correctness problem of the different variants of transmission protocols. We show that deciding correctness for delayed and queued transmission protocols is TOWER-

hard, even in the single-instance case, and that the general correctness problem is TOWER-hard for the three variants (immediate, delayed, queued) of transmission protocols.

In order to establish these lower bounds, we make use of the fact that the reachability problem for *VASS* (vector addition systems with states) is TOWER-hard [24]. A VASS of some fixed dimension $k \in \mathbb{N}$ can be described as a pair (Q, T) where Q is a finite set of states, and $T \subseteq Q \times \mathbb{Z}^k \times Q$ is a transition relation. We write $q \overset{\mathbf{v}}{\to} r$ whenever $(q, \mathbf{v}, r) \in T$. Furthermore, for two vectors $\mathbf{w}, \mathbf{w}' \in \mathbb{N}^k$ and states $q, q' \in Q$, we write $(q, \mathbf{w}) \to (q', \mathbf{w}')$ whenever there exists a vector $\mathbf{v}$ such that $q \overset{\mathbf{v}}{\to} q'$ and $\mathbf{w}' = \mathbf{w} + \mathbf{v}$. As usual, by $\overset{*}{\to}$ we denote the reflexive-transitive closure of $\to$. The reachability problem for VASS is the following problem: Given vectors $\mathbf{v}, \mathbf{w} \in \mathbb{N}^k$ in the dimension k of a given VASS, and given states q, r, does $(q, \mathbf{v}) \overset{*}{\to} (r, \mathbf{w})$ hold?

We call a VASS (Q, T) a ± 1-VASS, if every transition $q \overset{\mathbf{v}}{\to} q'$ in T satisfy that all components of $\mathbf{v}$ but one are equal to 0, and this component has value 1 or -1. For a given ± 1-VASS $\mathcal{N}$ of some dimension k, and $1 \leq m \leq k$, we write $q \overset{m++}{\longrightarrow} q'$ whenever $q \overset{\mathbf{v}}{\to}_{\mathcal{N}} q'$ holds for some $\mathbf{v}, q, q'$ such that $v_m = 1$. Likewise, we write $q \overset{m--}{\longrightarrow} q'$ whenever $q \overset{\mathbf{v}}{\to}_{\mathcal{N}} q'$ holds for some $\mathbf{v}, q, q'$ such that $v_m = -1$. The following proposition holds:

Proposition 7.1 *For every unary-encoded VASS $\mathcal{N} = (Q, T)$ and unary-encoded configurations $(q_0, \mathbf{v}_0)$, $(q, \mathbf{v})$, one can construct in polynomial time a ± 1-VASS $\mathcal{N}' = (Q', T')$ with distinct states $r_0, r \in Q'$ such that*

$$(q_0, \mathbf{v}_0) \overset{*}{\to}_{\mathcal{N}} (q, \mathbf{v}) \iff (r_0, \mathbf{0}) \overset{*}{\to}_{\mathcal{N}'} (r, \mathbf{0}) .$$

Proof The reduction is rather straightforward; details can be found in the "Appendix E". □

To simplify the coming proofs, we introduce nondeterministic delayed-transmission protocols. The definition of the nondeterministic version is identical to the deterministic version except that δ_s now maps to sets of message/state pairs, δ_r maps to a non-empty set of states, and the scheduler must choose nondeterministically from these sets whenever a message is sent or received.

Nondeterminism adds no expressive power to delayed-transmission protocols, as the following proposition shows:

Proposition 7.2 *For every nondeterministic DT protocol $\mathcal{P}$ there exists a deterministic DT protocol $\mathcal{P}'$ that computes the same predicate as $\mathcal{P}$. Moreover, $\mathcal{P}'$ can be constructed in polynomial time.*

Proof Let $\mathcal{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$. In order to simulate the nondeterminism of $\mathcal{P}$ in $\mathcal{P}'$, each state $q \in Q$ is annotated with a round counter i ranging from 1 to n,

where n is the maximal number of nondeterministic choices per state. When an agent sends/receives a message from M, the counter i determines the choice to be made. Additionally, agents may send and receive a special message increment. Whenever an agent receives the message increment, its round counter is incremented by one, that is, i is set to $(i \bmod n) + 1$. To ensure full simulation of nondeterminism, we must ensure that there are always enough increment messages in circulation. We achieve this by letting every agent emit an increment message at the start of the computation, and enforcing re-emission of increment messages after receiving an increment message. Whether an agent must send an increment message is governed by an additional bit, which the agent stores in its state. We provide the full construction in the "Appendix E". □

We show:

Proposition 7.3 *Let $\mathcal{N} = (Q^{\mathcal{N}}, T^{\mathcal{N}})$ be a ± 1-VASS and let $r_0, r \in Q^{\mathcal{N}}$. It is possible to construct in polynomial time a (nondeterministic) DT protocol $\mathcal{P}$ and an initial configuration C_0 of $\mathcal{P}$ such that $(r_0, \mathbf{0}) \overset{*}{\to} (r, \mathbf{0})$ holds if and only if $\mathcal{P}$ does not converge to 1 for the initial configuration C_0.*

Proof Intuitively, the protocol $\mathcal{P}$ simulates the ± 1-VASS in a population of size 1, with the current control state of $\mathcal{N}$ being stored in the state of the single agent, and the current counting vector represented in the message pool by messages denoted $1, \ldots, k$. For example, if the configuration of the machine is $q, (6, 4)$, then the agent is in state q, and the message pool contains 6 messages denoted by 1, and 4 messages denoted by 2. Decrementing/incrementing a counter is implemented by sending/receiving messages.

When the agent reaches state r, it can nondeterministically guess that the current vector is $\mathbf{0}$, and then alternate indefinitely between a false and a true state, say $r_\perp$ and $r_\top$, which constitutes a non-stabilizing fair execution in the case where $r_0, \mathbf{0} \overset{*}{\to} r, \mathbf{0}$ holds. If the agent makes a wrong guess, then the message pool is non-empty at that time, and by fairness the agent eventually receives a message which lets the the agent turn to a permanent true state, say, $\top$. This ensures that every fair execution converges to 1 in the case where $r_0, \mathbf{0} \overset{*}{\not\to} r, \mathbf{0}$.

Let us now define $\mathcal{P}$ formally. Given the ± 1-VASS $\mathcal{N}$ of some dimension k and the states r_0, r, the protocol $\mathcal{P} = (Q, M, \delta_s, \delta_r, \Sigma, \iota, o)$ is constructed as follows:

- $Q \overset{\text{def}}{=} Q^{\mathcal{N}} \cup \{r_\top, r_\perp, \top\}$
- $M \overset{\text{def}}{=} \{1, \ldots, k\} \cup \{\epsilon\}$

- δ_s is given by:

$$\delta_s(r) \stackrel{\text{def}}{=} \{(q', m) \mid r \xrightarrow{m++} q'\} \cup \{(r_\perp, \epsilon)\}$$

$$\delta_s(r_\perp) \stackrel{\text{def}}{=} \{(r_\top, \epsilon)\}$$

$$\delta_s(r_\top) \stackrel{\text{def}}{=} \{(r_\perp, \epsilon)\}$$

$$\delta_s(q) \stackrel{\text{def}}{=} \{(q', m) \mid q \xrightarrow{m++} q'\} \quad \text{for every } q \in Q^{\mathcal{N}} \setminus \{r\}.$$

- δ_r is given by:

$$\delta_r(r_\top, \epsilon) = \delta_r(r_\perp, \epsilon) \stackrel{\text{def}}{=} \{r_\top\}$$

$$\delta_r(q, m) \stackrel{\text{def}}{=} \{q' \mid q \xrightarrow{m--} q'\} \quad \text{if } q \xrightarrow{m--} q' \text{ for some } q'$$

$$\delta_r(q, m) \stackrel{\text{def}}{=} \{\top\} \qquad\qquad \text{in all remaining cases.}$$

- $\Sigma \stackrel{\text{def}}{=} \{r_0\}$
- $\iota = \text{id}$
- $o(r_\perp) \stackrel{\text{def}}{=} 0$ and $o(q') \stackrel{\text{def}}{=} 1$ for every $q' \neq r_\perp$.

We define the initial configuration by setting $C_0 \stackrel{\text{def}}{=} \wr r_0 \wr$.

We associate a configuration $C \in \text{Pop}(\{1, \ldots, k\})$ with its corresponding vector in $\mathbb{N}^k$ via the bijection $\varphi \colon \text{Pop}(Q) \to \mathbb{N}^k$ given by $\varphi(C) \stackrel{\text{def}}{=} (C(1), \ldots, C(k))$. By construction, for every sequence of states $q_1, \ldots, q_m \in Q^{\mathcal{N}}$, and every sequence of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{N}^k$ we have:

$$(r_0, \mathbf{0}) \to (q_1, \mathbf{v}_1) \to (q_2, \mathbf{v}_2) \to \ldots \to (q_m, \mathbf{v}_m)$$
$$\Longleftrightarrow$$
$$\wr r_0 \wr \to \left(\wr q_1 \wr + \varphi^{-1}(\mathbf{v}_1)\right) \to \ldots \to \left(\wr q_m \wr + \varphi^{-1}(\mathbf{v}_m)\right).$$

It remains to prove that $\mathcal{P}$ does not converge to 1 for $C_0 = \wr r_0 \wr$ if and only if $r_0, \mathbf{0} \xrightarrow{*} r, \mathbf{0}$. We only prove the direction ($\Leftarrow$); the converse direction is similar. Assume $r_0, \mathbf{0} \xrightarrow{*} r, \mathbf{0}$ holds. Then by the previous consideration we have: $C_0 \xrightarrow{*} \wr r \wr$. Thus we obtain:

$$C_0 \xrightarrow{*} \wr r \wr \to \wr r_\perp, \epsilon \wr \to \wr r_\top \wr \to \wr r_\perp, \epsilon \wr \to \wr r_\top \wr \to \ldots$$

The above execution is fair, but does not converge to a consensus, as $o(r_\top) \neq o(r_\perp)$. Hence $\mathcal{P}$ does not converge to 1 for C_0, which concludes the proof for this direction.

Formally, the population should have at least two agents. One of the ways to resolve this problem is to say that we have an extra state $\perp$ with output 0, and an extra agent starting in the state $\perp$. It never sends messages, and if it ever receives a message, it switches to $\top$. We can let $\top$ send a special message $m_\top$ turning the other agent into $\top$. If there is a finite execution producing $r_\perp$ and leaving no messages, it can happen despite existence of the extra $\perp$ agent; otherwise we reach $\top$ like we did before. $\qquad\square$

Combining the previously established propositions, we obtain:

Theorem 7.4 *The single-instance correctness problem is TOWER-hard for DT and QT protocols.*

Proof Since delayed-transmission protocols are a subclass of queued-transmission protocols, it suffices to show the claim for delayed-transmission protocols.

By propositions 7.1 and 7.3, the TOWER-hard reachability problem for VASS is polynomially Turing-reducible to 1-instance correctness of delayed-transmission protocols. This shows the theorem. $\qquad\square$

We establish the same hardness result for the general correctness problem:

Theorem 7.5 *The correctness problem for DT and QT protocols is TOWER-hard.*

Proof Since delayed-transmission protocols are a subclass of queued-transmission protocols, we only need to prove the theorem for delayed-transmission protocols. In the "Appendix E", we prove the following claim: For every delayed-transmission protocol $\mathcal{P} = (Q, M, \delta_r, \delta_s, \Sigma, \iota, o)$ and every initial configuration $C \in \text{Pop}(I)$, one can construct in polynomial time a delayed-transmission protocol $\mathcal{P}' = (Q', M', \delta'_r, \delta'_s, \Sigma, \iota', o')$ such that $\mathcal{P}'$ computes constant 1 if and only if $\mathcal{P}$ converges to 1 for the single instance C. By Theorem 7.4, the claim entails Theorem 7.5, and we are done. $\qquad\square$

Perhaps surprisingly, even in the restricted setting of immediate-transmission protocols, the general correctness problem remains TOWER-hard:

Theorem 7.6 *The correctness problem for IT protocols is TOWER-hard.*

Proof Let $\mathcal{N} = (Q, T)$ be a ± 1-VASS and let $q, r \in Q$. We claim that we can construct in polynomial time an immediate-transmission protocol $\mathcal{P}$ that computes constant 1 if and only if $q, \mathbf{0} \xrightarrow{*} r, \mathbf{0}$ does *not* hold. The claim entails the theorem by Proposition 7.1 and TOWER-hardness of VASS-reachability. In the "Appendix E" we provide a construction that shows the claim. $\qquad\square$

On the other hand, the single-instance correctness problem for immediate transmission protocols is not TOWER-hard. It is in fact PSPACE-complete.

Theorem 7.7 *The single-instance correctness problem for IT protocols is* PSPACE-*complete.*

Proof Let $\mathcal{P} = (Q, \delta, \Sigma, \iota, o)$ be an IT protocol, φ a predicate over $\text{Pop}(\Sigma)$ and C_0 a configuration. We reuse the

notation of Sect. 6, and let C_0 be a configuration in I_b for $b \in \{0, 1\}$, i.e. a fair execution starting in C_0 must converge to b if the protocol is correct. The proof is the same as for single-instance correctness of IO protocols in Theorem 6.4: using the correctness characterization of Proposition 2.12, we guess a configuration C of size $|C_0|$ and check that it is in the intersection $post^*(I_b) \cap \overline{pre^*(St_b)}$ using NPSPACE procedures. The only difference with the IO proof lies in the step relation, which remains checkable in polynomial time.

PSPACE-hardness follows from the fact that IO protocols are IT protocols, and the hardness result of Theorem 3.3. □

7.2 Decidability of correctness for PP and DT protocols

We present a generic result showing that the correctness problem is decidable for a class of protocols satisfying certain properties. All protocol models considered in the paper, with the exception of QT, satisfy the properties. The proof follows closely the one of [32] for standard population protocols. However, the presentation emphasizes the role played by each of the properties, allowing us to pinpoint why the proof of [32] can be generalized to DT protocols, but not to QT protocols. While we leave the decidabililty of correctness for QT open, we also argue that the notion of fairness chosen in [7], and also used in our paper, is questionable for QT, making the correctness problem for QT less interesting than for the other five models.

Recall the property defined in Sect. 2.6: a protocol is *well-behaved* if every fair execution contains a bottom configuration. We introduce some further properties of protocols:

Definition 7.8 A protocol $\mathscr{P} = (Conf, \Sigma, Step, I, O)$ is

- *finitely generated* if $Conf \subseteq \mathbb{N}^k$ for some $k \geq 0$, and there is a finite set $\Delta \subseteq \mathbb{Z}^k$ such that $(C, C') \in Step$ iff $C' - C \in \Delta$; we say that $Step$ is *generated* by Δ.
- *input-Presburger* if for every effectively Presburger set $L \subseteq \mathrm{Pop}(\Sigma)$ of inputs the set $I(L) \subseteq \mathrm{Pop}(Q)$ is an effectively computable Presburger set of configurations.
- *output-Presburger* if $O^{-1}(0)$ and $O^{-1}(1)$ are effectively Presburger sets of configurations.

We call a protocol that is well-behaved, finitely generated, and input/output-Presburger a *WFP-protocol*.

Recall the characterization of correctness for well-behaved protocols that we obtained in Proposition 2.13.

Proposition 7.9 *Let $\mathscr{P}$ be a well-behaved generalized protocol and let φ be a predicate. $\mathscr{P}$ computes φ iff for every $b \in \{0, 1\}$ the set $\mathscr{B} \setminus \mathscr{B}_b$ is not reachable from I_b.*

We show that this reachability condition is decidable for WFP-protocols. Observe that a finitely generated protocol

$\mathscr{P} = (Conf, \Sigma, Step, I, O)$ can be easily represented as a VAS. Indeed, if $Conf \subseteq \mathbb{N}^k$ and $Step$ is generated by Δ, then the VAS has dimension k and has Δ as set of transitions. Using this fact, and the powerful result stating the decidability of the reachability problem in a VAS between effectively Presburger sets of configurations, we obtain:

Proposition 7.10 ([32]) *Let $\mathscr{C}, \mathscr{C}'$ be two effectively Presburger sets of configurations of a finitely generated protocol. It is decidable if some of configuration of $\mathscr{C}'$ is reachable from some configuration of $\mathscr{C}$.*

By Proposition 7.10, in order to prove the decidability of correctness it suffices to show that the sets $I(\varphi^{-1}(b))$ and $\mathscr{B} \setminus \mathscr{B}_b$ of a WFP-protocol are effectively Presburger sets. $I(\varphi^{-1}(b))$ holds by the definition of WFP-protocols (recall that $\varphi^{-1}(b)$ is always a Presburger set). It remains to show that $\mathscr{B} \setminus \mathscr{B}_b$ is effectively Presburger. Since effectively Presburger sets are closed under boolean operations, it suffices to show that $\mathscr{B}$ and $\mathscr{B}_b$ are effectively Presburger. This is a nontrivial result, but already proved in [32]:

Proposition 7.11 ([32], Proposition 14) *There is an algorithm that takes as input a finitely generated, output-Presburger protocol, and returns Presburger predicates denoting the sets $\mathscr{B}, \mathscr{B}_0,$ and $\mathscr{B}_1$.*

So we finally obtain:

Theorem 7.12 *The correctness problem is decidable for WFP-protocols.*

Applying Theorem 7.12 we can easily prove that the correctness problem is decidable for PP and DT. Indeed, PP protocols and DT protocols are WFP as they are well-behaved by Lemma 2.9, and finitely generated and input/output Presburger by hypothesis. Since IT and IO are subclasses of PP and DO is a subclass of DT, the proof is valid for them as well.

Corollary 7.13 *The correctness problem is decidable for PP, DT, and their subclasses.*

However, queued-transmission protocols are not necessarily well-behaved (as shown in Example 2.10), and so not necessarily WFP. Currently, to the best of our knowledge the decidability of the well-specification and correctness problems for queued-transmission protocols is open. At the same time, Example 2.10 shows that our fairness condition is questionable for queued-transmission models: An execution $C_0, C_1, \ldots$ in which only one agent acts, even if other agents have enabled actions in C_i for every $i \geq 0$, can still be fair. Is the fairness notion of [7] adequate for queued-transmission protocols?

7.3 Correctness in probabilistic models

In [7], Angluin et al. state that the fairness condition "may be viewed as an attempt to capture useful probability 1 properties in a probability-free model". Indeed, population protocols are often introduced in a probabilistic setting, which assigns a probability to the set of executions that converge to a value. Once a probabilistic model is fixed, we have two different definitions of when a protocol $\mathscr{P}$ computes a predicate φ:

- $\mathscr{P}$ *f-computes* φ if for every input $\sigma \in \text{Pop}(\Sigma)$, every fair execution starting at $I(\sigma)$ converges to $\varphi(\sigma)$.
- $\mathscr{P}$ *p-computes* φ if for every input $\sigma \in \text{Pop}(\Sigma)$, the set of all executions starting at $I(\sigma)$ that converge to $\varphi(\sigma)$ has probability 1.

The question whether the fairness condition is adequate for a class of protocols can now be rephrased as: Do f-computation and p-computation coincide for the class? In this section we examine this question in some detail.

In order to formalize a probabilistic protocol model we must specify the random experiment that determines the next step carried out by the protocol. For standard population protocols there is agreement in the literature on the experiment: At each step two agents of the population are chosen uniformly at random, and they interact. However, for the delayed and queued-transmission models there is no canonical experiment. We consider the following family of random experiments parameterized by a probability p.

Definition 7.14 Let $\mathscr{P} = (Q, M, \delta_s, \delta_r, I, O)$ be a queued-transmission protocol, and let $0 < p < 1$. For every state $q \in Q$, let $R(q)$ denote the set of messages that an agent can receive in state q. The *s:p/r:(1-p) probabilistic model* [3] is described by the following random experiment. Assume the current configuration is C. First, choose an agent uniformly at random, and let q be its current state; then:

- with probability p, let the agent send the message specified by the send function;
- with probability $1 - p$: if $R(q) \neq \emptyset$, choose a message from the multiset $\bigcup_{m \in R(q)} C(m)$ uniformly at random, and let the agent receive it; otherwise, the agent does nothing.

Recall that in the delayed-transmission model we have $R(q) = M$ for every state q, i.e., agents can never refuse receiving a message.

---

[3] Short for "send with probability p, receive with probability $(1 - p)$".

In the rest of the section we examine the relation between f-computation and p-computation for our protocol models, and obtain the following results:

- For standard population protocols and their subclasses, f-computation and p-computation coincide.
- For delayed-transmission protocols and s:p/r:(1-p) models, f-computation and p-computation coincide iff $p \leq 1/2$.
- For queued-transmission protocols, f-computation and p-computation are incomparable notions under fairly general conditions on probabilistic models. In particular, there are protocols that f-compute a predicate but do not p-compute any predicate in any s:p/r:(1-p) model, and vice-versa.

Standard population protocols. Recall that in the probabilistic model at each step two agents are chosen uniformly at random. We have:

Proposition 7.15 *Let $\mathscr{P}$ be a standard population protocol, and let φ be a predicate. $\mathscr{P}$ f-computes φ iff $\mathscr{P}$ p-computes φ.*

Proof By Proposition 2.13, $\mathscr{P}$ f-computes φ iff for every input a the set $\mathscr{B} \setminus \mathscr{B}_{\varphi(a)}$ is not reachable from $I(a)$. We show that this is the case iff $\mathscr{P}$ p-computes φ.

Since every configuration of a standard population protocol has a finite number of successors, an execution starting at $I(a)$ almost surely visits a bottom configuration. So $\mathscr{P}$ p-computes φ if the set of executions visiting $\mathscr{B}_{\varphi(a)}$ has probability 1. Since every finite execution leading from $I(a)$ to a configuration of $\mathscr{B}$ has positive probability, this is the case iff $\mathscr{B} \setminus \mathscr{B}_{\varphi(a)}$ is not reachable from $I(a)$. □

Delayed-transmission protocols. We show that for delayed-transmission protocols and s:p/r:(1-p)-models f-computation and p-computation coincide iff $p \leq 1/2$.

Lemma 7.16 *Let $\mathscr{P}$ be a delayed-transmission protocol in the s:p/r:(1-p) model with $p \leq 1/2$. With probability 1, an execution of $\mathscr{P}$ visits infinitely often configurations with no messages in transit.*

Proof We prove that the number k of messages in transit behaves similarly to a random walk in which the probability of reducing k is at least as high as the probability of increasing it.

For a configuration C, let $\text{Pr}(C)$ denote the probability that an execution starting from C only visits configurations with at least one message in transit. Further, let $\text{Pr}(n, k)$ be the maximum value of $\text{Pr}(C)$ among all configurations with n agents and k messages in transit. Observe that $\text{Pr}(n, 0) = 0$,

because in this case C itself has no messages in transit. We prove that $\Pr(n, k) = 0$ for every $k \geq 0$, which is equivalent to the statement of the lemma.

Let n and $k > 0$, and let C_{max} be a configuration with n agents and k messages satisfying $\Pr(C_{max}) = \Pr(n, k)$. A step from configuration C_{max} consumes a message with probability at least $\frac{1}{2}$ (in a delayed transmission protocol an agent can always receive any message), and produces a message with probability $0 \leq p \leq \frac{1}{2}$. So we have

$$\Pr(n, k) = \Pr(C_{max})$$
$$\leq \frac{1}{2} \Pr(n, k - 1) + p \Pr(n, k + 1)$$
$$+ \left(\frac{1}{2} - p \right) \Pr(n, k)$$

which can be rewritten as

$$\Pr(n, k) \leq \frac{\frac{1}{2} \Pr(n, k - 1) + p \Pr(n, k + 1)}{\frac{1}{2} + p}$$

The right side is the weighted average of $\Pr(n, k - 1)$ and $\Pr(n, k + 1)$, with weight p between 0 and $\frac{1}{2}$. It can be bounded by the weighted average for one of the extremal values of p, and so we have $\Pr(n, k) < \Pr(n, k - 1)$ or $\Pr(n, k) \leq \frac{1}{2} \Pr(n, k - 1) + \frac{1}{2} \Pr(n, k + 1)$. Rewriting the second case, we finally obtain that the following disjunction holds for all $n, k > 0$:

$$\Pr(n, k) < \Pr(n, k - 1) \quad \text{or}$$
$$\Pr(n, k + 1) - \Pr(n, k) \geq \Pr(n, k) - \Pr(n, k - 1) .$$

Assume there is a smallest number z such that $\Pr(n, z) > 0$ and $\Pr(n, z - 1) = 0$. Then, by the disjunction above and $\Pr(n, z) - \Pr(n, z - 1) = \Pr(n, z)$, we have $\Pr(n, z + i) \geq (i + 1)\Pr(n, z)$ for every $i \geq 0$ (easy induction on i). This contradicts that $1 \geq \Pr(n, z + i)$ holds for every $i \geq 0$, and so z does not exist. Since $\Pr(n, 0) = 0$ by definition, we have $\Pr(n, k) = 0$ for every $k \geq 0$. □

Proposition 7.17 *Let $\mathscr{P}$ be a delayed-transmission protocol in a s:p/r:(1-p) model with $p \leq 1/2$, and let φ be a predicate. $\mathscr{P}$ f-computes φ iff $\mathscr{P}$ p-computes φ.*

Proof Assume $\mathscr{P}$ f-computes φ. We show that it p-computes φ. For this it suffices to show that for every initial configuration C_0 the set of fair executions starting at C_0 has probability 1, or, in other words, that an execution is fair with probability 1.

Fix an initial configuration C_0, and let C be an arbitrary configuration. Let $\mathscr{Z}$ be the set of configurations reachable from C_0 with zero messages in transit. Since the number of agents remains constant, $\mathscr{Z}$ is finite. For each $Z \in \mathscr{Z}$, either

C is unreachable from Z, or there is a shortest sequence of transitions leading from Z to C (possibly not unique). Such a sequence has a positive probability of occurring from Z. Let p_{min} be the minimal probability of all the probabilities of shortest paths from any $Z \in \mathscr{Z}$ to C, and ℓ be the maximum length of a shortest path.

By Lemma 7.16, an execution starting at C_0 reaches a configuration $Z_1 \in \mathscr{Z}$ with probability 1. Either C is unreachable from Z_1, or the probability of reaching C in at most ℓ steps is at least p_{min}. If C is not reached in ℓ steps but remains reachable, with probability 1 we reach a configuration $Z_2 \in \mathscr{Z}$ from Z_1. Iterating this reasoning, we observe that the execution visits a sequence of configurations $Z_1, Z_2, \ldots \in \mathscr{Z}$ such that for every Z_i, the probability that in the next ℓ steps C is reached or becomes unreachable is at least p_{min}. Therefore, the event "C becomes unreachable or it is reached infinitely often" has probability 1. So an execution is fair with probability 1.

Assume $\mathscr{P}$ does not f-compute φ. We show that it does not p-compute φ. Since $\mathscr{P}$ does not f-compute φ, there is a fair execution π that does not converge to the value specified by φ, call it b. Let C_0 be the initial configuration of π and, as above, let $\mathscr{Z}$ be the finite set of configurations reachable from C_0 with zero messages in transit. Further, let $Rec(\pi)$ be the set of configurations of $\mathscr{Z}$ that occur in π infinitely often.

Since $\mathscr{P}$ is a delayed-transmission protocol, every configuration of π can reach some configuration of $\mathscr{Z}$. Therefore, by fairness and finiteness of $\mathscr{Z}$, $Rec(\pi) \neq \emptyset$, and $Rec(\pi)$ is closed under reachability. We claim that an execution that reaches $Rec(\pi)$ converges to b with probability 0. Since there is a positive probability that a execution reaches $Rec(\pi)$, it follows that $\mathscr{P}$ does not p-compute φ. To prove the claim, observe that, since π does not converge to b, some configuration C reachable from $Rec(\pi)$ is not a b-consensus. Since $Rec(\pi)$ is finite, there exists $p > 0$ such that C is reachable from every configuration of $Rec(\pi)$ with probability at least p. Therefore, an execution that reaches $Rec(\pi)$ visits C infinitely many times with probability 1, and so it converges to b with probability 0. □

Proposition 7.18 *There is a delayed-transmission protocol $\mathscr{P}$ that p-computes the value 0 on a certain input in every s:p/r:(1-p) model with $p > 1/2$, but that does not f-compute any value on the same input.*

Proof Consider the protocol with states $\{q_0, q_1, q_2\}$; output function given by $O(q_0) = O(q_1) = 0$ and $O(q_2) = 1$; messages $\{a, b\}$; and transitions

$$q_0 \xrightarrow{a+} q_0 \quad q_1 \xrightarrow{b+} q_0 \quad q_2 \xrightarrow{b+} q_1$$
$$q_0 \xrightarrow{a-} q_0 \quad q_1 \xrightarrow{a-} q_1 \quad q_2 \xrightarrow{a-} q_2$$
$$q_0 \xrightarrow{b-} q_1 \quad q_1 \xrightarrow{b-} q_2 \quad q_2 \xrightarrow{b-} q_2$$

Consider the input configuration $\langle q_2 \rangle$.

For the sake of simplicity we allow configurations with a single agent. The behaviour is qualitatively the same for multiple agents (as required by the definition of population), up to some technicalities in probablility calculations.

In each configuration of each execution the sum of the index of the state and the number of messages of type b is equal to 2. This protocol does not f-compute any value on $\langle q_2 \rangle$ because the configuration $\langle q_2 \rangle$ with no messages is reachable from each configuration in the execution, as well as the configuration $\langle q_0, b, b \rangle$ with 2 messages of type b. These two configurations occur infinitely often in each fair execution and have different output values.

The proof that an execution from $\langle q_2 \rangle$ converges with probability 1 if $p > \frac{1}{2}$ is based on the following observations.

- The number of messages changes independently of the configuration change, so it is a biased random walk with linear growth.
- The state q_0 can always be reached with probability at least $1/4$, and so it is reached infinitely many times.
- Going from q_0 to q_2 requires receiving two bs without sending in-between.
- The probability to receive two bs is proportional to $1/n^2$, where n is the number of messages. Since the series $\sum_{i=1}^{\infty} 1/n^2$ converges, so with probability 1 the state q_2 is only observed a finite number of times.

We show that therefore q_2 occurs only a finite number of times with probability 1, and that the protocol p-computes value 0. The rest of the proof presents this argument in detail; it is purely technical and can be found in the "Appendix F". □

Queued-transmission protocols. Unfortunately, in queued-transmission protocols there is no useful relation between f-computation and p-computation. We show this with the help of two examples. The first one computes a predicate in every model from a general class, but does not f-compute any predicate. The second f-computes a predicate, but does not compute a predicate in any probabilistic model from the same general class.

Definition 7.19 A probabilistic model of execution for queued-transmission protocols is

- *positive* if for every configuration C every step $C \to C'$ has positive probability.
- *markovian* if for every configuration C the probability of a step $C \to C'$ is independent of the previous history.
- *bounded* if for every $n \geq 1$ and $\alpha > 0$ there is $c(n, \alpha) > 0$ with the following property. Consider any configuration with n agents and at least one message in transit. If the fraction of messages receivable by at least one agent is larger than α, the probability of receiving a message is at least $c(n, \alpha)$.
- *uniform* if for every configuration C and agent a, every message in transit that can be received by a at C is received with the same probability.

Remark 7.20 Each s:p/r:(1-p) model is positive, markovian, bounded, and uniform.

In the following constructions we again use single-agent configurations. We implicitly assume that an agent in a special state that can neither send nor receive is always added to the configuration to obtain a valid population.

Proposition 7.21 *There is a queued-transmission protocol $\mathscr{P}$ that p-computes the value 1 on a certain input in all positive, bounded, and markovian models, but that does not f-compute any value on this input.*

Proof Consider the protocol with states $\{q_0, q_1\}$; messages $M = \{a\}$; transitions $q_0 \xrightarrow{a+} q_0$ and $q_0 \xrightarrow{a-} q_1$; and output function given by $O(q_0) = O$ and $O(q_1) = 1$. Consider the input configuration $\langle q_0 \rangle$.

In this protocol, the unique agent sends messages until it receives a message and moves to q_1. Note that all the messages are receivable by the agent in state q_0. In any positive, bounded markovian model the agent eventually reaches q_1 with probability 1 and stays there. So the protocol p-computes the value 1 on input $\langle q_0 \rangle$. We show that the protocol does not f-compute any value on this input, because it has a fair execution converging to 0 and fair executions converging to 1. The fair executions converging to 1 are those in which the agent reaches q_1. The unique fair execution converging to 0 is the one in which the agent stays in q_0 forever. To prove that this execution is fair observe that (a) along the execution the number of messages grows continuously, and (b) every configuration reachable from a configuration of the execution with m messages in transit has at least $m - 1$ messages in transit. So no configuration of the protocol is reachable from infinitely many configurations of the execution. □

Proposition 7.22 *There is a queued-transmission protocol $\mathscr{P}$ that f-computes the value 1 on a certain input, but that does not p-compute any value on this input in any positive, markovian and uniform model.*

Proof Consider the protocol with states $\{q_0, q_1, q_2, q_0^+, q_1^+, q_2^+, q_3^+, q^-, q\}$, messages $M = \{p, m, c\}$, and transitions

$$q_0 \xrightarrow{p+} q_1 \qquad q_1 \xrightarrow{m+} q_2$$
$$q_2 \xrightarrow{p-} q_0^+ \qquad q_2 \xrightarrow{m-} q^-$$
$$q_0^+ \xrightarrow{c+} q_1^+ \qquad q_1^+ \xrightarrow{c+} q_2^+ \qquad q_2^+ \xrightarrow{c+} q_3^+$$
$$q_3^+ \xrightarrow{m-} q \qquad q^- \xrightarrow{p-} q$$
$$q \xrightarrow{c-} q_0$$

The output function maps q to 1 and all other states to 0. Consider the input configuration $\langle q_0 \rangle$.

In this protocol, starting from $\langle q_0 \rangle$, every configuration can reach the configuration $\langle q \rangle$ in which the unique agent is in state q, and there are no messages in transit. So every fair execution eventually reaches $\langle q \rangle$ and, since no message can be sent from q, stays in it forever. Therefore, the protocol f-computes the value 1 on input $\langle q_0 \rangle$. We now show that the protocol does not p-compute any value on the same input in any positive, markovian, uniform model. Indeed, after reaching the state q_0 the execution must proceed to reach the state q_2 creating two messages of types p and m. The only way to proceed is to receive either p or m, which in uniform models is equally likely. Afterwards, both p and m are consumed, and either three messages of type c or none are created. To proceed, the agent needs to receive a message of type c. The number of messages of type c follows a random walk with possible changes $+2$ and -1 until it tries to go below zero. There is a positive probability that it will never return to zero and grow linearly. In this case all the states will be observed infinitely many times, so the protocol does not compute any value. □

These propositions show that the correctness problem for probabilistic queued-transmission protocols cannot be reduced to the same problem for the fairness model. So in the queued-transmission model fairness does not capture useful probability 1 properties, which questions the interest of the fairness-based model in a probability-free model. At the same time, it opens the question of the decidability of correctness for probabilistic queued-transmission protocols. Cummings, Doty and Soloveichik have recently proved that Chemical Reaction Networks can compute with probability 1 a superset of the Turing-computable functions [23], and using this result we can easily prove that correctness is undecidable.

Theorem 7.23 *In any positive, markovian, and uniform probabilistic model, the single-instance correctness problem for queued-transmission protocols is undecidable.*

Proof We only sketch the argument. According to [23], binary chemical reaction networks with uniform rates can p-compute all recursively enumerable predicates (in fact even more, see [23]). In such a network we are initially given set of chemical reactions, like e.g. $A+B \rightarrow 2C+D+E$, a multiset of molecules of different species (A, B, C, …). At every step, two molecules are picked uniformly at random and allowed to interact according to one of the reactions, which results in an arbitrary number of product molecules. A binary chemical reaction network can be modelled by a queued-transmission protocol with a single agent. Molecules are modeled by messages. The agent sends an initial set of messages, which corresponds to the initial multiset of molecules, and moves to a new state, from which it repeatedly receives two randomly chosen messages, and sends the results of the reaction. At each stop the agent can either only send or only receive, and if it can receive it can receive any message. Uniformity and Markov property guarantee that each pair of messages is selected with equal probability regardless of the details of the model, and positivity ensures that the protocol will make progress in modelling the chemical reaction network. As every binary reaction network can be modeled in such a way, and the problem of checking whether a Turing machine computes the constant true function is undecidable, the result follows. □

8 Related models and approaches

We have studied the correctness problem for the population protocol models introduced by Angluin et al. in [7]. Section 2 of [7] presents a detailed comparison with other models, focusing on expressivity questions. In this section we discuss work on models that are related to those of [7], and moreover address verification questions.

The IO and DO observation models of [7] are closely related to Reconfigurable Broadcast Networks (RBN), introduced by Delzanno et al. in [27], and further studied in [11,26]. [4] In RBNs, networks of finite-state agents communicate through broadcast. The network is modeled as an undirected graph $G = (V, E)$, with an agent at each node of V. An agent in state q can execute a transition $q \xrightarrow{a!!} q'$, which broadcasts the message a to all neighbours, and updates the state of the agent to q'. All neighbours of the agent must react to the message according to transitions of the form $r \xrightarrow{a??} r'$ for every state r. The crucial feature of RBNs is that between any two broadcasts the network can nondeterministically reconfigure itself into *any* other network with the same set of nodes. This makes RBNs equivalent to symmetric, fully connected networks in which agents nondeterministically choose whether to react to a broadcast or not. Symmetry makes the agents indistinguishable, and so the configuration of an RBN is completely determined by the number of agents in each state. As a consequence, given an instance of an IO protocol with n agents, one can construct an equivalent RBN as follows. The network has n nodes. For every transition $q_1 \xrightarrow{q_2} q_3$ of the IO protocol, we add to the network transitions $q_2 \xrightarrow{a!!} q_2$, $q_1 \xrightarrow{a??} q_3$, and $q \xrightarrow{a??} q$ for every $q \neq q_1$. So IO protocols are a special case of RBNs. However, the analysis problems we study are more general than the ones studied in [11,26,27]. The parameterized reachability problem studied in [27] corresponds to the problem whether a given counting set is reachable from a cube $[L, U]$ such that $L(q) = 0$ and $U(q) \in \{0, \infty\}$ for every state q (i.e.,

---

[4] We thank an anonymous reviewer for pointing this out.

from configurations that can put arbitrarily many agents in some states, and no agent in others). We solve the more general problem of reachability between two arbitrary counting sets. Further, our solution allows us to prove that counting sets are closed under reachability, a question not considered in [27]. The results of [11] on minimal length of covering executions have the same flavour as our Shortening Theorem for IO, but only consider the case in which the configuration C to be covered satisfies $C(q) \in \{0, 1\}$ for every state q. We conjecture that at least some of our results extend to RBNs, and leave this question for future research.

The standard population protocol model is closely related to Petri nets and Vector Addition Systems. The decidability of correctness for PP is proved in [32] using results of Leroux and others on reversible and cyclic Petri nets [39,40]. The TOWER-hard lower bound is also proved in [32] by reduction to the reachability problem for Petri nets, which is shown to be TOWER-hard in [25]. Practical verification algorithms for PP have been given in [14,15,17]. The complexity of other verification problems beyond correctness is studied in [31].

Population protocols are also closely related to Chemical Reaction Networks [47]. Our result on the undecidability of correctness of queued-transmission protocols in positive, markovian and uniform probabilistic models is based on the results on the computational power of Chemical Reaction Networks by Cummings et al. [23].

After Angluin et al. proved in [7] that population protocols can only compute Presburger predicates, several models have been proposed that increase the expressive power. These include community protocols [36], passively mobile logarithmic space machines (PALOMA)[18], mediated protocols [44], clocked population protocols [8] and broadcast population protocols [16]. All these models can compute all predicates $\mathbb{N}^k \to \{0, 1\}$ in NSPACE($\log n$) or more, where n is the number of agents. This makes the correctness problem for all these models undecidable. To prove this we can for example reduce from the halting problem for Turing machines started on empty tape. Indeed, given a machine T, the predicate $\varphi_T(n)$ that holds for n if the computation of T on empty tape terminates and visits at most $\log n$ cells is a symmetric predicate in NSPACE($\log n$), and so it can be computed by a protocol. So T fails to terminate iff the protocol computes the false predicate.

From a verification point of view, the correctness problem for population protocols is a so-called parameterized verification problem, in which one has to show that a system of identical agents satisfies a property independently of the number of agents. Parameterized verification problems have been intensely studied, and we refer the reader to [1,12,30] for survey articles. Most work, however, concerns the verification of safety or liveness under adversarial schedulers; in other words, the property must hold even if the scheduler that selects which agents interact at each step tries to break

it. Correctness of population protocols is however a liveness property under stochastic schedulers, which choose the agents at random. This distinguishes our work from recent contributions to parameterized verification [38,41].

9 Conclusion

We have determined the computational complexity of the correctness problem for population protocols with different communication mechanisms, completing a research program initiated in [32]. We have followed the classification used by Angluin et al. in [7] to study the expressive power of the models.

Our main results concern the observation-based models IO and DO. A first surprise is the fact that checking correctness of a protocol *for all inputs* is not harder than checking it *for one input*. Further, both problems have the same complexity as many standard verification problems for concurrent systems, which are typically PSPACE-complete [46]. Moreover, our upper bounds are obtained by means of algorithms that suggest clean verification procedures. In particular, they show that the verification of properties of IO and DO protocols can be achieved by conducting symbolic state space exploration with counting sets represented by counting constraints. This opens the door to efficient implementations using SMT-solving technology [9].

From a more theoretical point of view, we have derived our upper bounds from a number of fundamental results about the dynamics of the IO and DO models. We have encapsulated them in the Pruning, Shortening, and Closure Theorems, which could be of independent interest. In particular, the connection between IO protocols and models for enzymatic reactions is intriguing [43].

The second surprise is the huge complexity gap between observation-based and transmission-based models. Thanks to the recent result by Czerwinski et al. [24], we can show that the correctness problem is TOWER-hard for all transmission-based models. This is in contrast with the limited computational power of the model, and raises the question whether there exists a natural model of computation by indistinguishable agents which is able to compute all Presburger predicates, and has a more manageable correctness problem. Another important insight is the fact that for all delayed-transmission models the problem is already TOWER-hard in the single-instance case. This already makes the application of model-checking technology to checking correctness for a few instances very difficult, and suggests a number of questions for further research.

Our investigation leaves one question open, namely whether the correctness problem is decidable for queued-transmission problems. We have explained that for this model the fairness assumption used by Angluin et al. in [7] is

questionable, since it can no longer be seen as an "over-approximation" of the probabilistic behavior of the system. However, settling the question can be relevant for stochastic models with assumptions concerning the size of the pool of messages.

A Appendix for Sect. 3.1

The following definition and lemma are introduced to help prove that our IO protocol implementation of a Turing machine does indeed simulate its functioning. To recall the notation, let us start with an illustration of transitions modelling a single step of the Turing machine into the protocol. The fragment of the protocol is represented as a diagram with some of the states and transitions of the IO protocol.

Figure 1 illustrates transitions involved in modelling a single step of a Turing machine that reads 0, writes 1, moves head to the right and switches the control state from q to q'.

Definition A.1 A configuration of $\mathscr{P}_M$ is a *modelling configuration* if the following conditions hold.

1. For every $1 \leq n \leq K$ exactly one of the $2|\Sigma|$ states $on[\sigma, n], off[\sigma, n]$ is populated, and it is populated with a single agent.
   (Intuitively: every cell is either *on* or *off* and contains exactly one symbol.)
2. Exactly one of all the head states is populated (again, with a single agent).
3. If a cell state $on[\sigma, n]$ is populated, then a head state $at[q, n]$ or $move[q, \sigma', n, d]$ is populated for some σ' and d.
4. If a head state $move[q, \sigma, n, d]$ is populated, either $on[\sigma', n]$ is populated for some σ', of $off[\sigma, n]$ is populated.

Remark A.2 Note that for every configuration c of M the configuration C_c described in Definition 3.1 is a modelling configuration.

Lemma A.3 *For every modelling configuration C of $\mathscr{P}_M$:*

(1) C enables at most one transition.
(2) If C enables no transitions, then it populates states $on[\sigma, n]$ and $at[q, n]$ for some $q \in Q$, $\sigma \in \Sigma$, and $1 \leq n \leq K$.
(3) If $C \rightarrow C'$, then C' is also a modelling configuration.

Proof (1) All possible transitions require agents at two states, one of type $on[\cdot, n]$ or $off[\cdot, n]$ and one of type $at[\cdot, n]$ or $move[\cdot, n, \cdot, \cdot]$, with the same n. But the modelling condition requires that there can be at most one such pair.
(2) If a $move[\cdot, \cdot, \cdot, \cdot]$ state is populated, a transition is always possible by definition of the list of $move[\cdot, \cdot, \cdot, \cdot]$ states. The same for the case where a $at[\cdot, \cdot]$ state is populated but no $on[\cdot, \cdot]$ case is populated. If there are populated states of types $on[\cdot, n]$ and $at[\cdot, n]$, the transition may fail to exist if either the Turing machine halts or if it goes outside the allocated space.
(3) Every transition consumes and produces one agent at $off[\cdot, n]$ or $on[\cdot, n]$ state, and the new state has the same n. Every transition consumes and produces one agent at $move[\cdot, \cdot, \cdot, \cdot]$ or $at[\cdot, \cdot]$ state. If an $on[\cdot, n]$ state becomes populated after a transition, it has the same n as the populated $at[\cdot, n]$ state of both configurations (before and after); if an $on[\cdot, n]$ state stays populated, the agent is moved from a $at[\cdot, n]$ to a $move[\cdot, n, \cdot, \cdot]$ state with the same n. When $move[q, \sigma, n, d]$ becomes populated, the transition needs a populated $on[\cdot, n]$ state. When $move[q, \sigma, n, d]$ stays populated, the transition populates a $off[\sigma, n]$ state. □

B Appendix for Sect. 3.2

Lemma B.1 *Let Γ be a circuit and let $\widehat{\mathscr{P}}_\Gamma$ be its evaluation protocol. Let C_0 be the initial configuration that puts exactly one agent in state $\iota(n)$ for every node n. A fair execution starting at C_0 eventually reaches a configuration C where each input agent is in a state with value 0 or 1, and these values do not change afterwards. The tail of the execution starting at C converges to a stable consensus equal to the output of Γ on these assigned inputs.*

Proof Every input node can change its own current value if it is $\square$ and cannot otherwise; by fairness of the execution and definition of how an input node can update its $\square$ value, the

input nodes will eventually all change their values from $\square$ and keep these afterwards.

By induction over the depth of an operation node, we can see that the value of each operation node eventually converges to the value of this node in circuit C, without ever holding the opposite value; moreover, once a node adopts a value, the value stays stable. Once the output value converges, each node will eventually learn it.

Notice that since the transitions of the circuit evaluation protocol always depend only on current values of nodes, the DO protocol cannot have a problem with lack of old messages. $\square$

C Appendix for Sect. 4.2

Lemma C.1 *Let $\mathscr{P}$ be an IO protocol. For every configuration C, C' the following holds: $C \xrightarrow{*} C'$ iff there exists a well-structured and realizable history in $\mathscr{P}$ with C and C' as initial and final configurations.*

Proof One direction is obvious by definition: if we have a realizable history, it also describes an execution. Let us prove the other direction.

Informally, we just implement the "de-anonymisation" of the agents, that is the assignment of a trajectory to each agent (in an arbitrary but consistent way). A formal proof can be given by induction in the number of transitions in the execution.

Base case. If there are no transitions, we create a multiset of trajectories of length one such that the initial states of the trajectories are exactly the states (with multiplicity) of the initial marking of the execution. This is well-structured because there are no steps.

Induction step. Consider a sequence of transitions and a corresponding well-structured history. Now let us add a single enabled transition. To build the new history, we choose an arbitrary trajectory of the existing history such that this trajectory ends in the state corresponding to the source state of the added transition. Such a trajectory exists because the transition is enabled and therefore its source state must be populated (one agent at least must be in the source state). We extend the chosen trajectory with a step from the source state to the destination state of the added transition, and we extend the rest of the trajectories with one horizontal step each. We obtain a multiset of trajectories of same length, thus constituting a history. It is realizable using the considered sequence of transitions followed by the new enabled transition. As we add only a single non-horizontal step at that moment of time, we do not break the well-structuring condition. $\square$

Lemma C.2 *4 Let $\mathscr{P}$ be an IO protocol. A well-structured history is realizable in $\mathscr{P}$ iff it is compatible with $\mathscr{P}$.*

Proof Let H be a well-structured history of $\mathscr{P}$.

Assume that H is realizable. Let $\tau \in H$, and let $\tau(i)\tau(i+1) = qq'$ be an arbitrary non-horizontal step of τ. Since H is well-structured, for every trajectory τ', if $\tau'(i)\tau'(i+1)$ is non-horizontal then $\tau'(i)\tau'(i+1) = qq'$. Since H is realizable, C_H^i enables a transition $q \xrightarrow{o} q'$ of $\mathscr{P}$. So $C_H^i(o) \geq 1$, and therefore there is a trajectory $\tau' \in H$ such that $\tau'(i) = o$. By the definition of step in IO protocols we have $\tau' \neq \tau$. Since H is well-structured, the i-th step of τ' is horizontal, and so $\tau'(i+1) = o$.

Now assume that H is compatible with $\mathscr{P}$. We prove that H is realizable by induction on the length n of H. If $n = 1$, there is nothing to show. If $n > 1$, let H' be the result of removing the last state from every trajectory of H. It follows immediately from the definitions that H' is compatible with $\mathscr{P}$. So there exist transitions $t_1, \ldots, t_{n-2}$ of $\mathscr{P}$ and numbers $k_1, \ldots, k_{n-2} \geq 0$ such that

$$C_H^1 \xrightarrow{t_1^{k_1}} C_H^2 \cdots C_H^{n-2} \xrightarrow{t_{n-2}^{k_{n-2}}} C_H^{n-1} .$$

We show that there is a transition t_n and $k_n \geq 0$ such that $C_H^{n-1} \xrightarrow{t_{n-1}^{k_{n-1}}} C_H^n$. Consider the last steps of all trajectories of H. If they are all horizontal, then $C_H^{n-1} = C_n$. So we can choose t_n as any transition of $\mathscr{P}$, and $k_n := 0$. If at least one of them is non-horizontal, let $s \geq 1$ be the number of non-horizontal steps. Since H is well-structured, all non-horizontal steps are equal, say $q\,q'$. Further, $\mathscr{P}$ has a transition $t = q \xrightarrow{o} q'$ and a trajectory $\tau' \neq \tau$ such that $\tau'(i) = \tau'(i+1) = o$. So we can choose $t_{n-1} := t$ and $k_{n-1} := s$. $\square$

Theorem C.3 (Quadratic Pruning Theorem) *Let $\mathscr{P}$ be an IO protocol, let L' and L be multisets of states of $\mathscr{P}$, and let $C' \xrightarrow{*} C$ be an execution of N such that $C' \geq L'$ and $C \geq L$. There exist configurations D' and D such that*

$$
\begin{array}{ccc}
C' & \xrightarrow{\ *\ } & C \\
\geq & & \geq \\
D' & \xrightarrow{\ *\ } & D \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|D'| = |D| \leq |L| + |L'| + 2|Q|^2$.

Proof The proof is similar to the proofs of Lemma 4.18 and Theorem 4.20. The main difference is the following. In Lemma 4.18 we keep trajectories that belong to small bunches, and prune each large bunch separately. To prove the quadratic lower bound we keep trajectories from and to small states, then prune all the remaining trajectories together. The

state is called small if it has less than $|Q|$ incoming or out-going trajectories.

Let $L' \leq C' \xrightarrow{*} C' \geq C$. By Lemma 4.12, there is a well-structured realizable history H with C' and C as initial and final configurations, respectively. Let $H_0 \subset H$ be an arbitrary minimal sub(multi)set of H with initial multiset of states at least L' and final multiset of states at least L. Let also $H' = H - H_0$. We further reduce H' by repeatedly removing all the trajectories with initial or final state having less than $|Q|$ trajectories still in H'. We can perform at most $2|Q|$ steps like that, removing at most $|Q| - 1$ trajectories per step. At the end, we will add back these trajectories as well as those of H_0.

Now we can define Q as the set of all states reached by the remaining trajectories in H', and $f(q)$ and $l(q)$ for $q \in Q$ as the earliest and the latest moment in time when this state has been used by any of the trajectories (possibly on different trajectories, and possibly on trajectories with different initial and final state).

We now build a trajectory for every $q \in Q$ by reaching it by the moment $f(q)$ and leaving it after $l(q)$. As all the trajectories in H' have initial and final state with at least $|Q|$ trajectories in H', the set of trajectories that we build will have the initial and final configurations covered by the corresponding configurations of H'.

The rest of the proof is identical to the proofs of Lemma 4.18 and Theorem 4.20. □

Lemma C.4 *Let $\mathscr{P}$ be an MFDO protocol. A well-structured history is realizable in $\mathscr{P}$ iff it is compatible with $\mathscr{P}$.*

Proof Let H be a well-structured history of $\mathscr{P}$.

Assume that H is realizable. Let $\tau \in H$, and let $\tau(i)\tau(i+1) = qq'$ be an arbitrary non-horizontal step of τ. Since H is well-structured, for every trajectory τ', if $\tau'(i)\tau'(i+1)$ is non-horizontal then $\tau'(i)\tau'(i+1) = qq'$. Since H is realizable, C_H^i enables a transition $q \xrightarrow{o} q'$ of $\mathscr{P}$, and so $o \in \mathscr{S}_H^i$. So H is compatible with $\mathscr{P}$.

Now assume that H is compatible with $\mathscr{P}$. We prove that H is realizable by induction on the length n of H. If $n = 1$, there is nothing to show. If $n > 1$, let H' be the result of removing the last state from every trajectory of H. It follows immediately from the definitions that H' is compatible with $\mathscr{P}$. So there exist transitions $t_1, \ldots, t_{n-2}$ of $\mathscr{P}$ and numbers $k_1, \ldots, k_{n-2} \geq 0$ such that

$$C_H^1 \xrightarrow{t_1^{k_1}} C_H^2 \cdots C_H^{n-2} \xrightarrow{t_{n-2}^{k_{n-2}}} C_H^{n-1}.$$

We show that there is a transition t_n and $k_n \geq 0$ such that $C_H^{n-1} \xrightarrow{t_{n-1}^{k_{n-1}}} C_H^n$. Consider the last steps of all trajectories of H. If they are all horizontal, then $C_H^{n-1} = C_n$. So we can choose t_n as any transition of $\mathscr{P}$, and $k_n := 0$. If at

least one of them is non-horizontal, let $s \geq 1$ be the number of non-horizontal steps. Since H is well-structured, all non-horizontal steps are equal, say $q\,q'$, and $\mathscr{P}$ has a transition $t = q \xrightarrow{o} q'$ such that $o \in \mathscr{S}_H^{n-1}$. So we can choose $t_{n-1} := t$ and $k_{n-1} := s$. □

Theorem C.5 (Linear MFDO Pruning) *Let $\mathscr{P} = (Q, \delta)$ be an MFDO protocol, let L' and L be multisets of states of $\mathscr{P}$, and let $C' \xrightarrow{*} C$ be an execution of $\mathscr{P}$ such that $C' \geq L'$ and $C \geq L$.*

$$
\begin{array}{ccc}
C' & \xrightarrow{*} & C \\
\geq & & \geq \\
D' & \xrightarrow{*} & D \\
\geq & & \geq \\
L' & & L
\end{array}
$$

and $|D'| = |D| \leq |L| + |L'| + |Q|$.

Proof Let H be a well-structured and realizable history for the execution $L' \leq C' \xrightarrow{*} C \geq L$. For every $q \in \mathscr{S}_H$, let $f(q)$ be the smallest index i such that $q \in \mathscr{S}_H^i$, that is, $f(q)$ is the earliest moment at which q is visited. Pick a trajectory τ_q of H such that q is reached by τ_q at moment $f(q)$ (we may pick the same trajectory for two different states).

Let H' be union of the set $\{\tau_q \mid q \in \mathscr{S}_H\}$ of trajectories, and an arbitrary sub(multi)set of trajectories of H such that the initial configuration of H' covers L' and the final configuration of H' covers L. We need at most $|L| + |L'|$ of these additional trajectories.

It follows immediately from the definition that H' is a history, covers L' and L by its initial and final configuration, and has at most $|Q| + |L| + |L'|$ trajectories. Let us show H' is well structured and realizable. By Lemma 4.28 it suffices to show that H' is well structured and compatible with $\mathscr{P}$. It is well-structured because $H' \subseteq H$, which is a well-structured history. Let us show that it is compatible with $\mathscr{P}$. By the definition of compatibility (Definition 4.26), and since $H' \subseteq H$, it suffices to show that $\mathscr{S}_H^i = \mathscr{S}_{H'}^i$ holds for every i. But this follows from the fact that, by the definition of H', each state is *first* visited in H' at the same moment that it is *first* visited in H. □

Example C.6 Consider the well-structured and realizable extended history of Fig. 4, leading from $(1, 4, 0)$ to $(0, 0, 5)$, which covers configuration $(0, 0, 2)$. The set of states visited by the trajectories is equal to Q. Figure 7 is annotated with the first moment $f(q)$ for every $q \in Q$. We pick the trajectories τ_a and τ_b drawn in dashed lines in Fig. 7, and choose $H' = \{\tau_a, \tau_b\}$. We have $C_{H'}^1 = (1, 1, 0) \xrightarrow{t_2 t_1} (0, 0, 2) = C_{H'}^4$ and $C_{H'}^4 \geq (0, 0, 2)$.

Theorem C.7 (Quadratic MFDO Shortening) *Let $\mathscr{P} = (Q, \delta)$ be an MFDO protocol, and let $C \xrightarrow{*} C'$ be an execution of $\mathscr{P}$. There exists a sequence ξ such that $C \xrightarrow{\xi} C'$ and $|\xi|_a \leq |Q|^2$.*

Proof *(Outline)* We optimise separately the construction for the last segment and for the other ones.

We require that:

(i) the trajectories of H'_j are in bijective correspondence with trajectories of H (and therefore H_j),

(ii) the set of visited states $\mathscr{S}_{H'_j}$ is the same as the set $\mathscr{S}_{H_j} = \mathscr{S}_H^{T_j}$ of states visited by H_j, and

(iii) for each trajectory $\tau' \in H'_j$ corresponding to $\tau \in H_j$, there is a path from final state of τ' to final state of τ visiting and observing only states in $\mathscr{S}_{H'_j}$.

In other words, instead of saying that the corresponding trajectories reach the same states, we say that the corresponding trajectories *could* reach the same states.

This can be maintained in the same way as in the proof of theorem 4.33 with two changes. We extend only one trajectory with the shortest path to the newly reachable state (this is possible because if state q' is reached by $\tau \in H_{j+1}$ starting from q at the moment T_j, the corresponding $\tau' \in H'_j$ can reach q and then q' from q). The remaining trajectories are extended with horizontal steps, and the new reachability requirement is also satisfied by transitivity of reachability.

There is a linear number of non-last segments, and each will correspond to a linear-length replacement segment. Therefore the aggregated length of all the segments (except the last) together is quadratic.

In the last segment we need to make all the trajectories to reach the final configuration of H from the final configuration in H'_n. Note that there is some feasible multiset of such trajectories because of the condition (iii). Also observe that as we don't change the set of visited states, the steps of the trajectories do not depend on each other.

Consider the multiset of trajectories leading from the final configuration of H'_n to the final configuration of H (maybe violating the initial trajectory correspondence) with the short-

est total number of steps across the trajectories. In such a multiset a union of all trajectories doesn't contain any cycles, as otherwise we would be able to cut and reconnect the trajectories to remove the steps along the cycle. Therefore we can consider topological ordering corresponding to the union of these trajectories. As each trajectory traverses the states in the ascending order according to the topological ordering, running the steps in the lexicographic order of the source and target states correctly traverses each trajectory. As all the equal steps are ran at the same time, we obtain quadratic aggregated length for the final segment. □

D Appendix for Sect. 5

Proposition D.1 ([33], Proposition 5) *Let Γ_1, Γ_2 be counting constraints.*

– *There exists a counting constraint Γ with $[\![\Gamma]\!] = [\![\Gamma_1]\!] \cup [\![\Gamma_2]\!]$ such that $\|\Gamma\|_u \leq \max\{\|\Gamma_1\|_u, \|\Gamma_2\|_u\}$ and $\|\Gamma\|_l \leq \max\{\|\Gamma_1\|_l, \|\Gamma_2\|_l\}$.*

– *There exists a counting constraint Γ with $[\![\Gamma]\!] = [\![\Gamma_1]\!] \cap [\![\Gamma_2]\!]$ such that $\|\Gamma\|_u \leq \|\Gamma_1\|_u + \|\Gamma_2\|_u$ and $\|\Gamma\|_l \leq \|\Gamma_1\|_l + \|\Gamma_2\|_l$.*

– *There exists a counting constraint Γ with $[\![\Gamma]\!] = \mathbb{N}^n \setminus [\![\Gamma_1]\!]$ such that $\|\Gamma\|_u \leq n\|\Gamma_1\|_l$ and $\|\Gamma\|_l \leq n\|\Gamma_1\|_u + n$.*

Proof *(Adapted from Proposition 5 of [33].)*
Union. Let Γ be the union of the two counting constraints Γ_1, Γ_2, i.e. the set of the cube representations of Γ_1 and of Γ_2. It is still a counting constraint as a set of cube representations, and the result follows from the definition of representation and norms.

Intersection. For this proof, we consider a cube representation (L, U) as a collection of constraints over $n = |Q|$ variables $x_1, \ldots, x_n$ of the form $[l_i \leq x_i]$ or $[x_i \leq u_i]$ with $l_i \in \mathbb{N}$ and $u_i \in \mathbb{N} \cup \infty$. Each variable x_i is associated to a state $q_i \in Q$, for an arbitrary ordering of Q, and it intuitively denotes the number of agents in q_i. A cube representation can now be seen as a conjunction of such constraints, one lower bound and one upper bound for each x_i for $i \in \{1, \ldots, n\}$. We call such a $2n$-conjunction a *minterm*. Counting constraints Γ_1, Γ_2 are thus disjunctions of minterms, noted γ_1, γ_2 respectively. The intersection of Γ_1, Γ_2 is the conjunction $\gamma_1 \wedge \gamma_2$.

We rearrange this conjunction into a disjunction of minterms by using the following steps: Put $\gamma_1 \wedge \gamma_2$ in disjunctive normal form. Remove conjunctions containing the unsatisfiable constraints $l \leq x_i \wedge x_i \leq u$ with $l > u$. Remove redundant constraints inside conjunctions, e.g., replace $(l_1 \leq x \wedge l_2 \leq x)$ by $\max\{l_1, l_2\} \leq x$. If a conjunction does not contain a lower bound (upper bound) for x_i, add $0 \leq x_i$ ($x_i \leq \infty$), thus making it a minterm. The disjunction of
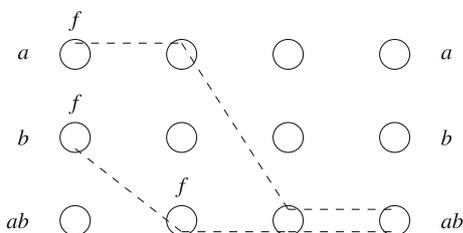
Fig. 7 History H of Fig. 4 after pruning

these minterms is the counting constraint Γ we are looking for, and the norm bounds follow from the fact that the new bounds are a mix of the old bounds.

Complement. We reuse the constraint and minterm formalism above. The complement is represented by the negation of the disjunction of minterms. We rearrange it into a disjunction of minterms using the rules above as well as $[\![\neg(x_i \le c)]\!] = [\![x_i \ge c + 1]\!]$; and $[\![\neg(x_i \ge c)]\!] = [\![x_i \le c - 1]\!]$ if $c \in \mathbb{N} \setminus \{0\}$ and remove the enclosing conjunction otherwise. We obtain n-conjunctions with lower bounds of the form $u + 1$, with $u \le \|\Gamma_1\|_u$ an upper bound in a minterm of the original constraint. This yields $\|\Gamma\|_l \le n\|\Gamma_1\|_u + n$ and the reasoning is similar for the u-norm. $\qquad\square$

Theorem 5.4 (IO Closure) *Let $\mathscr{P}$ be an IO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set of configurations of $\mathscr{P}$ represented by a counting constraint Γ. Then $pre^*(\mathscr{S})$ is also a counting set, and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre^*(\mathscr{S})$ and*

$$\|\Gamma'\|_u \le \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \le \|\Gamma\|_l + |Q|^3$$

The same holds for $post^*$.

Proof Consider a finite decomposition into cubes $\mathscr{S} = \cup_{i=1}^{k} \mathscr{C}_i$ of counting set $\mathscr{S}$, which exists by definition of a counting set.

Lemma 5.3 states that for every cube $\mathscr{C}$ of this decomposition, for every configuration C' in $pre^*(\mathscr{C})$, there is a "small" cube $\mathscr{C}'$ such that $C' \in \mathscr{C}'$ and $\mathscr{C}' \subseteq pre^*(\mathscr{C})$. So $pre^*(\mathscr{C}) = \cup_{C' \in pre^*(\mathscr{C})} \mathscr{C}'$. By the norm restrictions on the representation of $\mathscr{C}'$, there are only a finite number of such "small" cubes. So $pre^*(\mathscr{C})$ is a finite union of cubes, and by extension $pre^*(\mathscr{S}) = \cup_{i=1}^{k} pre^*(\mathscr{C}_i)$ is too. Thus by definition, $pre^*(\mathscr{S})$ is a counting set.

Let Γ be the counting constraint defined as the set of the representations of the $\mathscr{C}_i$. Let Γ' be the counting constraint defined as the set of the representations of the "small" cubes whose unions equal the $pre^*(\mathscr{C}_i)$. Then by the bounds in Lemma 5.3 and by definition of the norms, $\|\Gamma'\|_u \le \|\Gamma\|_u$ and $\|\Gamma'\|_l \le \|\Gamma\|_l + |Q|^3$.

The results also hold for $post^*(\mathscr{S})$, as the pruning theorem and our use of it are symmetric. $\qquad\square$

Lemma D.2 *Let $\mathscr{P}$ be an IO protocol with state set Q and let $\mathscr{C} \subseteq Pop(Q)$ be a cube. For all $C' \in pre^*(\mathscr{C})$, there exists a cube $\mathscr{C}'$ such that*

1. $C' \in \mathscr{C}' \subseteq pre^*(\mathscr{C})$, and
2. $\|\mathscr{C}'\|_l \le \|\mathscr{C}\|_l + |Q|^3$ and $\|\mathscr{C}'\|_u \le \|\mathscr{C}\|_u$.

Proof The proof is exactly the same as for Lemma 5.3, as adding a copy of a trajectory into a well-structured realizable

history produces a realizable history for MFDO protocols as well. $\qquad\square$

Theorem 5.6 (MFDO Closure) *Let $\mathscr{P}$ be an MFDO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set defined by a counting constraint Γ. Then $pre^*(\mathscr{S})$ is also a counting set and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre^*(\mathscr{S})$, and*

$$\|\Gamma'\|_u \le \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \le \|\Gamma\|_l + |Q|^3$$

The same holds for $post^*$.

Proof The proof is the same as for Theorem 5.4, except that Lemma 5.5 is used instead of Lemma 5.3. $\qquad\square$

Corollary D.3 (DO Closure) *Let $\mathscr{P}$ be a DO protocol with a set Q of states, and let $\mathscr{S}$ be a counting set of zero-message configurations defined by a counting constraint Γ. Then $pre_z^*(\mathscr{S})$ is also a counting set and there exists a counting constraint Γ' satisfying $[\![\Gamma']\!] = pre_z^*(\mathscr{S})$, and*

$$\|\Gamma'\|_u \le \|\Gamma\|_u \text{ and } \|\Gamma'\|_l \le \|\Gamma\|_l + |Q|^3$$

The same holds for $post_z^*$.

Proof The set $pre_z^*(\mathscr{S})$ is the set of zero-message configurations $Z' \in \mathscr{Z}$ such that there exists $Z \in \mathscr{S}$ and $Z' \xrightarrow{*} Z$ in DO protocol $\mathscr{P}$. By Lemma 4.4, $Z' \xrightarrow{*} Z$ in DO protocol $\mathscr{P}$ if and only if $Z' \xrightarrow{*} Z$ in the corresponding MFDO protocol. Since $\mathscr{S}$ can also be seen as a counting set of MFDO configurations (by our usual overloading of configurations of $\mathscr{Z}$), $pre_z^*(\mathscr{S})$ in $\mathscr{P}$ is equal to $pre^*(\mathscr{S})$ in the corresponding MFDO protocol. Thus we obtain the result by application of Theorem 5.6.

The result for $post_z^*(\mathscr{S})$ is proved in the same way. $\qquad\square$

E Appendix for Sect. 7.1

Proposition E.1 *For every unary-encoded VASS $\mathcal{N} = (Q, T)$ and unary-encoded configurations $(q_0, \mathbf{v}_0)$, $(q, \mathbf{v})$, one can construct in polynomial time a ± 1-VASS $\mathcal{N}' = (Q', T')$ with distinct states $r_0, r \in Q'$ such that*

$$(q_0, \mathbf{v}_0) \xrightarrow{*}_{\mathcal{N}} (q, \mathbf{v}) \iff (r_0, \mathbf{0}) \xrightarrow{*}_{\mathcal{N}'} (r, \mathbf{0}) .$$

Proof Let (Q, T) have dimension $k \in \mathbb{N}$. The ± 1-VASS (Q', T') of the same dimension is constructed as follows: add to Q new states r and r_0, and add to T the transitions $r_0 \xrightarrow{\mathbf{v}_0} q_0$ and $q \xrightarrow{-\mathbf{v}} r$. Then replace every transition of the form $q \xrightarrow{(w_1,...,w_m)} q'$ by

$$q \xrightarrow{(w_1, 0, \ldots, 0)} q_{w_1}$$
$$\xrightarrow{(0, w_2, 0, \ldots, 0)} q_{w_2}$$
$$\ldots$$
$$\xrightarrow{(0, \ldots, 0, w_{m-1}, 0)} q_{w_{m-1}}$$
$$\xrightarrow{(0, \ldots, 0, w_m)} q'.$$

where $q_{w_1}, \ldots, q_{w_{m-1}}$ are newly added states. Then replace every transition of the form $q \xrightarrow{\mathbf{0}} q'$ by $q \xrightarrow{1++} q'' \xrightarrow{1--} q'$, where q'' is a newly added state. Finally, replace every transition of the form $q \xrightarrow{\mathbf{w}} q'$ where $w_m \neq 0$ for a fixed $m \in \{1, \ldots, k\}$ by the following:

$$q \xrightarrow{m++} q_1 \xrightarrow{m++} q_2 \xrightarrow{m++} \ldots \xrightarrow{m++} q_{w_{m-1}} \xrightarrow{m++} q' \quad \text{if } w_i > 0,$$
$$q \xrightarrow{m--} q_1 \xrightarrow{m--} q_2 \xrightarrow{m--} \ldots \xrightarrow{m--} q_{w_{m-1}} \xrightarrow{m--} q' \quad \text{otherwise,}$$

where $q_1, \ldots, q_{w_{m-1}}$ are newly added states. The resulting construction is a ± 1-VASS and clearly satisfies the properties in our claim. $\qquad \square$

Proposition E.2 *For every nondeterministic DT protocol $\mathscr{P}$ there exists a deterministic DT protocol $\mathscr{P}'$ that computes the same predicate as $\mathscr{P}$. Moreover, $\mathscr{P}'$ can be constructed in polynomial time.*

Proof Fix some linear order $<$ on $Q \cup (M \times Q)$, and let $n \overset{\text{def}}{=} |Q \times M|$. We define the deterministic protocol $\mathscr{P}' = (Q', M', \delta'_s, \delta'_r, \Sigma \iota', o')$ as follows:

– $Q' \overset{\text{def}}{=} Q \times \{1, \ldots, n\} \times \{0, 1\}$. An agent in state $(q, i, b) \in Q'$ simulates an agent from $\mathscr{P}$ in state q, picks choice i to resolve nondeterminism, and may send an `increment` message if and only if $b = 1$.
– $M' \overset{\text{def}}{=} M \cup \{\texttt{increment}\}$
– δ'_s is defined as follows:
  – For every $q \in Q$ such that $\delta_s(q) = \{(m_1, q_1), \ldots, (m_k, q_k)\}$ for some $(m_1, q_1) < \ldots < (m_k, q_k)$, define:

  $$\delta'_s((q, i, 0)) \overset{\text{def}}{=} (m_j, (q_j, i, 0)) \text{ with } j$$
  $$= (i \bmod k) + 1 \text{ and } j > 0.$$

  This implements the resolution of nondeterminism for outgoing messages from M.
  – For every $(q, i, 1) \in Q'$, define:

  $$\delta'_s((q, i, 1)) \overset{\text{def}}{=} (\texttt{increment}, (q, i, 0))$$

  This enforces that whenever the last bit is set to 1, an agent will send an `increment` message exactly once.

– δ'_r is defined as follows:
  – For every $(q, m) \in Q \times M$ such that $\delta_s((q, m)) = \{q_1, \ldots, q_k\}$ for some $q_1 < \ldots < q_k$, and every i, b, define:

  $$\delta'_r((q, i, b), m) \overset{\text{def}}{=} (q_j, i, b) \text{ with } j = (i \bmod k) + 1$$

  This resolves the nondeterminism for incoming messages from M.
  – Define for every $(q, i, b) \in Q'$:

  $$\delta'_r((q, i, b), \texttt{increment}) \overset{\text{def}}{=} (q, (i \bmod n) + 1, 1)$$

  This implements the incrementation of the round counter after receiving an `increment` message. Moreover, b is set to 1, so that at least one `increment` will eventually be put back into the message pool.

– $\iota'(a) \overset{\text{def}}{=} (\iota(a), 1, 1)$ for every $a \in \Sigma$.
– $o'((q, i, b)) = o(q)$ for every $(q, i, b) \in Q'$.

$\mathscr{P}'$ can be constructed in polynomial time. It remains to prove that $\mathscr{P}$ and $\mathscr{P}'$ compute identical predicates. To this end, fix some input $X \in \text{Pop}(\Sigma)$ and let $b \in \{0, 1\}$. We must show that every fair execution of $\mathscr{P}$ starting in $I(X)$ stabilizes to b if and only if every fair of $\mathscr{P}'$ starting in $I'(X)$ stabilizes to b. Before we prove this equivalence, let us introduce some notation. For every $C \in \text{Pop}(Q')$, we define the projection $\pi(C) \in \text{Pop}(Q)$ through

$$\pi(C)(q) \overset{\text{def}}{=} \sum_{(i,b) \in \{1, \ldots, n\} \times \{0,1\}} C((q, i, b)).$$

For a given $C \in \text{Pop}(Q')$, we write $C(b = 1)$ as shorthand for

$$\sum_{(q,i,1) \in Q'} C((q, i, b)).$$

We make the following observations that easily follow from the construction of $\mathscr{P}'$:

1. For every $C, C' \in \text{Pop}(Q')$ such that $C(b = 1) > 0 \vee C(\texttt{increment}) > 0$ and $C \xrightarrow{*} C'$, it must hold that $C'(b = 1) > 0 \vee C'(\texttt{increment}) > 0$.
2. For every $C \in \text{Pop}(Q')$ and every $C' \in \text{Pop}(Q)$, we have: If $\pi(C) \xrightarrow{*} C'$ and $C(b = 1) > 0 \vee C(\texttt{increment}) > 0$, then there exists some $C'' \in \text{Pop}(Q')$ satisfying $\pi(C'') = C'$ and $C \xrightarrow{*} C'$.
3. For every $C, C' \in \text{Pop}(Q')$ such that $C \to C'$, we have $\pi(C) \xrightarrow{*} \pi(C')$.

4. For every $C' \in \text{Pop}(Q')$ and every $C \in \text{Pop}(Q)$, $\{C'' \in \text{Pop}(Q') \mid C' \xrightarrow{*} C'' \wedge \pi(C'') = C\}$ is finite.

Let us now prove the equivalence.

($\Leftarrow$) Let $C'_0, C'_1, C'_2, \ldots$ be a fair execution of $\mathscr{P}'$ starting in $C'_0 = I'(X)$. By definition of ι', we have $C'_0(b = 1) > 0$. By Observation 1, this gives

$$C'_i(b = 1) > 0 \vee C'_i(\texttt{increment}) > 0 \text{ for every } i. \qquad (5)$$

Now consider the sequence of configurations $C_0, C_1, C_2, \ldots = \pi(C'_0), \pi(C'_1), \pi(C'_2), \ldots$, and let $\hat{i}_1 < \hat{i}_2 < \hat{i}_3 < \ldots$ be the maximal sequence of indices such that $C_{\hat{i}_1} = C_0$ and $C_{\hat{i}_1} \to C_{\hat{i}_2} \to \ldots$. By Observation 3, such a sequence of indices exists. Moreover $\rho = C_{\hat{i}_1} C_{\hat{i}_2} C_{\hat{i}_3} \ldots$ is a fair execution of $\mathscr{P}$: By (5), Observation 2, and Observation 4, for every C that can be reached from infinitely many configurations in ρ, there exists a configuration $C' \in \pi^{-1}(C)$ that can be reached from C'_i for infinitely many i. By fairness of $C'_0, C'_1, C'_2, \ldots$, we thus obtain that every configuration which can be reached infinitely often in ρ is reached infinitely often. Hence ρ is fair. Moreover, by definition of o', $C'_0, C'_1, C'_2, \ldots$ and ρ converge to the same consensus. This proves the direction ($\Leftarrow$).

($\Rightarrow$). The converse direction can be proven analogously. $\square$

We prove the claim made in the proof of Theorem 7.5.

Proposition E.3 *For every delayed-transmission protocol $\mathscr{P}$ and every initial configuration C of $\mathscr{P}$, one can construct in polynomial time a delayed-transmission protocol $\mathscr{P}'$ such that $\mathscr{P}'$ computes the constant predicate 1 if and only if $\mathscr{P}$ stabilizes to 1 for the single instance C.*

Proof Fix $\mathscr{P} = (Q, M, \delta_r, \delta_s, \Sigma, \iota, o)$ and C. Let $\mathsf{C} \stackrel{\text{def}}{=} \{C' \in \text{Pop}(Q) \mid C' \leq C\}$.

Each agent in $\mathscr{P}'$ carries a state of $\mathscr{P}$ and simulates interactions from $\mathscr{P}$. Moreover, each agent carries a boolean flag $b \in \{0, 1\}$. The flag b indicates that the initial configuration is $\geq C$. Initially, b is set to 0 for every agent. If b is equal to 1 and the agent carries some state $q \in Q$, its opinion is equal to $o(q)$. If $b = 0$, the agent has opinion 1. This ensures that the computation stabilizes to 1 if the initial configuration is strictly smaller than C.

In order to be able to detect whether the initial configuration is $\geq C$, the agents additionally store configuration from C. Initially, if an agent carries the state q from Q, it stores the configuration $\{q\}$. Agents can transfer states from one stored configuration to another agent through message passing. If the initial configuration is equal to C in $\mathscr{P}$, by fairness a single agent will eventually store C in the corresponding execution of $\mathscr{P}'$, while all other agents store an empty configuration. When an agent stores C, it knows that the initial

configuration must be $\geq C$, and in this case it is allowed to send a message that flips the flag b of any receiving agent to 1, and by fairness, eventually all agents have their flag b set to 1. If the initial configuration is $> C$, then at some point a state is transferred to an agent that already stores C. Such an agent assumes an error state, say $\top$, that maps to opinion 1, and eats up all other states via message passing. This ensures that every execution starting in a configuration $> C$ stabilizes to 1.

Formally, the delayed-transmission protocol $\mathscr{P}' = (Q', M', \delta'_r, \delta'_s, \Sigma, \iota', o')$ is constructed as follows:

– $Q' \stackrel{\text{def}}{=} (Q \times \mathsf{C} \times \{0, 1\}) \cup \{\top\}$
– $M' \stackrel{\text{def}}{=} M \cup \{\top, \texttt{one}\} \cup Q$
– δ'_r is defined as follows:

  – For every transition $q \xrightarrow{m} r$ in δ and every $C \in \mathsf{C}$ and every $b \in \{0, 1\}$, add:

  $$(q, C, b) \xrightarrow{m} (r, C, b)$$

  – For every $(q, C, b) \in Q \times \mathsf{C} \times \{0, 1\}$, add:

  $$(q, C, b) \xrightarrow{\texttt{one}} (q, C, 1)$$

  – For every $m \in Q$ and every $(q, C', b) \in Q \times \mathsf{C} \times \{0, 1\}$ s.t. $(C' + \{q\}) \in \mathsf{C}$, add:

  $$(q, C', b) \xrightarrow{m} (q, C' + \{q\}, b)$$

  – All remaining transitions to be defined transition to $\top$.

– δ'_s is defined as follows:

  – For every $(q, C', b) \in Q \times \mathsf{C} \times \{0, 1\}$ and every $q' \in Q$ such that $\{q'\} \leq C'$, add:

  $$(q, C', b) \xrightarrow{q'} (q, C' - \{q'\}, b)$$

  – For every $q \in Q$, and every $b \in \{0, 1\}$, add:

  $$(q, C, b) \xrightarrow{\texttt{one}} (q, C, b)$$

  – For every transition $q \xrightarrow{m} r$ in δ_s, and every $C' \in \mathsf{C}$, add:

  $$(q, C', 1) \xrightarrow{m} (r, C', 1)$$

  – Further add:

  $$\top \xrightarrow{\top} \top$$

- $\iota'(a) \stackrel{\text{def}}{=} (\iota(a), \langle\iota(a)\rangle, 0)$ for every $a \in \Sigma$.
- Set $o'(q, C, 1) \stackrel{\text{def}}{=} O(q)$ for every $(q, C) \in Q \times C$, and $o'(\mathbf{q}) \stackrel{\text{def}}{=} 1$ for every other $\mathbf{q}$. □

We prove the claim made in the proof of Theorem 7.6.

Proposition E.4 *Let $\mathcal{N} = (Q, T)$ be a ± 1-VASS and let $r_0, r \in Q$. It is possible to construct in polynomial time an immediate-transmission protocol $\mathcal{P}$ that computes constant 1 if and only if $(r_0, \mathbf{0}) \stackrel{*}{\to} (r, \mathbf{0})$ does not hold.*

Proof Let the dimension of $\mathcal{N} = (Q, T)$ be k. Like in the proof of Proposition 7.3, we represent the control state of $\mathcal{N}$ in a single agent. The remaining agents either represent a reservoir of tokens by assuming states of the form $\texttt{free}_i$ or $\texttt{token}_i$ for every vector component $1 \leq i \leq k$, or they are of the form $t, \mathbf{t}$ for any given $t \in T$, or in some additional helper state. A configuration $q, \mathbf{v}$ of $\mathcal{N}$ is represented in a configuration C of $\mathcal{P}$ satisfying

$$C(q) = 1,$$
$$C(q') = 0 \qquad \text{for every } q' \in Q \setminus \{q\},$$
$$C(\texttt{token}_i) = v_i \qquad \text{for every } 1 \leq i \leq k.$$

The states $\texttt{free}_i, \mathbf{t}, t$ for every $t \in T$, and other helper states, may be populated by arbitrarily many agents. When the control agent interacts with an agent of the form $\mathbf{t}$, a transition of $\mathcal{N}$ is simulated. For example, a transition $t \in T$ of the form $q \stackrel{i++}{\longrightarrow} q'$ is implemented in $\mathcal{P}$ by a sequence of two transitions, namely $(t, q) \to (\mathbf{t}, t)$ followed by $(t, \texttt{free}_i) \to (q', \texttt{token}_i)$ Similarly, a transition $t \in T$ of the form $q \stackrel{i--}{\longrightarrow} q'$ is implemented in $\mathcal{P}$ by the sequence consisting of $(t, q) \to (\mathbf{t}, t)$ followed by $(t, \texttt{token}_i) \to (q', \texttt{free}_i)$. Thus, incrementation at position i is implemented by turning $\texttt{free}_i$ into $\texttt{token}_i$, and symmetrically, decrementation is implemented by transforming $\texttt{token}_i$ into $\texttt{free}_i$. Initially, no agent is in a state of the form $\texttt{token}_i$, which reflects the fact that the initial vector of $\mathcal{N}$ in the reachability query equals $\mathbf{0}$.

Moreover, there are states $\texttt{final}$ and $\overline{\texttt{final}}$. When the agent representing the control state of $\mathcal{N}$ assumes r, it can non-deterministically guess that the current vector is $\mathbf{0}$, and signal this guess via transitioning to state $\texttt{final}$ through the step $\overline{\texttt{final}}, r \to \overline{\texttt{final}}, \texttt{final}$. The state $\texttt{final}$ is the only state that maps to false. If the guess was right, then the agent permanently remains in state $\texttt{final}$, and thus the protocol does not compute constant 1. If the guess was wrong, then by fairness the agent eventually meets some agent in state $\texttt{token}_i$ for some i, and then turns into some error state, say $\top$, that maps to true and that converts all other states to $\top$, thus ensuring that the protocol eventually stabilizes to 1.

Formally, we define $\mathcal{P} = (Q^{\mathcal{P}}, \delta^{\mathcal{P}}, I^{\mathcal{P}}, O^{\mathcal{P}})$ as follows:

- We add the following states to $Q^{\mathcal{P}}$:
  - For every $1 \leq i \leq k$, add states $\texttt{free}_i$ and $\texttt{token}_i$.
  - Add an "error" state $\top$.
  - Add "final" states $\texttt{final}, \overline{\texttt{final}}$.
  - For every state $q \in Q$, add q to $Q^{\mathcal{P}}$.
  - For every transition $t \in T$, add t and $\mathbf{t}$.
- We define $\delta^{\mathcal{P}}(x, y) = (\delta_1(x), \delta_2(x, y))$ by adding the following transitions:
  - For every $t: q_1 \to q_2 \in T$, add:
    $$\mathbf{t}, q_1 \to \mathbf{t}, t.$$
  - For every $t: q_1 \stackrel{i++}{\longrightarrow} q_2$, add:
    $$t, \texttt{free}_i \to q_2, \texttt{token}_i.$$
  - For every $t: q_1 \stackrel{i--}{\longrightarrow} q_2$, add:
    $$t, \texttt{token}_i \to q_2, \texttt{free}_i.$$
  - Add:
    $$\overline{\texttt{final}}, r \to \overline{\texttt{final}}, \texttt{final}$$
  - For every $x \in Q^{\mathcal{P}}$, add:
    $$\top, x \to \top, \top.$$
    This ensures that the $\top$ eats up every other state.
  - For every $t: q_1 \stackrel{i++}{\longrightarrow} q_2$, and every $x \neq \texttt{free}_i$, add:
    $$t, x \to q_2, \top.$$
  - For every $t: q_1 \stackrel{i--}{\longrightarrow} q_2$, and every $x \neq \texttt{token}_i$, add:
    $$t, x \to q_2, \top.$$
  - For every $1 \leq i \leq k$, add:
    $$\texttt{token}_i, \texttt{final} \to \texttt{token}_i, \top.$$
    This ensures that an agent only remains in $\texttt{final}$ if the current marking is $\mathbf{0}$, otherwise everyone is sent to $\top$.
  - For every $x, y \in Q \cup T \cup \{\texttt{final}\}$, set:
    $$\delta_2(x, y) = \top.$$
    This ensures that at most one agent is in a control state of $\mathbb{N}$, otherwise everyone is sent to $\top$.
  - In all remaining cases, set $\delta_1(x) = x$ and $\delta_2(x, y) = y$

- $I \stackrel{\text{def}}{=} \{r_0\} \cup \{\texttt{free}_i \mid 1 \le i \le k\}$
- $O(\texttt{final}) \stackrel{\text{def}}{=} 0$ and $O(x) = 1$ for every $x \ne \texttt{final}.\square$

F Appendix for Sect. 7.3

Proposition F.1 *There is a delayed-transmission protocol $\mathscr{P}$ that p-computes the value 0 on a certain input in every s:p/r:(1-p) model with $p > 1/2$, but that does not f-compute any value on the same input.*

Proof Recall the given protocol with states $\{q_0, q_1, q_2\}$; output function given by $O(q_0) = O(q_1) = 0$ and $O(q_2) = 1$; messages $\{a, b\}$; transitions

$$q_0 \xrightarrow{a+} q_0 \quad q_1 \xrightarrow{b+} q_0 \quad q_2 \xrightarrow{b+} q_1$$
$$q_0 \xrightarrow{a-} q_0 \quad q_1 \xrightarrow{a-} q_1 \quad q_2 \xrightarrow{a-} q_2$$
$$q_0 \xrightarrow{b-} q_1 \quad q_1 \xrightarrow{b-} q_2 \quad q_2 \xrightarrow{b-} q_2$$

and the input configuration $\wr q_0 \wr$.

In each configuration of each execution the sum of the index of the state and the number of messages of type b is equal to 2. This protocol does not f-compute any value on $\wr q_0 \wr$ because the configuration with no messages and the agent in the state q_2 is reachable from each configuration in the execution, as well as the configuration with 2 messages of type b and the agent in the state q_0. These two configurations occur infinitely often in each fair execution and have different output values.

The proof that an execution from $\wr q_0 \wr$ converges with probability 1 if $p > \frac{1}{2}$ is based on the following observations.

- The number of messages changes independently of the configuration change, so it is a biased random walk with linear growth.
- The state q_0 can always be reached with probability at least 1/4, and so it is reached infinitely many times.
- Going from q_0 to q_2 requires receiving two bs without sending in-between.
- The probability to receive two bs is proportional to $1/n^2$, where n is the number of messages. Since the series $\sum_{i=1}^{\infty} 1/n^2$ converges, so with probability 1 the state q_2 is only observed a finite number of times.

Consider a step $C \to C'$. If C has no messages in transit, then the number or messages increases by 1; otherwise there is probability $p > \frac{1}{2}$ to increase the number of messages by 1 and probability $1 - p$ to decrease the number of messages. This is a biased random walk. Let X_i be the random variable equal to the number of messages at the i-th step. The expected value of X_i is $(2p - 1)i$ and the standard deviation grows proportionally to $\sqrt{i}$, and so in

particular $\lim_{c \to +\infty} \Pr[\forall i > c : X_i > (p - \frac{1}{2})i] = 1$. For any given configuration the probability of reaching a configuration with state q_0 is 1 (it is enough to send a message two times in a row which has probability larger than $\frac{1}{4}$ at each step), therefore state q_0 occurs infinitely often with probability 1. To reach state q_2 from state q_1 before reaching q_0 the agent needs to receive messages without transmitting until it receives the only message of type b. The probability of reaching q_2 from q_1 before either returning to q_0 or getting below k messages is less than $\sum_{j=1}^{\infty}(1 - p)^j \frac{1}{k} < \sum_{j=1}^{\infty} (\frac{1}{2})^j \frac{1}{k} = \frac{1}{k}$. Note that probability of the transition from q_0 to q_1 with at least k messages in transit is at most $(1 - p)\frac{2}{k} < \frac{1}{k}$. Therefore the probability of the agent starting at q_0 and reaching q_2 before either returning to q_2 or having fewer than k messages is at most $\frac{1}{k^2}$.

Let $N_{q_0 \to q_2}^j$ be the random variable equal to 1 if q_2 is visited after the j-th visit to q_0, before q_0 is visited again, and before the number of messages in transit goes below $(p - \frac{1}{2})j$ (and 0 otherwise). Let $N_{q_0 \to q_2}$ be the sum $\sum_{j=1}^{\infty} N_{q_0 \to q_2}^j$. We have shown that $E\left(N_{q_0 \to q_2}^j\right) \in O(\frac{1}{((p-\frac{1}{2})j)^2})$ and therefore $E\left(N_{q_0 \to q_2}\right) = c \cdot \sum_{j=0}^{\infty} \frac{1}{((p-\frac{1}{2})j)^2}$ for some constant c, which is finite.

Consider executions that reach q_2 after at least N different returns to q_0. Such an execution must either have the number of messages go below $(p - \frac{1}{2})i$ at the moment $i > \frac{N}{2}$, or have the value $N_{q_0 \to q_2}$ to be at least $\frac{N}{2}$. The probability of either option tends to zero when N grows. Therefore q_2 occurs only a finite number of times with probability 1, and so the execution converges to 0. So the protocol computes the value 0 on input $\wr q_0 \wr$. $\square$

References

1. Abdulla, P.A., Prasad Sistla, A., Talupur, M.: Model checking parameterized systems. In: Handbook of Model Checking, pp. 685–725. Springer (2018)
2. Alistarh, D., Aspnes, J., Eisenstat, D., Gelashvili, R., Rivest, R.L.: Time-space trade-offs in population protocols. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2560–2579 (2017)
3. Alistarh, D., Aspnes, J., Gelashvili, R.: Space-optimal majority in population protocols. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2221–2239 (2018)
4. Alistarh, D., Gelashvili, R.: Recent algorithmic advances in population protocols. SIGACT News **49**(3), 63–73 (2018)
5. Anglin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In Proceedings of the 23$^{\text{rd}}$ Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 290–299 (2004)
6. Anglin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006)

7. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distrib. Comput. **20**(4), 279–304 (2007)

8. Aspnes, J.: Clocked population protocols. In: PODC, pp. 431–440. ACM (2017)

9. Barrett, C.W., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Model Checking, pp. 305–343. Springer (2018)

10. Berman, L.: The complexity of logical theories. Theor. Comput. Sci. **11**(1), 71–77 (1980)

11. Bertrand, N., Bouyer, P., Majumdar, A.: Reconfiguration and message losses in parameterized broadcast networks. In: CONCUR, volume 140 of LIPIcs, pp. 32:1–32:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)

12. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers (2015)

13. Blondin, M., Esparza, J., Genest, B., Helfrich, M., Jaax, S.: Succinct population protocols for presburger arithmetic. In: STACS, volume 154 of *LIPIcs*, pp. 40:1–40:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

14. Blondin, M., Esparza, J., Helfrich, M., Kucera, A., Meyer, P.J.: Checking qualitative liveness properties of replicated systems with stochastic scheduling. In CAV (2), volume 12225 of Lecture Notes in Computer Science, pp. 372–397. Springer (2020)

15. Blondin, M., Esparza, J., Jaax, S.: Peregrine: A tool for the analysis of population protocols. In: CAV (1), volume 10981 of Lecture Notes in Computer Science, pp. 604–611. Springer (2018)

16. Blondin, M., Esparza, J., Jaax, S.: Expressive power of broadcast consensus protocols. In CONCUR, volume 140 of LIPIcs, pp. 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)

17. Blondin, M., Esparza, J., Jaax, S., Meyer, P.J.: Towards efficient verification of population protocols. In PODC, pages 423–430. ACM (2017)

18. Chatzigiannakis, I., Michail, O., Nikolaou, S., Pavlogiannis, A., Spirakis, P.G.: Passively mobile communicating logarithmic space machines. Technical report (2010)

19. Chatzigiannakis, I., Michail, O., Spirakis. P.G.: Algorithmic verification of population protocols. In: SSS '10, volume 6366 of LNCS, pp. 221–235. Springer (2010)

20. Cheng, A.: Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. Theor. Comput. Sci. **147**(1 & 2), 117–136 (1995)

21. Clement, J., Delporte-Gallet, C., Fauconnier, H., Sighireanu, M.: Guidelines for the verification of population protocols. In: ICDCS '11, pp. 215–224 (2011)

22. Cooper, D.C.: Theorem proving in arithmetic without multiplication. Mach. Intell. **7**, 91–99 (1972)

23. Cummings, R., Doty, D., Soloveichik, D.: Probability 1 computation with chemical reaction networks. Natural Comput. **15**(2), 245–261 (2016)

24. Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., Mazowiecki, F.: The reachability problem for petri nets is not elementary. In: STOC, pp. 24–33. ACM (201)

25. Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., Mazowiecki, F.: The reachability problem for petri nets is not elementary. In: Moses, C., Edith, C., (eds.) Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019, pp. 24–33. ACM (2019)

26. Delzanno, G., Sangnier, A., Traverso, R.: Adding data registers to parameterized networks with broadcast. Fundam. Informaticae **143**(3–4), 287–316 (2016)

27. Delzanno, G., Sangnier, A., Traverso, R., Zavattaro, G.: On the complexity of parameterized reachability in reconfigurable broadcast networks. In: FSTTCS, volume 18 of *LIPIcs*, pp. 289–300. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2012)

28. Elsässer, R.: Radzik, T: Recent results in population protocols for exact majority and leader election. Bull. EATCS **2018**, 126 (2018)

29. Esparza, J.: Decidability and complexity of petri net problems-an introduction. In: P, N., volume 1491 of Lecture Notes in Computer Science, pp. 374–428. Springer (1996)

30. Esparza, J.: Parameterized verification of crowds of anonymous processes. In: Dependable Software Systems Engineering, volume 45 of NATO Science for Peace and Security Series - D: Information and Communication Security, pp. 59–71. IOS Press (2016)

31. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Model checking population protocols. In FSTTCS, volume 65 of LIPIcs, pp. 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2016)

32. Esparza, J., Ganty, P., Leroux, J., Majumdar, R.: Verification of population protocols. Acta Informatica **54**(2), 191–215 (2017)

33. Esparza, J., Ganty, P., Majumdar, R., Weil-Kennedy, C.: Verification of immediate observation population protocols. In: CONCUR, volume 118 of LIPIcs, pp. 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)

34. Esparza, J., Raskin, M.A., Weil-Kennedy, C.: Parameterized analysis of immediate observation petri nets. In: Petri Nets, volume 11522 of Lecture Notes in Computer Science, pp. 365–385. Springer (2019)

35. Fischer, M.J., Rabin, M.O.: Super-exponential complexity of Presburger arithmetic. Technical report, MIT (1974)

36. Guerraoui, R., Ruppert, E.: Even small birds are unique: Population protocols with identifiers (2007)

37. Haase, C.: A survival guide to presburger arithmetic. ACM SIGLOG News **5**(3), 67–82 (2018)

38. Lengál, O., Lin, A.W., Majumdar, R., Rümmer, P.: Fair termination for parameterized probabilistic concurrent systems. In: TACAS (1), volume 10205 of Lecture Notes in Computer Science, pp. 499–517 (2017)

39. Leroux, J.: Presburger vector addition systems. In: LICS, pp. 23–32. IEEE Computer Society (2013)

40. Leroux, J.: Vector addition system reversible reachability problem. Log. Methods Comput. Sci. **9**, 1 (2013)

41. Lin, A.W., Rümmer, P.: Liveness of randomised parameterised systems under arbitrary schedulers. In: CAV (2), volume 9780 of Lecture Notes in Computer Science, pp. 112–133. Springer (2016)

42. Lipton, R.: The reachability problem is exponential-space hard. Technical Report 62, Department of Computer Science, Yale University, January (1976)

43. Marwan, W., Wagler, A., Weismantel, R.: Petri nets as a framework for the reconstruction and analysis of signal transduction pathways and regulatory networks. Nat. Comput. **10**(2), 639–654 (2011)

44. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: Mediated population protocols. Theor. Comput. Sci. **412**(22), 2434–2450 (2011)

45. Pang, J., Luo, Z., Deng, Y.: On automatic verification of self-stabilizing population protocols. In: TASE '08, pp. 185–192. IEEE Computer Society (2008)

46. Piterman, N., Pnueli, A.: Temporal logic and fair discrete systems. In: Handbook of Model Checking, pp. 27–73. Springer (2018)

47. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Nat. Comput. **7**(4), 615–633 (2008)

48. Stockmeyer, L.J.: The polynomial-time hierarchy. Theor. Comput. Sci. **3**(1), 1–22 (1976)

49. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: towards flexible verification under fairness. In: CAV '09, volume 5643 of LNCS, pp. 709–714. Springer (2009)