Population Protocols with Unordered Data

Michael Blondin
Department of Computer Science, Université de Sherbrooke, Canada

François Ladouceur
Department of Computer Science, Université de Sherbrooke, Canada

Abstract

Population protocols form a well-established model of computation of passively mobile anonymous agents with constant-size memory. It is well known that population protocols compute Presburger-definable predicates, such as absolute majority and counting predicates. In this work, we initiate the study of population protocols operating over arbitrarily large data domains. More precisely, we introduce *population protocols with unordered data* as a formalism to reason about anonymous crowd computing over unordered sequences of data. We first show that it is possible to determine whether an unordered sequence from an infinite data domain has a datum with absolute majority. We then establish the expressive power of the "immediate observation" restriction of our model, namely where, in each interaction, an agent observes another agent who is unaware of the interaction.

2012 ACM Subject Classification Theory of computation → Distributed computing models; Theory of computation → Automata over infinite objects

Keywords and phrases population protocols, unordered data, colored Petri nets

Funding *Michael Blondin*: Supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fonds de recherche du Québec – Nature et technologies (FRQNT).
François Ladouceur: Supported by a scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the Fondation J.A. De Sève.

Acknowledgements We thank Manuel Lafond for his ideas and feedback in the early phase of our research. We further thank the anonymous reviewers for their comments and insightful suggestions.

1 Introduction

Context. Population protocols form a well-established model of computation of passively mobile anonymous agents with constant-size memory [1]. Population protocols allow, *e.g.*, for the formal analysis of chemical reaction networks and networks of mobile sensors (see [23] for a review article on population protocols and more generally on dynamic networks).

In a population protocol, anonymous agents hold a mutable state from a finite set. They collectively seek to evaluate a predicate on the initial global state of the population. At each discrete moment, a scheduler picks two agents who jointly update their respective states according to their current states. Such a scheduler is assumed to be "fair" (or, equivalently, to pick pairs of agents uniformly at random). Let us illustrate the model with a classical protocol for the aboslute majority predicate. Consider a population of ℓ (anonymous) agents, each initialized with either Y or N, that seek to compute whether the number of Y exceeds the number of N, *i.e.*, to collectively evaluate the predicate $\varphi(\#Y, \#N) \coloneqq (\#Y > \#N)$. For example, a population of $\ell = 5$ agents may be initialized to $\{\!\{Y, N, Y, Y, N\}\!\}$. An update of two agents occurs according to these four rules:

strong to weak	*propagation of winning side*	*tiebreaker*
$\{\!\{Y, N\}\!\} \to \{\!\{n, n\}\!\}$	$\{\!\{Y, n\}\!\} \to \{\!\{Y, y\}\!\}$ $\{\!\{N, y\}\!\} \to \{\!\{N, n\}\!\}$	$\{\!\{y, n\}\!\} \to \{\!\{n, n\}\!\}$

A possible execution from the aforementioned population is $\{\!\{Y, N, Y, Y, N\}\!\} \rightarrow \{\!\{Y, N, Y, n, n\}\!\}$ $\rightarrow \{\!\{Y, n, n, n, n\}\!\} \rightarrow \{\!\{Y, y, n, n, n\}\!\} \rightarrow \cdots \rightarrow \{\!\{Y, y, y, y, y\}\!\}$.

Agents in states $\{Y, y\}$ believe that the output of φ should be *true*, while agents in $\{N, n\}$ believe that it should be *false*. Thus, in the above execution, a lasting *true*-consensus has been reached by the population (although no agent is locally certain of it).

It is well known that population protocols compute precisely the predicates definable in Presburger arithmetic, namely first-order logic over the naturals with addition and order. This was first shown through convex geometry [2], and reproven using the theory of vector addition systems [15]. For example, this means that, given voting options $\{1, \ldots, k\}$, there is a population protocol that determines whether some option i has an absolute majority, *i.e.*, whether more than $\ell/2$ of the ℓ agents initially hold a common $i \in \{1, \ldots, k\}$.

Since k must be stored in the state-space, such a majority protocol can only handle a fixed number of voting options. As rules also depend on k, this means that a whole population would need to be reconfigured in order to handle a larger k, *e.g.* if new voting options are made available. This is conceptually impractical in the context of flocks of anonymous mobile agents. Instead, we propose that the input of an agent can be modeled elegantly as drawn from an infinite set $\mathbb{D}$, with rules independent from $\mathbb{D}$.

Contribution. In this work, we initiate the study of population protocols operating over arbitrarily large domains. More precisely, we propose a more general model where each agent carries a read-only datum from an infinite domain $\mathbb{D}$ together with a mutable state from a finite set. In this setting, a population can, *e.g.*, seek to determine whether there is an absolute majority datum. For example, if $\mathbb{D} := \{1, 2, 3, \ldots\}$, then the population initialized with $\{\!\{(1, x), (1, x), (2, x), (3, x), (1, x)\}\!\}$ should reach a lasting *true*-consensus, while it should reach a lasting *false*-consensus from $\{\!\{(1, x), (1, x), (2, x), (3, x), (2, x)\}\!\}$.

As in the standard model, a fair scheduler picks a pair of agents. An interaction occurs according to a rule of the form $\{\!\{p, q\}\!\} \xrightarrow{d \sim e} \{\!\{p', q'\}\!\}$, where $d, e \in \mathbb{D}$ are the data values of the two agents, and where $\sim \in \{=, \neq\}$ compares them. As for states, we assume that arbitrarily many agents may be initialized with the same datum; and that agents can only compare data through (in)equality. So, $\mathbb{D}$ is not a set of (unique) identifiers and hence agents remain anonymous as in the standard model.

To illustrate our proposed model of computation, we first show that it can compute the absolute majority predicate. This means that a *single* protocol can handle *any* number of options in an absolute majority vote. From the perspective of distributed computing, this provides a framework to reason about anonymous crowd computing over unordered sequences of data. From the standpoint of computer-aided verification, this opens the possibility of formally analyzing single protocols (*e.g.* modeled as colored Petri nets) rather than resorting to parameterized verification, which is particularly difficult in the context of counter systems.

As a stepping stone towards pinpointing the expressive power of *population protocols with unordered data*, we then characterize *immediate observation* protocols. In this well-studied restriction, rules have the form $\{\!\{p, q\}\!\} \xrightarrow{d \sim e} \{\!\{p, q'\}\!\}$, *i.e.* an agent updates its state by observing another agent (who is unaware of it). In standard population protocols, this class is known to compute exactly predicates from $\mathbf{COUNT}_*$ [2]. The latter is the Boolean closure of predicates of the form $\#q \geq c$, where $\#q$ counts the number of agents in state q, and $c \in \mathbb{N}$ is a constant. In our case, we show that immediate observation protocols compute exactly *interval predicates*, which are Boolean combinations of such *simple interval predicates*:

$$\exists \text{ pairwise distinct } d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \#(d_i, q_j) \in T(i,j), \tag{1}$$

where $\#(d_i, q_j)$ counts agents in state q_j with datum d_i, and each $T(i,j) \subseteq \mathbb{N}$ is an interval.

In order to show that immediate observation protocols do not compute more than interval predicates, we exploit the fact that (finitely supported) data vectors are well-quasi-ordered. While our approach is inspired by [2], it is trickier to simultaneously deal with the several sources of unboundedness: number of data values, number of agents with a given datum, and number of agents with a given state. As a byproduct, we show that the absolute majority predicate cannot be computed by immediate observation protocols.

To show the other direction, *i.e.* that interval predicates are computable by immediate observation protocols, we describe a protocol for simple interval predicates. In contrast with the standard setting, we need to implement existential quantification. This is achieved by a data leader election and a global leader election. We call the latter elected agent the "controller". Its purpose is to handle the bookkeeping of data leaders choosing their role in (1). A correction mechanism is carefully implemented so that the population only reaches a *true*-consensus upon locking a correct assignment to the existential quantification.

Related work. It has been observed by the verification and concurrency communities that population protocols can be recast as Petri nets. In particular, this has enabled the automatic formal analysis of population protocols [15, 7] and the discovery of bounds on their state complexity [12, 6]. Our inspiration comes from the other direction: we introduce protocols with data by drawing from the recent attention to colored Petri nets [17, 19, 18]. Our model corresponds to unordered data Petri nets where the color and number of tokens is invariant.

Population protocols for computing majority and plurality have been extensively studied (*e.g.*, see [14, 4, 5, 3] for recent results). To the best of our knowledge, the closest work is [16], where the authors propose space-efficient *families* of deterministic protocols for variants of the majority problem including plurality consensus. They consider the k voting options as "colors" specified by $\lceil \log k \rceil$ bits stored within the agents.

Other incomparable models of distributed systems with some sort of data include broadcast networks of register automata [13], distributed register automata [9], and distributed memory automata [10]. Such formalisms, inspired by register automata [20], allow identities, control structures and alternative communication mechanisms; none allowed in population protocols.

Paper organization. Section 2 provides basic definitions and introduces our model. In Section 3, we present a protocol that computes the absolute majority predicate. Section 4 establishes the expressive power of immediate observation protocols. We conclude in Section 5.

2 Preliminaries

We write $\mathbb{N}$ and $[a..b]$ to respectively denote sets $\{0, 1, 2, \ldots\}$ and $\{a, a+1, \ldots, b\}$. The *support* of a multiset $\boldsymbol{m}$ over E is $\mathrm{act}(\boldsymbol{m}) := \{e \in E : \boldsymbol{m}(e) > 0\}$ (We use the notation $\mathrm{act}(\boldsymbol{m})$ rather than $\mathrm{supp}(\boldsymbol{m})$ as we will later refer to "active states".) We write $\mathbb{N}^E$ to denote the set of multisets over E with finite support. The empty multiset, denoted $\mathbf{0}$, is such that $\mathbf{0}(e) = 0$ for all $e \in E$. Let $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{N}^E$. We write $\boldsymbol{m} \leq \boldsymbol{m}'$ iff $\boldsymbol{m}(e) \leq \boldsymbol{m}'(e)$ for all $e \in E$. We define $\boldsymbol{m} + \boldsymbol{m}'$ as the multiset such that $(\boldsymbol{m} + \boldsymbol{m}')(e) := \boldsymbol{m}(e) + \boldsymbol{m}'(e)$ for all $e \in E$. The difference, denoted $\boldsymbol{m} - \boldsymbol{m}'$, is defined similarly provided that $\boldsymbol{m} \geq \boldsymbol{m}'$.

2.1 Population protocols with unordered data

A *population protocol with unordered data*, over an infinite domain $\mathbb{D}$ equipped with equality, is a tuple (Q, δ, I, O) where

- Q is a finite set of elements called *states*,
- $\delta \subseteq Q^2 \times \{=, \neq\} \times Q^2$ is the set of *transitions*,
- $I \subseteq Q$ is the set of *initial states*, and
- $O \colon Q \to \{\mathit{false}, \mathit{true}\}$ is the *output* function.

We refer to an element of $\mathbb{D}$ as a *datum* or as a *color*. We will implicitly assume throughout the paper that δ contains $((p, q), \sim, (p, q))$ for all $p, q \in Q$ and $\sim \in \{=, \neq\}$.

A *form* $\boldsymbol{f}$ is an element from $\mathbb{N}^Q$. We denote the set of all forms by $\mathbb{F}$. Given $Q' \subseteq Q$, let $\boldsymbol{f}(Q') \coloneqq \sum_{q \in Q'} \boldsymbol{f}(q)$. A *configuration* is a mapping $\mathbf{C}$ from $\mathbb{D}$ to $\mathbb{F}$ such that $\operatorname{supp}(\mathbf{C}) \coloneqq \{d \in \mathbb{D} : \mathbf{C}(d) \neq \mathbf{0}\}$ is finite, and $\sum_{d \in \mathbb{D}, q \in Q} \mathbf{C}(d)(q) \geq 2$. We often write $\mathbf{C}(d)(q)$ as $\mathbf{C}(d, q)$. Informally, the latter denotes the number of agents with datum d in state q. We extend this notation to subsets of states: $\mathbf{C}(d, Q') \coloneqq \mathbf{C}(d)(Q')$. We naturally extend $+$, $-$ and $\leq$ to $\mathbb{D} \to \mathbb{F}$, *e.g.* $\mathbf{C} + \mathbf{C}'$ is such that $(\mathbf{C} + \mathbf{C}')(d) \coloneqq \mathbf{C}(d) + \mathbf{C}'(d)$ for all $d \in \mathbb{D}$.

Let $\mathbf{C}$ be a configuration. We define the *active states* as the set $\operatorname{act}(\mathbf{C}) \coloneqq \{q \in Q : \mathbf{C}(d, q) > 0 \text{ for some } d \in \mathbb{D}\}$. We say that $\mathbf{C}$ is *initial* if $\operatorname{act}(\mathbf{C}) \subseteq I$. Given $Q' \subseteq Q$, let $|\mathbf{C}|_{Q'} \coloneqq \sum_{d \in \mathbb{D}} \mathbf{C}(d, Q')$ and $|\mathbf{C}| \coloneqq |\mathbf{C}|_Q$. The *output* of $\mathbf{C}$ is defined by $O(\mathbf{C}) \coloneqq b$ if $O(q) = b$ for every $q \in \operatorname{act}(\mathbf{C})$; and by $O(\mathbf{C}) = \bot$ otherwise. Informally, $O(\mathbf{C})$ indicates whether all agents agree on some output b.

▶ **Example 1.** Let $\mathbb{D} \coloneqq \{\bullet, \blacksquare, \blacklozenge, \ldots\}$ and $Q \coloneqq \{p, q\}$. Let $\boldsymbol{f} \coloneqq \{\!\{p, p, q\}\!\}$ and $\boldsymbol{f}' \coloneqq \{\!\{q\}\!\}$. Let $\mathbf{C} \coloneqq \{\bullet \mapsto \boldsymbol{f}, \blacksquare \mapsto \boldsymbol{f}', \blacklozenge \mapsto \mathbf{0}, \ldots\}$. We have $\mathbf{C}(\bullet, p) = 2$, $\mathbf{C}(\bullet, q) = \mathbf{C}(\blacksquare, q) = 1$, $\mathbf{C}(\blacksquare, p) = \mathbf{C}(\blacklozenge, p) = \mathbf{C}(\blacklozenge, q) = 0$ and $|\mathbf{C}|_{\{q\}} = 2$. Configuration $\mathbf{C}$ represents a population of four agents carrying an immutable datum and a mutable state: $\{\!\{(\bullet, p), (\bullet, p), (\bullet, q), (\blacksquare, q)\}\!\}$. ◀

For the sake of brevity, given a form $\boldsymbol{f}$, let $\boldsymbol{f}_d \colon \mathbb{D} \to \mathbb{F}$ be defined by $\boldsymbol{f}_d(d) \coloneqq \boldsymbol{f}$ and $\boldsymbol{f}_d(d') \coloneqq \mathbf{0}$ for every $d' \neq d$. Furthermore, given a state q, let $\boldsymbol{q}_d \colon \mathbb{D} \to \mathbb{F}$ be defined by $\boldsymbol{q}_d(d)(q) \coloneqq 1$ and $\boldsymbol{q}_d(d')(q') \coloneqq 0$ for every $(d', q') \neq (d, q)$.

Let $\mathbf{C}$ be a configuration and let $t = ((p, q), \sim, (p', q')) \in \delta$. We say that transition t is *enabled* in $\mathbf{C}$ if there exist $d, e \in \mathbb{D}$ such that $d \sim e$, $\mathbf{C} \geq \boldsymbol{p}_d + \boldsymbol{q}_e$. If the latter holds, then t can be used to obtain the configuration $\mathbf{C}' \coloneqq \mathbf{C} - (\boldsymbol{p}_d + \boldsymbol{q}_e) + (\boldsymbol{p}'_d + \boldsymbol{q}'_e)$, which we denote $\mathbf{C} \xrightarrow{t} \mathbf{C}'$. We write $\mathbf{C} \to \mathbf{C}'$ to denote that $\mathbf{C} \xrightarrow{t} \mathbf{C}'$ holds for some $t \in \delta$. We further define $\xrightarrow{*}$ as the reflexive-transitive closure of $\to$.

▶ **Example 2.** Let $O(p) \coloneqq \mathit{false}$, $O(q) \coloneqq \mathit{true}$ and $t \coloneqq ((p, q), =, (q, q))$. Using the notation of Example 1 to represent configurations, we have:

$$\{\!\{(\bullet, p), (\bullet, p), (\bullet, q), (\blacksquare, q)\}\!\} \xrightarrow{t} \{\!\{(\bullet, p), (\bullet, q), (\bullet, q), (\blacksquare, q)\}\!\} \xrightarrow{t} \{\!\{(\bullet, q), (\bullet, q), (\bullet, q), (\blacksquare, q)\}\!\}.$$

Let $\mathbf{C}$, $\mathbf{C}'$ and $\mathbf{C}''$ denote the three configurations above. We have $O(\mathbf{C}) = O(\mathbf{C}') = \bot$ and $O(\mathbf{C}'') = \mathit{true}$. Moreover, transition t is not enabled in $\mathbf{C}''$ as no datum $d \in \mathbb{D}$ satisfies $\mathbf{C}''(d, p) \geq 1$ and $\mathbf{C}''(d, q) \geq 1$. So, the agents have "converged to a *true*-consensus". ◀

An *execution* is an infinite sequence of configurations $\mathbf{C}_0 \mathbf{C}_1 \cdots$ such that $\mathbf{C}_0 \to \mathbf{C}_1 \to \cdots$. We say that such an execution *converges* to output $b \in \{\mathit{false}, \mathit{true}\}$ if there exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = b$. An execution $\mathbf{C}_0 \mathbf{C}_1 \cdots$ is *fair* if, for every configuration $\mathbf{C}'$, it is the case that $|\{i \in \mathbb{N} : \mathbf{C}_i \xrightarrow{*} \mathbf{C}'\}| = \infty$ implies $|\{i \in \mathbb{N} : \mathbf{C}_i = \mathbf{C}'\}| = \infty$. In words, fairness states that if $\mathbf{C}'$ is reachable infinitely often, then it appears infinitely often along the execution. Informally, this means that some "progress" cannot be avoided forever.

Let Σ be a nonempty finite set. An *input* is some $\mathbf{M} \in \mathbb{N}^{\mathbb{D} \times \Sigma}$ with $\sum_{d \in \mathbb{D}, \sigma \in \Sigma} \mathbf{M}(d, \sigma) \geq 2$. An input $\mathbf{M}$ is translated, via a bijective *input mapping* $\iota \colon \Sigma \to I$, into the initial configuration $\iota(\mathbf{M}) \coloneqq \sum_{d \in \mathbb{D}, \sigma \in \Sigma} \sum_{j=1}^{\mathbf{M}(d,\sigma)} \boldsymbol{\iota(\sigma)}_d$. We say that a protocol *computes* a predicate φ if, for every input $\mathbf{M}$, every fair execution starting in $\iota(\mathbf{M})$ converges to output $\varphi(\mathbf{M})$. By abuse of notation, we sometimes write $\varphi(\mathbf{C}_0)$ for $\varphi(\iota^{-1}(\mathbf{C}_0))$.

▶ **Example 3.** Let $\mathbb{D} \coloneqq \{●, ■, ◆, \ldots\}$, $\Sigma \coloneqq \{x_1, \ldots, x_4\}$, $I \coloneqq \{q_1, \ldots, q_4\}$ and $\iota(x_i) \coloneqq q_i$. The input $\mathbf{M} \coloneqq \{\!\{(●, x_1), (●, x_1), (●, x_2), (●, x_2), (●, x_2), (●, x_4), (■, x_1), (■, x_3)\}\!\}$ yields the initial configuration $\iota(\mathbf{M}) = \{● \mapsto \{\!\{q_1, q_1, q_2, q_2, q_2, q_4\}\!\}, ■ \mapsto \{\!\{q_1, q_3\}\!\}, ◆ \mapsto \mathbf{0}, \ldots\}$.

Observe that, as for standard protocols, the set of predicates computed by population protocols with unordered data is closed under Boolean operations. Given a protocol that computes φ, we obtain a protocol that computes $\neg\varphi$ by changing the value of $O(q)$ to $\neg O(q)$ for all $q \in Q$. Given predicates ψ_1 and ψ_2, respectively computed by protocols $(Q_1, \delta_1, I_1, O_1)$ and $(Q_2, \delta_2, I_2, O_2)$, it is easy to obtain a protocol computing $\psi = \psi_1 \land \psi_2$ by having both protocols run in parallel. This is achieved by defining $(Q \coloneqq Q_1 \times Q_2, \delta, I \coloneqq I_1 \times I_2, O)$ where

- δ contains $(((p_1, p_2), (q_1, q_2)), \sim, ((p_1', p_2), (q_1', q_2)))$ for every $((p_1, q_1), \sim, (p_1', q_1')) \in \delta_1$;
- δ contains $(((p_1, p_2), (q_1, q_2)), \sim, ((p_1, p_2'), (q_1, q_2')))$ for every $((p_2, q_2), \sim, (p_2', q_2')) \in \delta_2$;
- $O(q_1, q_2) = O_1(q_1) \land O_2(q_2)$.

3 A protocol for the majority predicate

Let $\Sigma \coloneqq \{x\}$. In this section, we present a protocol for the absolute majority predicate defined as $\varphi_{\text{maj}}(\mathbf{M}) \coloneqq \exists d \in \mathbb{D} : \mathbf{M}(d, x) > \sum_{d' \neq d} \mathbf{M}(d', x)$. Since each input pair has the form (d, x) with $d \in \mathbb{D}$, we omit the "dummy element" x in the informal presentation of the protocol. Note that for the sake of brevity, we use the term majority instead of absolute majority for the remainder of this paper.

Our protocol is not unlike the classical (sequential) Boyer–Moore algorithm [11]: we seek to elect a color as the majority candidate, and then check whether it indeed has the majority. It is intended to work in stages. In the *pairing stage*, each unpaired agent seeks to form a pair with an unpaired agent of a distinct color. For example, if the initial population is $\{\!\{●, ●, ●, ●, ■, ■, ◆\}\!\}$, then we (non-deterministically) end up with either of these two pairings:

paired agents	*agents left unpaired*
$\{\!\{$●–■, ●–■, ●–◆$\}\!\}$	$\{\!\{$●$\}\!\}$
$\{\!\{$●–■, ■–◆$\}\!\}$	$\{\!\{$●, ●, ●$\}\!\}$

The agents left unpaired must all have the same color d, *e.g.* "●" in the above example. Moreover, if the population has a majority color, then it must be d.

Since the agents are anonymous and have a finite memory, they cannot actually remember with whom they have been paired. Thus, once a candidate color has been elected, *e.g.* "●" in the above example, there is a *grouping stage*. In the latter, unpaired agents indicate to agents of their color that they are part of the candidate majority group. This is done by internally storing the value "Y", which stands for "Yes". Similarly, unpaired agents indicate to agents of a distinct color that they are part of the candidate minority group using "N". Once this is over, the *majority stage* takes place using the classical protocol from the introduction.

Two issues arise from this idealized description. First, the protocol is intended to work in stages, but they may occur concurrently due to their distributed nature. For this reason, we add a correction mechanism:

- If an unpaired agent of the candidate majority color d finds a paired agent of color d (resp. $d' \neq d$) with "N" (resp. "Y"), then it flips it to "Y" (resp. "N");
- If an unpaired agent of the candidate majority color d finds a paired agent of color d (resp. $d' \neq d$) with either "n" or "y", then it flips it to "$\overline{Y}$" (resp. "$\overline{N}$").

The intermediate value $\overline{Y}$ (resp. $\overline{N}$) must be reverted to Y (resp. N) by finding an agent that has initially played role Y (resp. N) and is then reset to its original value.

The second issue has to do with the fact that, in even-size populations, all agents may get paired. In that case, no unpaired agent is left to group the agents. To address this, each agent carries an "even bit" to indicate its belief on whether some unpaired agent remains.

3.1 States

The set of states is defined as $Q := \{\mathit{false}, \mathit{true}\}^3 \times \{Y, N, \overline{Y}, \overline{N}, y, n\}$. To ease the reader's understanding, we manipulate states with four "macros". Each macro has a set of possible values; each state is a combination of values for the different macros.

name	*values for $q \in Q$*	*value for $q \in I$*
pair(q)	$\{\mathit{false}, \mathit{true}\}$	*false*
grp(q)	$\{\mathit{false}, \mathit{true}\}$	*true*

name	*values for $q \in Q$*	*value for $q \in I$*
even(q)	$\{\mathit{false}, \mathit{true}\}$	*false*
maj(q)	$\{Y, N, \overline{Y}, \overline{N}, y, n\}$	Y

The input mapping is defined by $\iota(x) := q_I$, where q_I is the unique state of I. Informally, pair(q) indicates whether the agent has been paired; grp(q) indicates whether the agent belongs to the candidate majority group; maj(q) is the current value of the majority computation; and even(q) is the even bit.

3.2 Transitions and stages

We describe the protocol by introducing rules corresponding to each stage. Note that a rule is a structure on which transitions can be based; therefore, a single rule can yield multiple transitions of the same nature. For convenience, some lemmas are stated before they can actually be proven, as they require the full set of transitions to be defined first. Proofs in the appendix take into account the complete list of transitions.

As the set of transitions for the protocol is lengthy, we present it using a "precondition-update" notation where for any two agents in state $p, q \in Q$, respectively with colors $d_1, d_2 \in \mathbb{D}$, a single transition whose preconditions on p, q and d_1, d_2 are met is used. The result of such an interaction is the agent initially in state p updating its state to p', where p' is identical to p except for the specified macros; and likewise for q. To help the readability, the precondition and update of states p and q are on distinct lines in the forthcoming tables.

3.2.1 Pairing stage

The first rule is used for the pairing stage whose main goal is to match as many agents as possible with agents of a different color:

rule	*state precondition*	*color precondition*	*state update*

(1)	$\neg$pair(p)
	$\neg$pair(q)
		$d_1 \neq d_2$	pair(p') $\land$ even(p')
			pair(q') $\land$ even(q')

This rule gives rise to the following lemmas concerning the end of the pairing stage and the nature of unpaired agents, if they exist. For the remainder of the section, let us fix a fair execution $\mathbf{C}_0\mathbf{C}_1\cdots$ where $\mathbf{C}_0$ is initial. Moreover, let $P \coloneqq \{q \in Q : \mathrm{pair}(q)\}$ and $U \coloneqq Q \setminus P$.

▶ **Lemma 4.** *There exists $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_U = |\mathbf{C}_{\tau+1}|_U = \cdots$. Furthermore, for every $i \geq \tau$, all unpaired agents of $\mathbf{C}_i$ share the same color, i.e. the set $\{d \in \mathbb{D} : \mathbf{C}_i(d, U) > 0\}$ is either empty or a singleton.*

Let α denote the minimal threshold τ given by Lemma 4, which is informally the “end of the pairing stage”.

▶ **Lemma 5.** *Let $i \in \mathbb{N}$. If $\varphi_{maj}(\mathbf{C}_0)$ and d is the majority color, then $\mathbf{C}_i(d, U) > 0$.*

3.2.2 Grouping stage

The next set of transitions seeks to correctly set each agent’s group, representing its status in the computation of the majority. An agent is either part of the candidate majority group (*true*), or part of the candidate minority group (*false*). Note that this group (and its related majority computing value) are irrelevant if there are no unpaired agents in $\mathbf{C}_\alpha$; this special case is handled using the even bit, which is ignored for now.

rule	*state precondition*	*color precondition*	*state update*
(2)	$\neg\mathrm{pair}(p)$ $\mathrm{pair}(q) \land \mathrm{grp}(q) \land \mathrm{maj}(q) = Y$	$d_1 \neq d_2$	*none* $\neg\mathrm{grp}(q') \land \mathrm{maj}(q') = N$
(3)	$\neg\mathrm{pair}(p)$ $\mathrm{pair}(q) \land \neg\mathrm{grp}(q) \land \mathrm{maj}(q) = N$	$d_1 = d_2$	*none* $\mathrm{grp}(q') \land \mathrm{maj}(q') = Y$
(4)	$\neg\mathrm{pair}(p)$ $\mathrm{pair}(q) \land \mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$	$d_1 \neq d_2$	*none* $\mathrm{maj}(q') = \overline{N}$
(5)	$\neg\mathrm{pair}(p)$ $\mathrm{pair}(q) \land \neg\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$	$d_1 = d_2$	*none* $\mathrm{maj}(q') = \overline{Y}$

The forthcoming rules below are part of a two-rule combination whose aim is to rectify an error in grouping assignments. It allows agents who engaged in the computation within the candidate minority group (resp. majority group) who encountered a currently valid majority candidate of their color (resp. a different color) to reset their value to Y (resp. N) and their group to *true* (resp. *false*) by finding another agent, also engaged, to do the same. This, along with the rules described in the next subsection, ensures that the invariant below holds.

Let $Q_a \coloneqq \{q \in Q : \mathrm{maj}(q) = a\}$, $Q_M \coloneqq \{q \in Q : \mathrm{grp}(q)\}$ and $Q_m \coloneqq Q \setminus Q_M$.

▶ **Lemma 6.** *For every $i \in \mathbb{N}$, it is the case that $|\mathbf{C}_i|_{Q_Y} - |\mathbf{C}_i|_{Q_N} = |\mathbf{C}_i|_{Q_M} - |\mathbf{C}_i|_{Q_m}$.*

rule	*state precondition*	*color precondition*	*state update*
(6)	$\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{N}$ $\neg\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$	*none*	$\neg\mathrm{grp}(p') \land \mathrm{maj}(p') = N$ $\mathrm{maj}(q') = N$
(7)	$\neg\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{Y}$ $\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$	*none*	$\mathrm{grp}(p') \land \mathrm{maj}(p') = Y$ $\mathrm{maj}(q') = Y$
(8)	$\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{N}$ $\neg\mathrm{grp}(q) \land \mathrm{maj}(q) = \overline{Y}$	*none*	$\neg\mathrm{grp}(p') \land \mathrm{maj}(p') = n$ $\mathrm{grp}(q') \land \mathrm{maj}(q') = n$

We give the following example to help illustrate the necessity of the intermediate states $\overline{Y}, \overline{N}$ in the context of the suggested protocol.

▶ **Example 7.** Let us first consider a possible execution from the initial population ⦃●, ●, ■, ■, ◆⦄, for which there is no majority datum. Observe that since the number of agents is odd, in any execution, there will be a datum with an unpaired agent after the pairing stage. Assume, for the sake of our demonstration, that this datum is blue (◆). Entering the grouping stage, this blue agent will eventually let the other agents know that they are not part of the majority candidate group and, at some point, rule (11) will occur, leading to all agents permanently with $\mathrm{maj}(q) \in \{N, n\}$. This is summarized in these three snapshots (where the even bit is omitted for the sake of clarity):

input	pair	grp	maj
●	✗	✓	Y
●	✗	✓	Y
■	✗	✓	Y
■	✗	✓	Y
◆	✗	✓	Y

$\xrightarrow{*}$

input	pair	grp	maj
●	✓	✗	N
●	✓	✗	N
■	✓	✗	N
■	✓	✗	N
◆	✗	✓	Y

$\rightarrow$

input	pair	grp	maj
●	✓	✗	N
●	✓	✗	n
■	✓	✗	N
■	✓	✗	N
◆	✗	✓	n

Now, consider a population where a majority datum does indeed exist: ⦃●, ●, ●, ●, ■, ■, ◆⦄. Note that this population is strictly greater than the previous population. Therefore, we can promptly obtain a configuration similar to the one described above, where two more agents of datum red (●) have yet to participate in the computation. Since the computation must output *true*, the consensus on $\{N, n\}$ initiated by the blue agent has to be reverted.

In this case, after the final pairing is done via an interaction between the blue agent (◆) and one of the newly introduced red agents (●), the error handling first works through rule (4) or (5): the unpaired red agent (●) notifies the blue agent (◆) that its group is incorrect by setting its computation value to $\overline{N}$ and similarly, it notifies all red agents (●) who had previously participated in the (now incorrect) majority stage to switch their computation value to $\overline{Y}$. This is summarized in these three snapshots:

. . .

input	pair	grp	maj
●	✓	✗	N
●	✓	✗	n
■	✓	✗	N
■	✓	✗	N
◆	✓	✓	n
●	✓	✓	Y
●	✗	✓	Y

$\xrightarrow{*}$

input	pair	grp	maj
●	✓	✓	Y
●	✓	✗	$\overline{Y}$
■	✓	✗	N
■	✓	✗	N
◆	✓	✓	$\overline{N}$
●	✓	✓	Y
●	✗	✓	Y

$\rightarrow$

input	pair	grp	maj
●	✓	✓	Y
●	✓	✓	n
■	✓	✗	N
■	✓	✗	N
◆	✓	✗	n
●	✓	✓	Y
●	✗	✓	Y

This inevitably leads each incorrectly grouped agent to rectify its group bit as well as its computation value, accordingly, through rules (6), (7) or (8). We then have a configuration for which the grouping stage is over and where either the majority stage is not yet initiated, or it has been correctly initiated with the right majority candidate. ◀

The following lemmas show that the grouping stage eventually ends if there are unpaired agents in $\mathbf{C}_\alpha$. Moreover, they show that the majority candidate color d eventually propagates the majority group to agents of color d, and the minority group to agents of color $d' \neq d$.

▶ **Lemma 8.** *Let E be the set of states engaged in the majority computation, i.e. $E := \{q \in Q : \mathrm{maj}(q) \in \{y, n, \overline{Y}, \overline{N}\}\}$. Let $E_M := E \cap Q_M$ and $E_m := E \cap Q_m$. For every $i \in \mathbb{N}$, the following holds: $|\mathbf{C}_i|_{E_M} = |\mathbf{C}_i|_{E_m}$.*

▶ **Lemma 9.** *Let $d \in \mathbb{D}$. If $\mathbf{C}_\alpha(d, U) > 0$, then there exists some $\tau \geq \alpha$ such that, for all $i \geq \tau$, $d' \in \mathbb{D}$ and $q \in \mathrm{act}(\mathbf{C}_i(d'))$, the following holds: $\mathrm{grp}(q) = (d' = d)$.*

3.2.3 Majority stage

The last set of transitions emulates a standard population protocol for the majority predicate. Populations of even size without a majority give rise to a case requiring careful handling. Indeed, for such a population the pairing stage may leave no unmatched agent. Therefore, we give the following rules to fix this specific issue.

rule	*state precondition*	*color precondition*	*state update*
(9)	$\neg\mathrm{pair}(p)$	*none*	*none*
	$\mathrm{even}(q)$		$\neg\mathrm{even}(q')$
(10)	$\mathrm{pair}(p) \land \mathrm{even}(p)$	*none*	*none*
	$\mathrm{pair}(q) \land \neg\mathrm{even}(q)$		$\mathrm{even}(q')$

▶ **Lemma 10.** *There exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$, it is the case that $\mathrm{even}(q)$ holds iff $|\mathbf{C}_i|_U = 0$.*

For other populations, a unique candidate color for the majority exists following the pairing stage. For the predicate to be *true*, this candidate must have more agents than all of the other colors combined. This is validated (or invalidated) through the following rules.

rule	*state pre.*	*col. pre.*	*state update*
(11)	$\mathrm{maj}(p) = Y$	*none*	$\mathrm{maj}(p') = n$
	$\mathrm{maj}(q) = N$		$\mathrm{maj}(q') = n$
(12)	$\mathrm{maj}(p) = Y$	*none*	*none*
	$\mathrm{maj}(q) = n$		$\mathrm{maj}(q') = y$

rule	*state pre.*	*col. pre.*	*state update*
(13)	$\mathrm{maj}(p) = N$	*none*	*none*
	$\mathrm{maj}(q) = y$		$\mathrm{maj}(q') = n$
(14)	$\mathrm{maj}(p) = n$	*none*	*none*
	$\mathrm{maj}(q) = y$		$\mathrm{maj}(q') = n$

▶ **Lemma 11.** *If $\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$, it is the case that $\mathrm{maj}(q) \in \{Y, y\}$ and $\neg\mathrm{even}(q)$ hold.*

▶ **Lemma 12.** *If $\neg\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that either:*
- *$\mathrm{even}(q)$ holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$; or*
- *$\mathrm{maj}(q) \in \{N, n\}$ holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$.*

We define the output of a given state $q \in Q$ as $O(q) := (\mathrm{maj}(q) \in \{Y, y\} \land \neg\mathrm{even}(q))$. The correctness of the protocol follows immediately from Lemmas 11 and 12:

▶ **Corollary 13.** *There exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = \varphi_{maj}(\mathbf{C}_0)$.*

4 Immediate observation protocols

We say that a population protocol is *immediate observation (IO)* if each of its transitions has the form $((p, q), \sim, (p, q'))$, *i.e.* only one agent can update its state by "observing" the other agent. There is no restriction on $\sim$, but one can also imagine the datum to be observed.

In this section, we characterize the expressive power of immediate observation protocols. First, we establish properties of IO protocols regarding truncations, thereby allowing us to prove that the majority predicate is not computable. Then, we show that IO protocols do not compute more than interval predicates. Finally, we show that every interval predicate can be computed by an IO protocol. Before proceeding, let us define interval predicates.

Let $\dot{\exists} d_1, d_2, \ldots, d_n$ denote a disjoint existential quantification, *i.e.* it indicates that $d_i \neq d_j$ for all $i, j \in [1..n]$ such that $i \neq j$. A *simple interval predicate*, interpreted over inputs from $\mathbb{N}^{\mathbb{D} \times \Sigma}$, where $\Sigma = \{x_1, \ldots, x_m\}$, is a predicate of the form

$$\psi(\mathbf{M}) = \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} \mathbf{M}(d_i, x_j) \in T(i, j), \tag{2}$$

where $m, n \in \mathbb{N}_{>0}$, each $T(i, j) \subseteq \mathbb{N}$ is a nonempty interval, and for every $i \in [1..n]$, there exists $j \in [1..m]$ such that $0 \notin T(i, j)$. An *interval predicate* is a Boolean combination of simple interval predicates.

4.1 State and form truncations

Given configurations $\mathbf{C}, \mathbf{C}'$, we write $\mathbf{C} \sqsubseteq \mathbf{C}'$ if there exists an injection $\rho \colon \mathbb{D} \to \mathbb{D}$ such that $\mathbf{C}(d) \leq \mathbf{C}'(\rho(d))$ for every $d \in \mathbb{D}$. We write $\mathbf{C} \equiv \mathbf{C}'$ if $\mathbf{C} \sqsubseteq \mathbf{C}'$ and $\mathbf{C}' \sqsubseteq \mathbf{C}$. We say that a subset of configurations X is *upward closed* if $\mathbf{C} \in X$ and $\mathbf{C} \sqsubseteq \mathbf{C}'$ implies $\mathbf{C}' \in X$. We say that a set B is a *basis* of an upward closed set X if $X = \{\mathbf{C}' : \mathbf{C} \sqsubseteq \mathbf{C}' \text{ for some } \mathbf{C} \in B\}$.

A configuration $\mathbf{C}$ is said *unstable* if either $O(\mathbf{C}) = \bot$ or there exists $\mathbf{C}'$ such that $\mathbf{C} \xrightarrow{*} \mathbf{C}'$ with $O(\mathbf{C}) \neq O(\mathbf{C}')$. Let $\mathcal{U}$ denote the set of unstable configurations, and let $\mathcal{S}_b \coloneqq \{\mathbf{C} : \mathbf{C} \notin \mathcal{U}, O(\mathbf{C}) = b\}$ denote the set of stable configurations with output b. As in the case of standard protocols (without data) [1], it is simple to see that $\mathcal{U}$ is upward closed. Moreover, since $\sqsubseteq$ is a well-quasi-order, it follows that $\mathcal{U}$ has a finite basis.

This allows us to extend the notion of truncations from [1]. A *state truncation* to $k \geq 1$ of some form $\boldsymbol{f}$, denoted by $\tau_k(\boldsymbol{f})$, is the form such that $\tau_k(\boldsymbol{f})(q) \coloneqq \min(\boldsymbol{f}(q), k)$ for all $q \in Q$. The concept of state truncations is also extended to configurations: $\tau_k(\mathbf{C})$ is the configuration such that $\tau_k(\mathbf{C})(d) \coloneqq \tau_k(\mathbf{C}(d))$ for all $d \in \mathbb{D}$. From a sufficiently large threshold, the stability and output of a configuration remain unchanged under state truncations:

▶ **Lemma 14.** *Let ψ be a predicate computed by a population protocol with unordered data. Let S_b be the set of stable configurations with output b of the protocol. There exists $k \geq 1$ such that, for all $b \in \{0, 1\}$, we have $\mathbf{C} \in \mathcal{S}_b$ iff $\tau_k(\mathbf{C}) \in \mathcal{S}_b$.*

Given a configuration $\mathbf{C}$ and a form $\boldsymbol{f}$, let $\#_{\boldsymbol{f}}(\mathbf{C}) \coloneqq |\{d \in \mathbb{D} : \mathbf{C}(d) = \boldsymbol{f}\}|$. Due to the nature of immediate observation protocols, it is always possible to take a form $\boldsymbol{f}$ of color d present in an configuration $\mathbf{C}$, duplicate $\boldsymbol{f}$ with a fresh color d', and have the latter mimic the behaviour of the former.

▶ **Lemma 15.** *Let $\mathbf{C}$ and $\mathbf{C}'$ be configurations such that $\mathbf{C} \xrightarrow{*} \mathbf{C}'$. For every $d \in \operatorname{supp}(\mathbf{C})$ and $d' \in \mathbb{D} \setminus \operatorname{supp}(\mathbf{C})$, it is the case that $\mathbf{C} + (\mathbf{C}(d))_{d'} \xrightarrow{*} \mathbf{C}' + (\mathbf{C}'(d))_{d'}$.*

Combined with the fact that $\mathcal{U}$ has a finite basis, this allows to show that from some threshold, duplicating forms with fresh colors does not change the output of the population.

▶ **Lemma 16.** *Let ψ be a predicate computed by a population protocol with unordered data. Let $\boldsymbol{f}$ be a form with* $\mathrm{act}(\boldsymbol{f}) \subseteq I$. *There exists $h(\boldsymbol{f}) \in \mathbb{N}$ such that, for all initial configuration $\mathbf{C}_0$ and $d \in \mathbb{D} \setminus \mathrm{supp}(\mathbf{C}_0)$ with $\#_{\boldsymbol{f}}(\mathbf{C}_0) \geq h(\boldsymbol{f})$, it is the case that $\psi(\mathbf{C}_0 + \boldsymbol{f}_d) = \psi(\mathbf{C}_0)$.*

The *form truncation* of a configuration $\mathbf{C}$, denoted $\sigma(\mathbf{C})$, is an (arbitrary) configuration such that $\sigma(\mathbf{C}) \sqsubseteq \mathbf{C}$ and $\#_{\boldsymbol{f}}(\sigma(\mathbf{C})) = \min(\#_{\boldsymbol{f}}(\mathbf{C}), h(\boldsymbol{f}))$ for every form $\boldsymbol{f}$, where $h(\boldsymbol{f})$ is given by Lemma 16. By Lemma 16, $\psi(\mathbf{C}_0)$ holds iff $\psi(\sigma(\mathbf{C}_0))$ holds. Moreover, Lemma 16 allows us to show that IO protocols are less expressive than the general model.

▶ **Proposition 17.** *No IO population protocol computes the majority predicate φ_{maj}.*

Proof. For the sake of contradiction, suppose that some IO protocol computes φ_{maj}. Let q_I be the unique initial state and let $\boldsymbol{f} \coloneqq \{\!\!\{q_I\}\!\!\}$. Let $h(\boldsymbol{f})$ be given by Lemma 16. Let $\mathbf{C}_0$ be an initial configuration such that

- $\mathbf{C}_0(d) = \sum_{i=1}^{h(\boldsymbol{f})+1} \boldsymbol{f}$ holds for a unique datum $d \in \mathbb{D}$, and
- $\mathbf{C}_0(d') = \boldsymbol{f}$ holds for exactly $h(\boldsymbol{f})$ other data $d' \in \{d_1, d_2, \ldots, d_{h(\boldsymbol{f})}\}$.

We have $\varphi_{\mathrm{maj}}(\mathbf{C}_0) = \mathit{true}$, since d has $h(\boldsymbol{f})+1$ agents in a population of $2 \cdot h(\boldsymbol{f})+1$ agents. Let $\mathbf{C}'_0$ be the initial configuration obtained from $\mathbf{C}_0$ by adding a datum $d^* \notin \mathrm{supp}(\mathbf{C}_0)$ such that $\mathbf{C}'_0(d^*) = \boldsymbol{f}$. By Lemma 16, $\varphi_{\mathrm{maj}}(\mathbf{C}'_0) = \varphi_{\mathrm{maj}}(\mathbf{C}_0) = \mathit{true}$. However, datum d no longer has a majority in $\mathbf{C}'_0$, which is a contradiction. ◀

4.2 Predicates computed by IO protocols are interval predicates

▶ **Theorem 18.** *Let (Q, δ, I, O) be an immediate observation protocol with unordered data that computes a predicate ψ. The predicate ψ can be expressed as an interval predicate.*

Proof. We will express ψ as a finite Boolean combination of simple interval predicates.

Let h be the mapping given by Lemma 16. Let $\mathbb{T} \coloneqq \{\mathbf{C} : \psi(\mathbf{C}) = \mathit{true}\}$ and $\mathbb{T}_1 \coloneqq \{\mathbf{C} \in \mathbb{T} : \mathbf{C}(d, q) \leq k \text{ for all } d \in \mathbb{D}, q \in Q\}$. From Lemma 14, we learn that state truncations do not change the stability of a configuration. So, $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_1} \tau_k(\mathbf{C}) = \mathbf{C}'$ holds. Let $\mathbb{T}_2 \coloneqq \{\mathbf{C} \in \mathbb{T}_1 : \#_{\boldsymbol{f}}(\mathbf{C}) \leq h(\boldsymbol{f}) \text{ for all } \boldsymbol{f} \in \mathbb{F}\}$. It follows from Lemma 16 that $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_2} \sigma(\tau_k(\mathbf{C})) = \mathbf{C}'$ holds.

The latter is an infinite disjunction. Let us make it finite. Observe that if $\mathbf{C} \xrightarrow{*} \mathbf{C}'$ and $\overline{\mathbf{C}} \equiv \mathbf{C}$ hold, then there exists $\overline{\mathbf{C}'} \equiv \mathbf{C}'$ such that $\overline{\mathbf{C}} \xrightarrow{*} \overline{\mathbf{C}'}$. Moreover, note that equivalent configurations have the same output as they share the same active states. Indeed, $\mathbf{C} \equiv \overline{\mathbf{C}}$ iff $\bigwedge_{\boldsymbol{f} \in \mathbb{F}} \#_{\boldsymbol{f}}(\mathbf{C}) = \#_{\boldsymbol{f}}(\overline{\mathbf{C}})$. Hence, for every initial configuration $\mathbf{C} \equiv \overline{\mathbf{C}}$, we have $\psi(\mathbf{C}) = \psi(\overline{\mathbf{C}})$. Let $\mathbb{T}_2/{\equiv}$ be the set of all equivalence classes of $\equiv$ on $\mathbb{T}_2$, and let $\mathbb{T}_3$ be a set that contains one representative configuration per equivalence class of $\mathbb{T}_2/{\equiv}$. It is readily seen that $\psi(\mathbf{C})$ holds iff $\bigvee_{\mathbf{C}' \in \mathbb{T}_3} \sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$ holds.

Let us argue that $\mathbb{T}_3$ is finite. Let $\mathbb{F}_k \coloneqq \{\boldsymbol{f} \neq \mathbf{0} : \boldsymbol{f}(q) \leq k \text{ for all } q \in Q\}$. For every configuration $\mathbf{C} \in \mathbb{T}_1$, each form $\boldsymbol{f}$ with $\#_{\boldsymbol{f}}(\mathbf{C}) > 0$ belongs to $\mathbb{F}_k$. As $\mathbb{T}_2 \subseteq \mathbb{T}_1$, this also holds for configurations of $\mathbb{T}_2$. Given $\mathbf{C} \in \mathbb{T}_2$, we have $\#_{\boldsymbol{f}}(\mathbf{C}) \leq h(\boldsymbol{f})$ for all $\boldsymbol{f} \in \mathbb{F}_k$, and $\#_{\boldsymbol{f}}(\mathbf{C}) = 0$ for all $\boldsymbol{f} \notin \mathbb{F}_k$. Thus, as $\mathbb{F}_k$ is finite, we conclude that $\mathbb{T}_3$ is finite.

Let us now exploit our observations to express ψ as an interval predicate. Let us fix some $\mathbf{C}' \in \mathbb{T}_3$. It suffices to explain how to express "$\sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$". Indeed, as $\mathbb{T}_3$ is finite, we can conclude by taking the finite disjunction $\bigvee_{\mathbf{C}' \in \mathbb{T}_3} \sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$.

For every form $\boldsymbol{f} \in \mathbb{F}_k$, let $\mathrm{lt}(\boldsymbol{f}) \coloneqq \{q \in Q : \boldsymbol{f}(q) < k\}$, $\mathrm{eq}(\boldsymbol{f}) \coloneqq \{q \in Q : \boldsymbol{f}(q) = k\}$ and

$$\varphi_{\boldsymbol{f},d}(\mathbf{C}) \coloneqq \bigwedge_{q \in \mathrm{lt}(\boldsymbol{f})} (\mathbf{C}(d, q) = \boldsymbol{f}(q)) \wedge \bigwedge_{q \in \mathrm{eq}(\boldsymbol{f})} (\mathbf{C}(d, q) \geq \boldsymbol{f}(q)).$$

Observe that $\varphi_{\boldsymbol{f},d}(\mathbf{C})$ holds iff $\tau_k(\mathbf{C})(d) = \boldsymbol{f}$.

For every $\boldsymbol{f} \in \mathbb{F}_k$ such that $\#_{\boldsymbol{f}}(\mathbf{C}') < h(\boldsymbol{f})$, we define this formula, where $n \coloneqq \#_{\boldsymbol{f}}(\mathbf{C}')$:

$$\psi_{\boldsymbol{f}}(\mathbf{C}) \coloneqq \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}) \land \neg \dot{\exists} d_1, d_2, \ldots, d_{n+1} \in \mathbb{D} : \bigwedge_{i=1}^{n+1} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}).$$

For every $\boldsymbol{f} \in \mathbb{F}_k$ such that $\#_{\boldsymbol{f}}(\mathbf{C}') = h(\boldsymbol{f})$, we define this formula, where $n \coloneqq \#_{\boldsymbol{f}}(\mathbf{C}')$:

$$\psi_{\boldsymbol{f}}(\mathbf{C}) \coloneqq \dot{\exists} d_1, d_2, \ldots, d_n \in \mathbb{D} : \bigwedge_{i=1}^{n} \varphi_{\boldsymbol{f},d_i}(\mathbf{C}).$$

Observe that $\psi_{\boldsymbol{f}}$ is either a simple interval predicate or a Boolean combination of two simple interval predicates. Note that $\psi_{\boldsymbol{f}}(\mathbf{C})$ holds iff $\#_{\boldsymbol{f}}(\sigma(\tau_k(\mathbf{C}))) = \#_{\boldsymbol{f}}(\mathbf{C}')$ holds. This means that $\bigwedge_{\boldsymbol{f} \in \mathbb{F}_k} \psi_{\boldsymbol{f}}(\mathbf{C})$ holds iff $\sigma(\tau_k(\mathbf{C})) \equiv \mathbf{C}'$ holds, and hence we are done. ◀

4.3 An IO protocol for simple interval predicates

As Boolean combinations can be implemented (see end of Section 2.1), it suffices to describe a protocol for a simple interval predicate of the form (2). We refer to each $i \in [1..n]$ as a *role*. In the forthcoming set of states Q, we associate to each $q \in Q$ an element $\text{elem}(q) \in [1..m]$. Each agent's element is set through the input; *e.g.* an agent mapped from symbol $(\bullet, x_1)$ is initially in a state q such that $\text{elem}(q) = 1$. Let $Q_j \coloneqq \{q \in Q : \text{elem}(q) = j\}$. For any two configurations $\mathbf{C}_a$ and $\mathbf{C}_b$ of an execution, any datum $d \in \mathbb{D}$ and any element $j \in [1..m]$, the invariant $\mathbf{C}_a(d, Q_j) = \mathbf{C}_b(d, Q_j)$ holds. We say that $d \in \mathbb{D}$ *matches* role i in configuration $\mathbf{C}$ if $\mathbf{C}(d, Q_j) \in T(i, j)$ holds for all $j \in [1..m]$. Let $r \coloneqq \max(r_1, \ldots, r_n) + 1$, where

$$r_i \coloneqq \max(\{\min T(i, j) : j \in [1..m]\} \cup \{\max T(i, j) : j \in [1..m], \sup T(i, j) < \infty\}).$$

Agents will not need to count beyond value r to decide whether a role is matched.

As for the majority protocol, our simple interval protocol works in stages, each one being necessary to ensure properties and invariants for the subsequent stages. In the *election stage*, a unique controller for the population and a single leader per datum of the support are selected; the former seeks to distribute a set of roles to the latter.

All agents contribute to the tallying of their immutable element j through the *counting stage*. This is done using the "tower method" described in [2], whereby two agents of the same datum, element *and* value meet and allow one of the two agents to increment its value. The maximal value computed in that manner is subsequently communicated to the (unique) datum leader.

Once the leaders carry correct counts for each element of their respective datum, they undertake roles that they match in the *distribution stage*. These roles can be swapped for other roles (as long as requirements are met) through a process of interrogating the controller. The controller is constantly notified of selected roles and updates its list of tasks accordingly.

If a fully assigned task list is obtained by the controller, it spreads a *true*-output throughout the population in what we call the *output propagation stage*. If that is not possible, leaders are in a consistent state of trial-and-error for their role assignments, ultimately failing to completely fill the task list, leaving the controller free to propagate its *false*-output.

▶ **Example 19.** Consider $n = m = 2$ with $T(1, 1) \coloneqq [2..\infty)$, $T(1, 2) \coloneqq [0..4]$, $T(2, 1) \coloneqq \mathbb{N}$, $T(2, 2) \coloneqq [1..\infty)$. Let $\mathbf{M} \coloneqq \{\!\{(\bullet, x_1), (\bullet, x_1), (\bullet, x_1), (\bullet, x_2), (\blacksquare, x_1), (\blacklozenge, x_2)\}\!\}$. Note that $r = 5$. Datum "$\bullet$" could match roles 1 and 2, "$\blacksquare$" cannot match any role, and "$\blacklozenge$" could match role 2.

After executing the protocol for a while, we may end up with the configuration illustrated in the table below. The third, fourth, fifth and sixth agents contain the correct value for their datum and element: $\mathbf{M}(\bullet, x_1) = 3$ and $\mathbf{M}(\bullet, x_2) = \mathbf{M}(\blacksquare, x_1) = \mathbf{M}(\blacklozenge, x_2) = 1$. The second agent has been elected controller. The last three agents have been elected their respective datum's leader and have collected the correct counts for each element. Either the $\bullet$-leader or the $\blacklozenge$-leader (possibly both) has notified the controller that they play role 2.

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(\bullet, x_1)$	1					
$(\bullet, x_1)$	2		✓			[✗, ✓]
$(\bullet, x_1)$	**3**					
$(\bullet, x_2)$	**1**	✓		2	$[3, 1]$	
$(\blacksquare, x_1)$	**1**	✓			$[1, 0]$	
$(\blacklozenge, x_2)$	**1**	✓		2	$[0, 1]$	

The $\bullet$-leader may change its mind and decide to play role 1 after noticing the controller does not have its task 1 assigned. This switches its role to -2. Once the $\bullet$-leader notifies the controller, its role is set to 0 and (in doubt) the controller considers that role 2 is not assigned anymore. The $\bullet$-leader then changes its role to 1. Eventually the $\bullet$-leader and $\blacklozenge$-leader notify the controller that roles 1 and 2 are taken. This is summarized in these three snapshots:

input	⋯	role	⋯	task	⋯	role	⋯	task	⋯	role	⋯	task
$(\bullet, x_1)$												
$(\bullet, x_1)$				[✗, ✗]				[✗, ✗]				[✓, ✓]
$(\bullet, x_1)$												
$(\bullet, x_2)$		-2				0				1		
$(\blacksquare, x_1)$												
$(\blacklozenge, x_2)$		2				2				2		

◀

Note that while we rely on stages to describe our protocol, the distributed nature of the model implies that some stages may interfere with others. Therefore, we present here a list of potential problems and the way our protocol fixes them.

- While leader election is straightforward, role assignment for leaders can happen at any time before the actual leader is elected. This could lead to the controller being notified of a role assignment for which no current leader is assigned. Thus, when an agent loses its leadership status, it reverts its role to a negative value, meaning it will have to inform the controller of the change before returning to a passive value.
- A leader may take a role before having the correct counts. We provide a reset mechanism through which the leader falls into a "negative role". This forces it to then contact the controller and rectify the situation.
- A leader may have previously taken a role before realizing it does not actually meet the requirements. The leader is then forced to convey its mistake to the controller. But the controller it notifies may not ultimately be the population's controller. Therefore, after losing the controller status, an agent has to go to a negative controller state, meaning it must reset the controller's tasks before reverting to a passive value.
- There may be many leaders with the same role. To prevent deadlocks, we allow a leader to self-reassign to a new role if it notices the controller does not have the task filled.

4.3.1 States

The set of states is defined as $Q := \{\mathit{false}, \mathit{true}\}^{n+2} \times [1..m] \times [1..r]^{m+1} \times [-n..n] \times \{-1, 0, 1\}$. For the sake of readability, we specify and manipulate states with these macros:

name	*values for $q \in Q$*	*values for $q \in I$*
$\mathrm{elem}(q)$	$[1..m]$	$j \in [1..m]$
$\mathrm{val}(q)$	$[1..r]$	1
$\mathrm{out}(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*
$\mathrm{lead}(q)$	$\{\mathit{false}, \mathit{true}\}$	*true*
$\mathrm{role}(q)$	$[-n..n]$	0
$\mathrm{count}_\ell(q)$	$[1..r]$	1 if $\ell = j$, 0 otherwise
$\mathrm{ctrl}(q)$	$\{-1, 0, 1\}$	1
$\mathrm{task}_i(q)$	$\{\mathit{false}, \mathit{true}\}$	*false*

The input mapping is defined by $\iota(x_j) := p_j$, where p_j is the unique state of I with $\mathrm{elem}(p_j) = j$. Informally, $\mathrm{elem}(q) = j$ indicates that the agent holds the j-th element; $\mathrm{val}(q)$ is the current tally of element $\mathrm{elem}(q)$ for the datum of the agent; $\mathrm{lead}(q)$ and $\mathrm{ctrl}(q)$ respectively indicate whether an agent is a datum leader or a controller; $\mathrm{role}(q)$ indicates the role for a leader; $\mathrm{count}_j(q)$ allows a datum leader to maintain the highest count currently witnessed for element j; $\mathrm{task}_i(q)$ allows the controller to maintain a list of the currently matched roles; and $\mathrm{out}(q)$ is the current belief of an agent on the output of the protocol.

Note that the rules presented in this section are used to succinctly describe transitions. A single rule may induce several transitions. Furthermore, for the sake of brevity, we mark rules allowing *mirror transitions* with an asterisk ($*$) next to the rule number. Mirror transitions are transitions in which an agent may observe its own state and react accordingly. Thus, a $*$-rule generating transitions whose precondition formula is $A(p) \wedge B(q)$ also generates a transition whose precondition is $A(q) \wedge B(q)$, effectively making state p the state of any "dummy agent". Note that q is still the only state to be updated to q'.

4.3.2 Leader and controller election

The first two rules are meant to elect a unique leader per datum present in the population, and a unique global controller for the whole population. For the remainder of the section, let us fix a fair execution $\mathbf{C}_0\mathbf{C}_1 \cdots$ where $\mathbf{C}_0$ is initial.

rule	*state precondition*	*color precondition*	*state update*
(1)	$\mathrm{lead}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{role}(q') = -\lvert\mathrm{role}(q)\rvert$ $\neg\mathrm{lead}(q')$
(2)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = 1$	*none*	$\mathrm{ctrl}(q') = -1$

Note that rule (1) guarantees that the agent losing leadership has its role set to a non-positive value. Similarly, rule (2) pushes the non-controller into a temporary intermediate state for its controller value, *i.e.* -1. Let $Q_L := \{q \in Q : \mathrm{lead}(q)\}$ and $Q_C := \{q \in Q : \mathrm{ctrl}(q) = 1\}$. The following lemma identifies the end of both elections.

▶ **Lemma 20.** *There exists $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_{Q_C} = |\mathbf{C}_{\tau+1}|_{Q_C} = \cdots = 1$, and $|\mathbf{C}_\tau(d)|_{Q_L} = |\mathbf{C}_{\tau+1}(d)|_{Q_L} = \cdots = 1$ for every $d \in \mathbb{D}$.*

Let α denote the minimal value τ given by Lemma 20, which we refer to as the end of the election stage.

4.3.3 Element count by datum

The next rules allow to count how many agents with a common datum hold the same element. This count is ultimately communicated to the datum leader. Given $d \in \mathbb{D}$ and $\tau \in \mathbb{N}$, we say that a state $q \in Q$ is (d, j)*-valid* if $\mathrm{elem}(q) = j$ and $\mathrm{val}(q) = \min(\mathbf{C}_\tau(d, Q_j), r)$.

rule	*state precondition*	*color precondition*	*state update*
(3)	$\mathrm{elem}(p) = \mathrm{elem}(q)$ $\mathrm{val}(q) = \mathrm{val}(p) < r$	$d_1 = d_2$	$\mathrm{val}(q') = \mathrm{val}(q) + 1$
(4)*	$\mathrm{count}_{\mathrm{elem}(p)}(q) < \mathrm{val}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{count}_{\mathrm{elem}(p)}(q') = \mathrm{val}(p)$ `if` $(\mathrm{role}(q) > 0 \land$ $\mathrm{val}(p) \notin T(\mathrm{role}(q), \mathrm{elem}(p)))$: $\mathrm{role}(q') = -\mathrm{role}(q)$

Observe another correction mechanism; rule (4) guarantees that a leader with an assigned role $i > 0$ verifies that it can still assume role i after updating its count. The following lemmas explain that the correct counts are eventually provided to each datum leader.

▶ **Lemma 21.** *There exists $\tau \in \mathbb{N}$ such that, for every $\tau' \geq \tau$, $d \in \mathrm{supp}(\mathbf{C}_{\tau'})$ and $j \in [1..m]$, if $\mathbf{C}_0(d, Q_j) > 0$, then $\mathbf{C}_{\tau'}(d, q) > 0$ holds for some (d, j)-valid state q.*

▶ **Lemma 22.** *There exists $\tau \geq \alpha$ such that, for every $\tau' \geq \tau$, $d \in \mathrm{supp}(\mathbf{C}_{\tau'})$, $j \in [1..m]$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'}(d)) \cap Q_L$, it is the case that $\mathrm{count}_j(q) = \min(\mathbf{C}_{\tau'}(d, Q_j), r)$.*

Let τ' and τ'' denote the minimal values τ given by Lemmas 21 and 22. From now on, let $\beta := \max(\tau', \tau'')$.

4.3.4 Role distribution and task tracking

The following rules assign roles to leaders and allow leaders to reset their roles when possible, therefore preventing deadlocks. In rule (5), variable i can take any value from $[1..n]$.

rule	*state precondition*	*color precondition*	*state update*
(5)	$\mathrm{lead}(q)$ $\mathrm{role}(q) = 0$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i, j)$	*none*	$\mathrm{role}(q') = i$
(6)*	$\mathrm{ctrl}(p)$ $\mathrm{lead}(q)$ $\mathrm{role}(q) = i > 0$ $\bigvee_{i' \in [1..n] \setminus \{i\}} \big(\neg \mathrm{task}_{i'}(p) \land$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i', j)\big)$	*none*	$\mathrm{role}(q') = -i$

This induces the following result, informally meaning that if a leader has taken a role, then it currently *believes* it can fill this role.

▶ **Lemma 23.** *For every $\tau \in \mathbb{N}$, $j \in [1..m]$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \cap Q_L$ such that $\mathrm{role}(q) > 0$, it is the case that $\mathrm{count}_j(q) \in T(\mathrm{role}(q), j)$.*

These rules allow to update the controller's task list and reset roles when needed:

rule	*state precondition*	*color precondition*	*state update*
(7)*	$\text{role}(p) \neq 0$ $\text{ctrl}(q) = 1$	*none*	$\text{task}_{\lvert\text{role}(p)\rvert}(q') = (\text{role}(p) > 0)$
(8)*	$\text{ctrl}(p) = 1$ $\text{role}(q) < 0$ $\neg\text{task}_{\lvert\text{role}(q)\rvert}(p)$	*none*	$\text{role}(q') = 0$

To illustrate how rules (5) through (8) operate, we give the following example.

▶ **Example 24.** Recall Example 19, introduced earlier. Consider the configuration of its first snapshot. While we initially gave intuitions on how role reassignment might happen from this specific configuration, we give here a deeper analysis of the important configurations involved in this process.

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✗, ✓]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		2	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

In the above, the ●-leader currently believes (rightly so) that it can fill roles 1 and 2. Observe that the controller has task 2 assigned. However, its task 1 is still unassigned. Therefore, rule (6) allows the ●-leader to initiate its reassignment by setting its role to -2 through an interaction with the controller. This leads to the following configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✗, ✓]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		-2	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

Since its role is set to -2, the ●-leader now seeks to inform the controller that it should unassign role 2 from its task list. This is achieved on their next meeting through rule (7). We then have this next configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✗, ✗]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		-2	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

Note that this does not mean that no leader currently has its role set to 2; indeed, the ◆-leader still has its role set to 2. Let us now assume that, immediately after reaching this configuration, the ●-leader and the controller meet again. Since the ●-leader observes that the controller no longer has its task 2 assigned, it can assume that either it unassigned it, some other leader did, or it was never assigned. In any case, it can safely reset its role to 0 through rule (8), giving us the following configuration:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✗,✗]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		0	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

Observe that the ◆-leader could have met the controller before the ●-leader, thereby reassigning role 2 in the controller's task list and undoing the ●-leader's work. This would only delay the ●-leader's role resetting; through fairness, it would not endlessly prevent it.

From this last configuration, since the ●-leader's role is set to 0, it is now free to take any role it can fill through rule (5). Let us assume, for the sake of brevity, that it takes on role 1:

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✗,✗]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		1	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

Suppose the ◆-leader meets the controller before the ●-leader. Then, rule (7) assigns task 2 in the controller's task list. The ●-leader can no longer reset its role through rule (6) because the controller has task 2 already assigned. Therefore, when the ●-leader eventually meets the controller again, it finally assigns task 1 to its task list via rule (7).

input	val	lead	ctrl	role	count of $[\#x_1, \#x_2]$	task list for [role 1, role 2]
$(●, x_1)$	1					
$(●, x_1)$	2		✓			[✓,✓]
$(●, x_1)$	**3**					
$(●, x_2)$	**1**	✓		1	$[3, 1]$	
$(■, x_1)$	**1**	✓			$[1, 0]$	
$(◆, x_2)$	**1**	✓		2	$[0, 1]$	

◀

The following lemmas show that at some point in the execution, a configuration is reached where agents who are neither leaders nor controllers no longer interact with other agents.

▶ **Lemma 25.** *There exists some $\tau \in \mathbb{N}$ such that for every $\tau' \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'}) \setminus Q_L$, it is the case that* $\mathrm{role}(q) = 0$.

rule	*state precondition*	*color precondition*	*state update*
(9)	$\mathrm{ctrl}(p) = -1$ $\mathrm{ctrl}(q) = 1$	*none*	$\bigwedge_{i \in [1..m]} \neg\mathrm{task}_i(q')$
(10)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = -1$ $\bigwedge_{i \in [1..m]} \neg\mathrm{task}_i(p)$	*none*	$\mathrm{ctrl}(q') = 0$

▶ **Lemma 26.** *There exists $\tau \geq \alpha$ such that for every $\tau' \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'})$, it is the case that $\mathrm{ctrl}(q) \in \{0, 1\}$.*

Let τ' and τ'' denote the minimal values τ given by Lemmas 25 and 26. From now on, let $\gamma \coloneqq \max(\beta, \tau', \tau'')$. Informally, this delimits the configuration where there are no negative controllers, therefore preventing recurring resets of the controller's tasks through rule (9). Finally, the following lemma argues that past $\mathbf{C}_\gamma$, a controller can only have a task set to *true* if some leader is currently assuming the corresponding role (whether positive or negative).

▶ **Lemma 27.** *For every $\tau \geq \gamma$, $i \in [1..n]$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \cap Q_C$ such that $\mathrm{task}_i(q)$ holds, there exists $q' \in \mathrm{act}(\mathbf{C}_\tau)$ such that $|\mathrm{role}(q')| = i$.*

4.3.5 Output propagation

The last rule allows the controller to communicate to the other agents whether it currently has its task list completely assigned or not. Note that the output of a state q is precisely the value of $\mathrm{out}(q)$, *i.e.* $O(q) \coloneqq \mathrm{out}(q)$.

rule	*state precondition*	*color precondition*	*state update*
(11)*	$\mathrm{ctrl}(p)$	*none*	$\mathrm{out}(q') = \bigwedge_{i=1}^{n} \mathrm{task}_i(p)$

▶ **Lemma 28.** *It is the case that $\psi(\mathbf{C}_0)$ holds iff there exists some $\tau \geq \gamma$ such that for every $\tau' \geq \tau$, there exists $q \in \mathrm{act}(\mathbf{C}_{\tau'}) \cap Q_C$ such that $\bigwedge_{i \in [1..n]} \mathrm{task}_i(q)$ holds.*

▶ **Corollary 29.** *There exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = \psi(\mathbf{C}_0)$.*

5 Conclusion

In this article, we introduced population protocols with unordered data; we presented such a protocol that computes majority over an infinite data domain; and we established the expressive power of immediate observation protocols: they compute interval predicates.

This work initiates the study of population protocols operating over arbitrarily large domains. Hence, this opens the door to numerous exciting questions, *e.g.* on space-efficient and time-efficient protocols. In particular, the expressive power of our model remains open.

There exist results on logics over data multisets (*e.g.*, see [22, 24]). In particular, the author of [22] provides a decidable logic reminiscent of Presburger arithmetic. It appears plausible that population protocols with unordered data compute (perhaps precisely) this logic. While we are fairly confident that remainder and threshold predicates with respect to the data counts can be computed in our model, the existential quantification, arising in the (non-ambiguous) solved forms of [22], seems more challenging to implement than the one of simple interval predicates.

Our model further relates to logic and automata on data words: inputs of a protocol with data can be seen as data words where Q is the alphabet and $\mathbb{D}$ is the data domain. Importantly, these data words are commutative, *i.e.*, permutations do not change acceptance. For example, the logic $\mathsf{FO}^2(+1, \sim, <)$ of [8] allows to specify non-commutative properties such as "there is a block of a's followed by a block of b's". In this respect, this logic is too "strong". It is also too "weak" as it cannot express "for each datum, the number of a's is even". For this same reason, $\mathsf{EMSO}^2(+1, \sim)$, and equivalently weak data automata [21], is too "weak". The logic $\mathsf{EMSO}^2_\#(+1, \sim)$, and equivalently commutative data automata [25], can express the latter, but, again, the successor relation allows to express non-commutative properties on letters. Thus, while models related to data words have been studied and could influence research on the complete characterization of the expressive power of our model, we have yet to directly connect them to our model.

References

1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

2 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

3 Gregor Bankhamer, Petra Berenbrink, Felix Biermeier, Robert Elsässer, Hamed Hosseinpour, Dominik Kaaser, and Peter Kling. Population protocols for exact plurality consensus: How a small chance of failure helps to eliminate insignificant opinions. In *Proc. 41st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 224–234, 2022.

4 Petra Berenbrink, Felix Biermeier, Christopher Hahn, and Dominik Kaaser. Loosely-stabilizing phase clocks and the adaptive majority problem. In *Proc. 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 7:1–7:17, 2022.

5 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and $\Theta(\log n)$ states. In *Proc. 32nd International Symposium on Distributed Computing (DISC)*, pages 10:1–10:18, 2018.

6 Michael Blondin, Javier Esparza, Blaise Genest, Martin Helfrich, and Stefan Jaax. Succinct population protocols for Presburger arithmetic. In *Proc. 37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 40:1–40:15, 2020.

7 Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J. Meyer. Towards efficient verification of population protocols. *Formal Methods in System Design (FMSD)*, 57(3):305–342, 2021.

8 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic (TOCL)*, 12(4):27:1–27:26, 2011.

9 Benedikt Bollig, Patricia Bouyer, and Fabian Reiter. Identifiers in registers – describing network algorithms with logic. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 115–132, 2019.

10 Benedikt Bollig, Fedor Ryabinin, and Arnaud Sangnier. Reachability in distributed memory automata. In *Proc. 29th EACSL Annual Conference on Computer Science Logic (CSL)*, pages 13:1–13:16, 2021.

11 Robert S. Boyer and J. Strother Moore. Mjrty: A fast majority vote algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, 1991.

12 Philipp Czerner, Roland Guttenberg, Martin Helfrich, and Javier Esparza. Fast and succinct population protocols for Presburger arithmetic. In *Proc. 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 11:1–11:17, 2022.

13 Giorgio Delzanno, Arnaud Sangnier, and Riccardo Traverso. Adding data registers to parameterized networks with broadcast. *Fundamenta Informaticae*, 143(3-4):287–316, 2016.

14 David Doty, Mahsa Eftekhari, Leszek Gasieniec, Eric E. Severson, Przemyslaw Uznanski, and Grzegorz Stachowiak. A time and space optimal stable population protocol solving exact majority. In *Proc. 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1044–1055, 2021.

15 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017.

16 Leszek Gasieniec, David D. Hamilton, Russell Martin, Paul G. Spirakis, and Grzegorz Stachowiak. Deterministic population protocols for exact majority and plurality. In *Proc. 20th International Conference on Principles of Distributed Systems (OPODIS)*, pages 14:1–14:14, 2016.

17 Utkarsh Gupta, Preey Shah, S. Akshay, and Piotr Hofman. Continuous reachability for unordered data Petri nets is in PTime. In *Proc. 22nd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 260–276, 2019.

18 Piotr Hofman, Slawomir Lasota, Ranko Lazic, Jérôme Leroux, Sylvain Schmitz, and Patrick Totzke. Coverability trees for Petri nets with unordered data. In *Proc. 19th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 445–461, 2016.

19 Piotr Hofman, Jérôme Leroux, and Patrick Totzke. Linear combinations of unordered data vectors. In *Proc. 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11, 2017.

20 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

21 Ahmet Kara, Thomas Schwentick, and Tony Tan. Feasible automata for two-variable logic with successor on data words. In *Proc. 6th International Conference on Language and Automata Theory and Applications (LATA)*, volume 7183, pages 351–362, 2012.

22 Denis Lugiez. Multitree automata that count. *Theoretical Computer Science*, 333(1-2):225–263, 2005.

23 Othon Michail and Paul G. Spirakis. Elements of the theory of dynamic networks. *Communications of the ACM*, 61(2):72, 2018.

24 Ruzica Piskac and Viktor Kuncak. Decision procedures for multisets with cardinality constraints. In *Proc. 9th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 218–232, 2008.

25 Zhilin Wu. Commutative data automata. In *Proc. 26th International Workshop/21st Annual Conference of the EACSL on Computer Science Logic (CSL)*, volume 16, pages 528–542, 2012.

A Missing proofs of Section 3

Let us first recall all of the transitions in a single table:

Pairing stage			
rule	*state precondition*	*color precondition*	*state update*
(1)	$\neg\mathrm{pair}(p)$	$d_1 \neq d_2$	$\mathrm{pair}(p') \land \mathrm{even}(p')$
	$\neg\mathrm{pair}(q)$		$\mathrm{pair}(q') \land \mathrm{even}(q')$
Grouping stage			
rule	*state precondition*	*color precondition*	*state update*
(2)	$\neg\mathrm{pair}(p)$	$d_1 \neq d_2$	*none*
	$\mathrm{pair}(q) \land \mathrm{grp}(q) \land \mathrm{maj}(q) = Y$		$\neg\mathrm{grp}(q') \land \mathrm{maj}(q') = N$
(3)	$\neg\mathrm{pair}(p)$	$d_1 = d_2$	*none*
	$\mathrm{pair}(q) \land \neg\mathrm{grp}(q) \land \mathrm{maj}(q) = N$		$\mathrm{grp}(q') \land \mathrm{maj}(q') = Y$
(4)	$\neg\mathrm{pair}(p)$	$d_1 \neq d_2$	*none*
	$\mathrm{pair}(q) \land \mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$		$\mathrm{maj}(q') = \overline{N}$
(5)	$\neg\mathrm{pair}(p)$	$d_1 = d_2$	*none*
	$\mathrm{pair}(q) \land \neg\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$		$\mathrm{maj}(q') = \overline{Y}$
(6)	$\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{N}$	*none*	$\neg\mathrm{grp}(p') \land \mathrm{maj}(p') = N$
	$\neg\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$		$\mathrm{maj}(q') = N$
(7)	$\neg\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{Y}$	*none*	$\mathrm{grp}(p') \land \mathrm{maj}(p') = Y$
	$\mathrm{grp}(q) \land \mathrm{maj}(q) \in \{y, n\}$		$\mathrm{maj}(q') = Y$
(8)	$\mathrm{grp}(p) \land \mathrm{maj}(p) = \overline{N}$	*none*	$\neg\mathrm{grp}(p') \land \mathrm{maj}(p') = n$
	$\neg\mathrm{grp}(q) \land \mathrm{maj}(q) = \overline{Y}$		$\mathrm{grp}(q') \land \mathrm{maj}(q') = n$
Majority stage			
rule	*state precondition*	*color precondition*	*state update*
(9)	$\neg\mathrm{pair}(p)$	*none*	*none*
	$\mathrm{even}(q)$		$\neg\mathrm{even}(q')$
(10)	$\mathrm{pair}(p) \land \mathrm{even}(p)$	*none*	*none*
	$\mathrm{pair}(q) \land \neg\mathrm{even}(q)$		$\mathrm{even}(q')$
(11)	$\mathrm{maj}(p) = Y$	*none*	$\mathrm{maj}(p') = n$
	$\mathrm{maj}(q) = N$		$\mathrm{maj}(q') = n$
(12)	$\mathrm{maj}(p) = Y$	*none*	*none*
	$\mathrm{maj}(q) = n$		$\mathrm{maj}(q') = y$
(13)	$\mathrm{maj}(p) = N$	*none*	*none*
	$\mathrm{maj}(q) = y$		$\mathrm{maj}(q') = n$
(14)	$\mathrm{maj}(p) = n$	*none*	*none*
	$\mathrm{maj}(q) = y$		$\mathrm{maj}(q') = n$

Note that for the sake of brevity, for the remainder of the section, we use the notation $\mathbf{C} \xrightarrow{(n)} \mathbf{C}'$ to denote that $\mathbf{C} \xrightarrow{t} \mathbf{C}'$ using a transition t arising from rule (n).

▶ **Lemma 4.** *There exists $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_U = |\mathbf{C}_{\tau+1}|_U = \cdots$. Furthermore, for every $i \geq \tau$, all unpaired agents of $\mathbf{C}_i$ share the same color, i.e. the set $\{d \in \mathbb{D} : \mathbf{C}_i(d, U) > 0\}$ is either empty or a singleton.*

Proof. First, note that no rule reverses a state from paired to unpaired. This implies that for any two $\mathbf{C}, \mathbf{C}'$ such that $\mathbf{C} \xrightarrow{*} \mathbf{C}'$, it is the case that $|\mathbf{C}|_U \geq |\mathbf{C}'|_U$. This yields a threshold $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_U = |\mathbf{C}_{\tau+1}|_U = \cdots$, as the value cannot drop below zero.

Assume that there exist infinitely many $i \geq \tau$ such that $\mathbf{C}_i(d, U) > 0$ and $\mathbf{C}_i(d', U) > 0$ holds for two distinct colors $d, d' \in \mathbb{D}$. Rule (1) is enabled in each such $\mathbf{C}_i$. This implies the existence of a configuration $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_U > |\mathbf{C}'_i|_U$. By fairness, some configuration from $\{\mathbf{C}'_i : i \in \mathbb{N}\}$ is reached along the execution, which is a contradiction. ◀

▶ **Lemma 5.** *Let $i \in \mathbb{N}$. If $\varphi_{maj}(\mathbf{C}_0)$ and d is the majority color, then $\mathbf{C}_i(d, U) > 0$.*

Proof. Since d is the majority color and all agents are initially unpaired, we have

$$\mathbf{C}_0(d, U) > \sum_{d' \neq d} \mathbf{C}_0(d', U). \tag{3}$$

Moreover, for every $i > 0$ such that $\mathbf{C}_{i-1} \xrightarrow{(1)} \mathbf{C}_i$, at most of one state $q \in \operatorname{act}(\mathbf{C}_{i-1}(d))$ is paired while at least one color $d' \neq d$ has one of its states $q' \in \operatorname{act}(\mathbf{C}_{i-1}(d'))$ paired. Therefore, we either have $\mathbf{C}_i(d, U) = \mathbf{C}_{i-1}(d, U)$ or $\mathbf{C}_i(d, U) = \mathbf{C}_{i-1}(d, U) - 1$, and we have

$$\sum_{d' \neq d} \mathbf{C}_i(d', U) < \sum_{d' \neq d} \mathbf{C}_{i-1}(d', U).$$

Consequently, by (3), we have $\mathbf{C}_i(d, U) > \sum_{d' \neq d} \mathbf{C}_i(d', U) \geq 0$.

Recall that only rule (1) changes the pairing of a state. Thus, $\mathbf{C}_i(d, U) > 0$ holds for all $i \in \mathbb{N}$. ◀

▶ **Lemma 6.** *For every $i \in \mathbb{N}$, it is the case that $|\mathbf{C}_i|_{Q_Y} - |\mathbf{C}_i|_{Q_N} = |\mathbf{C}_i|_{Q_M} - |\mathbf{C}_i|_{Q_m}$.*

Proof. We prove the claim by induction on $i \geq 0$. We have $|\mathbf{C}_0|_{Q_N} = |\mathbf{C}_0|_{Q_m} = 0$ and $|\mathbf{C}_0|_{Q_Y} = |\mathbf{C}_0|_{Q_M} = |\mathbf{C}_0|$. Therefore, the claim holds initially.

Given a transition $t = ((p, q), \sim, (p', q')) \in \delta$ and $x \in \{Y, N, M, m\}$, we define $\Delta_x(t) := \{\!\{p', q'\}\!\}(Q_x) - \{\!\{p, q\}\!\}(Q_x)$. This table illustrates the value of $\Delta_x(t)$ for each t:

$t \in \delta$ from rule #	$\Delta_Y(t)$	$\Delta_N(t)$	$\Delta_M(t)$	$\Delta_m(t)$	$\Delta_Y(t) - \Delta_N(t)$	$\Delta_M(t) - \Delta_m(t)$
(2)	-1	1	-1	1	-2	-2
(3)	1	-1	1	-1	2	2
(6)	0	2	-1	1	-2	-2
(7)	2	0	1	-1	2	2
(11)	-1	-1	0	0	0	0
other rules	0	0	0	0	0	0

Let $i > 0$. Observe that, by definition, $|\mathbf{C}_i|_{Q_x} = |\mathbf{C}_{i-1}|_{Q_x} + \Delta_x(t)$ holds for every $t \in \delta$ and

$x \in \{Y, N, M, n\}$. Thus,

$$\begin{aligned}
|\mathbf{C}_i|_{Q_Y} - |\mathbf{C}_i|_{Q_N} &= (|\mathbf{C}_{i-1}|_{Q_Y} + \Delta_Y(t)) - (|\mathbf{C}_{i-1}|_{Q_N} + \Delta_N(t)) \\
&= \big(|\mathbf{C}_{i-1}|_{Q_Y} - |\mathbf{C}_{i-1}|_{Q_N}\big) + \big(\Delta_Y(t) - \Delta_N(t)\big) \\
&= \big(|\mathbf{C}_{i-1}|_{Q_M} - |\mathbf{C}_{i-1}|_{Q_m}\big) + \big(\Delta_Y(t) - \Delta_N(t)\big) \quad \text{(by induction hyp.)} \\
&= \big(|\mathbf{C}_{i-1}|_{Q_M} - |\mathbf{C}_{i-1}|_{Q_m}\big) + \big(\Delta_M(t) - \Delta_m(t)\big) \quad \text{(by above table)} \\
&= \big(|\mathbf{C}_{i-1}|_{Q_M} + \Delta_M(t)\big) - \big(|\mathbf{C}_{i-1}|_{Q_m} + \Delta_m(t)\big) \\
&= |\mathbf{C}_i|_{Q_M} - |\mathbf{C}_i|_{Q_m}.
\end{aligned}$$

◀

▶ **Lemma 8.** *Let E be the set of states engaged in the majority computation, i.e. $E \coloneqq \{q \in Q : \mathrm{maj}(q) \in \{y, n, \overline{Y}, \overline{N}\}\}$. Let $E_M \coloneqq E \cap Q_M$ and $E_m \coloneqq E \cap Q_m$. For every $i \in \mathbb{N}$, the following holds: $|\mathbf{C}_i|_{E_M} = |\mathbf{C}_i|_{E_m}$.*

Proof. We proceed by induction on $i \in \mathbb{N}$. The claim trivially holds for $\mathbf{C}_0$, as the unique initial state q_I satisfies $\mathrm{maj}(q_I) \notin E$, which implies $|\mathbf{C}_0|_E = |\mathbf{C}_0|_{E_M} = |\mathbf{C}_0|_{E_m} = 0$.

Let $i > 0$ and let t be such that $\mathbf{C}_{i-1} \xrightarrow{t} \mathbf{C}_i$. For $|\mathbf{C}_i|_{E_M}$ (resp. $|\mathbf{C}_i|_{E_m}$) to be different from $|\mathbf{C}_{i-1}|_{E_M}$ (resp. $|\mathbf{C}_{i-1}|_{E_m}$), t must be (6), (7) or (11). Let us take a look at the effect of each of these transitions:

$t \in \delta$ from rule #	$\lvert\mathbf{C}_i\rvert_{E_M} - \lvert\mathbf{C}_{i-1}\rvert_{E_M}$	$\lvert\mathbf{C}_i\rvert_{E_m} - \lvert\mathbf{C}_{i-1}\rvert_{E_m}$
(6)	-1	-1
(7)	-1	-1
(11)	1	1

Observe that rule (11) does not explicitly specify the groups of states p and q. However, no rule generates a state q such that $\mathrm{maj}(q) = Y$ (resp. $\mathrm{maj}(q) = N$) and $\neg\mathrm{grp}(q)$ (resp. $\mathrm{grp}(q)$), and q_I satisfies $\mathrm{maj}(q_I) = Y \land \mathrm{grp}(q_I)$; therefore, for states p and q of rule (11), $\mathrm{grp}(p)$ and $\neg\mathrm{grp}(q)$ implicitly hold. Since the effect is the same on each side of the equation for all transitions, the claim follows by induction hypothesis. ◀

▶ **Lemma 9.** *Let $d \in \mathbb{D}$. If $\mathbf{C}_\alpha(d, U) > 0$, then there exists some $\tau \geq \alpha$ such that, for all $i \geq \tau$, $d' \in \mathbb{D}$ and $q \in \mathrm{act}(\mathbf{C}_i(d'))$, the following holds: $\mathrm{grp}(q) = (d' = d)$.*

Proof. We claim that there exist $\tau, \tau', \tau'', \tau''' \geq \alpha$ such that

1. $\mathbf{C}_i(d, Q_{\overline{N}}) = 0$ for all $i \geq \tau$;
2. $\sum_{d' \neq d} \mathbf{C}_i(d', Q_{\overline{Y}}) = 0$ for all $i \geq \tau'$;
3. $\mathbf{C}_i(d, Q_m) = 0$ for all $i \geq \tau''$;
4. $\sum_{d' \neq d} \mathbf{C}_i(d', Q_M) = 0$ for all $i \geq \tau'''$.

Before proving the claims, observe that the lemma follows by taking $\max(\tau, \tau', \tau'', \tau''')$.

Claim 1. By definition of α and Lemma 4, there are no unpaired agents of a color $d' \neq d$ in $\mathbf{C}_\alpha$ to play the corresponding role in rule (4). Moreover, rule (4) is the only rule generating states from $Q_{\overline{N}}$. Thus, for every $i \geq \alpha$ and $\mathbf{C}_i \xrightarrow{*} \mathbf{C}$, we have $\mathbf{C}_i(d, Q_{\overline{N}}) \geq \mathbf{C}(d, Q_{\overline{N}})$.

It remains to show that the value decreases to zero. If $\mathbf{C}_i(d, Q_{\overline{N}}) > 0$ holds for only finitely many indices $i \geq \alpha$, then we are done. Otherwise, we must have $\mathbf{C}_i(d, Q_{\overline{N}} \cap Q_M) > 0$ for infinitely many $i \geq \alpha$. Indeed, rule (4) updates $\mathrm{maj}(q') = \overline{N}$ under the precondition $\mathrm{grp}(q)$, and no rule produces a state from $Q_{\overline{N}} \cap Q_m$. By Lemma 8, this implies the existence of a state from $\mathrm{act}(\mathbf{C}_i) \cap E_m$ enabling rule (6) or rule (8). Thus, there exists a configuration $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $\mathbf{C}_i(d, Q_{\overline{N}}) > \mathbf{C}'_i(d, Q_{\overline{N}})$. Hence, by fairness, we obtain $\mathbf{C}_\tau(d, Q_{\overline{N}}) = 0$ for some $\tau \geq \alpha$. Hence, $\mathbf{C}_\tau(d, Q_{\overline{N}}) = \mathbf{C}_{\tau+1}(d, Q_{\overline{N}}) = \cdots = 0$ (as the value cannot increase).

Claim 2. Similarly to Claim 1, as unpaired agents share color d, rule (5) is only enabled for agents of color d for $\mathbf{C}_\alpha$. So, as rule (5) is the only rule generating states from $Q_{\overline{Y}}$, we have

$$\sum_{d' \neq d} \mathbf{C}_i(d', Q_{\overline{Y}}) \geq \sum_{d' \neq d} \mathbf{C}(d', Q_{\overline{Y}}) \text{ for every } i \geq \alpha \text{ and } \mathbf{C}_i \xrightarrow{*} \mathbf{C}.$$

It remains to show that the value decreases to zero. If $\sum_{d' \neq d} \mathbf{C}_i(d', Q_{\overline{Y}}) > 0$ for only finitely many indices $i \geq \alpha$, then we are done. Otherwise, we must have $\mathbf{C}_i(d, Q_{\overline{Y}} \cap Q_m) > 0$ for infinitely many $i \geq \alpha$. Indeed, rule (5) updates $\text{maj}(q') = \overline{Y}$ under the precondition $\neg\text{grp}(q)$, and no rule produces a state from $Q_{\overline{Y}} \cap Q_M$. By Lemma 8, some state from $\text{act}(\mathbf{C}_i) \cap E_M$ enables rule (7) or rule (8). Thus, there exists a configuration $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and

$$\sum_{d' \neq d} \mathbf{C}_i(d', Q_{\overline{Y}}) > \sum_{d' \neq d} \mathbf{C}'_i(d', Q_{\overline{Y}}).$$

Thus, by fairness, we obtain $\sum_{d' \neq d} \mathbf{C}_{\tau'}(d', Q_{\overline{Y}}) = 0$ for some $\tau' \geq \alpha$. So, $\sum_{d' \neq d} \mathbf{C}_{\tau'}(d', Q_{\overline{Y}}) = \sum_{d' \neq d} \mathbf{C}_{\tau'+1}(d', Q_{\overline{Y}}) = \cdots = 0$ (as the value cannot increase).

Claim 3. Let $\alpha' := \max(\tau, \tau')$. By Claim 1, from $\mathbf{C}_{\alpha'}$, since no agent of color d is in a state from $Q_{\overline{N}}$, rules (6) and (8) can no longer change the group of an agent of color d to the minority group, *i.e.* set $\text{grp}(q)$ to *false*. Moreover, since agents of color $d' \neq d$ are all paired by Lemma 4, rule (2) can also no longer change an agent of color d to the minority group. Therefore, for any $\mathbf{C}'$ such that $\mathbf{C}_{\alpha'} \xrightarrow{*} \mathbf{C}'$, we have $\mathbf{C}_{\alpha'}(d, Q_m) \geq \mathbf{C}'(d, Q_m)$.

If $\mathbf{C}_i(d, Q_m) > 0$ for only finitely many indices $i \geq \alpha'$, then we are done. Thus, assume that $\mathbf{C}_i(d, Q_m) > 0$ for infinitely many $i \geq \alpha'$. If there exists $q \in \text{act}(\mathbf{C}_i(d)) \cap Q_m \cap P$, then, depending on the value of $\text{maj}(q)$, either rule (3); rule (5) followed by rule (7); or rule (5) followed by rule (8) is enabled. In each case, it yields $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $\mathbf{C}_i(d, Q_m \cap P) > \mathbf{C}'_i(d, Q_m \cap P)$. Thus, by fairness, we obtain $\tau'' \geq \alpha$ such that $\mathbf{C}_i(d, Q_m \cap P) = 0$ for all $i \geq \tau''$.

It remains to show that $\mathbf{C}_{\tau''}(d, Q_m \cap U) = 0$. Rules (2) and (3) have the precondition $\text{pair}(q)$ for the only modified state q'. Hence, they cannot change the group of an unpaired agent. While rules (6), (7) and (8) do not have such a precondition, they require state p to satisfy $\text{maj}(p) \in \{\overline{Y}, \overline{N}\}$. The only rules changing a state to satisfy that condition are (4) and (5). Since both rules only change a state q to q' satisfying $\text{maj}(q') \in \{\overline{Y}, \overline{N}\}$ if $\text{pair}(q)$ holds, unpaired agents are never in a state meeting the requirements to have their group changed. From these observations, we conclude that a state $q \in \text{act}(\mathbf{C}_{\tau''})$ cannot satisfy both $\neg\text{grp}(q)$ and $\neg\text{pair}(q)$. Thus, we have $\mathbf{C}_{\tau''}(d, Q_m \cap U) = 0$ as desired.

Claim 4. Let $\alpha' := \max(\tau, \tau')$. By Claim 2, from $\mathbf{C}_{\alpha'}$, no agent of color $d' \neq d$ is in a state from $Q_{\overline{Y}}$. Therefore, rules (7) and (8) can no longer change the group of an agent of color $d' \neq d$ to the majority group. Since there is no unpaired agent of color $d' \neq d$ by Lemma 4, rule (3) can also no longer change their group to the majority group. So, for any $\mathbf{C}'$ such that $\mathbf{C}_{\alpha'} \xrightarrow{*} \mathbf{C}'$ and for any $d' \neq d$, we have $\mathbf{C}_{\alpha'}(d', Q_M) \geq \mathbf{C}'(d', Q_M)$.

If $\sum_{d' \neq d} \mathbf{C}_i(d', Q_M) > 0$ for only finitely many indices $i \geq \alpha'$, then we are done. Thus, assume that there exist infinitely many $i \geq \alpha'$ such that for $d' \neq d$, there exists $q \in \text{act}(\mathbf{C}_i(d')) \cap Q_M$. Depending on the value of $\text{maj}(q)$, either rule (2); rule (4) followed by rule (6); or rule (4) followed by rule (8) is enabled. In each case, there exists a configuration $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $\mathbf{C}_i(d', Q_M) > \mathbf{C}'_i(d', Q_M)$. Therefore, we are done by fairness. ◀

▶ **Lemma 10.** *There exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \text{act}(\mathbf{C}_i)$, it is the case that* $\text{even}(q)$ *holds iff* $|\mathbf{C}_i|_U = 0$.

Proof. Note that from $\mathbf{C}_\alpha$ onwards, only rules (9) and (10) may change a state q to a state q' such that $\text{even}(q') \neq \text{even}(q)$. Let $Q_e \coloneqq \{q \in Q : \text{even}(q)\}$. We make a case distinction.

Case $|\mathbf{C}_\alpha|_U > 0$. Suppose that there exist infinitely many $i \geq \alpha$ such that, for some $q \in \text{act}(\mathbf{C}_i)$, $\text{even}(q)$ holds, as we are otherwise done. As rule (9) is enabled, there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_e} > |\mathbf{C}'_i|_{Q_e}$. By repeatedly invoking fairness, we obtain $\tau \geq \alpha$ such that $\mathbf{C}_\alpha \xrightarrow{*} \mathbf{C}_\tau$ and $|\mathbf{C}_\tau|_{Q_e} = 0$. Moreover, for any $\mathbf{C}'$ such that $\mathbf{C}_\tau \xrightarrow{*} \mathbf{C}'$, rules (9) and (10) are permanently disabled since $|\mathbf{C}'|_{Q_e} = 0$. So, for any $q \in \text{act}(\mathbf{C}')$, we have $\neg\text{even}(q)$.

Case $|\mathbf{C}_\alpha|_U = 0$. Since every agent is initially unpaired, we necessarily have $\alpha > 0$. For every $i \geq \alpha$, note that rule (9) is disabled in $\mathbf{C}_i$. Therefore, only rule (10) can change a state q to a state q' such that $\text{even}(q') \neq \text{even}(q)$. By minimality of α, we have

$$\mathbf{C}_{\alpha-1} \xrightarrow{(1)} \mathbf{C}_\alpha, \text{ and hence } |\mathbf{C}_\alpha|_{Q_e} \geq 2.$$

Since rule (9) is permanently disabled, we have $|\mathbf{C}_i|_{Q_e} > 0$ for every $i \geq \alpha$.

If $\text{act}(\mathbf{C}_i) \not\subseteq Q_e$ for only finitely many indices $i \geq \alpha$, then we are done. So, assume that there exist infinitely many i such that for some $q \in \text{act}(\mathbf{C}_i)$, $\neg\text{even}(q)$ holds. Rule (10) is enabled, which means that there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_e} < |\mathbf{C}'_i|_{Q_e}$. Therefore, by repeatedly invoking fairness, we obtain some $\tau \geq \alpha$ such that $\mathbf{C}_\alpha \xrightarrow{*} \mathbf{C}_\tau$ and $|\mathbf{C}_\tau|_{Q_e} = |\mathbf{C}_\tau|$. Thus, $\text{act}(\mathbf{C}_i) \subseteq Q_e$ holds for all $i \geq \tau$. ◀

▶ **Lemma 11.** *If $\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that for every $i \geq \tau$ and $q \in \text{act}(\mathbf{C}_i)$, it is the case that $\text{maj}(q) \in \{Y, y\}$ and $\neg\text{even}(q)$ hold.*

Proof. Let $d \in \mathbb{D}$ be the majority color. By Lemma 5, we have $\mathbf{C}_\alpha(d, U) > 0$. As there is a majority for d, we have $\mathbf{C}_i(d, Q) > \sum_{d' \neq d} \mathbf{C}_i(d', Q)$ for all $i \in \mathbb{N}$. Let τ be given by Lemma 9 for color d. The following holds for all $i \geq \tau$:

$$\text{act}(\mathbf{C}_i(d)) \subseteq Q_M \text{ and } \text{act}(\mathbf{C}_i(d')) \subseteq Q_m \text{ for all } d' \neq d.$$

We can infer from the previous statements that $|\mathbf{C}_i|_{Q_M} > |\mathbf{C}_i|_{Q_m}$ for all $i \geq \tau$. Combined with Lemma 6, we conclude that $|\mathbf{C}_i|_{Q_Y} > |\mathbf{C}_i|_{Q_N}$ for all $i \geq \tau$.

Let us show that the number of active states from Q_N decreases permanently to zero. Assume that there are infinitely many indices $i \geq \tau$ such that $|\mathbf{C}_i|_{Q_N} > 0$, as we are otherwise done. As there is at least one state $q \in \text{act}(\mathbf{C}_i) \cap Q_N$ and $|\mathbf{C}_i|_{Q_Y} > |\mathbf{C}_i|_{Q_N}$, rule (11) is enabled. Thus, there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_N} > |\mathbf{C}'_i|_{Q_N}$. So, by repeatedly invoking fairness, we obtain some $\tau' \geq \tau$ such that $|\mathbf{C}_{\tau'}|_{Q_N} = 0$. From $\mathbf{C}_{\tau'}$ onwards, rules (11) and (13) are disabled.

Let us now show that the number of active states from Q_n decreases permanently to zero. Assume that there exist infinitely many $i \geq \tau'$ such that $|\mathbf{C}_i|_{Q_n} > 0$, as we are otherwise done. Since there is at least one state $q \in \text{act}(\mathbf{C}_i) \cap Q_Y$, rule (12) is enabled. Therefore, there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_n} > |\mathbf{C}'_i|_{Q_n}$. By repeatedly invoking fairness, we obtain some τ'' such that $|\mathbf{C}_{\tau''}|_{Q_n} = 0$. From there, rules (12) and (14) are disabled.

Let us now prove that there are no active states q in $\mathbf{C}_\tau$ (and any $\mathbf{C}'$ reachable from $\mathbf{C}_\tau$) such that $\text{maj}(q) \in \{\overline{Y}, \overline{N}\}$. Recall the claims from Lemma 9; agents of d are no longer in states from $Q_{\overline{N}}$ and agents of $d' \neq d$ are no longer in states of $Q_{\overline{Y}}$. Furthermore, all agents are in the correct group, *i.e.* agents of d are in the majority group while agents of $d' \neq d$ are in the minority group. Assume that for $i \geq \tau$, an agent of $d' \neq d$ (resp. d) is in a state $q \in \text{act}(\mathbf{C}_i)$ such that $\text{maj}(q) = \overline{N}$ (resp. $\text{maj}(q) = \overline{Y}$). This implies the existence of a configuration $\mathbf{C}'$ reachable from $\mathbf{C}_i$ such that some state $q' \in \text{act}(\mathbf{C}'(d'))$

(resp. $q' \in \mathrm{act}(\mathbf{C}'(d))$) satisfies $\mathrm{grp}(q')$ (resp. $\neg\mathrm{grp}(q')$). By fairness, such a configuration is reached, which is a contradiction to the definition of τ from Lemma 9. Thus, for any state $q \in \mathrm{act}(\mathbf{C}_{\tau''})$, we have $\mathrm{maj}(q) \in \{Y, y\}$.

Finally, by Lemma 10, there exists $\tau''' \geq \tau''$ such that for every $i \geq \tau'''$ and $q \in \mathrm{act}(\mathbf{C}_i)$, it is the case that $\neg\mathrm{even}(q)$ holds. ◀

▶ **Lemma 12.** *If $\neg\varphi_{maj}(\mathbf{C}_0)$ holds, then there exists $\tau \geq \alpha$ such that either:*

- $\mathrm{even}(q)$ *holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$; or*
- $\mathrm{maj}(q) \in \{N, n\}$ *holds for every $i \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_i)$.*

Proof. If $|\mathbf{C}_\alpha|_U = 0$, then, by Lemmas 4 and 10, there exists $u \geq \alpha$ such that for any $i \geq u$, each state $q \in \mathrm{act}(\mathbf{C}_i)$ satisfies $\mathrm{even}(q)$.

Now, assume this is not the case. By Lemma 4, there is a unique color $d \in \mathbb{D}$ such that $|\mathbf{C}_\alpha(d)|_U > 0$. As there is no majority color, we have $\mathbf{C}_i(d, Q) \leq \sum_{d' \neq d} \mathbf{C}_i(d', Q)$ for every $i \in \mathbb{N}$. Let $\tau \geq \alpha$ be given by Lemma 9 and taken as minimal. By Lemma 9, for every $i \geq \tau$:

$$\mathrm{act}(\mathbf{C}_i(d)) \subseteq Q_M \text{ and } \mathrm{act}(\mathbf{C}_i(d')) \subseteq Q_m \text{ for all } d' \neq d.$$

So, we have $|\mathbf{C}_i|_{Q_M} \leq |\mathbf{C}_i|_{Q_m}$ for all $i \geq \tau$. By Lemma 6, this implies that $|\mathbf{C}_i|_{Q_Y} \leq |\mathbf{C}_i|_{Q_N}$ for all $i \geq \tau$.

Let us show that the number of active states from Q_Y decreases permanently to zero. Assume that there exist infinitely many indices $i \geq \tau$ such that $|\mathbf{C}_i|_{Q_Y} > 0$. As there is at least one state $q \in \mathrm{act}(\mathbf{C}_i) \cap Q_Y$ and $|\mathbf{C}_i|_{Q_Y} \leq |\mathbf{C}_i|_{Q_N}$, rule (11) is enabled. So, there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_Y} > |\mathbf{C}'_i|_{Q_Y}$. By repeatedly invoking fairness, we obtain some $\tau' \geq \tau$ with $|\mathbf{C}_{\tau'}|_{Q_Y} = 0$. Let τ' be minimal. Rules (11) and (12) are now disabled.

Since rule (12) is disabled, we have $|\mathbf{C}_i|_{Q_y} \geq |\mathbf{C}_{i+1}|_{Q_y}$ for all $i \geq \tau'$. Let us show that the number of active states from Q_y decreases permanently to zero. Two subcases arise.

First, consider the case where $|\mathbf{C}_{\tau'}|_{Q_N} > 0$. Assume that there exist infinitely many $i \geq \tau'$ such that $|\mathbf{C}_i|_{Q_y} > 0$, as we are otherwise done. Since there exists at least one state $q \in \mathrm{act}(\mathbf{C}_i) \cap Q_N$, rule (13) is enabled. Therefore, there exists $\mathbf{C}'_i$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'_i$ and $|\mathbf{C}_i|_{Q_y} > |\mathbf{C}'_i|_{Q_y}$. By repeatedly invoking fairness, we obtain some $\tau'' \geq \tau'$ such that $|\mathbf{C}_{\tau''}|_{Q_y} = 0$. This disables rules (13) and (14).

Let us now consider the remaining case where $|\mathbf{C}_{\tau'}|_{Q_N} = 0$. From Lemma 6, it follows that for any $v \geq \tau$, $|\mathbf{C}_v|_{Q_M} = |\mathbf{C}_v|_{Q_m}$. Since each agent is initially in a state q such that $\mathrm{grp}(q)$ holds, since $|\mathbf{C}_\alpha|_U > 0$ and since $\varphi_{\mathrm{maj}}(\mathbf{C}_0) = \mathit{false}$, at least one agent has its group changed in the execution. There exists a transition $t = ((p, q), \sim, (p', q'))$ and data $d_1, d_2 \in \mathbb{D}$ such that $(\mathrm{grp}(p) \neq \mathrm{grp}(p')) \vee (\mathrm{grp}(q) \neq \mathrm{grp}(q'))$ holds, $\mathbf{C}_{\tau-1} \xrightarrow{t} \mathbf{C}_\tau$ and $\mathbf{C}_\tau = \mathbf{C}_{\tau-1} - (\boldsymbol{p}_{d_1} + \boldsymbol{q}_{d_2}) + (\boldsymbol{p}'_{d_1} + \boldsymbol{q}'_{d_2})$. Observe that t must be a transition arising from a rule (2), (3), (6), (7) or (8). We identify two subcases.

Subcase: transition t arises from rule (2), (3), (6) or (7). Recall that for any $v \geq \tau$, groups are stabilized in $\mathbf{C}_v$, and hence $|\mathbf{C}_v|_{Q_Y} = |\mathbf{C}_v|_{Q_N}$ since $|\mathbf{C}_v|_{Q_M} = |\mathbf{C}_v|_{Q_m}$. We know from the definition of the rules that either $|\mathbf{C}_\tau|_{Q_N} > 0$ (which implies $|\mathbf{C}_\tau|_{Q_Y} > 0$) or $|\mathbf{C}_\tau|_{Q_Y} > 0$ (which implies $|\mathbf{C}_\tau|_{Q_N} > 0$). Therefore,

$$\mathbf{C}_{\tau'-1} \xrightarrow{(11)} \mathbf{C}_{\tau'} \text{ and } |\mathbf{C}_{\tau'-1}|_{Q_Y} = |\mathbf{C}_{\tau'-1}|_{Q_N} = 1.$$

By definition of rule (12), it follows that $|\mathbf{C}_{\tau'}|_{Q_n} \geq 2$.

Subcase: transition t arises from rule (8). By definition of (8), we have $|\mathbf{C}_\tau|_{Q_n} \geq 2$. If $|\mathbf{C}_\tau|_{Q_Y} = |\mathbf{C}_\tau|_{Q_N} = 0$, then $\tau = \tau'$. Otherwise, we have

$$|\mathbf{C}_\tau|_{Q_Y} = |\mathbf{C}_\tau|_{Q_N} > 0 \text{ and } \mathbf{C}_{\tau'-1} \xrightarrow{(11)} \mathbf{C}_{\tau'}.$$

Again, it follows that $|\mathbf{C}_{\tau'}|_{Q_n} \geq 2$.

In both subcases, we have $|\mathbf{C}_{\tau'}|_{Q_n} > 0$. Observe that rule (13) is disabled. Thus, only rule (14) can now change a state q to a state q' such that $\mathrm{maj}(q') \neq \mathrm{maj}(q)$. Since $|\mathbf{C}_{\tau'}|_{Q_n} > 0$, rule (14) is enabled for any $j \geq \tau'$ such that $|\mathbf{C}_j|_{Q_y} > 0$. So, by repeatedly invoking fairness, we obtain some $\tau'' \geq \tau'$ such that $|\mathbf{C}_{\tau''}|_{Q_y} = 0$.

Using the results shown in the proof of Lemma 11, we know there are no active states q in $\mathbf{C}_\tau$ (and in any $\mathbf{C}'$ reachable from $\mathbf{C}_\tau$) with $\mathrm{maj}(q) \in \{\overline{Y}, \overline{N}\}$. Thus, we are done, as $\mathrm{maj}(q) \in \{N, n\}$ holds for every $i \geq \tau''$ and $q \in \mathrm{act}(\mathbf{C}_i)$. ◀

B Missing proofs of Section 4.1

▶ **Lemma A1** (monotonicity). *Let $\mathbf{C}$, $\overline{\mathbf{C}}$ and $\mathbf{C}'$ be configurations such that $\mathbf{C} \xrightarrow{*} \overline{\mathbf{C}}$ and $\mathbf{C} \sqsubseteq \mathbf{C}'$. There exists $\overline{\mathbf{C}'}$ such that $\mathbf{C}' \xrightarrow{*} \overline{\mathbf{C}'}$ and $\overline{\mathbf{C}} \sqsubseteq \overline{\mathbf{C}'}$.*

Proof. We consider the case where $\mathbf{C} \xrightarrow{t} \overline{\mathbf{C}}$. The general case "$\xrightarrow{*}$" follows by induction. Let $\rho\colon \mathbb{D} \to \mathbb{D}$ be an injection such that $\mathbf{C}(x) \leq \mathbf{C}'(\rho(x))$ for all $x \in \mathbb{D}$. Let $t = ((p, q), \sim, (p', q'))$ be the transition used in $\mathbf{C}$ with $d, e \in \mathbb{D}$. As $\mathbf{C} \geq \boldsymbol{p}_d + \boldsymbol{q}_e$, we have $\mathbf{C}' \geq \boldsymbol{p}_{\rho(d)} + \boldsymbol{q}_{\rho(e)}$. So, we can use t in $\mathbf{C}'$ to obtain $\overline{\mathbf{C}'} := \mathbf{C}' - (\boldsymbol{p}_{\rho(d)} + \boldsymbol{q}_{\rho(e)}) + (\boldsymbol{p}'_{\rho(d)} + \boldsymbol{q}'_{\rho(e)})$. We have $\overline{\mathbf{C}}(d) \leq \overline{\mathbf{C}'}(\rho(x))$ for all $x \in \mathbb{D}$, and hence $\mathbf{C}' \sqsubseteq \overline{\mathbf{C}'}$. ◀

▶ **Lemma A2.** *The set $\mathcal{U}$ is upward closed and has a finite basis.*

Proof. It suffices to show that $\mathcal{U}$ is upward closed. Indeed, since $\sqsubseteq$ is a well-quasi-order (*e.g.*, see [18]), it follows that $\mathcal{U}$ has a finite basis.

Let $\mathbf{C}$ and $\mathbf{C}'$ be configurations such that $\mathbf{C} \in \mathcal{U}$ and $\mathbf{C} \sqsubseteq \mathbf{C}'$. If $O(\mathbf{C}') = \bot$, then we are done. Therefore, assume that $O(\mathbf{C}') = b \in \{0, 1\}$. Since $\mathrm{act}(\mathbf{C}) \subseteq \mathrm{act}(\mathbf{C}')$, we have $O(\mathbf{C}) = b$. As $\mathbf{C} \in \mathcal{U}$, there exists a configuration $\overline{\mathbf{C}}$ such that $\mathbf{C} \xrightarrow{*} \overline{\mathbf{C}}$ and $O(\overline{\mathbf{C}}) \neq b$. By Lemma A1, there exists $\overline{\mathbf{C}'}$ such that $\mathbf{C}' \xrightarrow{*} \overline{\mathbf{C}'}$ and $\overline{\mathbf{C}} \sqsubseteq \overline{\mathbf{C}'}$. As $\mathrm{act}(\overline{\mathbf{C}}) \subseteq \mathrm{act}(\overline{\mathbf{C}'})$, we either have $O(\overline{\mathbf{C}'}) = O(\overline{\mathbf{C}})$ or $O(\overline{\mathbf{C}'}) = \bot$. So, we obtain $\mathbf{C}' \in \mathcal{U}$. ◀

▶ **Lemma A3.** *Let $k \geq 1$ and let $\mathbf{C}, \mathbf{C}'$ be configurations such that $\mathbf{C} \sqsubseteq \mathbf{C}'$. It is the case that $\tau_k(\mathbf{C}) \sqsubseteq \mathbf{C}$ and $\tau_k(\mathbf{C}) \sqsubseteq \tau_k(\mathbf{C}')$.*

Proof. For every $d \in \mathbb{D}$ and $q \in Q$, we have $\tau_k(\mathbf{C})(d, q) = \min(\mathbf{C}(d, q), k) \leq \mathbf{C}(d, q)$. Thus, $\tau_k(\mathbf{C}) \sqsubseteq \mathbf{C}$ trivially holds (using the identity function).

Let $\rho\colon \mathbb{D} \to \mathbb{D}$ be an injection such that $\mathbf{C}(d) \leq \mathbf{C}'(\rho(d))$ for every $d \in \mathbb{D}$. For every $d \in \mathbb{D}$ and $q \in Q$, we have

$$\tau_k(\mathbf{C})(d, q) = \min(\mathbf{C}(d, q), k) \leq \min(\mathbf{C}'(\rho(d), q), k) = \tau_k(\mathbf{C}')(\rho(d), q).$$

Thus, $\tau_k(\mathbf{C}) \sqsubseteq \tau_k(\mathbf{C}')$ (using injection ρ). ◀

▶ **Lemma 14.** *Let ψ be a predicate computed by a population protocol with unordered data. Let S_b be the set of stable configurations with output b of the protocol. There exists $k \geq 1$ such that, for all $b \in \{0, 1\}$, we have $\mathbf{C} \in \mathcal{S}_b$ iff $\tau_k(\mathbf{C}) \in \mathcal{S}_b$.*

Proof. By Lemma A2, the set $\mathcal{U}$ has a finite basis $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$. Let $k := \max\{\mathbf{C}_i(d, q) : d \in \mathbb{D}, q \in Q\}$. Note that $\tau_k(\mathbf{C}_i) = \mathbf{C}_i$ for all $i \in [1..n]$. Let us first show that $\mathbf{C} \in \mathcal{U}$ iff $\tau_k(\mathbf{C}) \in \mathcal{U}$.

$\Rightarrow$) As $\mathbf{C} \in \mathcal{U}$, we have $\mathbf{C}_i \sqsubseteq \mathbf{C}$ for some $i \in [1..n]$. By Lemma A3, we have $\tau_k(\mathbf{C}_i) \sqsubseteq \tau_k(\mathbf{C})$. Since $\mathbf{C}_i = \tau_k(\mathbf{C}_i)$, we obtain that $\mathbf{C}_i \sqsubseteq \tau_k(\mathbf{C})$ and hence $\tau_k(\mathbf{C}) \in \mathcal{U}$.

$\Leftarrow$) Since $\tau_k(\mathbf{C}) \in \mathcal{U}$, we have $\mathbf{C}_i \sqsubseteq \tau_k(\mathbf{C})$ for some $i \in [1..n]$. By Lemma A3, we have $\tau_k(\mathbf{C}) \sqsubseteq \mathbf{C}$. Consequently, $\mathbf{C}_i \sqsubseteq \mathbf{C}$, which implies $\mathbf{C} \in \mathcal{U}$.

It remains to prove the main claim. As $\mathcal{S}_0 \cup \mathcal{S}_1$ is the complement of $\mathcal{U}$, we have $\mathbf{C} \in \mathcal{S}_0 \cup \mathcal{S}_1$ iff $\tau_k(\mathbf{C}) \in \mathcal{S}_0 \cup \mathcal{S}_1$. Moreover, $O(\mathbf{C}) = O(\tau_k(\mathbf{C}))$ since $\text{act}(\mathbf{C}) = \text{act}(\tau_k(\mathbf{C}))$. Therefore, $\mathbf{C} \in \mathcal{S}_b$ iff $\tau_k(\mathbf{C}) \in \mathcal{S}_b$. ◀

▶ **Lemma 15.** *Let* $\mathbf{C}$ *and* $\mathbf{C}'$ *be configurations such that* $\mathbf{C} \xrightarrow{*} \mathbf{C}'$. *For every* $d \in \text{supp}(\mathbf{C})$ *and* $d' \in \mathbb{D} \setminus \text{supp}(\mathbf{C})$, *it is the case that* $\mathbf{C} + (\mathbf{C}(d))_{d'} \xrightarrow{*} \mathbf{C}' + (\mathbf{C}'(d))_{d'}$.

Proof. Let d, d' be as described and let $\mathbf{C} = \mathbf{C}_0 \xrightarrow{t_1} \mathbf{C}_1 \xrightarrow{t_2} \cdots \xrightarrow{t_i} \mathbf{C}_i = \mathbf{C}'$. We prove the claim by induction on i. As the claim trivially holds for $i = 0$, let us consider the case where $n > i$. We make a case distinction.

Case $\mathbf{C}_i(d) = \mathbf{C}_{i-1}(d)$. We have

$$\begin{aligned} \mathbf{C}_0 + (\mathbf{C}_0(d))_{d'} &\xrightarrow{*} \mathbf{C}_{i-1} + (\mathbf{C}_{i-1}(d))_{d'} && \text{(by induction hypothesis)} \\ &= \mathbf{C}_{i-1} + (\mathbf{C}_i(d))_{d'} \\ &\rightarrow \mathbf{C}_i + (\mathbf{C}_i(d))_{d'} && (\text{by } \mathbf{C}_{i-1} \rightarrow \mathbf{C}_i). \end{aligned}$$

Case $\mathbf{C}_i(d) \neq \mathbf{C}_{i-1}(d)$. We show that $(\mathbf{C}_{i-1}(d))_{d'} \xrightarrow{t} (\mathbf{C}_i(d))_{d'}$. By assumption, there is a transition $t = ((p, q), \sim, (p, q')) \in \delta$ such that $\mathbf{C}_{i-1} \xrightarrow{t} \mathbf{C}_i$, $q \in \text{act}(\mathbf{C}_{i-1}(d))$ and $q \neq q'$. The latter implies $(\mathbf{C}_{i-1}(d))_{d'} \geq \mathbf{q}_{\mathbf{d}'}$. Let us make a case distinction on $\sim$.

- Assume that $\sim$ is "$=$". If $p = q$, we have $\mathbf{C}_{i-1}(d, q) \geq 2$. Thus, we have $(\mathbf{C}_{i-1}(d))_{d'} \xrightarrow{t} (\mathbf{C}_i(d))_{d'}$. Otherwise, we have $\mathbf{C}_{i-1}(d, p) \geq 1$ and $\mathbf{C}_{i-1}(d, q) \geq 1$. This also implies that $(\mathbf{C}_{i-1}(d))_{d'} \xrightarrow{t} (\mathbf{C}_i(d))_{d'}$ holds.
- Assume that $\sim$ is "$\neq$". Some color $d'' \in \text{supp}(\mathbf{C}_0)$ satisfies $p \in \text{act}(\mathbf{C}_{i-1}(d''))$ and hence $p \in \text{act}(\mathbf{C}_i(d''))$. Observe that $d'' \in \text{supp}(\mathbf{C}_0)$ implies that $d'' \neq d'$. Therefore, we have $(\mathbf{C}_{i-1}(d))_{d'} \xrightarrow{t} (\mathbf{C}_i(d))_{d'}$. ◀

▶ **Lemma 16.** *Let* ψ *be a predicate computed by a population protocol with unordered data. Let* $\boldsymbol{f}$ *be a form with* $\text{act}(\boldsymbol{f}) \subseteq I$. *There exists* $h(\boldsymbol{f}) \in \mathbb{N}$ *such that, for all initial configuration* $\mathbf{C}_0$ *and* $d \in \mathbb{D} \setminus \text{supp}(\mathbf{C}_0)$ *with* $\#_{\boldsymbol{f}}(\mathbf{C}_0) \geq h(\boldsymbol{f})$, *it is the case that* $\psi(\mathbf{C}_0 + \boldsymbol{f}_d) = \psi(\mathbf{C}_0)$.

Proof. Let $\boldsymbol{f}$ be a form with $\text{act}(\boldsymbol{f}) \subseteq I$, let $\{\mathbf{C}_1, \ldots, \mathbf{C}_n\}$ be a basis of $\mathcal{U}$ given by Lemma A2, and let $m := \max(|\text{supp}(\mathbf{C}_1)|, |\text{supp}(\mathbf{C}_2)|, \ldots, |\text{supp}(\mathbf{C}_n)|)$. We define $h(\boldsymbol{f})$ as follows:

$$h(\boldsymbol{f}) := (m - 1) \cdot |Q|^{\sum_{q \in Q} \boldsymbol{f}(q)} + 1.$$

Let $\mathbf{C}_0$ be an initial configuration such that $\#_{\boldsymbol{f}}(\mathbf{C}_0) \geq h(\boldsymbol{f})$. Let $b := \psi(\mathbf{C}_0)$. Let $\mathbf{C} \in \mathcal{S}_b$ be such that $\mathbf{C}_0 \xrightarrow{*} \mathbf{C}$. By the pigeonhole principle, there exists $d \in \text{supp}(\mathbf{C}_0)$ such that

- $\mathbf{C}_0(d) = \boldsymbol{f}$, and
- $\mathbf{C}(d) = \boldsymbol{f}'$ with $\#_{\boldsymbol{f}'}(\mathbf{C}) \geq m$.

Let us now inject the form $\boldsymbol{f}$ with a new datum $d' \in \mathbb{D} \setminus \text{supp}(\mathbf{C}_0)$. By Lemma 15, we have

$$\mathbf{C}_0 + \boldsymbol{f}_{d'} = \mathbf{C}_0 + (\mathbf{C}_0(d))_{d'} \xrightarrow{*} \mathbf{C} + (\mathbf{C}(d))_{d'} = \mathbf{C} + \boldsymbol{f}'_{d'}.$$

Let $\mathbf{C}' := \mathbf{C} + \boldsymbol{f}'_{d'}$. As $\mathbf{C}(d) = \boldsymbol{f}'$ and $\mathbf{C} \in \mathcal{S}_b$, we have $O(\mathbf{C}) = O(\boldsymbol{f}'_{d'}) = b$ and so $O(\mathbf{C}') = b$.

Let us now show that $\mathbf{C}' \in \mathcal{S}_b$. For the sake of contradiction, suppose that $\mathbf{C}' \in \mathcal{U}$. There exists $i \in [1..n]$ such that $\mathbf{C}_i \sqsubseteq \mathbf{C}'$. By the latter, there exists an injection $\rho \colon \mathbb{D} \to \mathbb{D}$ such that $\mathbf{C}_i(x) \leq \mathbf{C}'(\rho(x))$ for every $x \in \mathbb{D}$. Since

$$\#_{\boldsymbol{f}'}(\mathbf{C}') > \#_{\boldsymbol{f}'}(\mathbf{C}) \geq m \geq |\text{supp}(\mathbf{C}_i)|,$$

$d^* \in \mathbb{D}$ such that $\mathbf{C}'(d^*) = \boldsymbol{f}'$, and either d^* does not belong to the image of ρ or $\mathbf{C}_i(\rho^{-1}(d^*)) = \mathbf{0}$. Let us show that $\mathbf{C}_i \sqsubseteq \mathbf{C}$, which yields the contradiction $\mathbf{C} \in \mathcal{U}$. We make a case distinction.

Case $\rho^{-1}(d')$ undefined. We trivially have $\mathbf{C}_i(x) \leq \mathbf{C}'(\rho(x)) = \mathbf{C}(\rho(x))$ for all $x \in \mathbb{D}$.

Case $d^* = d'$. Since $\mathbf{C}_i(\rho^{-1}(d')) = \mathbf{C}_i(\rho^{-1}(d^*)) = \mathbf{0} = \mathbf{C}(d') = \mathbf{C}(d^*)$, and as $\mathbf{C}$ and $\mathbf{C}'$ only differ on d', we have $\mathbf{C}_i(x) \leq \mathbf{C}(\rho(x))$ for every $x \in \mathbb{D}$. Thus, $\mathbf{C}_i \sqsubseteq \mathbf{C}$.

Case $d^* \neq d'$. Let $\rho' \colon \mathbb{D} \to \mathbb{D}$ be the injection given by ρ, but with d^* playing the role of d' (and possibly the converse to ensure that ρ' is indeed an injection), *i.e.*

$$\rho'(x) \coloneqq \begin{cases} d' & \text{if } \rho^{-1}(d^*) \text{ is defined and } x = \rho^{-1}(d^*), \\ d^* & \text{if } x = \rho^{-1}(d'), \\ \rho(x) & \text{otherwise.} \end{cases}$$

As $d^* \neq d'$, we have $\mathbf{C}(d^*) = \mathbf{C}'(d^*) = \boldsymbol{f}'$. Thus,

$$\mathbf{C}_i(\rho^{-1}(d')) \leq \mathbf{C}'(d') = \boldsymbol{f}' = \mathbf{C}(d^*) = \mathbf{C}(\rho'(\rho^{-1}(d'))).$$

Furthermore, if $\rho^{-1}(d^*)$ is defined, then we have $\mathbf{C}_i(\rho^{-1}(d^*)) = \mathbf{0} = \mathbf{C}(d') = \mathbf{C}(\rho'(\rho^{-1}(d^*)))$. Thus, since $\mathbf{C}$ and $\mathbf{C}'$ only differ on datum d', we have $\mathbf{C}_i(x) \leq \mathbf{C}(\rho'(x))$ for all $x \in \mathbb{D}$, which implies $\mathbf{C}_i \sqsubseteq \mathbf{C}$. ◀

C Missing proofs of Section 4.3

Let us first recall all of the transitions in a single table:

Leader and controller election

rule	*state precondition*	*color pre.*	*state update*
(1)	$\mathrm{lead}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{role}(q') = -\lvert\mathrm{role}(q)\rvert$ $\neg\mathrm{lead}(q')$
(2)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = 1$	*none*	$\mathrm{ctrl}(q') = -1$

Element count by datum

rule	*state precondition*	*color pre.*	*state update*
(3)	$\mathrm{elem}(p) = \mathrm{elem}(q)$ $\mathrm{val}(q) = \mathrm{val}(p) < r$	$d_1 = d_2$	$\mathrm{val}(q') = \mathrm{val}(q) + 1$
(4)*	$\mathrm{count}_{\mathrm{elem}(p)}(q) < \mathrm{val}(p)$ $\mathrm{lead}(q)$	$d_1 = d_2$	$\mathrm{count}_{\mathrm{elem}(p)}(q') = \mathrm{val}(p)$ `if` $(\mathrm{role}(q) > 0 \wedge$ $\mathrm{val}(p) \notin T(\mathrm{role}(q), \mathrm{elem}(p)))$: $\mathrm{role}(q') = -\mathrm{role}(q)$

Role distribution and task tracking

rule	*state precondition*	*color pre.*	*state update*
(5)	$\mathrm{lead}(q)$ $\mathrm{role}(q) = 0$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i,j)$	*none*	$\mathrm{role}(q') = i$
(6)*	$\mathrm{ctrl}(p)$ $\mathrm{lead}(q)$ $\mathrm{role}(q) = i > 0$ $\bigvee_{i' \in [1..n] \setminus \{i\}} \big(\neg\mathrm{task}_{i'}(p) \wedge$ $\bigwedge_{j \in [1..m]} \mathrm{count}_j(q) \in T(i', j)\big)$	*none*	$\mathrm{role}(q') = -i$
(7)*	$\mathrm{role}(p) \neq 0$ $\mathrm{ctrl}(q) = 1$	*none*	$\mathrm{task}_{\lvert\mathrm{role}(p)\rvert}(q') = (\mathrm{role}(p) > 0)$
(8)*	$\mathrm{ctrl}(p) = 1$ $\mathrm{role}(q) < 0$ $\neg\mathrm{task}_{\lvert\mathrm{role}(q)\rvert}(p)$	*none*	$\mathrm{role}(q') = 0$
(9)	$\mathrm{ctrl}(p) = -1$ $\mathrm{ctrl}(q) = 1$	*none*	$\bigwedge_{i \in [1..m]} \neg\mathrm{task}_i(q')$
(10)	$\mathrm{ctrl}(p) = 1$ $\mathrm{ctrl}(q) = -1$ $\bigwedge_{i \in [1..m]} \neg\mathrm{task}_i(p)$	*none*	$\mathrm{ctrl}(q') = 0$

Output propagation

rule	*state precondition*	*color pre.*	*state update*
(11)*	$\mathrm{ctrl}(p)$	*none*	$\mathrm{out}(q') = \bigwedge_{i=1}^{n} \mathrm{task}_i(p)$

Note that for the sake of brevity, for the remainder of the section, we use the notation $\mathbf{C} \xrightarrow{(n)} \mathbf{C}'$ to denote that $\mathbf{C} \xrightarrow{t} \mathbf{C}'$ using a transition t arising from rule (n).

▶ **Lemma 20.** *There exists $\tau \in \mathbb{N}$ such that $|\mathbf{C}_\tau|_{Q_C} = |\mathbf{C}_{\tau+1}|_{Q_C} = \cdots = 1$, and $|\mathbf{C}_\tau(d)|_{Q_L} = |\mathbf{C}_{\tau+1}(d)|_{Q_L} = \cdots = 1$ for every $d \in \mathbb{D}$.*

Proof. We prove the two parts of the claim independently. The validity of the claim follows by taking the maximum among the two thresholds τ.

Threshold for Q_L. Note that $\mathbf{C} \xrightarrow{(1)} \mathbf{C}'$ implies $|\mathbf{C}'|_{Q_L} = |\mathbf{C}|_{Q_L} - 1 > 0$, and $|\mathbf{C}|_{Q_L} = |\mathbf{C}'|_{Q_L}$ otherwise. Therefore, there exist $\tau \in \mathbb{N}$ and $k \geq 1$ such that $|\mathbf{C}_\tau|_{Q_L} = |\mathbf{C}_{\tau+1}|_{Q_L} = \cdots = k$. If $k = 1$, then we are done. Thus, for the sake of contradiction, assume that $k \geq 2$. Rule (1) is enabled in each of $\mathbf{C}_\tau, \mathbf{C}_{\tau+1}, \ldots$. By fairness, this implies that for some $\tau' \geq \tau$, we have $|\mathbf{C}_{\tau'}|_{Q_L} > |\mathbf{C}_{\tau'+1}|_{Q_L}$, which is a contradiction.

Threshold for Q_C. The proof is the same, but using rule (2). ◀

▶ **Lemma 21.** *There exists $\tau \in \mathbb{N}$ such that, for every $\tau' \geq \tau$, $d \in \operatorname{supp}(\mathbf{C}_{\tau'})$ and $j \in [1..m]$, if $\mathbf{C}_0(d, Q_j) > 0$, then $\mathbf{C}_{\tau'}(d, q) > 0$ holds for some (d, j)-valid state q.*

Proof. Note that rule (3) is the only rule modifying the value of a state; it requires two states with the same value and it increases the value of a single state by 1. Therefore, for any $\mathbf{C}, \mathbf{C}'$ and $d \in \mathbb{D}$ such that $\mathbf{C} \to \mathbf{C}'$ with $\mathbf{C}' = \mathbf{C} - \boldsymbol{q}_d + \boldsymbol{q}'_d$ for some states $q, q' \in Q$, either $\operatorname{val}(q') = \operatorname{val}(q)$, or $\operatorname{val}(q') = \operatorname{val}(q) + 1$ holds. Further observe that no rule changes a state q to a state q' such that $\operatorname{elem}(q') \neq \operatorname{elem}(q)$, *i.e.* $\operatorname{elem}(q)$ is immutable for any q.

Let $d \in \mathbb{D}$ and $j \in [1..m]$ be such that $\mathbf{C}_0(d, Q_j) > 0$. We define

$$Q_{j,k} \coloneqq \{q \in Q : \operatorname{elem}(q) = j \wedge \operatorname{val}(q) = k\} \text{ and } Q_{j,>k} \coloneqq \bigcup_{k'=k+1}^{r} Q_{j,k'}.$$

We claim that, for every $1 \leq k < \min(\mathbf{C}_0(d, Q_j) + 1, r)$, there exists $\tau \in \mathbb{N}$ such that, for every $\tau' \geq \tau$, the following holds:

$$|\mathbf{C}_{\tau'}(d)|_{Q_{j,k}} = 1 \text{ and } |\mathbf{C}_{\tau'}(d)|_{Q_{j,>k}} = \mathbf{C}_0(d, Q_j) - k.$$

Before proving the claim, let us see how it allows to conclude. If $\mathbf{C}_0(d, Q_j) \leq r$, then this confirms the existence of an agent of d in a (d, j)-valid state. Note that if $k = r - 1$, then $Q_{j,>k}$ is in fact $Q_{j,r}$. Therefore, if $\mathbf{C}_0(d, Q_j) > r$, then agents will accumulate in some states of $Q_{j,r}$. In that case, there will be more than one agent of d in a (d, j)-valid state. Let us now prove the claim by induction on k.

Base case ($k = 1$). Since all states $q \in I$ satisfy $\operatorname{val}(q) = 1$, we gather from the previous observations that for $\mathbf{C}_i, \mathbf{C}'$ such that $\mathbf{C}_i \xrightarrow{*} \mathbf{C}'$, $\mathbf{C}_i(d, Q_{j,1}) \geq \mathbf{C}'(d, Q_{j,1})$ holds. Note that the fact that two states with the same value are needed for rule (3) to decrease $\mathbf{C}'(d, Q_{j,1})$ implies that $\mathbf{C}'(d, Q_{j,1}) \geq 1$. Note also that for any $q \in \operatorname{act}(\mathbf{C}_i)$, either $\operatorname{val}(q) = 1$ or $\operatorname{val}(q) > 1$. By the above observations, there exist $\ell \in \mathbb{N}$ and $\kappa \geq 1$ such that $|\mathbf{C}_\ell(d)|_{Q_{j,1}} = |\mathbf{C}_{\ell+1}(d)|_{Q_{j,1}} = \cdots = \kappa$. If $\kappa = 1$, then $|\mathbf{C}_\ell(d)|_{Q_{j,>1}} = \mathbf{C}_0(d, Q_j) - 1$ and we are done. Otherwise, rule (3) is enabled in each of $\mathbf{C}_\ell, \mathbf{C}_{\ell+1}, \ldots$. By fairness, this implies the existence of $\ell' \geq \ell$ such that

$$|\mathbf{C}_{\ell'}(d)|_{Q_{j,1}} > |\mathbf{C}_{\ell'+1}(d)|_{Q_{j,1}} \text{ and } |\mathbf{C}_{\ell'}(d)|_{Q_{j,>1}} < |\mathbf{C}_{\ell'+1}(d)|_{Q_{j,>1}},$$

which is a contradiction.

Induction step. Assume that the claim is true for k such that $1 < k < r - 1$ and $k \leq \mathbf{C}_0(d, Q_j)$. Let us show that the claim holds for $k+1$. If $k = \mathbf{C}_0(d, Q_j)$, then $k+1 \not< \min(\mathbf{C}_0(d, Q_j)+1, r)$, and hence we are trivially done as the claim does not have to hold for $k+1$. Thus, let us assume that $k < \mathbf{C}_0(d, Q_j)$. By induction hypothesis, there exists $v \in \mathbb{N}$ such that, for every $v' \geq v$, the following holds:

$$|\mathbf{C}_{v'}(d)|_{Q_{j,k}} = 1 \text{ and } |\mathbf{C}_{v'}(d)|_{Q_{j,>k}} = \mathbf{C}_0(d, Q_j) - k.$$

Since values are incremented by 1 (*i.e.* there are no gaps in values), at least one agent is in a state of $Q_{j,k+1}$. Suppose that for some $w \geq v$, there exist states $p, q \in Q_{j,k+1}$ such that $\mathbf{C}_w \geq \boldsymbol{p}_d + \boldsymbol{q}_d$. Then, rule (3) is enabled, implying the existence of a configuration $\mathbf{C}'$ such that $\mathbf{C}_w \xrightarrow{*} \mathbf{C}'$,

$$|\mathbf{C}_w(d)|_{Q_{j,k+1}} > |\mathbf{C}'(d)|_{Q_{j,k+1}} \text{ and } |\mathbf{C}_w(d)|_{Q_{j,>k+1}} < |\mathbf{C}'(d)|_{Q_{j,>k+1}}.$$

By fairness, this implies the existence of some w' with $|\mathbf{C}_{w'}(d)|_{Q_{j,k+1}} = 1$. We are done since

$$\begin{aligned}
|\mathbf{C}_{w'}(d)|_{Q_{j,>k+1}} &= |\mathbf{C}_{w'}(d)|_{Q_{j,>k}} - |\mathbf{C}_{w'}(d)|_{Q_{j,k+1}} && \text{(by definition of } Q_{j,\cdot}) \\
&= |\mathbf{C}_{w'}(d)|_{Q_{j,>k}} - 1 \\
&= (\mathbf{C}_0(d, Q_j) - k) - 1 \\
&= \mathbf{C}_0(d, Q_j) - (k+1).
\end{aligned}$$

◀

▶ **Lemma 22.** *There exists $\tau \geq \alpha$ such that, for every $\tau' \geq \tau$, $d \in \operatorname{supp}(\mathbf{C}_{\tau'})$, $j \in [1..m]$ and $q \in \operatorname{act}(\mathbf{C}_{\tau'}(d)) \cap Q_L$, it is the case that $\operatorname{count}_j(q) = \min(\mathbf{C}_{\tau'}(d, Q_j), r)$.*

Proof. Note that only rule (4) changes a state q to a state q' such that for some $j \in [1..m]$, $\operatorname{count}_j(q') \neq \operatorname{count}_j(q)$. Moreover, using rule (4) leads to $\operatorname{count}_j(q') > \operatorname{count}_j(q)$.

Let $d \in \mathbb{D}$ and $j \in [1..m]$. If $\mathbf{C}_0(d, Q_j) = 0$, then the leader's count for that element is never updated, *i.e.* it never meets a state q of its datum such that $\operatorname{elem}(q) = j$ and therefore its count is always 0 for element j. Otherwise, let τ be the threshold given by Lemma 21, let $v \geq \max(\alpha, \tau)$ and let $q \in \operatorname{act}(\mathbf{C}_v(d))$ be a (d, j)-valid state given by Lemma 21. If there exists $p \in \operatorname{act}(\mathbf{C}_v(d)) \cap Q_L$ such that $\operatorname{count}_j(p) < \operatorname{val}(q)$, then, rule (4) is enabled. This implies the existence of a configuration $\mathbf{C}'$ such that $\mathbf{C}_v \xrightarrow{*} \mathbf{C}'$ and, for any $q' \in \operatorname{act}(\mathbf{C}'(d)) \cap Q_L$, $\operatorname{count}_j(q') = \min(\mathbf{C}_0(d, Q_j), r)$ holds. Therefore, by fairness, there exists v' such that

$$\forall q' \in \operatorname{act}(\mathbf{C}_{v'}(d)) \cap Q_L, j \in [1..m] : \operatorname{count}_j(q') = \min(\mathbf{C}_0(d, Q_j), r).$$

Note that the precondition of rule (4) is no longer met once the leader has the correct count since no other state can have a higher count for j in d. Since this is valid for any datum and any j, there exists a configuration such that for any datum, the datum's leader has the correct counts for all elements. ◀

▶ **Lemma 23.** *For every $\tau \in \mathbb{N}$, $j \in [1..m]$ and $q \in \operatorname{act}(\mathbf{C}_\tau) \cap Q_L$ such that $\operatorname{role}(q) > 0$, it is the case that $\operatorname{count}_j(q) \in T(\operatorname{role}(q), j)$.*

Proof. Initially, each state $q \in \operatorname{act}(\mathbf{C}_0)$ satisfies $\operatorname{role}(q) = 0$. Observe that only rule (5) can assign a strictly positive role to a state. Formally, for a datum $d \in \mathbb{D}$, a role $i \in [1..n]$ and configurations $\mathbf{C}_\tau, \mathbf{C}'$ such that $\mathbf{C}_\tau \xrightarrow{t} \mathbf{C}'$ and $\mathbf{C}' = \mathbf{C}_\tau - \boldsymbol{q}_d + \boldsymbol{q}'_d$ for some $q, q' \in Q$ and transition t arising from rule (5), the leader's state q must satisfy $\operatorname{count}_j(q) \in T(i, j)$ for all $j \in [1..m]$. Thus, a leader must have the correct counts for a role before self-assigning the aforementioned role.

Futher observe that rule (4) is the only rule changing a count for a leader. Formally, for a datum $d \in \mathbb{D}$, a role $i \in [1..n]$ and configurations $\mathbf{C}_\tau, \mathbf{C}'$ such that $\mathbf{C}_\tau \xrightarrow{t} \mathbf{C}'$ and

$\mathbf{C}' = \mathbf{C}_\tau - \boldsymbol{q}_d + \boldsymbol{q}'_d$ for some $q, q' \in Q$ and transition t arising from rule (4), this leader must enter a new state q' such that $\mathrm{role}(q') < 0$ if it no longer satisfies the count condition for the corresponding interval. Therefore, after updating its count, the leader either has a strictly negative assigned role or it still satisfies the role it previously assumed.

Therefore, for any $\tau \in \mathbb{N}$, all leaders with strictly positive roles have the correct counts for their role in $\mathbf{C}_\tau$. ◀

▶ **Lemma A4.** *For every $\tau \in \mathbb{N}$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \setminus Q_L$, it is the case that $\mathrm{role}(q) \leq 0$.*

Proof. Observe that only rule (5) can assign a strictly positive role to a state. Formally, for a datum $d \in \mathbb{D}$ and configurations $\mathbf{C}_\tau, \mathbf{C}'$ such that $\mathbf{C}_\tau \xrightarrow{(5)} \mathbf{C}'$ and $\mathbf{C}' = \mathbf{C}_\tau - \boldsymbol{q}_d + \boldsymbol{q}'_d$ for some $q, q' \in Q$, state q must satisfy $\mathrm{lead}(q)$. Therefore, non-leaders are never directly assigned a strictly positive role.

Further note that only rule (1) removes leadership from a state. Formally, for a datum $d \in \mathbb{D}$ and configurations $\mathbf{C}_\tau, \mathbf{C}'$ such that $\mathbf{C}_\tau \xrightarrow{(1)} \mathbf{C}'$ and $\mathbf{C}' = \mathbf{C}_\tau - \boldsymbol{q}_d + \boldsymbol{q}'_d$ for $q, q' \in Q$, state q' must satisfy $\mathrm{role}(q') < 0$. Hence, non-leaders are assigned a strictly negative role when their leadership is stripped away. Note that leadership removal happens only once.

Therefore, non-leaders can never have a strictly positive assigned role. ◀

▶ **Lemma 25.** *There exists some $\tau \in \mathbb{N}$ such that for every $\tau' \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'}) \setminus Q_L$, it is the case that $\mathrm{role}(q) = 0$.*

Proof. From Lemma A4, we know non-leaders cannot have a strictly positive assigned role. Let $Q_0 \coloneqq \{q \in Q \colon \neg\mathrm{lead}(q) \wedge \mathrm{role}(q) = 0\}$. Note that only rule (8) changes the role of a non-leader, and it changes it to 0. This, along with the fact that leadership removal is definitive, implies that $|\mathbf{C}_j|_{Q_0} \geq |\mathbf{C}_i|_{Q_0}$ holds for every $j > i$.

Assume that for some $v \in \mathbb{N}$, there is some state $q \in \mathrm{act}(\mathbf{C}_v)$ such that $\neg\mathrm{lead}(q)$ holds and $\mathrm{role}(q) < 0$. Let $i \coloneqq |\mathrm{role}(q)|$. Then, rule (7) is enabled in $\mathbf{C}_v$. This means that there exists a configuration $\mathbf{C}'$ and a state $q \in \mathrm{act}(\mathbf{C}') \cap Q_C$ such that $\mathbf{C}_v \xrightarrow{(7)} \mathbf{C}'$ and $\mathrm{task}_i(q) = \mathit{false}$. Rule (8) is enabled in $\mathbf{C}'$. This implies the existence of a configuration $\mathbf{C}''$ such that $\mathbf{C}' \xrightarrow{*} \mathbf{C}''$ and $|\mathbf{C}_v|_{Q_0} < |\mathbf{C}''|_{Q_0}$.

By fairness, some configuration $\mathbf{C}_{v'}$ also satisfies this condition. Thus, for some $\tau \geq v'$, we have $|\mathbf{C}_\tau|_{Q_0} = |\mathbf{C}_\tau|_{Q \setminus Q_L}$. Since it has been established that no rule changes a non-leader's role to a non-0 value, this implies that for any $\tau' \geq \tau$, non-leaders are in states $q \in Q_0$. ◀

▶ **Lemma 26.** *There exists $\tau \geq \alpha$ such that for every $\tau' \geq \tau$ and $q \in \mathrm{act}(\mathbf{C}_{\tau'})$, it is the case that $\mathrm{ctrl}(q) \in \{0, 1\}$.*

Proof. Observe that rule (2) is the only rule assigning a strictly negative controller value. We know that from $\mathbf{C}_\alpha$, there exists a unique controller. Let $Q_{-1} \coloneqq \{q \in Q : \mathrm{ctrl}(q) = -1\}$. Note that for any $j > i \geq \alpha$, since rule (2) is disabled, we have $|\mathbf{C}_j|_{Q_{-1}} \leq |\mathbf{C}_i|_{Q_{-1}}$.

Assume that for some $v \geq \alpha$, there exists a state $q \in \mathrm{act}(\mathbf{C}_v)$ satisfying $\mathrm{ctrl}(q) = -1$. Then, we know that rule (9) is enabled in $\mathbf{C}_v$, implying the existence of a configuration $\mathbf{C}'$ such that $\mathbf{C}_v \xrightarrow{(9)} \mathbf{C}'$ and

$$\bigwedge_{i \in [1..m]} \neg\mathrm{task}_i(q) \qquad\qquad \text{for some } q \in \mathrm{act}(\mathbf{C}') \cap Q_C$$

This means that for $\mathbf{C}'$, rule (10) is enabled. Therefore, there exists a configuration $\mathbf{C}''$ such that $\mathbf{C}' \xrightarrow{(10)} \mathbf{C}''$ and $|\mathbf{C}_v|_{Q_{-1}} > |\mathbf{C}''|_{Q_{-1}}$. Hence, by fairness, there exists $\tau \geq v$ such that $|\mathbf{C}_\tau|_{Q_{-1}} = 0$. Since no rule can change a state q with $\mathrm{ctrl}(q) = 0$ to a state q' with $\mathrm{ctrl}(q) \neq 0$, this holds for any $\tau' \geq \tau$. ◀

▶ **Lemma 27.** *For every $\tau \geq \gamma$, $i \in [1..n]$ and $q \in \mathrm{act}(\mathbf{C}_\tau) \cap Q_C$ such that $\mathrm{task}_i(q)$ holds, there exists $q' \in \mathrm{act}(\mathbf{C}_\tau)$ such that $|\mathrm{role}(q')| = i$.*

Proof. Note that only rule (7) can assign *true* to some task i for the controller. Let $R_i \coloneqq \{q \in Q : |\mathrm{role}(q)| = i\}$.

Let $i \in [1..n]$ and let $q \in Q_C$ satisfy $\neg\mathrm{task}_i(q)$ and $q' \in Q_C$ satisfy $\mathrm{task}_i(q')$. Let some $v \geq \gamma$ be such that $\mathbf{C}_v \xrightarrow{(7)} \mathbf{C}_{v+1}$, $\mathbf{C}_{v+1} = \mathbf{C}_v - \boldsymbol{q}_d + \boldsymbol{q}'_d$. Rule (7) implies the existence of some state $p \in \mathrm{act}(\mathbf{C}_{v+1}) \cap R_i$. Note that for $\mathbf{C}_\gamma$, a unique controller has been permanently elected.

Let $w \geq v + 1$. For $\mathbf{C}_w \to \mathbf{C}_{w+1}$ to satisfy $|\mathbf{C}_w|_{R_i}| > |\mathbf{C}_{w+1}|_{R_i}|$, we must have $\mathbf{C}_w \xrightarrow{(8)} \mathbf{C}_{w+1}$. For that to be the case, there has to be some state $q'' \in \mathrm{act}(\mathbf{C}_w) \cap Q_C$ such that $\neg\mathrm{task}_i(q'')$ holds. Thus, for any $\mathbf{C}_w$ for which there exists a state $q^* \in \mathrm{act}(\mathbf{C}_w) \cap Q_C$ satisfying $\mathrm{task}_i(q^*)$, there exists a state $p \in \mathrm{act}(\mathbf{C}_w) \cap R_i$. ◀

▶ **Lemma 28.** *It is the case that $\psi(\mathbf{C}_0)$ holds iff there exists some $\tau \geq \gamma$ such that for every $\tau' \geq \tau$, there exists $q \in \mathrm{act}(\mathbf{C}_{\tau'}) \cap Q_C$ such that $\bigwedge_{i\in[1..n]} \mathrm{task}_i(q)$ holds.*

Proof of ⇒). By definition, $\mathbf{C}_\beta$ has a unique controller elected and a unique leader for each datum, carrying the correct counts for its datum's initial states. Since $\psi(\mathbf{C}_0)$ is *true*, a subset of these leaders can fulfill the n roles simultaneously. While the roles might not be correctly distributed at $\mathbf{C}_\beta$, we show that for some configuration $\mathbf{C}$ reachable from $\mathbf{C}_\beta$, the controller's tasks will all be *true*. Let us consider the following lemmas and their implications:
- from Lemma 23, a leader with an assigned role always *thinks* it can currently fill this role,
- from Lemma 25, all non-leaders eventually have their roles set to 0, leaving only the elected leaders with (possibly) assigned roles,
- from Lemma 26 and Lemma 27, eventually, the unique controller can only have a task i set to *true* if a leader has its role set to $\pm i$.

Therefore, for configuration $\mathbf{C}_\gamma$, only leaders have a non-zero role. These leaders also correctly *think* they can fulfill the roles to which they are assigned. Finally, there is a unique controller and no agent is in a negative controller state.

We now define a *valid assignment* A as an injective function $A\colon [1..n] \to \mathbb{D}$ such that if $A(i) = d$, then d matches role i. For a configuration $\mathbf{C}$ such that $\mathbf{C}_0 \xrightarrow{*} \mathbf{C}$ and a valid assignment A, let $s_A(\mathbf{C})$ be the set of correctly assigned colors in $\mathbf{C}$ with respect to A, *i.e.*

$$s_A(\mathbf{C}) \coloneqq \{d \in \mathbb{D} \colon (\exists i \in [1..n] : A(i) = d) \land (\exists q \in Q_L : \mathbf{C}(d, q) > 0 \land \mathrm{role}(q) = i)\}.$$

Moreover, we say that a configuration $\mathbf{C}$ is *fixed* if:
- $\mathbf{C}_\gamma \xrightarrow{*} \mathbf{C}$,
- there exists $q \in \mathrm{act}(\mathbf{C}) \cap Q_C$ such that $\bigwedge_{i=1}^n \mathrm{task}_i(q)$, and
- $\mathrm{role}(q) \geq 0$ for every $q \in \mathrm{act}(\mathbf{C})$.

Note that a strictly negative role can only be assigned by rule (1) (leader election), rule (4) (count change) and rule (6) (self-reassignment). Observe that rules (1), (4) and (6) are disabled for a fixed configuration. We also know that the controller can only change one of its tasks to *false* if it observes a negative role through rule (7) or a negative controller through rule (9). Since a fixed configuration does not have a strictly negative role, nor can it generate one, rule (7) cannot change any of the controller's tasks to *false*. Furthermore, rule (9) is disabled for a fixed configuration. Hence, a fixed configuration can only reach fixed configurations.

Now, let A be a valid assignment. For any $v \geq \gamma$, we observe two cases.

Case: $\exists A' : |s_{A'}(\mathbf{C}_v)| = n$. There exists a fixed configuration $\mathbf{C}'$, reachable from $\mathbf{C}_v$.

Case: $\forall A' : |s_{A'}(\mathbf{C}_v)| < n$. This implies that some role i is not taken by any leader. If $A(i)$'s leader has no assigned role, then by rule (5) there exists a configuration $\mathbf{C}'$, reachable from $\mathbf{C}_v$, such that $|s_A(\mathbf{C}_v)| < |s_A(\mathbf{C}')|$ and $\text{role}(q) = i$ for some $q \in \text{act}(\mathbf{C}'(A(i))) \cap Q_L$.

Otherwise, we observe two cases:

- The controller has its task i set to *false*, allowing self-reassignment of $A(i)$'s leader through rule (6).
- The controller has its task i set to *true*. Since role i is not taken by any leader, by Lemma 27, for the controller to have its task i set to *true*, some $q \in \text{act}(\mathbf{C}_v) \cap Q_L$ satisfies $\text{role}(q) = -i$. Thus, by rule (7), there exists a configuration $\mathbf{C}'$ reachable from $\mathbf{C}_v$ such that $\neg\text{task}_i(q)$ for some $q \in \text{act}(\mathbf{C}') \cap Q_C$. The latter allows self-reassignment of $A(i)$'s leader through rule (6).

Consequently, there also exists a configuration $\mathbf{C}''$, reachable from $\mathbf{C}'$, such that $|s_A(\mathbf{C}_v)| < |s_A(\mathbf{C}'')|$ and $\text{role}(q) = i$ for some $q \in \text{act}(\mathbf{C}''(A(i))) \cap Q_L$.

By fairness, since there exists a finite number of configurations and a finite number of valid assignments for $\text{supp}(\mathbf{C}_0)$, either for some v, $|s_A(\mathbf{C}_v)| = n$ holds and, by fairness, $\mathbf{C}_v$ leads to a fixed configuration $\mathbf{C}_\tau$; or for some v and some valid assignment $A' \neq A$, $|s_{A'}(\mathbf{C}_v)| = n$ holds, and again, by fairness, this leads to a fixed configuration $\mathbf{C}_\tau$. Since we know that fixed configurations can only reach fixed configurations and that they also guarantee the controller has its task list completely assigned, this is true for any $\tau' \geq \tau$. ◀

Proof of $\Leftarrow$). We prove the contrapositive. Assume $\psi(\mathbf{C}_0) = \textit{false}$. Let $W(\mathbf{C})$ be the set of data $d \in \text{supp}(\mathbf{C}_0)$ such that for some $q \in \text{act}(\mathbf{C}(d)) \cap Q_L$,

- $|\text{role}(q)| = i$ holds for some $i \in [1..n]$, and
- $\mathbf{C}(d, Q_j) \notin T(i, j)$ holds for some $j \in [1..m]$.

Informally, $W(\mathbf{C})$ is the set of data in $\mathbf{C}$ with a leader having a wrongly assigned role (positive or negative), *i.e.*, a role the data cannot match. Since $|\mathbf{C}_\beta|_{Q_L} = |\text{supp}(\mathbf{C}_0)|$ and the number of leaders never increases, for any $\mathbf{C}'$ such that $\mathbf{C}_\beta \xrightarrow{*} \mathbf{C}'$, we have $W(\mathbf{C}') \leq |\text{supp}(\mathbf{C}_0)|$. Furthermore, by Lemma 22, the counts are stable and correct for leaders past $\mathbf{C}_\beta$. The latter implies that for any $t \geq \beta$ and datum d, if a role i is assigned by rule (5) to d's leader, then d matches role i. Therefore, for any $i, j \geq \beta$ with $i \leq j$, we have $W(\mathbf{C}_j) \leq W(\mathbf{C}_i)$.

Assume that there exists some $\pi \geq \beta$ such that $W(\mathbf{C}_\pi) > 0$. Then, for some $d \in \text{supp}(\mathbf{C}_0)$, there exists a state $q \in \text{act}(\mathbf{C}_\pi(d)) \cap Q_L$ that satisfies $|\text{role}(q)| = i$, where i is a role d does not match. Since $\pi \geq \beta$, by Lemma 22, state q must satisfy $\bigwedge_{j \in [1..m]} \text{count}_j(q) = \min(\mathbf{C}_0(d, Q_j), r)$, *i.e.*, it must have the right counts for each elements for d.

Thus, by Lemma 23, $\text{role}(q)$ must be strictly negative. This means that rule (7) is enabled in $\mathbf{C}_\pi$. So, there exist $\mathbf{C}'$ and $q \in \text{act}(\mathbf{C}') \cap Q_C$ such that $\mathbf{C}_\pi \xrightarrow{*} \mathbf{C}'$ and $\neg\text{task}_i(q)$ holds. Note that rule (8) is now enabled in $\mathbf{C}'$. Therefore, there exists a configuration $\mathbf{C}''$ such that $\mathbf{C}' \xrightarrow{(8)} \mathbf{C}''$ and $W(\mathbf{C}'') < W(\mathbf{C}_\pi)$. Thus, by fairness, there exists $u \geq \pi$ such that $W(\mathbf{C}_u) = 0$.

From Lemma 27, for any $v \geq \beta$, for all $q \in \text{act}(\mathbf{C}_v) \cap Q_C$, if $\text{task}_i(q)$ holds, then there exists some state $q' \in \text{act}(\mathbf{C}_v) \cap Q_L$ such that $\text{role}(q') = \pm i$. Recall from Lemma 25 that any non-leader eventually has a role of 0. Furthermore, observe that

- for $\mathbf{C}_u$ described above, for any $d \in \mathbb{D}$, if d's leader has an assigned role i, then d matches role i; and
- for every $w \geq \max(u, \gamma)$, since $\psi(\mathbf{C}_0)$ is *false*, there is no valid assignment A such that $|s_A(\mathbf{C}_w)| = n$.

Thus, no $\mathbf{C}_w$ satisfies the following for all $i \in [1..n]$:

$$\exists d \in \text{supp}(\mathbf{C}_0), q \in \text{act}(\mathbf{C}_w(d)) \cap Q_L : |\text{role}(q)| = i.$$

Therefore, there exists $\tau \in \mathbb{N}$ such that, for any $\mathbf{C}'$ with $\mathbf{C}_\tau \xrightarrow{*} \mathbf{C}'$, there exists $i \in [1..n]$ such that, for every $q \in \mathrm{act}(\mathbf{C}') \cap Q_C$, it is the case that $\neg\mathrm{task}_i(q)$ holds. ◀

▶ **Corollary 29.** *There exists $\tau \in \mathbb{N}$ such that $O(\mathbf{C}_\tau) = O(\mathbf{C}_{\tau+1}) = \cdots = \psi(\mathbf{C}_0)$.*

Proof. *Case: $\psi(\mathbf{C}_0) = true$.* Let $Q_{true} \coloneqq \{q \in Q \colon \mathrm{out}(q)\}$. Note that only one controller exists for any configuration reachable from $\mathbf{C}_\alpha$. Recall the notion of fixed configurations from the proof of Lemma 28. For a fixed configuration $\mathbf{C}$, if there exists a state $q \in \mathrm{act}(\mathbf{C})$ such that $\neg\mathrm{out}(q)$ holds, since the controller has its task list completely assigned, then there exists a configuration $\mathbf{C}'$ reachable from $\mathbf{C}$ such that $|\mathbf{C}|_{Q_{true}} < |\mathbf{C}'|_{Q_{true}}$. Note that the controller never propagates the *false* output in a fixed configuration; it follows that $|\mathbf{C}_v|_{Q_{true}} \leq |\mathbf{C}^*|_{Q_{true}}$ for any $\mathbf{C}^*$ reachable from $\mathbf{C}_v$. Invoking fairness, this implies that there also is a configuration $\mathbf{C}''$ reachable from $\mathbf{C}'$ such that $|\mathbf{C}''|_{Q_{true}} = |\mathbf{C}''|$. From the proof of Lemma 28, we also know that if $\psi(\mathbf{C}_0) = true$, then a fixed configuration is eventually reached. Therefore, some configuration $\mathbf{C}_\tau$ such that $O(\mathbf{C}_\tau) = true$ is reached; it remains *true* for any configuration $\mathbf{C}_{\tau'}$ such that $\mathbf{C}_\tau \xrightarrow{*} \mathbf{C}_{\tau'}$.

Case: $\psi(\mathbf{C}_0) = false$. From Lemma 28, we know that for some $u \in \mathbb{N}$ and for any $v \geq u$, at least one task is unassigned for the controller in $\mathbf{C}_v$. Assume that there exists some $v \geq u$ such that some $q \in \mathrm{act}(\mathbf{C}_v)$ satisfies $\mathrm{out}(q)$. Then, since the controller has at least one task that is unassigned, rule (11) is enabled (with the controller propagating the *false* output) and there exists a configuration $\mathbf{C}'$ reachable from $\mathbf{C}_v$ such that $|\mathbf{C}_v|_{Q_{true}} > |\mathbf{C}'|_{Q_{true}}$. Note that the controller never propagates the *true* output for $\mathbf{C}_v$; it follows that $|\mathbf{C}_v|_{Q_{true}} \geq |\mathbf{C}^*|_{Q_{true}}$ for any $\mathbf{C}^*$ reachable from $\mathbf{C}_v$. Thus, by fairness, this implies that there exists a configuration $\mathbf{C}''$ reachable from $\mathbf{C}'$ such that $|\mathbf{C}''|_{Q_{true}} = 0$. So, some configuration $\mathbf{C}_\tau$ such that $O(\mathbf{C}_\tau) = false$ is reached. Any configuration $\mathbf{C}_{\tau'}$ reachable from $\mathbf{C}_\tau$ also satisfies $O(\mathbf{C}_\tau) = false$. ◀