

Succinct Population Protocols for Presburger Arithmetic

Michael Blondin 

Département d'informatique, Université de Sherbrooke, Sherbrooke, Canada
michael.blondin@usherbrooke.ca

Javier Esparza 

Fakultät für Informatik, Technische Universität München, Garching bei München, Germany
esparza@in.tum.de

Blaise Genest 

Univ Rennes, CNRS, IRISA, France
blaise.genest@irisa.fr

Martin Helfrich 

Fakultät für Informatik, Technische Universität München, Garching bei München, Germany
helfrich@in.tum.de

Stefan Jaax 

Fakultät für Informatik, Technische Universität München, Garching bei München, Germany
jaax@in.tum.de

Abstract

In [5], Angluin *et al.* proved that population protocols compute exactly the predicates definable in Presburger arithmetic (PA), the first-order theory of addition. As part of this result, they presented a procedure that translates any formula φ of quantifier-free PA with remainder predicates (which has the same expressive power as full PA) into a population protocol with $2^{\mathcal{O}(\text{poly}(|\varphi|))}$ states that computes φ . More precisely, the number of states of the protocol is exponential in both the bit length of the largest coefficient in the formula, and the number of nodes of its syntax tree.

In this paper, we prove that every formula φ of quantifier-free PA with remainder predicates is computable by a leaderless population protocol with $\mathcal{O}(\text{poly}(|\varphi|))$ states. Our result shows that, contrary to the case of time complexity, where protocols with leaders can be exponentially faster than leaderless protocols [1], protocols with and without leaders have both polynomial state complexity.

Our proof is based on several new constructions, which may be of independent interest. Given a formula φ of quantifier-free PA with remainder predicates, a first construction produces a succinct protocol (with $\mathcal{O}(|\varphi|^3)$ leaders) that computes φ ; this completes the work initiated in [8], where we constructed such protocols for a fragment of PA. For large enough inputs, we can get rid of these leaders. If the input is not large enough, then it is small, and we design another construction producing a succinct protocol with *one* leader that computes φ . Our last construction gets rid of this leader for small inputs.

2012 ACM Subject Classification Theory of computation \rightarrow Distributed computing models; Theory of computation \rightarrow Automata over infinite objects; Theory of computation \rightarrow Logic and verification

Keywords and phrases Population protocols, Presburger arithmetic, state complexity

Funding *Javier Esparza, Martin Helfrich, Stefan Jaax:* Supported by an ERC Advanced Grant (787367: PaVeS)

Michael Blondin: Supported by a Quebec–Bavaria project funded by the Fonds de recherche du Québec (FRQ), 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)

1 Introduction

Population protocols [3, 4] are a model of distributed computation by indistinguishable, mobile finite-state agents, intensely investigated in recent years (see e.g. [2, 10]). Initially introduced to model networks of passively mobile sensors, they have also been applied to the analysis of chemical reactions under the name of chemical reaction networks (see e.g. [16]).

In a population protocol, a collection of agents, called a *population*, randomly interact in pairs to decide whether their initial configuration satisfies a given property, e.g. whether there are initially more agents in some state A than in some state B . Since agents are indistinguishable and finite-state, their configuration at any time moment is completely characterized by the mapping that assigns to each state the number of agents that currently populate it. A protocol is said to *compute a predicate* if for every initial configuration where the predicate holds, the agents eventually reach consensus 1, and they eventually reach consensus 0 otherwise.

In a seminal paper, Angluin *et al.* proved that population protocols compute exactly the predicates definable in Presburger arithmetic (PA) [5]. As part of the result, for every Presburger predicate Angluin *et al.* construct a (leaderless) protocol that computes it. The construction makes use of the fact that PA has a quantifier elimination procedure (see e.g. [13]); every Presburger formula φ can be transformed into an equivalent boolean combination of *threshold predicates* of the form $\alpha \cdot x > \beta$ and *remainder predicates* of the form $\alpha \cdot x \equiv \beta \pmod{m}$, where α is an integer vector, and β, m are integers. Slightly abusing language¹, we call the set of these boolean combinations *quantifier-free Presburger arithmetic* (QFPA). Using that PA and QFPA have the same expressive power, Angluin *et al.* first construct protocols for all threshold and remainder predicates, and then show that the predicates computed by protocols are closed under negation and conjunction.

The construction of [5] is simple and elegant, but it produces large protocols. Given a formula φ of QFPA, let n be the number of bits of the largest coefficient of φ in absolute value, and let m be the number of atomic formulas of φ , respectively. The number of states of the protocols of [5] grows exponentially in both n and m . In terms of $|\varphi|$ (defined as the sum of the number of variables, n , and m) they have $\mathcal{O}(2^{\text{poly}(|\varphi|)})$ states. This raises the question of whether *succinct protocols*, i.e., protocols with $\mathcal{O}(\text{poly}(|\varphi|))$ states, exist for every formula φ of QFPA. In this paper we give an affirmative answer by proving that every formula of QFPA has a succinct and leaderless protocol.

Succinct protocols are the state-complexity counterpart of *fast protocols*, defined as protocols running in polylogarithmic parallel time in the size of the population. Angluin *et al.* showed that every predicate has a fast protocol with a leader [6], but Alistarh *et al.*, based on work by Doty and Soloveichik [9], proved in [1] that some predicates need linear parallel time. Our result shows that, unlike for time complexity, there is no exponential gap between the state complexity of leaderless protocols and protocols with leaders.

The proof of our result overcomes a number of obstacles. Designing succinct *leaderless* protocols is particularly hard for inputs with very few input agents, because there are less resources to simulate leaders. So we produce two completely different families of protocols, one for small inputs with $\mathcal{O}(|\varphi|^3)$ agents, and a second for larger inputs with $\Omega(|\varphi|^3)$ agents, and combine them appropriately.

Large inputs. The family for large inputs is based on our previous work ([8], see below). However, in order to obtain leaderless protocols we need a new succinct construction

¹ Remainder predicates cannot be directly expressed in Presburger arithmetic without quantifiers.

for boolean combinations of atomic predicates. This obstacle is overcome by designing new protocols for threshold and remainder predicates that work under *reversible dynamic initialization*. Intuitively, agents are allowed to dynamically “enter” and “leave” the protocol through the initial states (dynamic initialization). Further, every interaction can be undone (reversibility), until a certain condition is met, after which the protocol converges to the correct output for the current input. We expect protocols with reversible dynamic initialization to prove useful in other contexts, since they allow a protocol designer to cope with “wrong” non-deterministic choices.

Small inputs. The family of protocols for small inputs is designed from scratch. We exploit that there are few inputs of small size. So it becomes possible to design one protocol for each possible size of the population, and combine them appropriately. Once the population size is fixed, it is possible to design agents that check if they have interacted with all other agents. This is used to simulate the *concatenation operator* of sequential programs, which allows for boolean combinations and succinct evaluation of linear combinations. Although the resulting protocols use a single leader, they can be made leaderless via a dedicated leader election construction.

Relation to previous work. In [8], we designed succinct protocols with leaders for systems of linear equations. More precisely, we proved the following result: There is a protocol with $\mathcal{O}((m+k)(n+\log m))$ states and $\mathcal{O}(m(n+\log m))$ leaders that computes the predicate $A\mathbf{x} \geq \mathbf{c}$, where $A \in \mathbb{Z}^{m \times k}$ and n is the number of bits of the largest entry in A and \mathbf{c} , in absolute value. If we represent $A\mathbf{x} \geq \mathbf{c}$ as a formula φ of QFPA, this result yields a protocol with $\mathcal{O}(|\varphi|^2)$ states and $\mathcal{O}(|\varphi|^2)$ leaders that computes φ . However, in [8] we were not able to find succinct protocols for formulas with remainder predicates, and all parts of the construction for systems of equations made extensive use of leaders.

Organization. Sections 2 and 3 introduce basic notation and definitions. Section 4 presents the main result. Sections 5 and 6 present the constructions of the protocols for large and small inputs, respectively. Section 7 presents conclusions. For space reasons, several proofs are only sketched. Detailed proofs are presented in the appendices.

2 Preliminaries

Notation. We write \mathbb{Z} to denote the set of integers, \mathbb{N} to denote the set of non negative integers $\{0, 1, \dots\}$, $[n]$ to denote $\{1, 2, \dots, n\}$, and \mathbb{N}^E to denote the set of all multisets over E , *i.e.* unordered vectors with components labeled by E . The *size* of a multiset $\mathbf{v} \in \mathbb{N}^E$ is defined as $|\mathbf{v}| \stackrel{\text{def}}{=} \sum_{e \in E} \mathbf{v}(e)$. The set of all multisets over E with size $s \geq 0$ is $E^{(s)} \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathbb{N}^E : |\mathbf{v}| = s\}$. We sometimes write multisets using set-like notation, *e.g.* $\{a, 2 \cdot b\}$ denotes the multiset \mathbf{v} such that $\mathbf{v}(a) = 1$, $\mathbf{v}(b) = 2$ and $\mathbf{v}(e) = 0$ for every $e \in E \setminus \{a, b\}$. The empty multiset $\{\}$ is instead denoted $\mathbf{0}$ for readability. For every $\mathbf{u}, \mathbf{v} \in \mathbb{N}^E$, we write $\mathbf{u} \geq \mathbf{v}$ if $\mathbf{u}(e) \geq \mathbf{v}(e)$ for every $e \in E$. Moreover, we write $\mathbf{u} + \mathbf{v}$ to denote the multiset $\mathbf{w} \in \mathbb{N}^E$ such that $\mathbf{w}(e) \stackrel{\text{def}}{=} \mathbf{u}(e) + \mathbf{v}(e)$ for every $e \in E$. The multiset $\mathbf{u} \ominus \mathbf{v}$ is defined analogously with $-$ instead of $+$, provided that $\mathbf{u} \geq \mathbf{v}$.

Presburger arithmetic. *Presburger arithmetic* (PA) is the first-order theory of \mathbb{N} with addition, *i.e.* $\text{FO}(\mathbb{N}, +)$. For example, the PA formula $\psi(x, y, z) = \exists x' \exists z' (x = x' + x') \wedge (y = z + z') \wedge \neg(z' = 0)$ states that x is even and that $y > z$. It is well-known that for every formula of PA there is an equivalent formula of quantifier-free Presburger arithmetic (QFPA) [15], the theory with syntax given by the grammar

$$\varphi(\mathbf{v}) ::= \mathbf{a} \cdot \mathbf{v} > b \mid \mathbf{a} \cdot \mathbf{v} \equiv_c b \mid \varphi(\mathbf{v}) \wedge \varphi(\mathbf{v}) \mid \varphi(\mathbf{v}) \vee \varphi(\mathbf{v}) \mid \neg \varphi(\mathbf{v})$$

where $\mathbf{a} \in \mathbb{Z}^X$, $b \in \mathbb{Z}$, $c \in \mathbb{N}_{\geq 2}$, and \equiv_c denotes equality modulo c . For example, the formula $\psi(x, y, z)$ above is equivalent to $(x \equiv_2 0) \wedge (y - z \geq 1)$. Throughout the paper, we refer to any formula of QFPA, or the predicate $\mathbb{N}^X \rightarrow \{0, 1\}$ it denotes, as a *predicate*. Predicates of the form $\mathbf{a} \cdot \mathbf{v} > b$ and $\mathbf{a} \cdot \mathbf{v} \equiv_c b$ are *atomic*, and they are called *threshold* and *remainder* predicates respectively. The *max-norm* $\|\varphi\|$ of a predicate φ is the largest absolute value among all coefficients occurring within φ . The *length* $\text{len}(\varphi)$ of a predicate φ is the number of boolean operators occurring within φ . The *bit length* of a predicate φ , over variables X , is defined as $|\varphi| \stackrel{\text{def}}{=} \text{len}(\varphi) + \log\|\varphi\| + |X|$. We lift these definitions to sets of predicates in the natural way: given a finite set P of predicates, we define its *size* $\text{size}(P)$ as the number of predicates in P , its *length* as $\text{len}(P) \stackrel{\text{def}}{=} \sum_{\varphi \in P} \text{len}(\varphi)$, its *norm* as $\|P\| \stackrel{\text{def}}{=} \max\{\|\varphi\| : \varphi \in P\}$, and its *bit length* as $|P| \stackrel{\text{def}}{=} \text{size}(P) + \text{len}(P) + \log\|P\| + |X|$. Note that $\text{len}(P) = 0$ iff P only contains atomic predicates.

3 Population protocols

A *population protocol* is a tuple $\mathcal{P} = (Q, T, L, X, I, O)$ where

- Q is a finite set whose elements are called *states*;
- $T \subseteq \{(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^Q \times \mathbb{N}^Q : |\mathbf{p}| = |\mathbf{q}|\}$ is a finite set whose elements are called *transitions*;
- $L \in \mathbb{N}^Q$ is the *leader multiset*;
- X is a finite set whose elements are called *input variables*;
- $I: X \rightarrow Q$ is the *input mapping*;
- $O: Q \rightarrow \{0, 1, \perp\}$ is the *output mapping*.

For readability, we often write $t: \mathbf{p} \mapsto \mathbf{q}$ to denote a transition $t = (\mathbf{p}, \mathbf{q})$. Given $\Delta \geq 2$, we say that t is Δ -way if $|\mathbf{p}| \leq \Delta$.

In the standard syntax of population protocols T is a subset of $\mathbb{N}^2 \times \mathbb{N}^2$, and $O: Q \rightarrow \{0, 1\}$. These differences are discussed at the end of this section.

Inputs and configurations. An *input* is a multiset $\mathbf{v} \in \mathbb{N}^X$ such that $|\mathbf{v}| \geq 2$, and a *configuration* is a multiset $C \in \mathbb{N}^Q$ such that $|C| \geq 2$. Intuitively, a configuration represents a population of agents where $C(q)$ denotes the number of agents in state q . The *initial configuration* $C_{\mathbf{v}}$ for input \mathbf{v} is defined as $C_{\mathbf{v}} \stackrel{\text{def}}{=} L + \{\mathbf{v}(x) \cdot I(x) : x \in X\}$.

The *support* and *b-support* of a configuration C are respectively defined as $\llbracket C \rrbracket \stackrel{\text{def}}{=} \{q \in Q : C(q) > 0\}$ and $\llbracket C \rrbracket_b \stackrel{\text{def}}{=} \{q \in \llbracket C \rrbracket : O(q) = b\}$. The *output* of a configuration C is defined as $O(C) \stackrel{\text{def}}{=} b$ if $\llbracket C \rrbracket_b \neq \emptyset$ and $\llbracket C \rrbracket_{\neg b} = \emptyset$ for some $b \in \{0, 1\}$, and $O(C) \stackrel{\text{def}}{=} \perp$ otherwise. Loosely speaking, if $O(q) = \perp$ then agents in state q have no output, and a population has output $b \in \{0, 1\}$ if all agents *with output* have output b .

Executions. A transition $t = (\mathbf{p}, \mathbf{q})$ is *enabled* in a configuration C if $C \geq \mathbf{p}$, and *disabled* otherwise. If t is enabled in C , then it can be *fired* leading to configuration $C' \stackrel{\text{def}}{=} C \ominus \mathbf{p} + \mathbf{q}$, which we denote $C \xrightarrow{t} C'$. For every set of transitions S , we write $C \xrightarrow{S} C'$ if $C \xrightarrow{t} C'$ for some $t \in S$. We denote the reflexive and transitive closure of \xrightarrow{S} by $\xrightarrow{S^*}$. If S is the set of all transitions of the protocol under consideration, then we simply write \rightarrow and $\xrightarrow{*}$.

An execution is a sequence of configurations $\sigma = C_0 C_1 \dots$ such that $C_i \rightarrow C_{i+1}$ for every $i \in \mathbb{N}$. We write σ_i to denote configuration C_i . The *output* of an execution σ is defined as follows. If there exist $i \in \mathbb{N}$ and $b \in \{0, 1\}$ such that $O(\sigma_i) = O(\sigma_{i+1}) = \dots = b$, then $O(\sigma) \stackrel{\text{def}}{=} b$, and otherwise $O(\sigma) \stackrel{\text{def}}{=} \perp$.

Computations. An execution σ is *fair* if for every configuration D the following holds:

$$\text{if } |\{i \in \mathbb{N} : \sigma_i \xrightarrow{*} D\}| \text{ is infinite, then } |\{i \in \mathbb{N} : \sigma_i = D\}| \text{ is infinite.}$$

In other words, fairness ensures that an execution cannot avoid a configuration forever. We say that a population protocol *computes* a predicate $\varphi: \mathbb{N}^X \rightarrow \{0, 1\}$ if for every $\mathbf{v} \in \mathbb{N}^X$ and every fair execution σ starting from $C_{\mathbf{v}}$, it is the case that $O(\sigma) = \varphi(\mathbf{v})$. Two protocols are *equivalent* if they compute the same predicate. It is known that population protocols compute precisely the Presburger-definable predicates [5, 11].

► **Example 1.** Let $\mathcal{P}_n = (Q, T, \mathbf{0}, \{x\}, I, O)$ be the protocol where $Q \stackrel{\text{def}}{=} \{0, 1, 2, 3, \dots, 2^n\}$, $I(x) \stackrel{\text{def}}{=} 1$, $O(a) = 1 \stackrel{\text{def}}{\iff} a = 2^n$, and T contains a transition, for each $a, b \in Q$, of the form $\langle a, b \rangle \mapsto \langle 0, a+b \rangle$ if $a+b < 2^n$, and $\langle a, b \rangle \mapsto \langle 2^n, 2^n \rangle$ if $a+b \geq 2^n$. It is readily seen that \mathcal{P}_n computes $\varphi(x) \stackrel{\text{def}}{=} (x \geq 2^n)$. Intuitively, each agent stores a number, initially 1. When two agents meet, one of them stores the sum of their values and the other one stores 0, with sums capping at 2^n . Once an agent reaches this cap, all agents eventually get converted to 2^n .

Now, consider the protocol $\mathcal{P}'_n = (Q', T', \mathbf{0}, \{x\}, I', O')$, where $Q' \stackrel{\text{def}}{=} \{0, 2^0, 2^1, \dots, 2^n\}$, $I'(x) \stackrel{\text{def}}{=} 2^0$, $O'(a) = 1 \stackrel{\text{def}}{\iff} a = 2^n$, and T' contains a transition for each $0 \leq i < n$ of the form $\langle 2^i, 2^i \rangle \mapsto \langle 0, 2^{i+1} \rangle$, and a transition for each $a \in Q'$ of the form $\langle a, 2^n \rangle \mapsto \langle 2^n, 2^n \rangle$. Using similar arguments as above, it follows that \mathcal{P}'_n also computes φ , but more succinctly: While \mathcal{P}_n has $2^n + 1$ states, \mathcal{P}'_n has only $n + 1$ states.

Types of protocols. A protocol $\mathcal{P} = (Q, T, L, X, I, O)$ is

- *leaderless* if $|L| = 0$, and *has* $|L|$ *leaders* otherwise;
- Δ -*way* if all its transitions are Δ -way;
- *simple* if there exist $\mathbf{f}, \mathbf{t} \in Q$ such that $O(\mathbf{f}) = 0$, $O(\mathbf{t}) = 1$ and $O(q) = \perp$ for every $q \in Q \setminus \{\mathbf{f}, \mathbf{t}\}$ (i.e., the output is determined by the number of agents in \mathbf{f} and \mathbf{t} .)

Protocols with leaders and leaderless protocols compute the same predicates [5]. Every Δ -way protocol can be transformed into an equivalent 2-way protocol with a polynomial increase in the number of transitions [8]. Finally, every protocol can be transformed into an equivalent simple protocol with a polynomial increase in the number of states (see Appendix A).

4 Main result

The main result of this paper is the following theorem:

► **Theorem 2.** *Every predicate φ of QFPA can be computed by a leaderless population protocol \mathcal{P} with $\mathcal{O}(\text{poly}(|\varphi|))$ states. Moreover, \mathcal{P} can be constructed in polynomial time.*

To prove Theorem 2, we first provide a construction that uses $\ell \in \mathcal{O}(|\varphi|^3)$ leaders. If there are at least $|\mathbf{v}| \geq \ell$ input agents \mathbf{v} (*large inputs*), we will show how the protocol can be made leaderless by having agents encode both their state and the state of some leader. Otherwise, $|\mathbf{v}| < \ell$ (*small inputs*), and we will resort to a special construction, with a single leader, that only works for populations of bounded size. We will show how the leader can be simulated collectively by the agents. Hence, we will construct succinct protocols computing φ for large and small inputs, respectively. Formally, we prove:

► **Lemma 3.** *Let φ be a predicate over variables X . There exist $\ell \in \mathcal{O}(|\varphi|^3)$ and leaderless protocols $\mathcal{P}_{\geq \ell}$ and $\mathcal{P}_{< \ell}$ with $\mathcal{O}(\text{poly}(|\varphi|))$ states such that:*

- (a) $\mathcal{P}_{\geq \ell}$ computes predicate $(|\mathbf{v}| \geq \ell) \rightarrow \varphi(\mathbf{v})$, and
- (b) $\mathcal{P}_{< \ell}$ computes predicate $(|\mathbf{v}| < \ell) \rightarrow \varphi(\mathbf{v})$.

Theorem 2 follows immediately from the lemma: it suffices to take the conjunction of both protocols, which only yields a quadratic blow-up on the number of states, using the

classical product construction [3]. The rest of the paper is dedicated to proving Lemma 3. Parts (a) and (b) are shown in Sections 5 and 6, respectively.

In the remainder of the paper, whenever we claim the existence of some protocol \mathcal{P} , we also claim polynomial-time constructibility of \mathcal{P} without mentioning it explicitly.

5 Succinct protocols for large populations

We show that, for every predicate φ , there exists a constant $\ell \in \mathcal{O}(|\varphi|^3)$ and a succinct protocol $\mathcal{P}_{\geq \ell}$ computing $(|\mathbf{v}| \geq \ell) \rightarrow \varphi(\mathbf{v})$. Throughout this section, we say that $n \in \mathbb{N}$ is *large* if $n \geq \ell$, and that a protocol *computes φ for large inputs* if it computes $(|\mathbf{v}| \geq \ell) \rightarrow \varphi(\mathbf{v})$.

We present the proof in a top-down manner, by means of a chain of statements of the form “ $A \leftarrow B$, $B \leftarrow C$, $C \leftarrow D$, and D ”. Roughly speaking, and using notions that will be defined in the forthcoming subsections:

- Section 5.1 introduces protocols with helpers, a special class of protocols with leaders. The section shows: φ is computable for large inputs by a succinct leaderless protocol (A), if it is computable for large inputs by a succinct protocol with helpers (B).
- Section 5.2 defines protocols that simultaneously compute a set of predicates. The section proves: (B) holds if the set P of atomic predicates occurring within φ is simultaneously computable for large inputs by a succinct protocol with helpers (C).
- Section 5.3 introduces protocols with reversible dynamic initialization. The section shows: (C) holds if each atomic predicate of P is computable for large inputs by a succinct protocol with helpers and reversible dynamic initialization (D).
- Section 5.4 shows that (D) holds by exhibiting succinct protocols with helpers and reversible dynamic initialization that compute atomic predicates for large inputs.

Due to space limitations, each section only gives a proof outline. Detailed proofs and some formal definitions are found in the appendix.

5.1 From protocols with helpers to leaderless protocols

Intuitively, a protocol with helpers is a protocol with leaders satisfying an additional property: adding more leaders does not change the predicate computed by the protocol. Formally, let $\mathcal{P} = (Q, T, L, X, I, O)$ be a population protocol computing a predicate φ . We say that \mathcal{P} is a protocol *with helpers* if for every $L' \succeq L$ the protocol $\mathcal{P}' = (Q, T, L', X, I, O)$ also computes φ , where $L' \succeq L \stackrel{\text{def}}{=} \forall q \in Q: (L'(q) = L(q) = 0 \vee L'(q) \geq L(q) > 0)$. If $|L| = \ell$, then we say that \mathcal{P} is a protocol *with ℓ helpers*.

► **Theorem 4.** *Let $\mathcal{P} = (Q, T, L, X, I, O)$ be a Δ -way population protocol with ℓ -helpers computing some predicate φ . There exists a 2-way leaderless population protocol with $\mathcal{O}(\ell \cdot |X| + (\Delta \cdot |T| + |Q|)^2)$ states that computes $(|\mathbf{v}| \geq \ell) \rightarrow \varphi(\mathbf{v})$.*

Proof sketch. By [8, Lemma 3], \mathcal{P} can be transformed into a 2-way population protocol (with helpers²) computing the same predicate φ , and with at most $|Q| + 3\Delta \cdot |T|$ states. Thus, we assume \mathcal{P} to be 2-way in the rest of the sketch.

For simplicity, assume $X = \{x\}$ and $L = \{3 \cdot q, 5 \cdot q'\}$; that is, \mathcal{P} has 8 helpers, and initially 3 of them are in state q , and 5 are in q' . We describe a leaderless protocol \mathcal{P}' that simulates \mathcal{P} for every input \mathbf{v} such that $|\mathbf{v}| \geq |L| = \ell$. Intuitively, \mathcal{P}' runs in two phases:

² Lemma 3 of [8] deals with leaders and not the more specific case of helpers. Nonetheless, computation under helpers is preserved as the input mapping of \mathcal{P} remains unchanged in the proof of the lemma.

- In the first phase each agent gets assigned a number between 1 and 8, ensuring that each number is assigned to at least one agent (this is the point at which the condition $|v| \geq \ell$ is needed). At the end of the phase, each agent is in a state of the form (x, i) , meaning that the agent initially represented one unit of input for variable x , and that it has been assigned number i . To achieve this, initially every agent is placed in state $(x, 1)$. Transitions are of the form $\{(x, i), (x, i)\} \mapsto \{(x, i+1), (x, i)\}$ for every $1 \leq i \leq 7$. The transitions guarantee that all but one agent is promoted to $(x, 2)$, all but one to $(x, 3)$, etc. In other words, one agent is “left behind” at each step.
- In the second phase, agents in state (x, i) move to state $\{I(x), q\}$ if $1 \leq i \leq 3$, and to state $\{I(x), q'\}$ if $4 \leq i \leq 8$. Intuitively, after this move each agent has been assigned two jobs: simultaneously simulate a regular agent of \mathcal{P} starting at state x , and a helper of L starting at state q or q' . Since in the first phase each number is assigned to at least one agent, \mathcal{P}' has at least 3 agents simulating helpers in state q , and at least 5 agents simulating helpers in state q' . There may be many more helpers, but this is harmless, because, by definition, additional helpers do not change the computed predicate.

The transitions of \mathcal{P}' are designed according to this double role of the agents of \mathcal{P}' . More precisely, for all multisets $\mathbf{p}, \mathbf{q}, \mathbf{p}', \mathbf{q}'$ of size two, $\{\mathbf{p}, \mathbf{q}\} \mapsto \{\mathbf{p}', \mathbf{q}'\}$ is a transition of \mathcal{P}' iff $(\mathbf{p} + \mathbf{q}) \rightarrow (\mathbf{p}' + \mathbf{q}')$ in \mathcal{P} . ◀

5.2 From multi-output protocols to protocols with helpers

A *k-output population protocol* is a tuple $\mathcal{Q} = (Q, T, L, X, I, O)$ where $O: [k] \times Q \rightarrow \{0, 1, \perp\}$ and $\mathcal{Q}_i \stackrel{\text{def}}{=} (Q, T, L, X, I, O_i)$ is a population protocol for every $i \in [k]$, where O_i denotes the mapping such that $O_i(q) \stackrel{\text{def}}{=} O(i, q)$ for every $q \in Q$. Intuitively, since each \mathcal{Q}_i only differs by its output mapping, \mathcal{Q} can be seen as a single population protocol whose executions have k outputs. More formally, \mathcal{Q} *computes* a set of predicates $P = \{\varphi_1, \varphi_2, \dots, \varphi_k\}$ if \mathcal{Q}_i computes φ_i for every $i \in [k]$. Furthermore, we say that \mathcal{Q} is *simple* if \mathcal{Q}_i is simple for every $i \in [k]$. Whenever the number k is irrelevant, we use the term *multi-output population protocol* instead of *k-output population protocol*.

► **Proposition 5.** *Assume that every finite set A of atomic predicates is computed by some $|A|$ -way multi-output protocol with $\mathcal{O}(|A|^3)$ helpers and states, and $\mathcal{O}(|A|^5)$ transitions. Every QFPA predicate φ is computed by some simple $|\varphi|$ -way protocol with $\mathcal{O}(|\varphi|^3)$ helpers and states, and $\mathcal{O}(|\varphi|^5)$ transitions.*

Proof sketch. Consider a binary tree decomposing the boolean operations of φ . We design a protocol for φ by induction on the height of the tree.

The case where the height is 0, and φ is atomic, is trivial. We sketch the induction step for the case where the root is labeled with \wedge , that is $\varphi = \varphi_1 \wedge \varphi_2$, the other cases are similar. By induction hypothesis, we have simple protocols $\mathcal{P}_1, \mathcal{P}_2$ computing φ_1, φ_2 , respectively. Let $\mathbf{t}_j, \mathbf{f}_j$ be the output states of \mathcal{P}_j for $j \in \{1, 2\}$ such that $O_j(t_j) = 1$ and $O_j(f_j) = 0$. We add two new states \mathbf{t}, \mathbf{f} (the output states of the new protocol) and an additional helper starting in state \mathbf{f} . To compute $\varphi_1 \wedge \varphi_2$ we add the following transitions for every $b_1 \in \{\mathbf{t}_1, \mathbf{f}_1\}, b_2 \in \{\mathbf{t}_2, \mathbf{f}_2\}$, and $b \in \{\mathbf{t}, \mathbf{f}\}$: $\{b_1, b_2, b\} \mapsto \{b_1, b_2, \mathbf{t}\}$ if $b_1 = \mathbf{t}_1 \wedge b_2 = \mathbf{t}_2$, and $\{b_1, b_2, b\} \mapsto \{b_1, b_2, \mathbf{f}\}$ otherwise. The additional helper computes the conjunction as desired. ◀

5.3 From reversible dynamic initialization to multi-output protocols

Let $P = \{\varphi_1, \dots, \varphi_k\}$ be a set of $k \geq 2$ atomic predicates over a set X of variables. We show how to produce a multi-output protocol \mathcal{P} for P from atomic protocols $\mathcal{P}_1, \dots, \mathcal{P}_k$ for each of $\varphi_1, \dots, \varphi_k$. \mathcal{P} cannot be a “product protocol” that executes $\mathcal{P}_1, \dots, \mathcal{P}_k$ synchronously. Indeed, the states of a “product protocol” are tuples (q_1, \dots, q_k) of states of $\mathcal{P}_1, \dots, \mathcal{P}_k$, and so their number is exponential in k . Instead, we want to execute $\mathcal{P}_1, \dots, \mathcal{P}_k$ asynchronously in parallel. The main obstacle is that this requires a separate input \mathbf{x} for each \mathcal{P}_i , and we only have one. \mathcal{P} cannot replicate the input \mathbf{x} , by using $(k-1) \cdot |\mathbf{x}|$ helpers, because the number of helpers cannot depend on \mathbf{x} . The solution is to define for each φ_i an equivalent atomic predicate $\tilde{\varphi}_i$ which needs at most $|\mathbf{x}|/k + k|X|$ input agents. Intuitively, for every k input agents, the protocol \mathcal{P} sends one input agent to each $\tilde{\varphi}_i$, $i \leq k$. Further, \mathcal{P} uses a bounded number of $O(k^2|X|)$ helpers to encode the remaining input agents of all $(\tilde{\varphi}_i)_{i \leq k}$.

Let us sketch this idea in more detail. Let $X = \{x_1, \dots, x_n\}$. We define $\bar{X} = \{\bar{x}_1, \dots, \bar{x}_n\}$ and $\underline{X} = \{\underline{x}_1, \dots, \underline{x}_n\}$. For each $\mathbf{x} \in \mathbb{N}^X$, define $\bar{\mathbf{x}} \in \mathbb{N}^{\bar{X}}$ and $\underline{\mathbf{x}} \in \mathbb{N}^{\underline{X}}$ with $\bar{\mathbf{x}}(i) := \lfloor \frac{\mathbf{x}(i)}{k} \rfloor$ and $\underline{\mathbf{x}}(i) := \mathbf{x}(i) \bmod k$ for all $i \leq n$. For each atomic predicate φ over X , define an equivalent atomic predicate $\tilde{\varphi}$ over $\bar{X} \cup \underline{X}$ as follows: If α_i is the linear coefficient associated with x_i in φ , then let $(k \cdot \alpha_i)$ be the coefficient associated with \bar{x}_i in $\tilde{\varphi}$, and let α_i be the coefficient associated with \underline{x}_i . For instance, if $\varphi(x_1, x_2) = 3x_1 - 2x_2 > 6$ and $k = 4$, we obtain $\tilde{\varphi}(\bar{x}_1, \underline{x}_1, \bar{x}_2, \underline{x}_2) = 12\bar{x}_1 + 3\underline{x}_1 - 8\bar{x}_2 - 2\underline{x}_2 > 6$. It is readily seen that $\tilde{\varphi}(\bar{\mathbf{x}}, \underline{\mathbf{x}}) = \varphi(\mathbf{x})$ holds for every \mathbf{x} , because $\sum_i \alpha_i \cdot \mathbf{x}(i) = \sum_i (k \cdot \alpha_i \cdot \bar{\mathbf{x}}(i) + \alpha_i \cdot \underline{\mathbf{x}}(i))$. In terms of number of input agents, we have $|\bar{\mathbf{x}}| = \lfloor \frac{|\mathbf{x}|}{k} \rfloor$ and $|\underline{\mathbf{x}}| \leq n \cdot (k-1)$.

Let us now describe the multi-output protocol \mathcal{P} for P . For every $1 \leq j \leq k$, let $\bar{x}_1^j, \dots, \bar{x}_n^j$ and $\underline{x}_1^j, \dots, \underline{x}_n^j$ denote the inputs of $\tilde{\varphi}_j$. Protocol \mathcal{P} repeatedly performs one of these two actions: either it removes k agents from x_i , and adds one agent to \bar{x}_i^j for each $j \leq k$; or it removes one agent from x_i and $k-1$ helpers, and adds one agent to \underline{x}_i^j for each $j \leq k$. These operations are chosen nondeterministically and do not change the number of agents. The protocol needs $n \cdot (k-1)^2$ helpers, as $k-1$ copies are needed. The obvious problem is that \mathcal{P} may or may not send the right number of agents \bar{y} and \underline{y} to $\bar{\mathbf{x}}$ and $\underline{\mathbf{x}}$. Assume, *e.g.*, that $n = 1$, $k = 5$, and $x = 17$. The protocol has $n \cdot (k-1)^2 = 16$ helpers. The protocol may correctly choose $\bar{y} := \bar{x} := \lfloor \frac{17}{5} \rfloor = 3$ and $\underline{y} := \underline{x} := (17 \bmod 5) = 2$; this gives a total of $(3+2) \cdot 5 = 25$ agents, consisting of the 17 agents for the input plus 8 helpers. However, it may also wrongly choose $\bar{y} := 2$ and $\underline{y} := 4$, with a total of $(2+4) \cdot 5 = 30$ agents, 17 input agents plus 13 helpers. In the second case, each \mathcal{P}_i wrongly computes $\tilde{\varphi}_i(2, 4) = \varphi_i(2 \cdot 5 + 4) = \varphi_i(14)$, instead of the correct value $\varphi_i(17)$.

The solution is to make sure that it is possible to restart the protocol if \mathcal{P} chooses a wrong distribution. This includes recovering agents that were assigned to atomic protocols. For this reason, we need to make sure that (i) atomic protocols can work with inputs agents that arrive over time and that might be taken back (*dynamic initialization*), and (ii) if \mathcal{P} chooses a wrong distribution, then atomic protocols can return to their initial configuration (*reversibility*). This allows \mathcal{P} to nondeterministically choose values for \bar{y} and \underline{y} again and again, until it chooses the right distribution, *i.e.*, $\bar{y} := (\lfloor \frac{x}{k} \rfloor)$ and $\underline{y} := (x \bmod k)$. Due to fairness, this will eventually happen. We need to make sure that when \mathcal{P} chooses the right distribution, the atomic protocols do not return to the initial configuration but actually compute their predicates. This is the case *iff* every input $\mathbf{x}(i)$ has been assigned to $\bar{x}_i, \underline{x}_i$ for all i . We then say that the *dynamic initialization terminates*. Indeed, for \mathbf{x} initial agents, when there is no more agent in X , we have by construction $k\bar{\mathbf{y}}(i) + \underline{\mathbf{y}}(i) = \mathbf{x}(i)$ for all i . Thus, $\tilde{\varphi}(\bar{\mathbf{y}}, \underline{\mathbf{y}}) = \varphi(k\bar{\mathbf{y}} + \underline{\mathbf{y}}) = \varphi(\mathbf{x})$ is correct. Hence, we ensure that the set T_\dagger of transitions making the system reversible is enabled *only if the set X of states is populated*: A transition

$p \mapsto q$ that should be disabled after the correct initialization is transformed into transitions $p + \langle x \rangle \mapsto q + \langle x \rangle$, one for each input state $x \in X$. Therefore, once we correctly assign the agents, all transitions of T_{\dagger} are disabled.

Reversible dynamic initialization. Let us now formally introduce the class of *protocols with reversible dynamic initialization* that enjoys all properties needed for our construction. A simple protocol with *reversible dynamic initialization* (*RDI-protocol* for short) is a tuple $\mathcal{P} = (Q, T_{\infty}, T_{\dagger}, L, X, I, O)$, where $\mathcal{P}_{\infty} = (Q, T_{\infty}, L, X, I, O)$ is a simple population protocol, and T_{\dagger} is the set of transitions making the system reversible, called the *RDI-transitions*.

Let $T \stackrel{\text{def}}{=} T_{\infty} \cup T_{\dagger}$, and let $\text{In} \stackrel{\text{def}}{=} \{\text{in}_x : x \in X\}$ and $\text{Out} \stackrel{\text{def}}{=} \{\text{out}_x : x \in X\}$ be the sets of *input* and *output transitions*, respectively, where $\text{in}_x \stackrel{\text{def}}{=} (\mathbf{0}, \langle I(x) \rangle)$ and $\text{out}_x \stackrel{\text{def}}{=} (\langle I(x) \rangle, \mathbf{0})$. An *initialization sequence* is a finite execution $\pi \in (T \cup \text{In} \cup \text{Out})^*$ from the *initial configuration* L' with $L' \succeq L$. The *effective input* of π is the vector \mathbf{w} such that $\mathbf{w}(x) \stackrel{\text{def}}{=} |\pi|_{\text{in}_x} - |\pi|_{\text{out}_x}$ for every $x \in X$. Intuitively, a RDI-protocol starts with helpers only, and is dynamically initialized via the input and output transitions.

Let $\mathbf{f}, \mathbf{t} \in Q$ be the unique states of \mathcal{P} with $O(\mathbf{f}) = 0$ and $O(\mathbf{t}) = 1$. For every configuration C , let $[C] \stackrel{\text{def}}{=} \{C' : C'(\mathbf{f}) + C'(\mathbf{t}) = C(\mathbf{f}) + C(\mathbf{t}) \text{ and } C'(q) = C(q) \text{ for all } q \in Q \setminus \{\mathbf{f}, \mathbf{t}\}\}$. Intuitively, all configurations $C' \in [C]$ are equivalent to C in all but the output states.

An RDI-protocol is required to be *reversible*, that is for every initialization sequence π with effective input \mathbf{w} , and such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, the following holds:

- if $C \xrightarrow{T^*} D$ and $D' \in [D]$, then $D' \xrightarrow{T^*} C'$ for some $C' \in [C]$, and
- $C(I(x)) \leq \mathbf{w}(x)$ for all $x \in X$.

Intuitively, an RDI-protocol can always reverse all sequences that do not contain input or output transitions. The reversal does not concern the states \mathbf{f} and \mathbf{t} . Further, an RDI-protocol can never have more agents in an input state than the effective number of agents it received via the input and output transitions.

An RDI-protocol \mathcal{P} *computes* φ if for every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, the standard population protocol \mathcal{P}_{∞} computes $\varphi(\mathbf{w})$ from C (that is with T_{\dagger} disabled). Intuitively, if the dynamic initialization terminates, the RDI-transitions T_{\dagger} become disabled, and then the resulting standard protocol \mathcal{P}_{∞} converges to the output corresponding to the dynamically initialized input.

► **Theorem 6.** *Assume that for every atomic predicate φ , there exists a $|\varphi|$ -way RDI-protocol with $\mathcal{O}(|\varphi|)$ helpers, $\mathcal{O}(|\varphi|^2)$ states and $\mathcal{O}(|\varphi|^3)$ transitions that computes φ . For every finite set P of atomic predicates, there exists a $|P|$ -way simple multi-output protocol, with $\mathcal{O}(|P|^3)$ helpers and states, and $\mathcal{O}(|P|^5)$ transitions, that computes P .*

5.4 Atomic predicates under reversible dynamic initialization

Lastly, we show that atomic predicates are succinctly computable by RDI-protocols:

► **Theorem 7.** *Every atomic predicate φ over variables X can be computed by a simple $|\varphi|$ -way population protocol with reversible dynamic initialization that has $\mathcal{O}(|\varphi|)$ helpers, $\mathcal{O}(|\varphi|^2)$ states, and $\mathcal{O}(|\varphi|^3)$ transitions.*

The protocols for arbitrary threshold and remainder predicates satisfying the conditions of Theorem 7, and their correctness proofs, are given in the Appendix. Note that the threshold protocol is very similar to the protocol for linear inequalities given in Section 6 of [8]. Thus, as an example, we will instead describe how to handle the remainder predicate $5x - y \equiv_7 4$. Note, that the predicate can be rewritten as $(5x + 6y \geq 4 \pmod{7}) \wedge (5x + 6y \not\geq 5 \pmod{7})$. As we can handle negations and conjunctions separately in Section 5.2, we will now explain

the protocol for $\varphi \stackrel{\text{def}}{=} 5x + 6y \geq 4 \pmod{7}$. The protocol is partially depicted in Figure 1 using Petri net conventions for the graphical representation.

The protocol has an *input state* \mathbf{x} for each variable $x \in X$, *output states* \mathbf{f} and \mathbf{t} , a *neutral state* 0 , and *numerical states* of the form $+2^i$ for every $0 \leq i \leq n$, where n is the smallest number such that $2^n > \|\varphi\|$. Initially, (at least) one helper is set to \mathbf{f} and (at least) $2n$ helpers set to 0 . In order to compute $5x + 6y \geq 4 \pmod{7}$ for $x := r$ and $y := s$, we initially place r and s agents in the states \mathbf{x} and \mathbf{y} , i.e., the agents in state \mathbf{x} encode the number r in unary, and similarly for \mathbf{y} . The blue transitions on the left of Figure 1 “convert” each agents in input states to a binary representation of their corresponding coefficient. In our example, agents in state \mathbf{x} are converted to $\mathbf{a}(x) = 5 = 0101_2$ by putting one agent in 4 and another one in 1 . Since two agents are needed to encode 5 , the transition “recruits” one helper from state 0 . Observe that, since the inputs can be arbitrarily large, but a protocol can only use a constant number of helpers, the protocol must reuse helpers in order to convert all agents in input states.

This happens as follows. If two agents are in the same power of two, say $+2^i$, then one of them can be “promoted” to $+2^{i+1}$, while the other one moves to state 0 , “liberating” one helper. This allows the agents to represent the overall value of all converted agents in the most efficient representation. That is, from any configuration, one can always reach a configuration where there is at most one agent in each place $2^0, \dots, 2^{n-1}$, there are at most the number of agents converted from input places in place 2^n , and hence there are at least n agents in place 0 , thus ready to convert some agent from the input place.

Similar to promotions, “demotions” to smaller powers of two can also happen. Thus, the agents effectively shift through all possible binary representations of the overall value of all converted agents. The \equiv_7 transition in Figure 1 allows 3 agents in states 4 , 2 and 1 to “cancel out” by moving to state 0 , and it moves the output helper to \mathbf{f} . Furthermore, there are RDI-transitions that allow to revert the effects of conversion and cancel transitions. These are not shown in Figure 1.

We have to show that this protocol computes φ under reversible dynamic initialization. First note, that while dynamic initialization has not terminated, all transitions have a corresponding reverse transition. Thus, it is always possible to return to wrong initial configurations. However, reversing the conversion transitions can create more agents in input states than the protocol effectively received. To forbid this, each input agent is “tagged” with its variable (see tokens in Figure 1). Therefore, in order to reverse a conversion transitions, the original input agent is needed. This implies, that the protocol is reversible.

Next, we need to argue that the protocol without the RDI-transitions computes φ once the dynamic initialization has terminated. The agents will shift through the binary

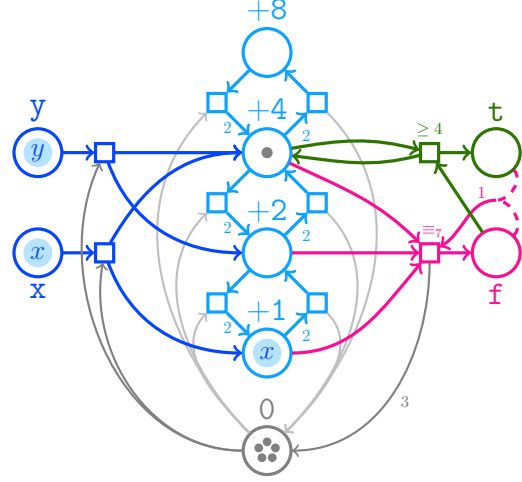


Figure 1 Partial representation of the protocol computing $5x + 6y \geq 4 \pmod{7}$ as a Petri net, where places (circles), transitions (squares) and tokens (smaller filled circles) represent respectively states, transitions and agents. Non-helper agents remember their input variable (labeled here within tokens). The depicted configuration is obtained from input $x = 2$, $y = 1$ by firing the bottom leftmost transition (dark blue).

representations of the overall value. Because of fairness, the \equiv_7 transition will eventually reduce the overall value to at most 6. There is a ≥ 4 -transition which detects the case where the final value is at least 4 and moves the output helper from **f** to state **t**. Notice that whenever transition \equiv_7 occurs, we reset the output by moving the output helper to state **f**.

6 Succinct protocols for small populations

We show that for every predicate φ and constant $\ell = \mathcal{O}(|\varphi|^3)$, there exists a succinct protocol $\mathcal{P}_{<\ell}$ that computes the predicate $(|\mathbf{v}| < \ell) \rightarrow \varphi(\mathbf{v})$. In this case, we say that $\mathcal{P}_{<\ell}$ *computes φ for small inputs*. Further, we say that a number $n \in \mathbb{N}$ (resp. an input \mathbf{v}) is small with respect to φ if $n \leq \ell$ (resp. $|\mathbf{v}| \leq \ell$). We present the proof strategy in a top-down manner.

- Section 6.1 proves: There is a succinct leaderless protocol \mathcal{P} that computes φ for small inputs (A), if for every small n some succinct protocol \mathcal{P}_n computes φ for all inputs of size n (B). Intuitively, constructing a succinct protocol for all small inputs reduces to the simpler problem of constructing a succinct protocol for all small inputs of a fixed size.
- Section 6.2 introduces halting protocols. It shows: There is a succinct protocol that computes φ for inputs of size n , if for every *atomic* predicate ψ of φ some halting succinct protocol computes ψ for inputs of size n (C). Thus, constructing protocols for arbitrary predicates reduces to constructing *halting* protocols for atomic predicates.
- Section 6.3 proves (C). Given a threshold or remainder predicate φ and a small n , it shows how to construct a succinct halting protocol that computes φ for inputs of size n .

6.1 From fixed sized protocols with one leader to leaderless protocols

We now define when a population protocol computes a predicate *for inputs of a fixed size*. Intuitively, it should compute the correct value for every initial configurations of this size; for inputs of other sizes, the protocol may converge to the wrong result, or may not converge.

► **Definition 8.** Let φ be a predicate and let $i \geq 2$. A protocol \mathcal{P} computes φ for inputs of size i , denoted “ \mathcal{P} computes $(\varphi \mid i)$ ”, if for every input \mathbf{v} of size i , every fair execution of \mathcal{P} starting at $C_{\mathbf{v}}$ stabilizes to $\varphi(C)$.

We show that if, for each small number i , some succinct protocol computes $(\varphi \mid i)$, then there is a single succinct protocol that computes φ for all small inputs.

► **Theorem 9.** Let φ be a predicate over a set of variables X , and let $\ell \in \mathbb{N}$. Assume that for every $i \in \{2, 3, \dots, \ell - 1\}$, there exists a protocol with at most one leader and at most m states that computes $(\varphi \mid i)$. Then, there is a leaderless population protocol with $\mathcal{O}(\ell^4 \cdot m^2 \cdot |X|^3)$ states that computes $(\mathbf{x} < \ell) \rightarrow \varphi(\mathbf{x})$.

Proof sketch. Fix a predicate φ and $\ell \in \mathbb{N}$. For every $2 \leq i < \ell$, let \mathcal{P}_i be a protocol computing $(\varphi \mid i)$. We describe the protocol $\mathcal{P} = (Q, T, X, I, O)$ that computes $(\mathbf{x} \geq \ell) \vee \varphi(\mathbf{x}) \equiv (\mathbf{x} < \ell) \rightarrow \varphi(\mathbf{x})$. The input mapping I is the identity. During the computation, agents never forget their initial state – that is, all successor states of an agent are annotated with their initial state. The protocol initially performs a leader election. Each provisional leader stores how many agents it has “knocked out” during the leader election in a counter from 0 to $\ell - 1$. After increasing the counter to a given value $i < \ell$, it resets the state of i agents and itself to the corresponding initial state of \mathcal{P}_{i+1} , annotated with X , and initiates a simulation of \mathcal{P}_{i+1} . When the counter of an agent reaches $\ell - 1$, the agent knows that the population size must be $\geq \ell$, and turns the population into a permanent 1-consensus. Now,

if the population size i is smaller than ℓ , then eventually a leader gets elected who resets the population to the initial population of \mathcal{P}_i . Since \mathcal{P}_i computes $(\varphi \mid i)$, the simulation of \mathcal{P}_i eventually yields the correct output. \blacktriangleleft

6.2 Computing boolean combinations of predicates for fixed-size inputs

We want to produce a population protocol \mathcal{P} for a boolean combination φ of atomic predicates $(\varphi_i)_{i \in [k]}$ for which we have population protocols $(\mathcal{P}_i)_{i \in [k]}$. As in Section 5.3, we cannot use a standard “product protocol” that executes $\mathcal{P}_1, \dots, \mathcal{P}_k$ synchronously because the number of states would be exponential in k . Instead, we want to simulate the *concatenation* of $(\mathcal{P}_i)_{i \in [k]}$. However, this is only possible if for all $i \in [k]$, the executions of \mathcal{P}_i eventually “halt”, i.e. some agents are eventually certain that the output of the protocol will not change anymore, which is not the case in general population protocols. For this reason we restrict our attention to “halting” protocols.

► **Definition 10.** Let \mathcal{P} be a simple protocol with output states \mathbf{f} and \mathbf{t} . We say that \mathcal{P} is a halting protocol if every configuration C reachable from an initial configuration satisfies:

- $C(\mathbf{f}) = 0 \vee C(\mathbf{t}) = 0$,
- $C \xrightarrow{*} C' \wedge C(q) > 0 \Rightarrow C'(q) > 0$ for every $q \in \{\mathbf{f}, \mathbf{t}\}$ and every configuration C' .

Intuitively, a halting protocol is a simple protocol in which states \mathbf{f} and \mathbf{t} behave like “final states”: If an agent reaches $q \in \{\mathbf{f}, \mathbf{t}\}$, then the agent stays in q forever. In other words, the protocol reaches consensus 0 (resp. 1) iff an agent ever reaches \mathbf{f} (resp. \mathbf{t}).

► **Theorem 11.** Let $k, i \in \mathbb{N}$. Let φ be a boolean combination of atomic predicates $(\varphi_j)_{j \in [k]}$. Assume that for every $j \in [k]$, there is a simple halting protocol $\mathcal{P}_j = (Q_j, L_j, X, T_j, I_j, O_j)$ with one leader computing $(\varphi_j \mid i)$. Then there exists a simple halting protocol \mathcal{P} that computes $(\varphi \mid i)$, with one leader and $\mathcal{O}(|X| \cdot (\text{len}(\varphi) + |Q_1| + \dots + |Q_k|))$ states.

Proof sketch. We only sketch the construction for $\varphi = \varphi_1 \wedge \varphi_2$; details can be found in the appendix. The main intuition is that, since \mathcal{P}_1 and \mathcal{P}_2 are halting, we can construct a protocol that, given an input \mathbf{v} , first simulates \mathcal{P}_1 on \mathbf{v} , and, after \mathcal{P}_1 halts, either halts if \mathcal{P}_1 converges to 0, or simulates \mathcal{P}_2 on \mathbf{v} if \mathcal{P}_1 converges to 1. Each agent remembers in its state the input variable it corresponds to, in order to simulate \mathcal{P}_2 on \mathbf{v} . \blacktriangleleft

6.3 Computing atomic predicates for fixed-size inputs

We describe a halting protocol that computes a given threshold predicate for fixed-size inputs. The protocols for the remainder predicates can be found in the appendix.

► **Theorem 12.** Let $\varphi(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \alpha \cdot \mathbf{x} - \beta \cdot \mathbf{y} > 0$. For every $i \in \mathbb{N}$, there exists a halting protocol with one leader and $\mathcal{O}(i^2(|\varphi| + \log i)^3)$ states that computes $(\varphi \mid i)$.

We first describe a sequential algorithm **Greater-Sum**(\mathbf{x}, \mathbf{y}), that for every input \mathbf{x}, \mathbf{y} satisfying $|\mathbf{x}| + |\mathbf{y}| = i$ decides whether $\alpha \cdot \mathbf{x} - \beta \cdot \mathbf{y} > 0$ holds. Then we simulate **Greater-Sum** by means of a halting protocol with i agents.

Since each agent can only have $\mathcal{O}(\log i + \log |\varphi|)$ bits of memory (the logarithm of the number of states), **Greater-Sum** must use at most $\mathcal{O}(i \cdot (\log i + \log |\varphi|))$ bits of memory, otherwise it cannot be simulated by the agents. Because of this requirement, **Greater-Sum** cannot just compute, store, and then compare $\alpha \cdot \mathbf{x}$ and $\beta \cdot \mathbf{y}$; this uses too much memory.

Greater-Sum calls procedures $\text{Probe}_1(j)$ and $\text{Probe}_2(j)$ that return the j -th bits of $\alpha \mathbf{x}$ and $\beta \mathbf{y}$, respectively, where $j = 1$ is the most significant bit. Since $|\mathbf{x}| \leq i$, and the

largest constant in α is at most $\|\varphi\|$, we have $\alpha \cdot \mathbf{x} \leq i \cdot \|\varphi\|$, and so $\alpha \cdot \mathbf{x}$ has at most $m \stackrel{\text{def}}{=} |\varphi| + \lfloor \log(i) \rfloor + 1$ bits, and the same holds for $\beta \mathbf{y}$. So we have $1 \leq j \leq m$. Let us first describe **Greater-Sum**, and then $\text{Probe}_1(j)$; the procedure $\text{Probe}_2(j)$ is similar.

Greater-Sum(\mathbf{x}, \mathbf{y}) loops through $j = 1, \dots, m$. For each j , it calls $\text{Probe}_1(j)$ and $\text{Probe}_2(j)$. If $\text{Probe}_1(j) > \text{Probe}_2(j)$, then it answers $\varphi(\mathbf{x}, \mathbf{y}) = 1$, otherwise it moves to $j + 1$. If **Greater-Sum** reaches the end of the loop, then it answers $\varphi(\mathbf{x}, \mathbf{y}) = 0$. Observe that **Greater-Sum** only needs to store the current value of j and the bits returned by $\text{Probe}_1(j)$ and $\text{Probe}_2(j)$. Since $j \leq m$, **Greater-Sum** only needs $\mathcal{O}(\log(|\varphi| + \log i))$ bits of memory.

$\text{Probe}_1(j)$ uses a decreasing counter $k = m, \dots, j$ to successively compute the bits $b_1(k)$ of $\alpha \cdot \mathbf{x}$, starting at the least significant bit. To compute $b_1(k)$, the procedure stores the carry $c_k \leq i$ of the computation of $b_1(k+1)$; it then computes the sum $s_k := c_k + \alpha(k) \cdot \mathbf{x}$ (where $\alpha(k)$ is the k -th vector of bits of α), and sets $b_k := s_k \bmod 2$ and $c_{k-1} := s_k \div 2$. The procedure needs $\mathcal{O}(\log(|\varphi| + \log i))$ bits of memory for counter k , $\log(i) + 1$ bits for encoding s_k , and $\mathcal{O}(\log(i))$ bits for encoding c_k . So it only uses $\mathcal{O}(\log(|\varphi| + \log i))$ bits of memory.

Let us now simulate **Greater-Sum**(\mathbf{x}, \mathbf{y}) by a halting protocol with one leader agent. Intuitively, the protocol proceeds in rounds corresponding to the counter k . The leader stores in its state the value j and the current values of the program counter, of counter k , and of variables b_k , s_k , and c_k . The crucial part is the implementation of the instruction $s_k := c_k + \alpha(k) \cdot \mathbf{x}$ of $\text{Probe}_1(j)$. In each round, the leader adds input agents one by one. As the protocol only needs to work for populations with i agents, it is possible for each agent to know if it already interacted with the leader in this round, and for the leader to count the number of agents it has interacted with this round, until it reaches i to start the next round. For further details of the construction in the general case, see Appendix C.3.

7 Conclusion and further work

We have proved that every predicate φ of quantifier-free Presburger arithmetic (QFPA) is computed by a leaderless protocol with $\text{poly}(|\varphi|)$ states. Further, the protocol can be computed in polynomial time. The number of states of previous constructions was exponential both in the bit-length of the coefficients of φ , and in the number of occurrences of boolean connectives. Since QFPA and PA have the same expressive power, every computable predicate has a succinct leaderless protocol. This result completes the work initiated in [8], which also constructed succinct protocols, but only for some predicates, and with the help of leaders.

Our result shows that there is no exponential gap in state complexity between leaderless protocols and protocols with leaders. This is in contrast with the situation for time complexity, where the exponential gap has been recently shown to exist [6, 1].

The question of whether protocols with $\text{poly}(|\varphi|)$ states exist for every PA formula φ , possibly with quantifiers, remains open. However, it is easy to prove that no algorithm for the construction of protocols from PA formulas can run in time $2^{p(n)}$ for any polynomial p (the proof is in Appendix D):

► **Theorem 13.** *For every polynomial p , every algorithm that accepts a formula φ of PA as input, and returns a population protocol computing φ , runs in time $2^{\omega(p(|\varphi|))}$.*

Therefore, if PA also has succinct protocols, then they are very hard to find.

Our succinct protocols for QFPA have slow convergence (in the usual parallel time model, see e.g. [2]), since they often rely on exhaustive exploration of a number of alternatives, until the right one is eventually hit. The question of whether every QFPA predicate has a succinct and fast protocol is very challenging, and we leave it open for future research.

References

- 1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2560–2579. SIAM, 2017.
- 2 Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, 2018. doi:10.1145/3289137.3289150.
- 3 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 290–299, 2004. doi:10.1145/1011767.1011810.
- 4 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.
- 5 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 292–299, 2006. doi:10.1145/1146381.1146425.
- 6 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.
- 7 Leonard Berman. The complexity of logical theories. *Theoretical Computer Science*, 11:71–77, 1980.
- 8 Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: On the minimal size of population protocols. In *Proc. 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 16:1–16:14, 2018. doi:10.4230/LIPIcs.STACS.2018.16.
- 9 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018.
- 10 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bulletin of the EATCS*, 126, 2018.
- 11 Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar. Verification of population protocols. *Acta Informatica*, 54(2):191–215, 2017. doi:10.1007/s00236-016-0272-3.
- 12 Christoph Haase. Subclasses of Presburger arithmetic and the weak EXP hierarchy. In *CSL-LICS*, pages 47:1–47:10. ACM, 2014.
- 13 Christoph Haase. A survival guide to Presburger arithmetic. *SIGLOG News*, 5(3):67–82, 2018.
- 14 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.
- 15 Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes rendus du I^{er} Congrès des mathématiciens des pays slaves*, pages 192–201, 1929.
- 16 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.

A

 Equivalence of simple and standard population protocols

Recall that a simple population protocol (SPP), has two unique states $\mathbf{f}, \mathbf{t} \in Q$ with outputs $O(\mathbf{f}) = 0$ and $O(\mathbf{t}) = 1$ and all other states q have output $O(q) = \perp$.

In the standard definition of population protocols used in the literature, all states q have an output $O(q) \in \{0, 1\}$. In this section we call such a protocol a *full output population protocols* (FOPP). In a FOPP, a configuration C is a consensus configuration if $O(p) = O(q)$ for every $p, q \in \llbracket C \rrbracket$. If C is a consensus configuration, then its output $O(C)$ is the unique output of its states, otherwise it is \perp . An execution $\sigma = C_0 C_1 \dots$ stabilizes to $b \in \{0, 1\}$ if $O(C_i) = O(C_{i+1}) = \dots = b$ for some $i \in \mathbb{N}$. The output of σ is $O(\sigma) = b$ if it stabilizes to b , and $O(\sigma) = \perp$ otherwise. A consensus configuration C is stable if every configuration C' reachable from C is a consensus configuration such that $O(C') = O(C)$. It is easy to see that a fair execution of a FOPP stabilizes to $b \in \{0, 1\}$ if and only if it contains a stable configuration whose output is b .

A FOPP \mathcal{P} computes a predicate $\varphi: \mathbb{N}^X \rightarrow \{0, 1\}$ if for every $\mathbf{v} \in \mathbb{N}^X$ every fair execution σ starting from $C_{\mathbf{v}}$ stabilizes to $\varphi(\mathbf{v})$.

In the rest of the section we show that every FOPP has an equivalent SPP, and vice versa. Both translations have linear blow-up.

FOPP \rightarrow SPP. Let $\mathcal{P} = (Q, T, L, X, I, O)$ be a FOPP computing a predicate φ . We obtain a SPP protocol \mathcal{P}' by adding two output states $\{\mathbf{f}, \mathbf{t}\}$ to \mathcal{P} , plus a new state \perp . The output function of \mathcal{P}' is the mapping $O' : q \mapsto (0 \text{ if } q = \mathbf{f} \text{ else } 1 \text{ if } q = \mathbf{t} \text{ else } \perp)$. The set L' of leaders of \mathcal{P}' is obtained by adding one leader to L , initially in state \perp . Finally, the set T' of transitions is obtained by adding to T , for all $b \in \{\mathbf{f}, \mathbf{t}, \perp\}$, a transition $\langle q, b \rangle \mapsto \langle q, \mathbf{f} \rangle$ for every state $q \in Q$ such that $O(q) = 0$ and a transition $\langle q, b \rangle \mapsto \langle q, \mathbf{t} \rangle$ for every state $q \in Q$ such that $O(q) = 1$.

We show that \mathcal{P}' also computes φ . Let $C'_0 C'_1 \dots$ be a fair execution of \mathcal{P}' from C'_0 . Projecting it onto the set of states of \mathcal{P} yields a fair execution of \mathcal{P} . Since \mathcal{P} computes φ , the execution outputs some $b \in \{0, 1\}$. Assume that $b = 0$ (the case $b = 1$ is symmetric). Let $i \in \mathbb{N}$ such that the output of every state populated by C_j is b for every $j \geq i$. Now, no matter the state populated by the additional leader in C_j (which is one of $\{\mathbf{f}, \mathbf{t}, \perp\}$), the transition $\langle q, b \rangle \mapsto \langle q, \mathbf{f} \rangle$ is enabled for every state q such that $C_j(q) = 0$. By fairness, the leader will thus eventually move to state \mathbf{f} and it will be stuck there, and \mathcal{P}' outputs 0 as well.

SPP \rightarrow FOPP. Let $\mathcal{P} = (Q, T, L, X, I, O)$ be an SPP with output states $\mathbf{f}, \mathbf{t} \in Q$ computing a predicate φ . Let \mathcal{P}' be the FOPP with two disjoint copies Q_0, Q_1 of Q as states. For $a \in Q$, let a_b denote the copy of a in Q_b , for $b \in \{0, 1\}$. We define $O'(q) = b$ for all $q \in Q_b$. The set T' of transitions is the following. First, for every transition $\langle x, y \rangle \mapsto \langle z, u \rangle$ of T , the set T' contains a transition $\langle x_b, y_c \rangle \mapsto \langle z_d, u_e \rangle$ for every $b, c, d, e \in \{0, 1\}$ such that if $b = c$ then $d = e = b = c$. Further, T' also contains a set T'' of transitions consisting of $\langle \mathbf{f}_1 \rangle \mapsto \langle \mathbf{f}_0 \rangle$, $\langle \mathbf{t}_0 \rangle \mapsto \langle \mathbf{t}_1 \rangle$, $\langle a_1, \mathbf{f}_0 \rangle \mapsto \langle a_0, \mathbf{f}_0 \rangle$ and $\langle a_0, \mathbf{t}_1 \rangle \mapsto \langle a_1, \mathbf{t}_1 \rangle$ for every $a \in Q$.

The input mapping and leader multiset of \mathcal{P}' are the “0” copies of the input mapping and leader multiset of \mathcal{P} . Hence, for any input \mathbf{v} the initial configuration $C'_{\mathbf{v}}$ of \mathcal{P}' is the “0” copy of the initial configuration $C_{\mathbf{v}}$ in \mathcal{P} .

We show that \mathcal{P}' also computes φ . Let $C'_0 C'_1 \dots$ be a fair execution of \mathcal{P}' from C'_0 . For every $i \in \mathbb{N}$, let $C_i = \pi(C'_i)$, where π is the mapping defined by $\pi(q_b) = b$ for every $q_b \in Q_0 \cup Q_1$. It is easy to see that $C_0 C_1 \dots$ is a fair execution of \mathcal{P} , with possible repetitions $C_i = C_{i+1}$ when the transition from C'_i to C'_{i+1} is in T'' . Hence $C_0 C_1 \dots$ eventually stabilizes to an output b . Assume that $b = 0$ (the case $b = 1$ is symmetric). By fairness, because of the

transitions $\{f_1\} \mapsto \{f_0\}$, and $\{a_1, f_0\} \mapsto \{a_0, f_0\}$, the execution $C'_0 C'_1 \dots$ eventually reaches, and gets trapped in, configurations of \mathbb{N}^{Q_0} . So the execution also stabilizes to the output 0.

B Proofs of Section 5: Protocols for large populations

B.1 Proof of Theorem 4

► **Theorem 4.** *Let $\mathcal{P} = (Q, T, L, X, I, O)$ be a Δ -way population protocol with ℓ -helpers computing some predicate φ . There exists a 2-way leaderless population protocol with $\mathcal{O}(\ell \cdot |X| + (\Delta \cdot |T| + |Q|)^2)$ states that computes $(|v| \geq \ell) \rightarrow \varphi(v)$.*

We first define a leaderless protocol $\overline{\mathcal{P}}$, introduce some auxiliary definitions and propositions, and finally prove that $\overline{\mathcal{P}}$ computes $(|v| \geq \ell) \rightarrow \varphi(v)$.

The protocol $\overline{\mathcal{P}}$. As mentioned in the main text, by [8, Lemma 3], \mathcal{P} can be transformed into a 2-way population protocol (with helpers) also computing φ , and with at most $|Q| + 3\lambda \cdot |T|$ states, where $\lambda \stackrel{\text{def}}{=} \max\{|p| : (p, q) \in T\}$. Thus, we assume that \mathcal{P} is 2-way in the rest of this section, implicitly keeping in mind the polynomial increase in the number of states.

Let $h_1, h_2, \dots, h_\ell \in Q$ be the helpers of \mathcal{P} in some arbitrary but fixed order. For example, if $L = \{p, 3 \cdot q\}$, then we can have $h_1 = p$, $h_2 = q$, $h_3 = q$ and $h_4 = q$. Let $\overline{\mathcal{P}} \stackrel{\text{def}}{=} (\overline{Q}, \overline{T}, \mathbf{0}, X, \overline{I}, \overline{O})$ be the population protocol such that:

$$\begin{aligned} \overline{Q} &\stackrel{\text{def}}{=} (X \times [\ell]) \cup Q^{(2)}, \\ \overline{T} &\stackrel{\text{def}}{=} \overline{T}_{\text{count}} \cup \overline{T}_{\text{init}} \cup \overline{T}_{\text{simul}}, \\ \overline{I} &\stackrel{\text{def}}{=} x \mapsto (x, 1), \\ \overline{O} &\stackrel{\text{def}}{=} \begin{cases} (x, i) \mapsto 1 & \text{for every } (x, i) \in X \times [\ell], \\ \mathbf{q} \mapsto O(\mathbf{q}) & \text{for every } \mathbf{q} \in Q^{(2)}, \end{cases} \end{aligned}$$

where

■ $\overline{T}_{\text{count}}$ is the set consisting of the following transitions:

$$\{(x, i), (y, i)\} \mapsto \{(x, i+1), (y, i)\} \quad \text{for every } x, y \in X \text{ and } i < \ell,$$

■ $\overline{T}_{\text{init}}$ is the set consisting of the following transitions:

$$\begin{aligned} \{(x, \ell), (y, i)\} &\mapsto \{(\overline{I}(x), h_\ell), (\overline{I}(y), h_i)\} && \text{for every } x, y \in X \text{ and } i \leq \ell, \\ \{\mathbf{q}, (y, i)\} &\mapsto \{\mathbf{q}, (\overline{I}(y), h_i)\} && \text{for every } y \in X, i \leq \ell, \text{ and } \mathbf{q} \in Q^{(2)}, \end{aligned}$$

■ $\overline{T}_{\text{simul}}$ is the consisting of the following transitions:

$$\{p, q\} \mapsto \{p', q'\} \quad \text{for every } p, q, p', q' \in Q^{(2)} \text{ such that } (p + q) \rightarrow (p' + q') \text{ in } \mathcal{P}.$$

Auxiliary definitions and propositions. The intended behavior of $\overline{\mathcal{P}}$ is to first fire $\overline{T}_{\text{count}}$, then $\overline{T}_{\text{init}}$, and then $\overline{T}_{\text{simul}}$. Although $\overline{\mathcal{P}}$ may fire sequences not respecting this order, there always exist an equivalent sequence respecting the order, in the following sense:

► **Proposition 14.** *For every configurations C and D such that $C \xrightarrow{*} D$, there exist $x \in \overline{T}_{\text{count}}^*$, $y \in \overline{T}_{\text{init}}^*$ and $z \in \overline{T}_{\text{simul}}^*$ such that $C \xrightarrow{xyz} D$.*

Proof. Let $w \in \overline{T}^*$ be such that $C \xrightarrow{w} D$. The sequence xyz is simply obtained by reordering the transitions of w . Firability of xyz follows from inspection of \overline{T} . ◀

Observe that firing $\overline{T}_{\text{count}}$, until no further possible, counts the number of agents up to ℓ :

► **Proposition 15.** *Let C and D be configurations such that C is initial, $C \xrightarrow{\overline{T}_{\text{count}}^*} D$ and $\overline{T}_{\text{count}}$ is disabled in D . We have $\llbracket D \rrbracket \cap (X \times \{j\}) \neq \emptyset \iff |C| \geq j$ for every $j \in [\ell]$.*

Proof. Let $P_j \stackrel{\text{def}}{=} X \times \{j\}$ for every $j \in [\ell]$. For every configuration E , let $\text{pos}(E) \stackrel{\text{def}}{=} \{j \in [\ell] : C(P_j) > 0\}$. We define a relation \prec on configurations:

$$E \prec E' \iff \text{pos}(E') = \text{pos}(E) \vee \text{pos}(E') = \text{pos}(E) \cup \{\max(\text{pos}(E)) + 1\}.$$

Observe that $E \xrightarrow{\overline{T}_{\text{count}}} E'$ implies $E \prec E'$. Consequently, since $\text{pos}(C) = \{1\}$, we have $\text{pos}(D) = \{1, 2, \dots, m\}$ for some $m \in [\ell]$. To complete the proof, it suffices to show that $m = \min(|C|, \ell)$.

Clearly, $m \leq \min(|D|, \ell) = \min(|C|, \ell)$ holds. Let us show that $m \geq \min(|C|, \ell)$. If $m = \ell$, then we are done. Therefore, assume $m < \ell$. Since $\overline{T}_{\text{count}}$ is disabled in D , we have $D(P_i) = 1$ for every $1 \leq i \leq m$ and $D(P_i) = 0$ for every $m < i \leq \ell$. Thus, $m = |D| = |C| \geq \min(|C|, \ell)$. ◀

For every configuration C of $\overline{\mathcal{P}}$, let \widehat{C} be the configuration of \mathcal{P} obtained by “projecting” C onto Q , i.e. the configuration such that

$$\widehat{C}(p) \stackrel{\text{def}}{=} \sum_{q \in Q^{(2)}} q(p) \cdot C(q) \quad \text{for every } p \in Q.$$

We extend this notation to executions, i.e., to sequences of configurations. The following correspondence follows immediately from the definitions:

► **Proposition 16.** *For every (fair) execution σ of $\overline{\mathcal{P}}$, $\widehat{\sigma}$ is a (fair) execution of \mathcal{P} .*

Main proof. We prove that $\overline{\mathcal{P}}$ computes φ .

Proof of Theorem 4. Let $\mathbf{v} \in \mathbb{N}^X$ and let σ be a fair execution of $\overline{\mathcal{P}}$ from $C_{\mathbf{v}}$. Observe that, by definition of \overline{T} , the number of transitions from $\overline{T} \setminus \overline{T}_{\text{simul}}$ occurring along σ must be finite. Let $i \in \mathbb{N}$ be some index such that $\overline{T} \setminus \overline{T}_{\text{simul}}$ is disabled in σ_j for every $j \geq i$. By Proposition 14, there exist $x \in \overline{T}_{\text{count}}^*$, $y \in \overline{T}_{\text{init}}^*$, $z \in \overline{T}_{\text{simul}}^*$, and configurations C and D such that $\sigma_0 \xrightarrow{x} C \xrightarrow{y} D \xrightarrow{z} \sigma_i$. By Proposition 15, the following holds for every $j \in [\ell]$:

$$\llbracket C \rrbracket \cap (X \times \{j\}) \neq \emptyset \iff |\mathbf{v}| \geq j. \quad (1)$$

Let us now show that $\overline{O}(\sigma)$ is as expected, by making a case distinction on whether $|\mathbf{v}| \geq \ell$.

Case $|\mathbf{v}| < \ell$. By (1), we have $C(x, \ell) = 0$ for every $x \in X$. Thus, we have $y = z = \varepsilon$ since no transition of $\overline{T}_{\text{init}} \cup \overline{T}_{\text{simul}}$ is enabled in C . This implies that $C = \sigma_i = \sigma_{i+1} = \dots \in X \times [\ell]$. Hence, $\overline{O}(\sigma) = \overline{O}(C) = 1$ which is the expected output.

Case $|\mathbf{v}| \geq \ell$. By (1), $C(x, \ell) > 0$ for some $x \in X$. Thus, fairness enforces sequence y to convert every agent from states $X \times [\ell]$ to states $\overline{Q}^{(2)}$. Thus, we have $D \geq \{h_1, h_2, \dots, h_\ell\}$ by (1), which implies $D \in (L' \ominus L) + C_{\mathbf{v}}$ for some $L' \succeq L$, and consequently $\sigma_i, \sigma_{i+1}, \dots \in \mathbb{N}^{\overline{Q}^{(2)}}$.

Let $m \stackrel{\text{def}}{=} |z|$ and let D_0, D_1, \dots, D_m be the configurations such that $D = D_0 \rightarrow D_1 \rightarrow \dots \rightarrow D_m = \sigma_i$. Let $\pi \stackrel{\text{def}}{=} D_0 D_1 \dots D_m \cdot \sigma$. By fairness of σ and by Proposition 16, $\widehat{\pi}$ is a fair execution of \mathcal{P} , which implies that $O(\widehat{\pi}) = \varphi(\mathbf{v})$. Therefore, we have $\overline{O}(\pi) = \varphi(\mathbf{v})$ by definition of \overline{O} . Since σ and π share a common (infinite) suffix, we have $\overline{O}(\sigma) = \overline{O}(\pi)$, which completes the proof. ◀

B.2 Proof of Proposition 5

► **Proposition 17.** *Assume that every finite set A of atomic predicates is computed by some $|A|$ -way multi-output protocol with $\mathcal{O}(|A|^3)$ helpers and states, and $\mathcal{O}(|A|^5)$ transitions. Every QFPA predicate φ is computed by some simple $|\varphi|$ -way protocol with $\mathcal{O}(|\varphi|^3)$ helpers and states, and $\mathcal{O}(|\varphi|^5)$ transitions.*

Proof. Let $\text{atomic}(P)$ be the set of atomic predicates in P . Consider a forest of binary trees of boolean operations encoding φ (negations have only one child), with atomic predicates at the leaves. There are at most $\text{len}(\varphi) + \text{size}(\varphi)$ nodes in that forest (roots correspond to different predicates of φ). Consider the set P' made of every predicate corresponding to nodes of the forest. We call such P' a *full* set of predicates. We have $\text{size}(P') \leq \text{len}(\varphi) + \text{size}(\varphi) \leq |\varphi|$, $\|P'\| = \|\varphi\|$ and $\text{len}(P') \leq \text{len}(\varphi)^2 \leq |\varphi|^2$. We prove by induction on $\text{len}(P')$ that every *full* P' is computed by some multi-output population protocol with $\mathcal{O}(\text{len}(P') + |\text{atomic}(P')|^5)$ helpers, states and transitions.

If $\text{len}(P') = 0$, then each predicate is atomic, and the claim is true by hypothesis.

Let P' be a full set with $\text{len}(P') = k > 0$, and assume that the claim holds for every full set P'' with $\text{len}(P'') < k$. Let $\varphi \in P'$ with $\text{len}(\varphi)$ maximal. Let us consider the case where $\varphi = \psi \wedge \psi'$ for some predicates ψ, ψ' . The case of disjunction and negation are handled similarly. Let $P'' \stackrel{\text{def}}{=} P' \setminus \{\varphi\}$. Note that $\text{len}(P'') < \text{len}(P')$, and that P'' is full because $\text{len}(\varphi)$ is maximal. Thus, by induction hypothesis, we obtain a simple multi-output population protocol $\mathcal{P}'' = (Q, T, L, X, I, O)$ that computes P'' . Assume w.l.o.g. that the indices of O associated to ψ and ψ' are $|P|$ and $|P| + 1$ respectively. Let $q_0, q_1, r_0, r_1 \in Q$ be the unique states such that $O_{|P|}(q_b) = b$ and $O_{|P|+1}(r_b) = b$ for $b \in \{0, 1\}$. These states exist since \mathcal{P}'' is simple. Let $\mathcal{P}' = (Q', T', L', X', I', O')$ be the multi-output protocol such that:

$$\begin{aligned} Q' &\stackrel{\text{def}}{=} Q \cup \{o_0, o_1\}, \\ T' &\stackrel{\text{def}}{=} \{(\downarrow q_a, r_b, o_{\neg c}, \downarrow q_a, r_b, o_c) : a, b, c \in \{0, 1\}, a \wedge b = c\}, \\ L' &\stackrel{\text{def}}{=} L + \downarrow o_0 \downarrow, \\ I' &\stackrel{\text{def}}{=} I, \\ O'_i &\stackrel{\text{def}}{=} \begin{cases} O_i & \text{for every } 1 \leq i < |P|, \\ q \mapsto (b \text{ if } q = o_b \text{ else } \perp) & \text{for } i = |P|. \end{cases} \end{aligned}$$

We claim that \mathcal{P}' computes P' . Note that \mathcal{P}' behaves exactly as \mathcal{P} on Q . This implies that \mathcal{P}' computes each predicate of $P' \setminus \{\varphi\}$. Thus, it suffices to show that it also computes φ . Let σ be a fair execution of \mathcal{P}' starting from some initial configuration C_v . Since \mathcal{P} is simple and computes both ψ and ψ' , there exists $i \in \mathbb{N}$ such that for every $j \geq i$:

$$\sigma_j(q_{\psi(v)}) > 0, \sigma_j(r_{\psi'(v)}) > 0 \text{ and } \sigma_j(q_{\neg\psi(v)}) = \sigma_j(r_{\neg\psi'(v)}) = 0.$$

Thus, by fairness, there exists $i' \geq i$ such that $\sigma_j(o_{\psi(v) \wedge \psi'(v)}) > 0$ and $\sigma_j(o_{\neg(\psi(v) \wedge \psi'(v))}) = 0$ for every $j \geq i'$. This implies that $O_{|P|}(\sigma) = \psi(v) \wedge \psi'(v) = \varphi(v)$.

Concerning the number of states and helpers, the protocol \mathcal{P}' uses two states plus the states of \mathcal{P}'' , and one helper plus the helpers of \mathcal{P}'' , which ends the proof by induction as $\text{atomic}(P'') = \text{atomic}(P') = \text{atomic}(P)$.

In terms of $|\varphi|$, we obtain a protocol with $\mathcal{O}(\text{len}(\varphi) + |\text{atomic}(\varphi)|^5) = \mathcal{O}(|\varphi|^5)$ helpers, states and transitions. ◀

B.3 Proof of Theorem 6

► **Theorem 6.** *Assume that for every atomic predicate φ , there exists a $|\varphi|$ -way RDI-protocol with $\mathcal{O}(|\varphi|)$ helpers, $\mathcal{O}(|\varphi|^2)$ states and $\mathcal{O}(|\varphi|^3)$ transitions that computes φ . For every finite set P of atomic predicates, there exists a $|P|$ -way simple multi-output protocol, with $\mathcal{O}(|P|^3)$ helpers and states, and $\mathcal{O}(|P|^5)$ transitions, that computes P .*

Let $P = \{\varphi_1, \varphi_2, \dots, \varphi_k\}$. For every $i \in [k]$, let $\mathcal{P}_i = (Q_i, T_{\infty i}, T_{\dagger i}, L_i, \bar{X} \cup \underline{X}, I_i, O_i)$ be the simple RDI-protocol with helpers computing $\tilde{\varphi}_i$. Recall that each \mathcal{P}_i has two input variables \bar{x} and \underline{x} for each input variable $x \in X$. Recall further that the transitions of $T_{\dagger i}$ are called RDI-transitions.

We first define a simple multi-output protocol \mathcal{P} . Then we introduce some auxiliary definitions and propositions, and finally we prove that \mathcal{P} computes P .

Notations. For every RDI-transition $t = (\mathbf{p}, \mathbf{q})$ and for every $x \in X$, let t^x be the transition defined as $t^x \stackrel{\text{def}}{=} (\mathbf{p} + \mathcal{I}x, \mathbf{q} + \mathcal{I}x)$. In other words, t^x has the same effect as t , but is “guarded” by X , i.e., it can only occur if some agent is in state x . We say that t^x is a *guarded transition*. Given a set U of transitions, we define the sets U^g of guarded transitions and U^{-1} of guarded reversal transitions as:

$$U^g \stackrel{\text{def}}{=} \{t^x : t \in U, x \in X\} \text{ and } U^{-1} \stackrel{\text{def}}{=} \{(\mathbf{q}, \mathbf{p}) : (\mathbf{p}, \mathbf{q}) \in U^g\}.$$

The protocol. The k -output population protocol with helpers $\mathcal{P} = (Q, T, L, X, I, O)$ is defined as follows:

- $Q \stackrel{\text{def}}{=} X \cup H \cup Q_1 \cup Q_2 \cup \dots \cup Q_k$, where $H \stackrel{\text{def}}{=} \{h_x : x \in X\}$.
Intuitively, X are the input states, H are auxiliary states used to distribute agents to the atomic protocols, and Q_1, \dots, Q_k are the states of the atomic protocols themselves.
- $T \stackrel{\text{def}}{=} S \cup S^{-1} \cup (T_{\infty 1} \cup T_{\infty 2} \cup \dots \cup T_{\infty k}) \cup (T_{\dagger 1} \cup T_{\dagger 2} \cup \dots \cup T_{\dagger k})^g$, where
 $S \stackrel{\text{def}}{=} \{\bar{s}_x, \underline{s}_x : x \in X\}$ and

$$\begin{aligned} \bar{s}_x &\stackrel{\text{def}}{=} \mathcal{I}k \cdot x \mathcal{I} \mapsto \mathcal{I}I_i(\bar{x}) : i \in [k] \mathcal{I}, \\ \underline{s}_x &\stackrel{\text{def}}{=} \mathcal{I}x, (k-1) \cdot h \mathcal{I} \mapsto \mathcal{I}I_i(\underline{x}) : i \in [k] \mathcal{I}. \end{aligned}$$

Intuitively, the transitions of S allow \mathcal{P} to distribute agents to $\mathcal{P}_1, \dots, \mathcal{P}_k$. Transition \bar{s}_x collects k agents from the input state of \mathcal{P} for x , and sends one agent to each of the input states of $\mathcal{P}_1, \dots, \mathcal{P}_k$ for \bar{x} . Similarly, \underline{s}_x collects one agent from x and $(k-1)$ helpers, and sends one agent to each of the input places of $\mathcal{P}_1, \dots, \mathcal{P}_k$ for \underline{x} .

Transitions of S^{-1} allow \mathcal{P} to collect agents back if they were not distributed properly. They are guarded to ensure that the agents are not collected when the distribution is correct.

The rest of the transitions are the transitions of $\mathcal{P}_1, \dots, \mathcal{P}_k$, with an additional guard on the transitions of $T_{\dagger 1}, \dots, T_{\dagger k}$. The guards ensure that $\mathcal{P}_1, \dots, \mathcal{P}_k$ stop returning agents to the input states once the correct distribution is achieved.

- $L \stackrel{\text{def}}{=} \mathcal{I}(k-1)^2 \cdot h_x : x \in X \mathcal{I} + L_1 + L_2 + \dots + L_k$.
The helpers of \mathcal{P} are those of $\mathcal{P}_1, \dots, \mathcal{P}_k$, plus $(k-1)^2$ helpers for each input variable.
- $I \stackrel{\text{def}}{=} x \mapsto x$.
- The output mapping for φ_i is given by $O(i, q) \stackrel{\text{def}}{=} (O_i(q) \text{ if } q \in Q_i \text{ else } \perp)$.

Auxiliary definitions and propositions.

- For every $i \in [k]$ and every configuration $C \in \mathbb{N}^Q$ of \mathcal{P} , let $C^i \in \mathbb{N}^{Q_i}$ be the configuration of \mathcal{P}_i such that $C^i(q) \stackrel{\text{def}}{=} C(q)$ for every $q \in Q_i$.
- For every $i \in [k]$ and every sequence $w \in T^*$, let w^i be the projection of w onto the transitions of $S \cup S^{-1} \cup T_{\infty i} \cup (T_{\dagger i})^g$.
- For every $w \in T^*$, let $\mathbf{w} \in \mathbb{Z}^{\overline{X} \cup \underline{X}}$ be the vector such that for every $x \in X$:

$$\mathbf{w}(\overline{x}) \stackrel{\text{def}}{=} |w|_{\overline{s}_x} - \sum_{t \in \{\overline{s}_x\}^g} |w|_t, \quad \mathbf{w}(\underline{x}) \stackrel{\text{def}}{=} |w|_{\underline{s}_x} - \sum_{t \in \{\underline{s}_x\}^g} |w|_t.$$

In other words, \mathbf{w} records the difference between the number of occurrences of transition \overline{s}_x and its guarded reversals, for each variable x , and similarly for \underline{s}_x .

- Observe that the set of input variables of \mathcal{P} is X , while the set of input variables of \mathcal{P}_i is $\overline{X} \cup \underline{X}$. Given $\mathbf{v} \in \mathbb{N}^X$ and $\mathbf{w} \in \mathbb{N}^{(\overline{X} \cup \underline{X})}$, we let $\mathbf{v} \equiv \mathbf{w}$ denote that $\mathbf{v}(x) = k \cdot \mathbf{w}(\overline{x}) + \mathbf{w}(\underline{x})$ for every $x \in X$.

Let us prove the following observations on the executions of \mathcal{P} :

► **Proposition 18.** *Let $\mathbf{v} \in \mathbb{N}^X$, $\hat{L} \succeq L$ and $\hat{C}_{\mathbf{v}} \stackrel{\text{def}}{=} \hat{L} + \{\mathbf{v}(x) \cdot I(x) : x \in X\}$. Let $\hat{C}_{\mathbf{v}} \xrightarrow{w} C$ be a finite execution of \mathcal{P} . We have:*

1. *There exists an initialization sequence from $\hat{C}_{\mathbf{v}}^i$ to C^i in \mathcal{P}_i with effective input $\mathbf{w} \geq \mathbf{0}$.*
2. *If $C(X) = 0$, then $\mathbf{v} \equiv \mathbf{w}$.*
3. *There exists a configuration D such that $C \xrightarrow{*} D$ and $D(X) = 0$.*

Proof.

1. Let $i \in [k]$. The only transitions that change the number of agents over the states of Q_i are those of $S \cup S^{-1} \cup T_{\infty i} \cup (T_{\dagger i})^g$. Transitions $S \cup S^{-1}$ have the same effect as the transitions of In and Out. Transitions $T_{\infty i} \cup T_{\dagger i}$ form precisely the set of transitions of \mathcal{P}_i , and the effects of the transitions of $T_{\dagger i}$ and $(T_{\dagger i})^g$ coincide. Moreover, we have $\hat{C}_{\mathbf{v}}^i = \hat{L}^i \succeq L^i = L_i$. Therefore, w_i yields an initialization sequence of \mathcal{P}^i from $\hat{C}_{\mathbf{v}}^i$ to C^i with effective input \mathbf{w} . Since \mathcal{P}^i is an RDI-protocol, $C^i(I_i(x)) \leq \mathbf{w}(x)$ holds for every $x \in \overline{X} \cup \underline{X}$. Hence, we must have $\mathbf{w} \geq \mathbf{0}$ as a configuration cannot hold any negative amount of agents.
2. An induction on $|w|$ shows that $C(x) = \mathbf{v}(x) - k \cdot \mathbf{w}(\overline{x}) - \mathbf{w}(\underline{x})$ for every $x \in X$. Thus, if $C(X) = 0$, then $\mathbf{v}(x) = k \cdot \mathbf{w}(\overline{x}) + \mathbf{w}(\underline{x})$ for every $x \in X$. Hence, since $\mathbf{w} \geq \mathbf{0}$ by (1), we have $\mathbf{v} \equiv \mathbf{w}$.
3. For every configuration A of \mathcal{P} , let $[A] \stackrel{\text{def}}{=} \{B : B^i \in [A^i]\}$ for every $i \in [k]\}$. Note that:

$$A(X) = B(X) \text{ for every } B \in [A]. \quad (2)$$

Let $C_j \xrightarrow{t_j} C_{j+1}$ be j^{th} step of $\hat{C}_{\mathbf{v}} \xrightarrow{w} C$. For every $D_{j+1} \in [C_{j+1}]$, we construct a sequence $w_j \in T^*$ such that $D_{j+1} \xrightarrow{w_j} D_j$ for some $D_j \in [C_j]$. In other words, we show how to reverse t_j , up to a possible redistribution of the output agents. The validity of the main claim follows by (2) and a straightforward induction. We may assume without loss of generality that $C_{j+1}(X) > 0$, as otherwise the main claim would already be satisfied. Since $C_{j+1}(X) > 0$, guarded transitions of \mathcal{P} are equivalent to their unguarded counterparts, *i.e.* a transition u is enabled at C_{j+1} if and only if u^g is enabled at C_{j+1} . Thus, we may reverse t_j as follows:

- If $t_j \in S$, then we pick $w_j \in S^{-1}$ as the guarded reversal of t_j ;

- If $t_j \in S^{-1}$, then we pick $w_j \in S$ as the counterpart transition of t_j ;
- If $t_j \in T_{\infty i} \cup T_{\dagger i}$ for some $i \in [k]$, then we proceed as follows. By (1), there is an initialization sequence from \hat{C}_v^i to C^i in \mathcal{P}_i with effective input \mathbf{w} . Moreover,

$$C^i \xrightarrow{*} C_j^i \xrightarrow{t_j} C_{j+1}^i \text{ in } \mathcal{P}_i.$$

Hence, since \mathcal{P}_i is an RDI-protocol, there exists $w_j \in (T_{\infty i} \cup T_{\dagger i})^*$ such that $D_{j+1}^i \xrightarrow{w_j} E$ in \mathcal{P}_i for some $E \in [C_j^i]$. Thus, we have $D_{j+1} \xrightarrow{w_j} D_j$ in \mathcal{P} for some $D_j \in [C_j]$. ◀

Main proof. We proceed to prove that \mathcal{P} indeed computes $\{\varphi_1, \varphi_2, \dots, \varphi_k\}$.

Proof of Theorem 6. Let $\mathbf{v} \in \mathbb{N}^X$, $\hat{L} \succeq L$, and let σ be a fair execution of \mathcal{P} starting from $\hat{C}_v \stackrel{\text{def}}{=} \hat{L} + \{\mathbf{v}(x) \cdot I(x) : x \in X\}$. By Proposition 18 (3) and by fairness, there exists $j \in \mathbb{N}$ such that $\sigma_j(X) = 0$. By definition of T , if X is emptied, then it remains permanently emptied, as none of the guarded reversals can be fired. Thus, we have:

$$\sigma_j(X) = \sigma_{j+1}(X) = \dots = 0. \quad (3)$$

Let $\hat{\sigma} \stackrel{\text{def}}{=} \sigma_j^i \sigma_{j+1}^i \dots$. Consider protocol \mathcal{P}_i for some i , and let \mathbf{w} be the effective input of the initialization sequence $\hat{\sigma}^i$ of \mathcal{P}_i . By (3), $\hat{\sigma}^i$ only contains transitions of $T_{\infty i}$, and is consequently a fair execution of protocol $\mathcal{P}_{\infty i}$. By hypothesis, and by definition of RDI-protocols, $\mathcal{P}_{\infty i}$ computes $\tilde{\varphi}_i$. Hence, we have $O_i(\hat{\sigma}^i) = \tilde{\varphi}_i(\mathbf{w})$. We are done since, by Proposition 18 (2), we have $\mathbf{v} \equiv \mathbf{w}$, which implies $\varphi_i(\mathbf{v}) = \tilde{\varphi}_i(\mathbf{w})$. ◀

B.4 Proof of Theorem 7

► **Theorem 7.** *Every atomic predicate φ over variables X can be computed by a simple $|\varphi|$ -way population protocol with reversible dynamic initialization that has $\mathcal{O}(|\varphi|)$ helpers, $\mathcal{O}(|\varphi|^2)$ states, and $\mathcal{O}(|\varphi|^3)$ transitions.*

In Section B.4.1 we describe the protocol for threshold predicates, and prove its correctness. Section B.4.2 does the same for remainder predicates.

B.4.1 Threshold protocols

Let us fix a threshold predicate φ over variables X . Without loss of generality³, we have $\varphi(\mathbf{v}) = \mathbf{a} \cdot \mathbf{v} \geq b$ where $\mathbf{a} \in \mathbb{Z}^X$ and $b > 0$. We construct a simple population protocol \mathcal{P}_{thr} that computes φ under reversible dynamic initialization, and prove its correctness.

Notations. Let n be the smallest number such that $2^n > \|\varphi\|$. Let $P \stackrel{\text{def}}{=} \{+2^i, -2^i : 0 \leq i \leq n\}$, $Z \stackrel{\text{def}}{=} \{0\}$, $N \stackrel{\text{def}}{=} P \cup Z$ and $B \stackrel{\text{def}}{=} \{\mathbf{f}, \mathbf{t}\}$, where P , Z , N and B respectively stand for “Powers of two”, “Zero”, “Numerical values” and “Boolean values”. For every set S and every $x \in X$, let $S_x \stackrel{\text{def}}{=} \{q_x : q \in S\}$ and $S_X \stackrel{\text{def}}{=} S \cup \bigcup_{x \in X} S_x$.

For every $d \in \mathbb{N}$, let $\text{bits}(d)$ denote the unique set $J \subseteq \mathbb{N}$ such that $d = \sum_{j \in J} 2^j$, e.g. $\text{bits}(13) = \text{bits}(1101_2) = \{3, 2, 0\}$. The *canonical representation* of an integer $d \in \mathbb{Z}$ is the multiset $\text{rep}(d)$ defined as follows:

³ If $b \leq 0$, then we can instead consider the equivalent predicate $\neg(-\mathbf{a} \cdot \mathbf{v} \geq -b + 1)$, construct a protocol for $-\mathbf{a} \cdot \mathbf{v} \geq -b + 1$ and handle the negation separately in Section 5.2.

$$\text{rep}(d) \stackrel{\text{def}}{=} \begin{cases} \wr +2^i : i \in \text{bits}(d) \wr & \text{if } d > 0, \\ \wr -2^i : i \in \text{bits}(|d|) \wr & \text{if } d < 0, \\ \wr 0 \wr & \text{if } d = 0. \end{cases}$$

The protocol. The RDI-protocol $\mathcal{P}_{\text{thr}} = (Q, T_\infty, T_\dagger, L, X, I, O)$ is defined as follows:

- $Q \stackrel{\text{def}}{=} X \cup N_X \cup B$.
Intuitively, the states of X are the “ports” through which the agents for each variable enter and exit the protocol.
- $I \stackrel{\text{def}}{=} x \mapsto \mathbf{x}$.
That is, the initial state for variable x is x .
- $L \stackrel{\text{def}}{=} \wr 2n \cdot 0, \mathbf{f} \wr$.
So, we have $2n$ helpers in state 0, and one helper in state \mathbf{f} , i.e., initially the protocol assumes that the predicate does not hold.
- $O(q) \stackrel{\text{def}}{=} q \mapsto (0 \text{ if } q = \mathbf{f} \text{ else } 1 \text{ if } q = \mathbf{t} \text{ else } \perp)$.
That is, the output of the protocol is completely determined by the number of agents in states \mathbf{t} and \mathbf{f}
- T_∞ is the following set of (“permanent”) transitions:

$$\begin{aligned} \text{add}_x : \wr x, |\text{rep}(\mathbf{a}(x))| \cdot 0 \wr &\mapsto \wr 0_x \wr + \text{rep}(\mathbf{a}(x)) && \text{for all } x \in X, \\ \text{up}_i^\circ : \wr \circ 2^i, \circ 2^i \wr &\mapsto \wr \circ 2^{i+1}, 0 \wr && \text{for all } 0 \leq i < n \text{ and } \circ \in \{+, -\}, \\ \text{down}_i^\circ : \wr \circ 2^i, 0 \wr &\mapsto \wr \circ 2^{i-1}, \circ 2^{i-1} \wr && \text{for all } 0 < i \leq n \text{ and } \circ \in \{+, -\}, \\ \text{cancel}_{i,q} : \wr +2^i, -2^i, q \wr &\mapsto \wr 0, 0, \mathbf{f} \wr && \text{for all } 0 \leq i \leq n \text{ and } q \in B, \\ \text{swap}_{p,q}^x : \wr p, q_x \wr &\mapsto \wr p_x, q \wr && \text{for all } p, q \in N \text{ and } x \in X, \\ \text{equal} : \text{rep}(b) + \wr \mathbf{f} \wr &\mapsto \text{rep}(b) + \wr \mathbf{t} \wr, \\ \text{false} : \wr \mathbf{f}, \mathbf{t} \wr &\mapsto \wr \mathbf{f}, \mathbf{f} \wr. \end{aligned}$$

Intuitively, add_x converts an agent which arrived via port x into the canonical representation of $\mathbf{a}(x)$. Transitions of the form up_i° , down_i° and $\text{cancel}_{i,q}$ allow the protocol to change the representation of a value, without changing the value itself. Transition equal allows the protocol to detect that the current value of $\mathbf{a} \cdot \mathbf{x}$, for the current input \mathbf{x} , is at least b , which moves a helper from state \mathbf{f} to \mathbf{t} .

- Finally, T_\dagger is the following set of RDI-transitions:

$$\begin{aligned} \text{add}_{x,q}^{-1} : \wr 0_x, q \wr + \text{rep}(\mathbf{a}(x)) &\mapsto \wr x, \mathbf{f}, |\text{rep}(\mathbf{a}(x))| \cdot 0 \wr && \text{for all } x \in X \text{ and } q \in B, \\ \text{cancel}_{i,q}^{-1} : \wr 0, 0, q \wr &\mapsto \wr +2^i, -2^i, \mathbf{f} \wr && \text{for all } 0 \leq i \leq n \text{ and } q \in B, \\ \text{reset} : \wr \mathbf{t} \wr &\mapsto \wr \mathbf{f} \wr. \end{aligned}$$

The first two transitions are needed to reverse the changes of add and cancel transitions while the dynamic initialization is not finished. Both types of transitions reset the output of the protocol by leaving an agent in the default output state \mathbf{f} . The reset transition resets the output by moving agents from \mathbf{t} to \mathbf{f} .

Let $\mathcal{P}_\infty = (Q, T_\infty, L, X, I, O)$. Let $T \stackrel{\text{def}}{=} T_\infty \cup T_\dagger$. For the sake of readability, we will sometimes omit the subscripts and superscripts from transitions names when they are irrelevant, e.g. “a swap transition is enabled” instead of “there exist $p, q \in N$ and $x \in X$ such that $\text{swap}_{p,q}^x$ is enabled”.

Size. Note that \mathcal{P}_{thr} has $|Q| = |X| + |N_X| + |B| = |X| + (2n+3) \cdot (|X|+1) + 2 \in \mathcal{O}(\log\|\varphi\| \cdot |X|)$ states and $|L| = 2n+1 \in \mathcal{O}(\log\|\varphi\|)$ helpers. Moreover, since families of transitions are parameterized by X , B , N or N^2 , and $\{+, -\}$, there are $\mathcal{O}(|N|^2 \cdot |X|) = \mathcal{O}(\log^2\|\varphi\| \cdot |X|) \subseteq \mathcal{O}(|\varphi|^3)$ transitions. Finally, each transition uses at most $\mathcal{O}(\text{rep}(\|\varphi\|)) = \mathcal{O}(\log\|\varphi\|) \subseteq \mathcal{O}(|\varphi|)$ states.

Auxiliary definitions and observations. Before proving that \mathcal{P}_{thr} works as intended, let us first introduce auxiliary definitions. Let $\text{val}: Q \rightarrow \mathbb{N}$ be the function that associates a value to each state as follows:

$$\begin{aligned} \text{val}(0) &= \text{val}(\mathbf{f}) = \text{val}(\mathbf{t}) \stackrel{\text{def}}{=} 0, \\ \text{val}(x) &\stackrel{\text{def}}{=} \mathbf{a}(x) && \text{for every } x \in X, \\ \text{val}(\circ 2^i) &\stackrel{\text{def}}{=} \circ 2^i && \text{for every } 0 \leq i \leq n \text{ and } \circ \in \{+, -\}, \\ \text{val}(q_x) &\stackrel{\text{def}}{=} \text{val}(q) && \text{for every } q \in N \text{ and } x \in X. \end{aligned}$$

So, for example, for the predicate $3x - 4y \geq 2$ we have $\text{val}(x) = 3$ and $\text{val}(y) = -4$. For every configuration C and every set of states $S \subseteq Q$, let

$$\text{val}_S(C) \stackrel{\text{def}}{=} \sum_{q \in S} \text{val}(q) \cdot C(q).$$

In particular, let $\text{val}(C) \stackrel{\text{def}}{=} \text{val}_Q(C)$. Intuitively, C can be seen as an encoding of the value $\text{val}(C)$. The following properties, relating values and configurations, can be derived from the above definitions:

► **Proposition 19.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, the following holds:*

1. $\text{val}(C) = \mathbf{a} \cdot \mathbf{w}$,
2. $|C| = C(N) + C(B) + |\mathbf{w}|$,
3. $C(N) \geq L(N)$ and $C(B) \geq L(B)$,
4. $C(N_x) + C(x) = \mathbf{w}(x)$ for every $x \in X$.

In particular, (2) states that the number of agents is always equal to the number of helpers plus the *net* amount of agents that dynamically entered the population.

Auxiliary propositions. We say that a configuration C is *clean* if the following holds for every $p, q \in P_X$:

- If $\text{val}(p) + \text{val}(q) = 0$, then $C(p) = 0$ or $C(q) = 0$.
For example, a configuration with agents in $+2^i_x$ and -2^i_y is not clean, since $\text{val}(+2^i_x) + \text{val}(-2^i_y) = 0$. Intuitively, no pair of agents can cancel in a clean configuration.
- If $\text{val}(p) = \text{val}(q)$ and $\text{val}(p) \notin \{-2^n, +2^n\}$, then $C(\{p, q\}) \leq 1$.
For example, a configuration with two agents in $+2^i_x$, where $i < n$, is not clean. Intuitively, in a clean configuration no agent can be promoted to a higher power of 2.

We show that any configuration can be cleaned using only permanent transitions. This implies that once the dynamic initialization has terminated, every fair execution visits clean configurations infinitely often.

► **Proposition 20.** *For every initialization sequence π such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, there exists a clean configuration D such that $C \xrightarrow{T_\infty^*} D$.*

Proof. If C is clean, then we pick $D \stackrel{\text{def}}{=} C$. Otherwise, at least one of the following holds:

- (a) $C(p) > 0$, $C(q) > 0$ and $\text{val}(p) + \text{val}(q) = 0$ for some $p, q \in P_X$;
- (b) $C(\{p, q\}) \geq 2$ for some $p, q \in P_X$ such that $\text{val}(p), \text{val}(q) \notin \{-2^n, +2^n\}$.

We claim there exists a configuration C' such that $C \xrightarrow{T_\infty^*} C'$ and $C(P_X) > C'(P_X)$. Let us show that if the claim is true then the result holds. If C' is clean, then we are done. Otherwise, this process is repeated until a clean configuration D has been reached. The process terminates as the number of agents in P_X cannot become negative.

Let us now prove the claim. Suppose (a) holds. By Proposition 19 (3), we have $C(N) \geq L(N) \geq 2$ and hence it is possible to consecutively fire at least two **swap** transitions. Note that they do not change the amount of agents in P_X . For this reason, we may assume without loss of generality that $p = +2^i$ and $q = -2^i$ for some $0 \leq i \leq n$. By Proposition 19 (3), we have $C(B) \geq L(B) > 0$. Thus, there exists $r \in B$ such that $C(r) > 0$. Therefore, firing transition $\text{cancel}_{i,r}$ decreases $C(P_X)$ by two.

Similarly, if case (b) holds, then we may assume without loss of generality that $C(o2^i) \geq 2$ for some $0 \leq i < n$ and $o \in \{+, -\}$. Thus, firing transition up_i^o decreases $C(P_X)$ by one. ◀

We now bound the number of agents in states from $X \cup P_X$ in a clean configuration.

► **Proposition 21.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, if C is clean, then $C(X) + C(P_X) \leq |\mathbf{w}| + n$.*

Proof. Let $S_{o2^n} \stackrel{\text{def}}{=} \{q \in P_X : \text{val}(q) = o2^n\}$ for both $o \in \{+, -\}$. Since C is clean, we have $C(S_{o2^n}) = 0$ for some $o \in \{+, -\}$. Let us consider the case where $o = -$. The other case is proven analogously.

Let $\mathbf{u} \in \mathbb{N}^X$ be such that $\mathbf{u}(x) \stackrel{\text{def}}{=} C(x)$ for every $x \in X$. Note that $|\mathbf{u}| = C(X)$, and that $\mathbf{u} \leq \mathbf{w}$ by Proposition 19 (4). Since C is clean, we have $C(P_X \setminus S_{+2^n}) \leq n$. Thus, it suffices to show that $C(S_{+2^n}) \leq |\mathbf{w}| - |\mathbf{u}|$. Suppose this is not the case. This yields a contradiction:

$$\begin{aligned}
 \text{val}_{P_X}(C) &> 2^n \cdot C(S_{+2^n}) - 2^n && \text{(since } C \text{ is clean)} \\
 &\geq 2^n \cdot (|\mathbf{w}| - |\mathbf{u}| + 1) - 2^n && \text{(by assumption)} \\
 &= 2^n \cdot (|\mathbf{w}| - |\mathbf{u}|) \\
 &\geq \mathbf{a} \cdot (\mathbf{w} - \mathbf{u}) && \text{(since } 2^n > \|\mathbf{a}\| \text{ and } \mathbf{w} \geq \mathbf{u}) \\
 &= \mathbf{a} \cdot \mathbf{w} - \mathbf{a} \cdot \mathbf{u} \\
 &= \text{val}(C) - \text{val}_X(C) && \text{(by Prop. 19 (1) and def. of } \mathbf{u}) \\
 &= \text{val}_{P_X}(C) && \text{(by def. of val)} \quad \blacktriangleleft
 \end{aligned}$$

The following corollary shows that the number of agents in state 0 can always be increased back to at least n . This will later be useful in arguing that the number of agents in X can eventually be decreased to zero.

► **Corollary 22.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, there exists a clean configuration D such that $C \xrightarrow{T_\infty^*} D$ and $D(0) \geq n$.*

Proof. By Proposition 20, there exists a clean configuration C' such that $C \xrightarrow{\pi' \in T_\infty^*} C'$. Let us first prove that $C'(Z_X) \geq n$. Note that $\pi\pi'$ is an initialization sequence with effective

input \mathbf{w} such that $L' \xrightarrow{\pi\pi'} C'$. Thus:

$$\begin{aligned}
C'(Z_X) &= |C'| - C'(X) - C'(P_X) - C'(B) && \text{(by def. of } Q) \\
&= (C'(N) + C'(B) + |\mathbf{w}|) - C'(X) - C'(P_X) - C'(B) && \text{(by Prop. 19 (2))} \\
&\geq (L(N) + C'(B) + |\mathbf{w}|) - C'(X) - C'(P_X) - C'(B) && \text{(by Prop. 19 (3))} \\
&\geq (L(N) + C'(B) + |\mathbf{w}|) - (|\mathbf{w}| + n) - C'(B) && \text{(by Prop. 21)} \\
&= L(N) - n \\
&\geq n && \text{(by def. of } L)
\end{aligned}$$

Now, by Proposition 19 (3), we have $C'(N) \geq L(N) \geq n$. Thus, using **swap** transitions, we can swap n agents from Z_X to 0. This way, we obtain a configuration D such that $C' \xrightarrow{T_\infty^*} D$ and $D(0) \geq n$. We are done since **swap** transitions preserve cleanness. \blacktriangleleft

For every configuration C , let

$$\text{val}^+(C) \stackrel{\text{def}}{=} \sum_{\substack{q \in P_X \\ \text{val}(q) > 0}} \text{val}(q) \cdot C(q), \quad \text{val}^-(C) \stackrel{\text{def}}{=} \sum_{\substack{q \in P_X \\ \text{val}(q) < 0}} \text{val}(q) \cdot C(q).$$

We now show that, once dynamic initialization has terminated, fair executions stabilize to configurations of a certain “normal form”.

► Proposition 23. *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \geq L$ and for every fair execution σ of \mathcal{P}_∞ starting from C , there exist $i \in \mathbb{N}$, $m_+ \geq 0$ and $m_- \leq 0$ such that:*

1. $\sigma_i(X) = \sigma_{i+1}(X) = \dots = 0$,
2. $\text{val}^\circ(\sigma_i) = \text{val}^\circ(\sigma_{i+1}) = \dots = m_\circ$ for both $\circ \in \{+, -\}$,
3. $m_+ = 0 \vee m_- = 0$.

Proof. For the sake of contradiction, assume there exist infinitely many indices j such that $\sigma_j(X) > 0$. Let $j \in \mathbb{N}$ be such an index. By Corollary 22, there exists a configuration C_j such that $\sigma_j \xrightarrow{T_\infty^*} C_j$ and $C_j(0) \geq n$. Hence, there exists $x \in X$ such that transition add_x is enabled in C_j . Since this holds for infinitely many indices and since X is finite, fairness implies that some **add** transition can be enabled infinitely often and hence occurs infinitely often along σ . This is impossible since the number of agents in X cannot be increased by any transition in T_∞ , and thus would eventually drop below zero. Therefore, there exists $h \in \mathbb{N}$ such that $\sigma_h(X) = \sigma_{h+1}(X) = \dots = 0$.

Since X is permanently empty from index h , the **add** transitions are permanently disabled. No other transition in T_∞ can increase the absolute value of val° for any $\circ \in \{+, -\}$. Thus, we have $|\text{val}^\circ(\sigma_h)| \geq |\text{val}^\circ(\sigma_{h+1})| \geq \dots$ for both $\circ \in \{+, -\}$. Therefore, there exist $i \geq h$, $m_+ \geq 0$ and $m_- \leq 0$ such that

$$\text{val}^+(\sigma_i) = \text{val}^+(\sigma_{i+1}) = \dots = m_+, \quad (4)$$

$$\text{val}^-(\sigma_i) = \text{val}^-(\sigma_{i+1}) = \dots = m_-. \quad (5)$$

It remains to show that $m_+ = 0$ or $m_- = 0$. For the sake of contradiction, suppose this is not the case. For every $j \geq i$, Corollary 22 yields a configuration C_j such that $\sigma_j \xrightarrow{T_\infty^*} C_j$ and $C_j(0) \geq n$. Thus, by fairness, there exist infinitely many indices $j \geq i$ such that $\sigma_j(0) \geq n$. Let j be such an index. Let $0 \leq d, d' \leq n$ be the largest indices for which there exist states

$q, q' \in P_X$ such that $\sigma_j(q) > 0$, $\sigma_j(q') > 0$, $\text{val}(q) = 2^d$ and $\text{val}(q') = -2^{d'}$. Note that these indices exist because $m_+ \neq 0$ and $m_- \neq 0$.

Assume without loss of generality that $d \geq d'$, as the other case is symmetric. By Proposition 19 (3), there exists $r \in B$ such that $C(r) > 0$. Since $C_j(0) \geq n \geq d - d'$, the sequence of transitions $\text{down}_d^+, \text{down}_{d-1}^+, \dots, \text{down}_{d'+1}^+$ can be fired from C_j . From there, we can fire $\text{cancel}_{d',r}$ which leads to a configuration D_j such that $|\text{val}^\circ(D_j)| < |m_\circ|$ for both $\circ \in \{+, -\}$. Since there are infinitely many such indices j , fairness implies that some such configuration D_j occurs (infinitely often) along σ , which contradicts both (4) and (5). \blacktriangleleft

Main proof. We are now ready to prove that \mathcal{P}_{thr} works as intended.

► **Theorem 24.** \mathcal{P}_{thr} computes φ with helpers and under reversible dynamic initialization.

Proof. We first show that \mathcal{P}_{thr} is input reversible, and then that it correctly computes φ .

Input reversibility. Let π be an initialization sequence with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$. By Proposition 19 (4), we have $C(I(x)) = C(x) = \mathbf{w}(x) - C(N_x) \leq \mathbf{w}(x)$ for every $x \in X$, which proves the first required property.

For every configuration C , let $\text{false}(C) \stackrel{\text{def}}{=} D$ where $D(t) \stackrel{\text{def}}{=} 0$, $D(f) \stackrel{\text{def}}{=} C(t) + C(f)$ and $D(q) \stackrel{\text{def}}{=} C(q)$ for every $q \in Q \setminus \{f, t\}$. Observe that for every configuration C , the following holds:

$$C \xrightarrow{\text{reset}^{C(\mathbf{v})}} \text{false}(C) \text{ and } \text{false}(C) \in [C]. \quad (6)$$

It remains to show that if $C \xrightarrow{\pi_{\triangleright} \in T^*} D$ and $D' \in [D]$, then $D' \xrightarrow{\pi_{\triangleleft} \in T^*} C'$ for some $C' \in [C]$. By (6), it is enough to argue that $\text{false}(D) \xrightarrow{T^*} \text{false}(C)$.

Let $C_i \xrightarrow{t_i} C_{i+1}$ be the i^{th} step of π_{\triangleright} . We argue that $\text{false}(C_{i+1}) \xrightarrow{t'_i} \text{false}(C_i)$ for some $t'_i \in T \cup \{\varepsilon\}$. By induction, this implies $\text{false}(D) \xrightarrow{T^*} \text{false}(C)$ as desired. If t_i is an *equal*, *false* or *reset* transition, then we already have $\text{false}(C_{i+1}) = \text{false}(C_i)$. Otherwise we revert the step as follows, where “ $s \mapsto u$ ” indicates that if $t_i = s$, then we reverse it with $t'_i = u$:

$$\begin{array}{ll} \text{add}_x \mapsto \text{add}_{x,\mathbf{f}}^{-1} & \text{for every } x \in X, \\ \text{add}_{x,q}^{-1} \mapsto \text{add}_x & \text{for every } x \in X \text{ and } q \in B, \\ \text{up}_i^\circ \mapsto \text{down}_{i+1}^\circ & \text{for every } 0 \leq i < n \text{ and } \circ \in \{+, -\}, \\ \text{down}_i^\circ \mapsto \text{up}_{i-1}^\circ & \text{for every } 0 < i \leq n \text{ and } \circ \in \{+, -\}, \\ \text{cancel}_{i,q} \mapsto \text{cancel}_{i,\mathbf{f}}^{-1} & \text{for every } 0 \leq i \leq n \text{ and } q \in B, \\ \text{cancel}_{i,q}^{-1} \mapsto \text{cancel}_{i,\mathbf{f}} & \text{for every } 0 \leq i \leq n \text{ and } q \in B, \\ \text{swap}_{p,q}^x \mapsto \text{swap}_{q,p}^x & \text{for every } p, q \in N \text{ and } x \in X. \end{array}$$

Note that t'_i is not the exact reverse transition of t_i , as it may differ over B . Indeed, t'_i may require an agent in state \mathbf{f} , which may not have been produced by t_i . However, this is not an issue since, by definition of *false* and by Proposition 19 (3), we have:

$$\text{false}(C_{i+1})(\mathbf{f}) = C_{i+1}(B) \geq L(B) > 0.$$

Thus, we have $\text{false}(C_{i+1}) \xrightarrow{t'_i} \text{false}(C_i)$ as desired, which completes the proof.

Correctness. Let π be an initialization sequence with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$. Let σ be a fair execution of \mathcal{P}_∞ starting from C . By Proposition 23, there exist $i \in \mathbb{N}$, $m_+ \geq 0$ and $m_- \leq 0$ such that:

- (a) $\sigma_i(X) = \sigma_{i+1}(X) = \dots = 0$,
- (b) $\text{val}^\circ(\sigma_i) = \text{val}^\circ(\sigma_{i+1}) = \dots = m_\circ$ for both $\circ \in \{+, -\}$,
- (c) $m_+ = 0 \vee m_- = 0$.

First, let us show that $O(\sigma) \in \{0, 1\}$. We make a case distinction on whether $m_+ \geq b$.

Case $m_+ \geq b$. We show that $O(\sigma) = 1$. Note that $m_+ \geq b > 0$ which implies that $m_- = 0$. Thus, by (b), no **cancel** transition is enabled in σ_j for every $j \geq i$. Thus, it suffices to show that $\sigma_j(\mathbf{f}) = 0$ for some $j \geq i$. For the sake of contradiction, suppose this is not the case. Let $j \geq i$ be such that $\sigma_j(\mathbf{f}) > 0$. We claim that σ_j can reach some configuration D_j enabling transition **equal**, *i.e.* larger or equal to $\text{rep}(b)$. This claim, together with fairness, yields a contradiction since this transition can move all agents in state \mathbf{f} to state \mathbf{t} .

Let us prove the claim. By Corollary 22, we have $\sigma_j \xrightarrow{T_\infty^*} C_j$ where $C_j(0) \geq n$. Note that $\text{val}^+(C_j) = \text{val}^+(\sigma_j)$. If $\text{val}^+(C_j) = b$, then, by cleanness, C_j contains precisely the binary representation of b , and hence $C_j \geq \text{rep}(b)$. Thus, assume $\text{val}^+(C_j) > b$. Let $0 \leq d \leq n$ be the largest exponent for which there exists a state $q \in P_X$ such that $C_j(q) > 0$, $\text{val}(q) = 2^d$ and $d \notin \text{bits}(b)$. Since $C_j(0) \geq n \geq d$, the sequence of transitions $\text{down}_d^+ \cdot \text{down}_{d-1}^+ \cdots \text{down}_1^+$ can be fired from C_j , which yields a configuration $D_j \geq \text{rep}(b)$.

Case $m_+ < b$. We show that $O(\sigma) = 0$. First note that **equal** is disabled in σ_j for every $j \geq i$, as otherwise we would have $\sigma_j \geq \text{rep}(b)$ which implies that $m_+ = \text{val}^+(\sigma_j') \geq b$. Thus, it suffices to show that there are infinitely many indices j such that $\sigma_j(\mathbf{f}) > 0$. Indeed, if this is the case, then, by fairness, **false** permanently moves all agents in state \mathbf{t} to state \mathbf{f} .

For the sake of contradiction, suppose the claim does not hold. Let $\sigma' \stackrel{\text{def}}{=} \pi\sigma$. Let $j \in \mathbb{N}$ be the largest index such that $\sigma_j'(\mathbf{f}) > 0$. Note that this configuration exists as $\sigma_0' = L' \succeq L$ and $L(\mathbf{f}) > 0$. The only transition that reduces the number of agents in \mathbf{f} is **equal**. Thus, $\sigma_j' \xrightarrow{\text{equal}} \sigma_{j+1}'$ and $\text{val}^+(\sigma_j') \geq b$. As finally, $m_+ < b$, some **cancel** or **add**⁻¹ transition must be fired in $\sigma_{j'}'$ for some $j' > j$. In both cases there is afterwards an agent in state \mathbf{f} . This contradicts the maximality of j .

We are done proving $O(\sigma) \in \{0, 1\}$. It remains to argue that $O(\sigma) = \varphi(\mathbf{w})$. We have:

$$\begin{aligned}
\mathbf{a} \cdot \mathbf{w} &= \text{val}(\sigma_i) && \text{(by Prop. 19 (1))} \\
&= \text{val}^+(\sigma_i) + \text{val}^-(\sigma_i) && \text{(by (a))} \\
&= m_+ + m_- && \text{(by (b))}
\end{aligned}$$

Recall that $m_+ \geq 0$, $m_- \leq 0$ and $(m_+ = 0 \vee m_- = 0)$. If $\mathbf{a} \cdot \mathbf{w} \geq b$, then we must have $m_+ \geq b > 0$ and $m_- = 0$. Therefore, the first case above holds, and hence $O(\sigma) = 1$, which is correct. If $\mathbf{a} \cdot \mathbf{w} < b$, then we must have $m_+ < b$. Therefore, the second case above holds, and hence $O(\sigma) = 0$, which is also correct. \blacktriangleleft

B.4.2 Remainder protocols

This section describes a family of protocols with helpers computing remainder predicates under reversible dynamic initialization. The construction, its correctness proof and its intermediary propositions are similar to those presented in Appendix B.4.1 for the case of threshold predicates. For completeness, we repeat and adapt them in full details.

Let us fix a remainder predicate φ over variables X . Let $\varphi(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{a} \cdot \mathbf{v} \equiv_m b$ where

$\mathbf{a} \in \mathbb{Z}^X$ and $m \in \mathbb{N}_{\geq 2}$ and $b \in \mathbb{Z}$. Without loss of generality⁴, we may assume $0 \leq b < m$ and $0 \leq \mathbf{a}(x) < m$ for each $x \in X$.

Instead of directly constructing a protocol for $\varphi(\mathbf{v})$, we rewrite the predicate. This yields a different but equivalent predicate $\varphi'(\mathbf{v})$:

$$\varphi'(\mathbf{v}) \stackrel{\text{def}}{=} \begin{cases} \mathbf{a} \cdot \mathbf{v} \not\geq 1 \pmod{m} & \text{if } b = 0, \\ \mathbf{a} \cdot \mathbf{v} \geq b \pmod{m} \wedge \mathbf{a} \cdot \mathbf{v} \not\geq b + 1 \pmod{m} & \text{if } b > 0. \end{cases}$$

As we can handle negations and conjunctions separately in Section 5.2, it is enough to describe a protocol for the predicate $\varphi(\mathbf{v}) \stackrel{\text{def}}{=} \mathbf{a} \cdot \mathbf{v} \geq b \pmod{m}$ where $\mathbf{a} \in \mathbb{N}^X$, $m \in \mathbb{N}_{\geq 2}$, $0 < b < m$ and $0 \leq \mathbf{a}(x) < m$ for each $x \in X$.

We construct a simple population protocol \mathcal{P}_{rem} with helpers that computes φ under reversible dynamic initialization, and prove its correctness.

Notation. Let n be the smallest number such that $2^n > \|\varphi\|$. Let $P \stackrel{\text{def}}{=} \{2^i : 0 \leq i \leq n\}$, $Z \stackrel{\text{def}}{=} \{0\}$, $N \stackrel{\text{def}}{=} P \cup Z$ and $B \stackrel{\text{def}}{=} \{\mathbf{f}, \mathbf{t}\}$, where P , Z , N and B respectively stand for “Powers of two”, “Zero”, “Numerical values” and “Boolean values”. For every set S and every $x \in X$, let $S_x \stackrel{\text{def}}{=} \{q_x : q \in S\}$ and $S_X \stackrel{\text{def}}{=} S \cup \bigcup_{x \in X} S_x$.

For every $d \in \mathbb{N}$, let $\text{bits}(d)$ denote the unique set $J \subseteq \mathbb{N}$ such that $d = \sum_{j \in J} 2^j$, *e.g.* $\text{bits}(13) = \text{bits}(1101_2) = \{3, 2, 0\}$. The *canonical representation* of an integer $d \in \mathbb{Z}$ is the multiset $\text{rep}(d)$ defined as follows:

$$\text{rep}(d) \stackrel{\text{def}}{=} \begin{cases} \{2^i : i \in \text{bits}(d)\} & \text{if } d > 0, \\ \{0\} & \text{if } d = 0. \end{cases}$$

The protocol. The RDI-protocol $\mathcal{P}_{\text{rem}} = (Q, T_\infty, T_\dagger, L, X, I, O)$ is defined as follows:

- $Q \stackrel{\text{def}}{=} X \cup N_X \cup B$.
Intuitively, the states of X are the “ports” through which the agents for each variable enter and exit the protocol.
- $I \stackrel{\text{def}}{=} x \mapsto \mathbf{x}$.
That is, the initial state for variable x is x .
- $L \stackrel{\text{def}}{=} \{2n \cdot 0, \mathbf{f}\}$.
So, we have $2n$ helpers in state 0, and one helper in state \mathbf{f} , i.e., initially the protocol assumes that the predicate does not hold.
- $O(q) \stackrel{\text{def}}{=} q \mapsto (0 \text{ if } q = \mathbf{f} \text{ else } 1 \text{ if } q = \mathbf{t} \text{ else } \perp)$.
That is, the output of the protocol is completely determined by the number of agents in states \mathbf{t} and \mathbf{f}

⁴ If this is not the case for some coefficient $\mathbf{a}(x)$, then we can replace it by $\mathbf{a}(x) \bmod m$, which yields an equivalent predicate.

■ T_∞ is the following set of (“permanent”) transitions:

$$\begin{array}{ll}
\text{add}_x : \langle x, |\text{rep}(\mathbf{a}(x))| \cdot 0 \rangle \mapsto \langle 0_x \rangle + \text{rep}(\mathbf{a}(x)) & \text{for every } x \in X, \\
\text{up}_i : \langle 2^i, 2^i \rangle \mapsto \langle 2^{i+1}, 0 \rangle & \text{for every } 0 \leq i < n, \\
\text{down}_i : \langle 2^i, 0 \rangle \mapsto \langle 2^{i-1}, 2^{i-1} \rangle & \text{for every } 0 < i \leq n, \\
\text{modulo}_q : \text{rep}(b) + \langle q \rangle \mapsto \text{rep}(b) + \langle \mathbf{f} \rangle & \text{for every } q \in B, \\
\text{swap}_{p,q}^x : \langle p, q_x \rangle \mapsto \langle p_x, q \rangle & \text{for every } p, q \in N \text{ and } x \in X, \\
\text{equal} : \text{rep}(b) + \langle \mathbf{f} \rangle \mapsto \text{rep}(b) + \langle \mathbf{t} \rangle, \\
\text{false} : \langle \mathbf{f}, \mathbf{t} \rangle \mapsto \langle \mathbf{f}, \mathbf{f} \rangle.
\end{array}$$

Intuitively, add_x converts an agent which arrived via port x into the canonical representation of $\mathbf{a}(x)$. Transitions of the form $\text{up}_i, \text{down}_i$ allow the protocol to change the representation of a value, without changing the value itself. The modulo transition reduces the overall value by m . Transition equal allows the protocol to detect that the current value is at least b , which moves a helper from state \mathbf{f} to \mathbf{t} .

■ Finally, T_\dagger is the following set of RDI-transitions:

$$\begin{array}{ll}
\text{add}_{x,q}^{-1} : \langle 0_x, q \rangle + \text{rep}(\mathbf{a}(x)) \mapsto \langle x, \mathbf{f}, |\text{rep}(\mathbf{a}(x))| \cdot 0 \rangle & \text{for every } x \in X \text{ and } q \in B, \\
\text{modulo}_q^{-1} : \text{rep}(b) + \langle q \rangle \mapsto \text{rep}(b) + \langle \mathbf{f} \rangle & \text{for every } q \in B, \\
\text{reset} : \langle \mathbf{t} \rangle \mapsto \langle \mathbf{f} \rangle.
\end{array}$$

The first two transitions are needed to reverse the changes of add and modulo transitions while the dynamic initialization is not finished. Both types of transitions reset the output of the protocol by leaving an agent in the default output state \mathbf{f} . The reset transition resets the output by moving agents from \mathbf{t} to \mathbf{f} .

Let $\mathcal{P}_\infty = (Q, T_\infty, L, X, I, O)$. Let $T \stackrel{\text{def}}{=} T_\infty \cup T_\dagger$. For the sake of readability, we will sometimes omit the subscripts and superscripts from transitions names when they are irrelevant, *e.g.* “a swap transition is enabled” instead of “there exist $p, q \in N$ and $x \in X$ such that $\text{swap}_{p,q}^x$ is enabled”.

Size. Note that \mathcal{P}_{rem} has $|Q| = |X| + |N_X| + |B| = |X| + (n+2) \cdot (|X|+1) + 2 = \mathcal{O}(\log\|\varphi\| \cdot |X|)$ states and $|L| = 2n + 1 = \mathcal{O}(\log\|\varphi\|)$ helpers. Moreover, since families of transitions are parameterized by X , B and N or N^2 , there are $\mathcal{O}(|N|^2 \cdot |X|) = \mathcal{O}(\log^2\|\varphi\| \cdot |X|) \subseteq \mathcal{O}(|\varphi|^3)$ transitions. Finally, each transition uses at most $\mathcal{O}(|\text{rep}(\|\varphi\|)|) = \mathcal{O}(\log\|\varphi\|) \subseteq \mathcal{O}(|\varphi|)$ states.

Auxiliary definitions and observations. Before proving that \mathcal{P}_{rem} works as intended, let us first introduce auxiliary definitions. Let $\text{val} : Q \rightarrow \mathbb{N}$ be the function that associates a value to each state as follows:

$$\begin{array}{ll}
\text{val}(0) = \text{val}(\mathbf{f}) = \text{val}(\mathbf{t}) \stackrel{\text{def}}{=} 0, \\
\text{val}(x) \stackrel{\text{def}}{=} \mathbf{a}(x) & \text{for every } x \in X, \\
\text{val}(2^i) \stackrel{\text{def}}{=} 2^i & \text{for every } 0 \leq i \leq n, \\
\text{val}(q_x) \stackrel{\text{def}}{=} \text{val}(q) & \text{for every } q \in N \text{ and } x \in X.
\end{array}$$

So, for example, for the predicate $5x + 6y \geq 4 \pmod{m}$ we have $\text{val}(x) = 5$ and $\text{val}(y) = 6$. For every configuration C and set of states $S \subseteq Q$, let

$$\text{val}_S(C) \stackrel{\text{def}}{=} \sum_{q \in S} \text{val}(q) \cdot C(q).$$

In particular, let $\text{val}(C) \stackrel{\text{def}}{=} \text{val}_Q(C)$. Intuitively, C can be seen as an encoding of the value $\text{val}(C)$. The following properties, relating values and configurations, can be derived from the above definitions:

► **Proposition 25.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, the following holds:*

1. $\text{val}(C) \equiv_m \mathbf{a} \cdot \mathbf{w}$,
2. $\text{val}(C) \leq \mathbf{a} \cdot \mathbf{w}$,
3. $|C| = C(N) + C(B) + |\mathbf{w}|$,
4. $C(N) \geq L(N)$ and $C(B) \geq L(B)$,
5. $C(N_x) + C(x) = \mathbf{w}(x)$ for every $x \in X$.

Auxiliary propositions. We say that a configuration C is *clean* if for every $p, q \in P_X$ with $\text{val}(p) = \text{val}(q)$ and $\text{val}(p) \neq 2^n$, it holds that $C(\{p, q\}) \leq 1$. Intuitively, in a clean configuration no agent can be promoted to a higher power of 2.

We show that any configuration can be cleaned using only permanent transitions. This implies that once the dynamic initialization has terminated, every fair execution visits clean configurations infinitely often.

► **Proposition 26.** *For every initialization sequence π such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, there exists a clean configuration D such that $C \xrightarrow{T_\infty^*} D$.*

Proof. If C is clean, then we pick $D \stackrel{\text{def}}{=} C$. Otherwise, we claim there exists a configuration C' such that $C \xrightarrow{*} C'$ and $C(P_X) > C'(P_X)$. If C' is clean, then we are done. Otherwise, this process is repeated until a clean configuration D has been reached. This must terminate as the number of agents in P_X cannot become negative.

Let us prove the claim. If C is not clean, then $C(\{p, q\}) \geq 2$ for some $p, q \in P_X$ such that $\text{val}(p) = \text{val}(q)$ and $\text{val}(p) \neq 2^n$. By Proposition 25 (4), we have $C(N) \geq L(N) \geq 2$ and hence it is possible to consecutively fire at least two **swap** transitions. Note that they do not change the amount of agents in P_X . For this reason, we may assume without loss of generality that $p = q = 2^i$ for some $0 \leq i \leq n$. Therefore, firing transition up_i decreases $C(P_X)$ by one. ◀

We now bound the number of agents in states from $X \cup P_X$ in a clean configuration.

► **Proposition 27.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, if C is clean, then $C(X) + C(P_X) \leq |\mathbf{w}| + n$.*

Proof. Let $S_{2^n} \stackrel{\text{def}}{=} \{q \in P_X : \text{val}(q) = 2^n\}$. Let $\mathbf{u} \in \mathbb{N}^X$ be such that $\mathbf{u}(x) \stackrel{\text{def}}{=} C(x)$ for every $x \in X$. Note that $|\mathbf{u}| = C(X)$, and that $\mathbf{u} \leq \mathbf{w}$ by Proposition 25 (5). Since C is clean, we have $C(P_X \setminus S_{2^n}) \leq n$. Thus, it suffices to show that $C(S_{2^n}) \leq |\mathbf{w}| - |\mathbf{u}|$. Suppose this is not the case. This yields a contradiction:

$$\begin{aligned}
 \text{val}_{P_X}(C) &\geq 2^n \cdot C(S_{2^n}) \\
 &> 2^n \cdot (|\mathbf{w}| - |\mathbf{u}|) && \text{(by assumption)} \\
 &\geq \mathbf{a} \cdot (\mathbf{w} - \mathbf{u}) && \text{(since } 2^n > \|\mathbf{a}\| \text{ and } \mathbf{w} \geq \mathbf{u}) \\
 &= \mathbf{a} \cdot \mathbf{w} - \mathbf{a} \cdot \mathbf{u} \\
 &\geq \text{val}(C) - \text{val}_X(C) && \text{(by Prop. 25 (2) and def. of } \mathbf{u}) \\
 &= \text{val}_{P_X}(C) && \text{(by def. of val)}
 \end{aligned}$$

◀

The following corollary shows that the number of agents in state 0 can always be increased back to at least n . This will later be useful in arguing that the number of agents in X can eventually be decreased to zero.

► **Corollary 28.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$, there exists a clean configuration D such that $C \xrightarrow{T_\infty^*} D$ and $D(0) \geq n$.*

Proof. By Proposition 26, there exist a clean configuration C' such that $C \xrightarrow{\pi' \in T_\infty^*} C'$. Let us first prove that $C'(Z_X) \geq n$. Note that $\pi\pi'$ is an initialization sequence with effective input \mathbf{w} such that $L \xrightarrow{\pi\pi'} C'$. Thus:

$$\begin{aligned}
 C'(Z_X) &= |C'| - C'(X) - C'(P_X) - C'(B) && \text{(by def. of } Q) \\
 &= (C'(N) + C'(B) + |\mathbf{w}|) - C'(X) - C'(P_X) - C'(B) && \text{(by Prop. 25 (3))} \\
 &\geq (L(N) + C'(B) + |\mathbf{w}|) - C'(X) - C'(P_X) - C'(B) && \text{(by Prop. 25 (4))} \\
 &\geq (L(N) + C'(B) + |\mathbf{w}|) - (|\mathbf{w}| + n) - C'(B) && \text{(by Prop. 27)} \\
 &= L(N) - n \\
 &\geq n && \text{(by def. of } L)
 \end{aligned}$$

Now, by Proposition 25 (4), we have $C'(N) \geq L(N) \geq n$. Thus, using **swap** transitions, we can swap n agents from Z_X to 0. This way, we obtain a configuration D such that $C' \xrightarrow{T_\infty^*} D$ and $D(0) \geq n$. We are done since **swap** transitions preserve cleanness. ◀

We now show that, once dynamic initialization has terminated, fair executions stabilize to configurations of a certain “normal form”.

► **Proposition 29.** *For every initialization sequence π with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$ and for every fair execution σ of \mathcal{P}_∞ starting from C , there exist $i \in \mathbb{N}, r \geq 0$ such that*

1. $\sigma_i(X) = \sigma_{i+1}(X) = \dots = 0$,
2. $\text{val}(\sigma_i) = \text{val}(\sigma_{i+1}) = \dots = r < m$.

Proof. For the sake of contradiction, assume there exist infinitely many indices j such that $\sigma_j(X) > 0$. Let $j \in \mathbb{N}$ be such an index. By Corollary 28, there exists a configuration C_j such that $\sigma_j \xrightarrow{T_\infty^*} C_j$ and $C_j(0) \geq n$. Hence, there exists $x \in X$ such that transition **add** _{x} is enabled in C_j . Since this holds for infinitely many indices and since X is finite, fairness implies that some **add** transition can be enabled infinitely often and hence occurs infinitely often along σ . This is impossible since the number of agents in X cannot be increased by any transition in T_∞ , and thus would eventually drop below zero. Therefore, there exists $h \in \mathbb{N}$ such that $\sigma_h(X) = \sigma_{h+1}(X) = \dots = 0$.

Transitions **modulo** _{t} and **modulo** _{f} reduce the value of a configuration by $m > 0$. **add** transitions are disabled in every configuration $\text{val}(\sigma_i)$ with $i \geq h$ and all other transitions in T_∞ do not change the value of a configuration. Moreover, the value of a configuration is always non-negative. Thus, there exist $i \geq h, r \geq 0$ such that $\text{val}(\sigma_i) = \text{val}(\sigma_{i+1}) = \dots = r$.

For the sake of contradiction, assume that $r \geq m$. As the execution is infinite but there are only finitely many different configurations for a fixed number of agents, there exists a configuration D with $D(X) = 0$ and $\text{val}(D) = r$ that occurs infinitely often in σ . We claim that D can reach a configuration that enables a **modulo** transition. This claim, together with fairness, yields a contradiction because the overall value would drop below r .

Let us prove the claim. By Corollary 28, we have $D \xrightarrow{*} D'$ where D' is clean and $D'(0) \geq n$. Furthermore, $D'(X) = 0$ as $D(X) = 0$ and the number of agents in X cannot be increased. By Proposition 25 (4), we have $D'(B) \geq L(B) = 1$. Thus, it suffices to show that $D' \xrightarrow{T^*_{\infty}} D''$ for some $D'' \geq \text{rep}(m)$.

If $\text{val}(D') = m$, then D' contains precisely the binary representation of m , because D' is clean and $D'(X) = 0$. Hence, $D' \geq \text{rep}(m)$. Thus, assume $\text{val}(D') > m$. Let $0 \leq d \leq n$ be the largest exponent for which there exists a state $q \in P_X$ such that $D(q) > 0$, $\text{val}(q) = 2^d$ and $d \notin \text{bits}(m)$. Since $D'(0) \geq n \geq d$, the sequence of transitions $\text{down}_d \cdot \text{down}_{d-1} \cdots \text{down}_1$ can be fired from D' , which yields a configuration $D'' \geq \text{rep}(m)$. \blacktriangleleft

Main proof. We are now ready to prove that \mathcal{P}_{rem} works as intended.

► **Theorem 30.** \mathcal{P}_{rem} computes φ under reversible dynamic initialization.

Proof. We first show that \mathcal{P}_{rem} is input reversible, and then that it correctly computes φ .

Input reversibility. Let π be an initialization sequence with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$. By Proposition 25 (5), we have $C(I(x)) = C(x) = \mathbf{w}(x) - C(N_x) \leq \mathbf{w}(x)$ for every $x \in X$, which proves the first required property.

For every configuration C , let $\text{false}(C) \stackrel{\text{def}}{=} D$ where $D(t) \stackrel{\text{def}}{=} 0$, $D(f) \stackrel{\text{def}}{=} C(t) + C(f)$ and $D(q) \stackrel{\text{def}}{=} C(q)$ for every $q \in Q \setminus \{f, t\}$. Observe that for every configuration C , the following holds:

$$C \xrightarrow{\text{reset}^{C(t)}} \text{false}(C) \text{ and } \text{false}(C) \in [C]. \quad (7)$$

It remains to show that if $C \xrightarrow{\pi_{\triangleright} \in T^*} D$ and $D' \in [D]$, then $D' \xrightarrow{\pi_{\triangleleft} \in T^*} C'$ for some $C' \in [C]$. By 7, it is enough to argue that $\text{false}(D) \xrightarrow{T^*} \text{false}(C)$.

Let $C_i \xrightarrow{t_i} C_{i+1}$ be the i^{th} step of π_{\triangleright} . We argue that $\text{false}(C_{i+1}) \xrightarrow{t'_i} \text{false}(C_i)$ for some $t'_i \in T \cup \{\varepsilon\}$. By induction, this implies $\text{false}(D) \xrightarrow{T^*} \text{false}(C)$ as desired. If t_i is an *equal*, *false* or *reset* transition, then we already have $\text{false}(C_{i+1}) = \text{false}(C_i)$. Otherwise we revert the step as follows, where “ $s \mapsto u$ ” indicates that if $t_i = s$, then we reverse it with $t'_i = u$:

$$\begin{array}{ll} \text{add}_x \mapsto \text{add}_{x,\mathbf{f}}^{-1} & \text{for every } x \in X, \\ \text{add}_{x,q}^{-1} \mapsto \text{add}_x & \text{for every } x \in X \text{ and } q \in B, \\ \text{up}_i \mapsto \text{down}_{i+1} & \text{for every } 0 \leq i < n, \\ \text{down}_i \mapsto \text{up}_{i-1} & \text{for every } 0 < i \leq n, \\ \text{modulo}_q \mapsto \text{modulo}_{\mathbf{f}}^{-1} & \text{for every } q \in B, \\ \text{modulo}_q^{-1} \mapsto \text{modulo}_{\mathbf{f}} & \text{for every } q \in B, \\ \text{swap}_{p,q}^x \mapsto \text{swap}_{q,p}^x & \text{for every } p, q \in N \text{ and } x \in X. \end{array}$$

Note that t'_i is not the exact reverse transition of t_i , as it may differ over B . Indeed, t'_i may require an agent in state \mathbf{f} , which may not have been produced by t_i . However, this is not an issue since, by definition of *false* and by Proposition 19 (3), we have:

$$\text{false}(C_{i+1})(\mathbf{f}) = C_{i+1}(B) \geq L(B) > 0.$$

Thus, we have $\text{false}(C_{i+1}) \xrightarrow{t'_i} \text{false}(C_i)$ as desired, which completes the proof.

Correctness. Let π be an initialization sequence with effective input \mathbf{w} such that $L' \xrightarrow{\pi} C$ for some $L' \succeq L$. Let σ be a fair execution of \mathcal{P}_{∞} starting from C . By Proposition 29, there exist $i \in \mathbb{N}, r \geq 0$ such that:

- (a) $\sigma_i(X) = \sigma_{i+1}(X) = \dots = 0$,
- (b) $\text{val}(\sigma_i) = \text{val}(\sigma_{i+1}) = \dots = r < m$.

Let us first show that $O(\sigma) \in \{0, 1\}$. We make a case distinction on whether $v \geq b$.

Case $r \geq b$. We show that $O(\sigma) = 1$. Note that $r < m$. The **modulo** transitions reduce the overall value by m . As the value of a configuration is never negative, no **modulo** transition can be fired again. Thus, it suffices to show that $\sigma_j(\mathbf{f}) = 0$ for some $j \geq i$. If $\sigma_i(\mathbf{f}) = 0$ then we are done. For the sake of contradiction, suppose this is not the case. As σ is infinite but there are only finitely many different configurations for a fixed number of agents, there exists a configuration D that occurs infinitely often in σ such that $D(X) = 0$, $\text{val}(D) = r$ and $D(\mathbf{f}) > 0$. We claim that D can reach a configuration that enables the transition **equal**. This claim, together with fairness, yields a contradiction because **equal** can move all agents from state \mathbf{f} to state \mathbf{t} .

Let us prove the claim. By Corollary 28, we have $D \xrightarrow{T_\infty^*} D'$ where D' is clean and $D'(0) \geq n$. Furthermore, $D'(X) = 0$ as $D(X) = 0$ and the number of agents in X cannot be increased by transitions in T_∞ . We show that $D' \xrightarrow{*} D''$ for some $D'' \geq \text{rep}(m)$. If $\text{val}(D') = m$, then D' contains precisely the binary representation of m , because D' is clean and $D'(X) = 0$. Hence, $D' \geq \text{rep}(m)$. Thus, assume $\text{val}(D') > m$. Let $0 \leq d \leq n$ be the largest exponent for which there exists a state $q \in P_X$ such that $D(q) > 0$, $\text{val}(q) = 2^d$ and $d \notin \text{bits}(m)$. Since $D'(0) \geq n \geq d$, the sequence of transitions $\text{down}_d \cdot \text{down}_{d-1} \cdots \text{down}_1$ can be fired from D' , which yields a configuration $D'' \geq \text{rep}(m)$. If $D''(\mathbf{f}) = 0$, then the claim holds because the only transition that reduces the number of agents in state \mathbf{f} is transition **equal**. If $D''(\mathbf{f}) > 0$, then the claim holds because D'' enables **equal**.

Case $r < b$. We show that $O(\sigma) = 0$. First note that **equal** is disabled in σ_j for every $j \geq i$, as otherwise we would have $\sigma_j \geq \text{rep}(b)$ which implies that $r = \text{val}(\sigma_j) \geq b$. Thus, it suffices to show that there are infinitely many indices j such that $\sigma_j(\mathbf{f}) > 0$. Indeed, if this is the case, then, by fairness, **false** permanently moves all agents in state \mathbf{t} to state \mathbf{f} .

For the sake of contradiction, suppose the claim does not hold. Let $\sigma' \stackrel{\text{def}}{=} \pi\sigma$. Let $j \in \mathbb{N}$ be the largest index such that $\sigma'_j(\mathbf{f}) > 0$. Note that this configuration exists as $\sigma'_0 = L' \succeq L$ and $L(\mathbf{f}) > 0$. The only transition that reduces the number of agents in \mathbf{f} is **equal**. Thus, $\sigma'_j \xrightarrow{\text{equal}} \sigma'_{j+1}$ and $\text{val}(\sigma'_j) \geq b$. As finally, $r < b$, some **modulo** or add^{-1} transition must be fired in $\sigma'_{j'}$ for some $j' > j$. In both cases there is afterwards an agent in state \mathbf{f} . This contradicts the maximality of j .

We are done proving $O(\sigma) \in \{0, 1\}$. It remains to argue that $O(\sigma) = \varphi(\mathbf{w})$. We have $m > r = \text{val}(\sigma_i) \equiv_m \mathbf{a} \cdot \mathbf{w}$ by Proposition 25 (1). If $\mathbf{a} \cdot \mathbf{w} \geq b \pmod{m}$, then $r \geq b$ and hence $O(\sigma) = 1$, which is correct. If $\mathbf{a} \cdot \mathbf{w} < b \pmod{m}$, then $r < b$ and hence $O(\sigma) = 0$, which is also correct. \blacktriangleleft

C Proofs of Section 6: Protocols for small populations

C.1 Proof of Theorem 9

Theorem 9. *Let φ be a predicate over a set of variables X , and let $\ell \in \mathbb{N}$. Assume that for every $i \in \{2, 3, \dots, \ell - 1\}$, there exists a protocol with at most one leader and at most m states that computes $(\varphi \mid i)$. Then, there is a leaderless population protocol with $\mathcal{O}(\ell^4 \cdot m^2 \cdot |X|^3)$ states that computes $(\mathbf{x} < \ell) \rightarrow \varphi(\mathbf{x})$.*

The proof proceeds in two steps. Lemma 31 shows that, under the assumptions of the proposition, there is a protocol with one leader computing φ for all small populations. Lemma

35 shows how to transform this protocol into a leaderless one. The bound on the number of states follows directly from the composition of the bounds given in the lemmas.

► **Lemma 31.** *Let φ be a predicate over a set of variables X and let $\ell \in \mathbb{N}$. Assume that for every $i \in \{2, 3, \dots, \ell - 1\}$, there exists a protocol with at most one leader and at most m states that computes $(\varphi \mid i)$. Then there exists a protocol with one leader and $\mathcal{O}(\ell \cdot m \cdot |X|)$ states that computes $\mathbf{x} < \ell \rightarrow \varphi(\mathbf{x})$.*

Proof. Let $\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\ell$ be such that each \mathcal{P}_i computes $(\varphi \mid i)$. Without loss of generality, assume the states of the \mathcal{P}_i to be pairwise disjoint. We construct a protocol $\mathcal{P} = (Q, T, L, X, I, O)$ with one leader and $\mathcal{O}(\ell \cdot m \cdot |X|)$ states that computes $\mathbf{x} \leq \ell \rightarrow \varphi(\mathbf{x})$. Intuitively, the protocol \mathcal{P} works as follows: the leader stores a lower-bound estimate of the current population size. When the leader meets a new agent it has not met, the leader increments its estimate. Whenever the estimate changes to some value i , the leader resets i agents in the population to initial of \mathcal{P}_i and lets the agents simulate the computation of \mathcal{P}_i . When the estimate reaches ℓ , the leader knows that the precondition $|X| < \ell$ is not satisfied, and it converts every agent to \top , a state that converts any other state to \top , thus yielding a stable 1-consensus. The agents' states are annotated with their initial input, which allows the leader to reset states to the correct value.

States and associated mappings. Let $Q_i \stackrel{\text{def}}{=} X \times Q^{\mathcal{P}_i}$ for every $i \in [\ell]$, where $Q^{\mathcal{P}_i}$ denotes the states of \mathcal{P}_i . The leader assumes a state from the *leader states* defined as $Q_L \stackrel{\text{def}}{=} \{0, 1, \dots, \ell\} \times \{0, 1\}$. The states of \mathcal{P} are defined as:

$$Q \stackrel{\text{def}}{=} X \cup (X \times (Q_1 \cup \dots \cup Q_{\ell-1})) \cup \{\top\} \cup Q_L.$$

For the size of the protocol we thus have

$$|Q| = |X| + |(X \times (Q_1 \cup \dots \cup Q_{\ell-1}))| + 1 + |Q_L| \leq |X| + |X| \cdot m \cdot \ell + 1 + 2 \cdot \ell,$$

which is in $\mathcal{O}(\ell \cdot m \cdot |X|)$.

We set the leader multiset to:

$$L \stackrel{\text{def}}{=} \lambda(0, 0)\$.$$

The input mapping I is defined as the identity function. The output mapping is given by:

$$\begin{aligned} O(x) &\stackrel{\text{def}}{=} 0 && \text{for every } x \in X, \\ O((x, q)) &\stackrel{\text{def}}{=} O^{\mathcal{P}_i}(q) && \text{for every } i \in [\ell] \text{ and every } (x, q) \in Q_i, \\ O((i, b)) &\stackrel{\text{def}}{=} b && \text{for every } (i, b) \in Q_L, \\ O(\top) &\stackrel{\text{def}}{=} 1. \end{aligned}$$

Transitions. The set of transitions T of \mathcal{P} is given by $T \stackrel{\text{def}}{=} T_\top \cup T_{\text{sim}} \cup T_{\text{incr}}$ where T_\top , T_{sim} , and T_{incr} are defined as follows.

■ T_\top contains precisely the transitions:

$$\begin{aligned} \text{true} : \quad & \lambda(\top, q)\$ \mapsto \lambda(\top, \top)\$ && \text{for every } q \in Q, \\ \text{threshold} : & \lambda(\ell, b)\$, q\$ \mapsto \lambda(\top, \top)\$ && \text{for every } b \in \{0, 1\}, q \in Q. \end{aligned}$$

Intuitively, T_\top contains transitions that ensure stabilization to 1 if $|X| < \ell$ is not satisfied: **threshold** initiates converting everyone to \top as soon as the threshold ℓ is reached in the leader agent. The transitions **true** then convert everyone to \top .

- T_{sim} is given by $\bigcup_{i \in [\ell]} T_{\text{sim}_i}$, and T_{sim_i} contains precisely the following transitions for every $x, y \in X$ and every $t: (\lambda(q, r), \lambda(q', r')) \in T^{\mathcal{P}_i}$:

$$\text{sim} : \lambda(x, q), (y, r) \mapsto \lambda(x, q'), (y, r')$$

Intuitively, the transitions in T_{sim_i} simulate the transitions of the individual protocols \mathcal{P}_i in \mathcal{P} .

- T_{conv} contains precisely the following transitions for every $x \in X$:

$$\begin{aligned} \text{incr}_i : \lambda(x, (i, b)) &\mapsto \lambda(x, I^{\mathcal{P}_{i+1}}(x)), (i+1, b) && \text{for every } i \in [1, \ell-1], b \in \{0, 1\}, \\ \text{conv}_i : \lambda(x, q), (i, b) &\mapsto \lambda(x, I^{\mathcal{P}_i}(x)), (i, b) && \text{for every } i \in [1, \ell-1], q \notin Q_i, \\ \text{bool}_i : \lambda(x, q), (i, b) &\mapsto \lambda(x, q), (i, O(q)) && \text{for every } i \in [1, \ell-1], q \in Q_i. \end{aligned}$$

Intuitively, the transitions in T_{conv} implement interactions with the leader whose role is to convert every agent to the current protocol: incr_i and conv_i take care of converting agents to the next protocol, while the transitions bool_i convert the leader's opinion to the opinion of the current protocol.

Correctness. Before we prove correctness of \mathcal{P} , we state without proof some propositions that follow by inspection of the transitions of \mathcal{P} :

► **Proposition 32.** *For every $C, C' \in \mathbb{N}^Q$, the following invariant holds: If $C \rightarrow C'$, then $C'(\top) \geq C(\top)$.*

► **Proposition 33.** *Let $\mathbf{v} \in \mathbb{N}^X$. In every fair run σ of \mathcal{P} starting in $C_{\mathbf{v}}$, the transition conv_i is taken precisely once in σ for every $i \leq \max(|\mathbf{v}|, \ell)$.*

► **Proposition 34.** *In every fair run σ of \mathcal{P} such that $\sigma_k((i, 0)) + \sigma_k((i, 1)) = 1$ for all but finitely many indices k , we have for all but finitely many indices k that $\sigma_k(\mathbf{q}) = 0$ for every $\mathbf{q} \in Q \setminus (Q_\ell \cup (X \times Q_i))$.*

We now prove correctness of \mathcal{P} . Let $\mathbf{v} \in \mathbb{N}^X$ and let σ be a fair execution of \mathcal{P} starting in $C_{\mathbf{v}}$. We consider two cases: $|\mathbf{v}| \geq \ell$ and $|\mathbf{v}| < \ell$.

Let us first consider the case $|\mathbf{v}| \geq \ell$. If $|\mathbf{v}| \geq \ell$, then by fairness of σ and Proposition 33 we have that the transitions incr_i are fired ℓ times in σ , and thus there is some j such that $\sigma_j((\ell, b)) > 0$ for some $b \in \{0, 1\}$. By fairness of σ and by construction of T , we then have that $\sigma_j \xrightarrow{\text{true}} \sigma_{j+1}$ and $\sigma_{j+1}(\top) > 0$ for some j , and by Proposition 32 true is fired infinitely often in σ , and thus σ stabilizes to a configuration where every agent is in state \top . Thus, if $|\mathbf{v}| \geq \ell$, then \mathcal{P} stabilizes to output 1.

Now consider the case $|\mathbf{v}| = m < \ell$. By fairness and Proposition 33, the transitions incr_i are fired m times in σ , until every agent leaves its initial state. Let j be the largest index such that $\sigma_j \xrightarrow{\text{incr}_{m-1}} \sigma_m$. Since the transitions incr_i are the only transitions that change the first component of the leader, we have $\sigma_k((m, 1)) = 1$ or $\sigma_k((m, 0)) = 1$ for every $k > j$. From this, Proposition 34 and fairness of σ , we have that eventually all non-leader agents are in a state from $X \times Q_m$. The transitions T_{sim} then guarantee by fairness of σ that the non-leader agents stabilize to the output of \mathcal{P}_m , while the transition bool_m ensures that the leader agent stabilizes to the same output, and thus, since by assumption \mathcal{P}_m stabilizes to $\varphi(\mathbf{v})$ if $|\mathbf{v}| = m$, and $|\mathbf{v}| = m$ holds by assumption, we have $O(\sigma) = \varphi(\mathbf{v})$. This completes the proof. ◀

The following lemma shows how to get rid of the leaders in halting protocols.

► **Lemma 35.** *Let $\ell \in \mathbb{N}$. For every protocol $\mathcal{P} = (Q, T, L, X, I, O)$ with leaders that computes some predicate φ , there exists a leaderless protocol \mathcal{P}' with $\mathcal{O}(|Q|^{|L|+1} \cdot |X| \cdot \ell^2)$ states that computes $(|x| < \ell) \rightarrow \varphi(x)$.*

Proof. Let $\mathcal{P} = (Q, T, L, X, I, O)$ be a protocol, with $|L|$ leaders initially in states $\langle l_1, \dots, l_{|L|} \rangle = L$, that computes some predicate φ . We construct a leaderless protocol $\mathcal{P}' = (Q', T', X, I', O')$, with $\mathcal{O}(|Q|^{|L|+1} \cdot |X| \cdot \ell^2)$ states, that computes $\varphi(x) \vee (|x| > \ell)$, which is equivalent to the desired predicate $(|x| < \ell) \rightarrow \varphi(x)$.

The $|L|$ leaders are simulated by one agent we refer to as *leader agent*. The leader agent is determined by a leader election. In general, the agents cannot know when the leader agent is finally elected, and so agents cannot wait for the leader election to be finished before starting their computation. However, as long as the population is sufficiently small, the leader agent may count the population size and reset the population to initial before starting the computation.

The leader agent simulates both the $|L|$ leaders plus an additional regular agent in a *multi-leader* state. Multi-leader states are defined as $Q_L \stackrel{\text{def}}{=} (Q^{|L|} \times [\ell] \times [\ell] \times X \times Q)$. Multi-leader states contain a representation of the states of the $|L|$ leaders, plus meta-data needed for additional bookkeeping: The leader agent stores a lower-bound estimate of the population size and a number that indicates how many agents need to be reset to initial. The estimate of the population size indicates how many agents need to be reset after the leader agent has been elected, while resetting agents to initial ensures that the computation starts from a sane configuration. In the case where the estimate of the population size exceeds ℓ , the leader agent moves to state \top that converts everyone to true, thereby ensuring stabilization to consensus 1. Multi-leader states are thus tuples of the form $\mathbf{l} = (\text{leader}_1, \dots, \text{leader}_{|L|}, \text{popsize}, \text{resetcounter}, \text{init}, q) \in Q_L$ where:

- leader_i is the current state of the i^{th} leader simulated by the leader agent (where $1 \leq i \leq |L|$),
- $\text{popsize} \in [\ell]$ is a counter for the population size,
- $\text{resetcounter} \in [\ell]$ counts how many agents have been reset,
- $\text{init} \in X$ stores the initial input of the regular agent simulated by the leader agent,
- $q \in Q$ stores the current state of the regular agent represented by the leader agent.

For every $\mathbf{l} \in Q_L$, we denote by $\mathbf{l}[\text{attr} := x]$ the state \mathbf{l}' that is identical to \mathbf{l} , except that $\mathbf{l}'(\text{attr}) = x$. For example, $\mathbf{l}[\text{popsize} := 10]$ denotes the state \mathbf{l}' that is identical to \mathbf{l} , except that $\mathbf{l}'(\text{popsize}) = 10$.

States. The set of states is

$$Q' \stackrel{\text{def}}{=} Q_L \cup (X \times Q \times \{\text{frozen}, \text{active}\}) \cup \{\top\}.$$

An agent is thus either:

- a leader in a multi-leader state of Q_L ;
- a leader or non-leader in state \top , which converts every agent to \top ; or
- a non-leader in a state of the form (x, q, s) with $s \in \{\text{frozen}, \text{active}\}$. The value of x is the initial input the agent came from, and it never changes. The value of q represents the current state of the agent from the original protocol. The value of s determines whether the agent can interact with other non-leader agents: If $s = \text{frozen}$, then the agent is “frozen” and cannot interact, otherwise it can interact freely with other agents.

Inputs. For every $x \in X$, we set the input mapping to:

$$I'(x) \stackrel{\text{def}}{=} (l_1, \dots, l_{|L|}, 1, 1, x, I(x)).$$

So initially every agent is a leader agent.

Outputs. We set the opinions of the states to:

$$\begin{aligned} O'(\mathbf{l}) &\stackrel{\text{def}}{=} O(\mathbf{l}(q)) && \text{for every } \mathbf{l} \in Q_L, \\ O'((i, q, x)) &\stackrel{\text{def}}{=} O(q) && \text{for every } (i, q, x) \in I \times Q \times \{\text{active}, \text{frozen}\}, \\ O'(\top) &\stackrel{\text{def}}{=} 1. \end{aligned}$$

Election of the leader agent. For every $\mathbf{l}, \mathbf{l}' \in Q_L$ s.t. $\max(\mathbf{l}(\text{popsize}), \mathbf{l}'(\text{popsize})) < \ell$, we add the following transition to T' :

$$\wr(\mathbf{l}, \mathbf{l}') \mapsto \wr(l_1, \dots, l_{|L|}, \mathbf{l}(\text{popsize}) + 1, 1, \mathbf{l}(\text{init}), \mathbf{l}(\text{init}), (\mathbf{l}'(\text{init}), I(\mathbf{l}'(\text{init})), \text{frozen})).$$

This implements a leader election; by fairness, one leader agent eventually remains.

Initiating conversion to \top . For every $\mathbf{l}, \mathbf{l}' \in Q_L$ s.t. $\max(\mathbf{l}(\text{popsize}), \mathbf{l}'(\text{popsize})) = \ell$, we add the following transition to T' :

$$\wr(\mathbf{l}, \mathbf{l}') \mapsto \wr(\top, \top).$$

This transition ensures that if the population size is at least ℓ , then all agents are eventually converted to \top , thus yielding a stable 1-consensus.

Conversion to 1-consensus. For every $q \in Q'$, we add the following transition to T' :

$$\wr(\top, q) \mapsto \wr(\top, \top).$$

This transition ensures that all agents eventually move to \top when one agent reaches \top .

Interactions with leaders. For every $\mathbf{l} \in Q_L$ s.t. $\mathbf{l}(\text{popsize}) = \mathbf{l}(\text{resetcounter})$, every $x \in X$ and every $r \in Q$, we add the following transitions to T' :

$$\begin{aligned} \wr(\mathbf{l}, (x, r, \text{active})) &\mapsto \wr(\mathbf{l}[q := q'], (x, r', \text{active})) && \text{for every } \wr(\mathbf{l}(q), r) \mapsto \wr(q', r') \in T, \\ \wr(\mathbf{l}, (x, r, \text{active})) &\mapsto \wr(\mathbf{l}[\text{leader}_i := \mathbf{l}'], (x, r', \text{active})) && \text{for every } 1 \leq i \leq |L|: \\ &&& \wr(\mathbf{l}(\text{leader}_i), r) \mapsto \wr(\mathbf{l}', r') \in T. \end{aligned}$$

This simulates interactions with leaders.

Interactions among regular agents. For every $(x, q), (y, r) \in X \times Q$ such that $\wr(q, r) \mapsto \wr(q', r') \in T$ for some q', r' , we add the following transition to T' :

$$\wr((x, q, \text{active}), (y, r, \text{active})) \mapsto \wr((x, q', \text{active}), (y, r', \text{active})).$$

This simulates interactions between non-leader agents.

Freezing agents. For every $\mathbf{l} \in Q_L$ such that $\mathbf{l}(\text{resetcounter}) < \mathbf{l}(\text{popsize})$, and every $(x, q) \in X \times Q$, we add the following transitions to T' :

$$\begin{aligned} \wr(\mathbf{l}, (x, q, \text{active})) &\mapsto \wr(\mathbf{l}[\text{resetcounter} := 1], (x, I(x), \text{frozen})), \\ \wr(\mathbf{l}, (x, q, \text{frozen})) &\mapsto \wr(\mathbf{l}[\text{resetcounter} := \mathbf{l}(\text{resetcounter}) + 1], (x, q, \text{active})). \end{aligned}$$

These transitions take care of freezing/activating agents and resetting agents to initial. Intuitively, the leader agent resets active agents by resetting their states to initial while simultaneously freezing them. Thus the following invariant is maintained: whenever an agent is frozen, it is in its initial state. The reset counter indicates how many frozen agents need to be activated: If the counter equals i , then $\mathbf{l}(\text{popsize}) - i$ agents must be activated. Ideally, the leader agents resets the population by first freezing all agents, one after another, and then activating each agent one by one. Of course, this order of steps cannot be guaranteed, but if it is violated, then by fairness $\mathbf{l}(\text{resetcounter})$ is eventually set to 1, and freezing/resetting is reinitiated. \blacktriangleleft

C.2 Proof of Theorem 11

► **Theorem 11.** *Let $k, i \in \mathbb{N}$. Let φ be a boolean combination of atomic predicates $(\varphi_j)_{j \in [k]}$. Assume that for every $j \in [k]$, there is a simple halting protocol $\mathcal{P}_j = (Q_j, L_j, X, T_j, I_j, O_j)$ with one leader computing $(\varphi_j \mid i)$. Then there exists a simple halting protocol \mathcal{P} that computes $(\varphi \mid i)$, with one leader and $\mathcal{O}(|X| \cdot (\text{len}(\varphi) + |Q_1| + \dots + |Q_k|))$ states.*

Proof. Without loss of generality, we assume that the state sets of $\mathcal{P}_1, \dots, \mathcal{P}_k$ are pairwise disjoint. For every $j \in [k]$, let \mathbf{f}_j and \mathbf{t}_j denote the output states of protocol \mathcal{P}_j . Further let l_j be the initial leader state of protocol \mathcal{P}_j , i.e. $\mathbf{l}_j = l_j$.

Remember that our final protocol \mathcal{P} should evaluate the outcomes of the individual protocols in succession. To this end, we enrich all states of $Q_1 \cup \dots \cup Q_k$ with a tag in $Y_{\text{tag}} \stackrel{\text{def}}{=} X \cup \{\square\}$. Intuitively, each agent is “tagged” with the input variable it corresponds to or with \square if it was the leader. This way, when a protocol \mathcal{P}_j halts, one can rewind to the initial configuration and start the next protocol \mathcal{P}_{j+1} .

Formally, for a given protocol \mathcal{P}_j , the *tagged protocol* \mathcal{P}_j^Y is given as $(Q_j^Y, L_j^Y, X, T_j^Y, I_j^Y, O_j^Y)$ where

$$\begin{aligned} Q_j^Y &\stackrel{\text{def}}{=} Y \times Q_j, \\ L_j^Y &\stackrel{\text{def}}{=} \mathbf{l}(\square, l_j), \\ T_j^Y &\stackrel{\text{def}}{=} \{\mathbf{l}(x, q), (y, r) \mapsto \mathbf{l}(x, q'), (y, r') \mid \mathbf{l}(q, r) \mapsto \mathbf{l}(q', r') \in T_j\}, \\ I_j^Y(x) &\stackrel{\text{def}}{=} (x, I_j(x)) \quad \text{for every } x \in X, \\ O_j^Y((x, q)) &\stackrel{\text{def}}{=} O_j(q) \quad \text{for every } (x, q) \in Q_j^Y. \end{aligned}$$

Notice that \mathcal{P}_j^Y is no longer simple, because we will have multiple states (x, \mathbf{f}_j) with output 0 and (x, \mathbf{t}_j) with output 1, one per $x \in Y_{\text{tag}}$. We will say that the intermediate tagged protocols \mathcal{P}_j^Y are *tagged-simple*. However, it is easy to recover a simple halting protocol from a tagged-simple halting protocol: we can add two states \mathbf{f}, \mathbf{t} and transitions $\mathbf{l}(x, \mathbf{f}) \mapsto \mathbf{l}(\mathbf{f})$ and $\mathbf{l}(x, \mathbf{t}) \mapsto \mathbf{l}(\mathbf{t})$ for all $x \in Y_{\text{tag}}$. It thus suffices to show how to combine the individual tagged-simple protocols to a tagged-simple protocol of appropriate size.

We show by induction on $\text{len}(\varphi)$: for every boolean combination of φ of atomic predicates $\varphi_1, \dots, \varphi_k$, there exists a tagged-simple halting protocol \mathcal{P}' with $\mathcal{O}(|X| \cdot (\text{len}(\varphi) + |Q_1| + \dots + |Q_k|))$ states and one leader that computes $(\varphi \mid i)$. By the previous remark, the claim entails the theorem to be shown.

The case $\text{len}(\varphi) = 0$ is trivial, since φ is computed by \mathcal{P}_j^Y for some j if $\text{len}(\varphi) = 0$ holds. For the induction, consider $\varphi = \varphi_1 \oplus \varphi_2$ for $\oplus \in \{\wedge, \vee\}$, and assume the existence of tagged-simple protocols $\mathcal{P}'_1, \mathcal{P}'_2$ that satisfy the claim for φ_1 and φ_2 , respectively. We construct a protocol $\mathcal{P}_{\oplus} = (Q, T, L, X, I, O)$ that computes $(\varphi \mid i)$ as follows.

States and associated mappings. We define states of \mathcal{P}_\oplus as:

$$Q \stackrel{\text{def}}{=} (Q'_1 \cup Q'_2) \cup \{(x, \mathbf{f}), (x, \mathbf{t}) \mid x \in Y_{\text{tag}}\}.$$

The leader multiset corresponds to the tagged leader multiset of \mathcal{P}'_1 :

$$L \stackrel{\text{def}}{=} \wr(\square, l_1)\S.$$

The output mapping is given by $O((x, \mathbf{f})) \stackrel{\text{def}}{=} 0$, $O((x, \mathbf{t})) \stackrel{\text{def}}{=} 1$ for every $x \in Y$, and $O(\mathbf{q}) \stackrel{\text{def}}{=} \perp$ for every other $\mathbf{q} \in Q$. The input mapping is defined as $I(x) \stackrel{\text{def}}{=} I'_1(x)$ for every $x \in X$.

Transitions. The set of transitions is $T = T'_1 \uplus T'_2 \uplus T'$, where T' is constructed as follows: For every $(x, q) \in Q'_1$, $(y, r) \in Q'_2$, we add the following transitions to T' :

$$\wr(x, q), (y, r)\S \mapsto \begin{cases} \wr I'_2(x), (y, r)\S & \text{if } x \neq \square \\ \wr(x, l'_2), (y, r)\S & \text{if } x = \square \end{cases}$$

These transitions make sure that once at least one agent is promoted to a state in the higher protocol \mathcal{P}'_2 , all agents eventually simulate the execution of protocol \mathcal{P}'_2 .

Moreover, we add the following transitions to T' : for every $x \in Y_{\text{tag}}$, depending on the operator \oplus :

- If $\oplus = \wedge$, we add the following transitions for each $x \in Y_{\text{tag}}$:

$$\begin{aligned} \wr(x, \mathbf{f}'_j)\S &\mapsto \wr(x, \mathbf{f})\S && \text{for every } j \in \{1, 2\}, \\ \wr(x, \mathbf{t}'_1)\S &\mapsto \begin{cases} \wr I'_2(x)\S & \text{if } x \neq \square \\ \wr(x, l'_2)\S & \text{if } x = \square \end{cases}, \\ \wr(x, \mathbf{t}'_2)\S &\mapsto \wr(x, \mathbf{t})\S. \end{aligned}$$

- If $\oplus = \vee$, we add the following transitions for each $x \in Y_{\text{tag}}$:

$$\begin{aligned} \wr(x, \mathbf{t}'_j)\S &\mapsto \wr(x, \mathbf{t})\S && \text{for every } j \in \{1, 2\}, \\ \wr(x, \mathbf{f}'_1)\S &\mapsto \begin{cases} \wr I'_2(x)\S & \text{if } x \neq \square \\ \wr(x, l'_2)\S & \text{if } x = \square \end{cases}, \\ \wr(x, \mathbf{f}'_2)\S &\mapsto \wr(x, \mathbf{f})\S. \end{aligned}$$

These transitions ensure that once an output state is reached in the simulation of a given protocol, then either its output is returned as final output (in the case where $\oplus = \vee$ and output of the protocol is 1, or $\oplus = \wedge$ and output of the protocol is 0), or the simulation of the second protocol is initiated, until its output is returned, and \mathcal{P} satisfies the claim by induction hypothesis. Note that each inductive call adds $2|Y_{\text{tag}}|$ states, which results in the bound given in our theorem. \blacktriangleleft

C.3 Proof of Theorem 12

We consider only the case $\varphi(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \alpha \cdot \mathbf{x} - \beta \cdot \mathbf{y} > 0$; the general case $> c \in \mathbb{Z}$ is easily adapted from there. We explain later how to adapt the proof to handle remainders predicate $\varphi(\mathbf{x}) \stackrel{\text{def}}{=} (\alpha \cdot \mathbf{x} \equiv_m b)$ with $m \in \mathbb{N}$ and $0 \leq b \leq m$.

► **Theorem 12.** *Let $\varphi(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} \alpha \cdot \mathbf{x} - \beta \cdot \mathbf{y} > 0$. For every $i \in \mathbb{N}$, there exists a halting protocol with one leader and $\mathcal{O}(i^2(|\varphi| + \log i)^3)$ states that computes $(\varphi \mid i)$.*

Input: Finite Multiset $Z \in \mathbb{N}^{\mathbb{N}}$,
bit position $j \in [m]$

Output: j^{th} bit of $\sum Z$

```

1: procedure PROBE( $Z, j$ )
2:    $\text{val} \leftarrow 0$ 
3:   for  $\text{pos} = 1$  to  $j$  do
4:      $\text{val} \leftarrow \text{val} \text{ div } 2$ 
5:     for all  $z \in Z$  do
6:        $\text{val} \leftarrow \text{val} + \text{bin}(z)_{\text{pos}}$ 
7:     end for
8:   end for
9:   return  $\text{val} \bmod 2$ 
10: end procedure

```

Input: Finite multisets $X, Y \in \mathbb{N}^{\mathbb{N}}$

Output: boolean ($\sum X > \sum Y$)

```

1: procedure GREATER-SUM( $X, Y$ )
2:   for  $\text{tgt} = m$  to  $1$  do
3:      $\text{val}_X \leftarrow \text{PROBE}(X, \text{tgt})$ 
4:      $\text{val}_Y \leftarrow \text{PROBE}(Y, \text{tgt})$ 
5:     if  $\text{val}_X \neq \text{val}_Y$  then
6:       return  $\text{val}_X > \text{val}_Y$ 
7:     end if
8:   end for
9:   return  $\text{false}$ 
10: end procedure

```

■ **Figure 2** Procedure PROBE(Z, j) probes the j^{th} bit of the sum of the elements of $Z \in \mathbb{N}^{\mathbb{N}}$. It implements a binary adder, but instead of storing the result of the addition, it only keeps the carry in variable val when moving from one bit position to the next.

Procedure GREATER-SUM(X, Y) compares the sums of the elements of $X, Y \in \mathbb{N}^{\mathbb{N}}$. It probes the bits of the two sums, starting with the most-significant bit, until it finds the first position at which the bits of the two sums differ. If there is no such position, the sums are equal and the algorithm returns *false*.

Proof. Let $A \stackrel{\text{def}}{=} \{\alpha_j : j \in \{1, \dots, |\alpha|\}\}$ and $B \stackrel{\text{def}}{=} \{\beta_j : j \in \{1, \dots, |\beta|\}\}$. Let m be the maximal bit length of any number in the set $\{x_1 + \dots + x_n : x_j \in A\} \cup \{y_1 + \dots + y_n : y_j \in B\}$. Note that $m \in \mathcal{O}(\log(i \cdot 2^{|\varphi|})) = \mathcal{O}(|\varphi| + \log i)$. For any $a < 2^m$, we write $\text{bin}(a)$ to denote the least-significant-bit-first binary representation of a , padded to length m with leading 0s. Whenever $\text{bin}(a) = b_m b_{m-1} \dots b_1$, we write $\text{bin}(a)_j$ to denote b_j for every $j \in [m]$.

Consider the sequential algorithm GREATER-SUM(X, Y) shown in Figure 2. We have $\alpha \cdot x - \beta \cdot y > 0$ iff GREATER-SUM(A, B) returns true. So it suffices to exhibit a protocol that simulates the execution of GREATER-SUM(A, B) for inputs of size i . Intuitively, the protocol has a leader that executes the procedure. The leader stores the values of the variables defined in GREATER-SUM. Regular agents store the input and one additional bit that indicates whether the leader has met the agent in the current round. The leader can set and unset this bit, which permits the implementation of a *for all* loop: the leader stores how many agent it has met in the current iteration of the loop. Whenever the leader encounters an agent whose bit is set to 0, it flips the bit to 1, increments its counter, and performs the variable updates defined in the body of the loop. When the counter value reaches i , the leader knows that the current iteration of the loop is complete. The leader then unsets all bits of the regular agents while decrementing its counter agent by agent, before starting the next iteration of the loop when the counter value reaches zero.

We now define the protocol $\mathcal{P} = (Q, L, T, I, O)$ formally.

States. Let $Q \stackrel{\text{def}}{=} Q' \cup Q_L$ where $Q' \stackrel{\text{def}}{=} (A \cup B) \times \{0, 1\}$ is the set of states for regular agents, and $\{\mathbf{f}, \mathbf{t}\} \subseteq Q_L$ is a set of leader states yet to be specified.

The leader multiset L contains exactly one leader. Unless the leader is in one of the output states $\{\mathbf{f}, \mathbf{t}\}$, it stores the following values:

- $\text{tgt} \in [m]$: the target bit position to be probed; corresponds to the loop counter tgt in line 2 of Procedure GREATER-SUM.
- $\text{pos} \in [m]$: the current bit position; corresponds to the loop counter pos in line 3 of Procedure PROBE.

- $\text{met} \in \{0, \dots, i-1\}$: the number of agents the leader has met in the current round. This is needed for the implementation of the *forall* loop in Procedure PROBE.
- $\text{reset} \in \{0, 1\}$: indicates whether the bit flag of each regular agent should be reset.
- $\text{val}_Z \in [i \cdot m]$ for every $Z \in \{X, Y\}$: storage for sum of bits from binary representations of numbers in A and B , respectively; corresponds to val_X val_Y in Procedure GREATER-SUM, and val in Procedure PROBE.

Initially, the variables of the leader are set as follows: $\text{tgt} = m, \text{pos} = 1, \text{met} = 0, \text{reset} = 0, \text{val}_X = \text{val}_Y = 0$. Note that this corresponds to the initial values of the variables in the procedures PROBE and GREATER-SUM. Thus, we set:

$$Q_L \stackrel{\text{def}}{=} [m] \times [m] \times \{0, \dots, i-1\} \times \{0, 1\} \times [i \cdot m],$$

$$L \stackrel{\text{def}}{=} \langle (m, 1, 0, 0, 0, 0) \rangle.$$

Size. The number of states is $|Q| = |Q'| + |Q_L| = 2 * |A \cup B| + 2m^3i^2 \in \mathcal{O}(i^2(|\varphi| + \log i)^3)$.

Input and output mappings. We define the input mapping I as:

$$I(x_j) \stackrel{\text{def}}{=} (\alpha_i, 0) \quad \text{for every } 1 \leq j \leq |\alpha|,$$

$$I(y_j) \stackrel{\text{def}}{=} (\beta_i, 0) \quad \text{for every } 1 \leq j \leq |\beta|.$$

The output mapping O is defined as:

$$O(\mathbf{f}) \stackrel{\text{def}}{=} 0,$$

$$O(\mathbf{t}) \stackrel{\text{def}}{=} 1,$$

$$O(q) \stackrel{\text{def}}{=} \perp \quad \text{for every } q \in Q \setminus \{\mathbf{f}, \mathbf{t}\}.$$

Transitions. For a state $\mathbf{q} = (\gamma, b) \in Q'$, let $\mathbf{q}(\gamma)$ denote γ and let $\mathbf{q}(b)$ denote b .

To implement resetting the bit flag of the regular agents, we add the following transitions for every $\mathbf{q} \in Q'$ and $\mathbf{l} \in Q_L$ where $\mathbf{q}(b) = 1$ and $\mathbf{l}(\text{reset}) = 1$:

$$\langle \mathbf{q}, \mathbf{l} \rangle \mapsto \langle \mathbf{q}[b := 0], \mathbf{l}[\text{met} := (\mathbf{l}(\text{met}) + 1 \bmod (i-1)); \text{reset} := \min(1, (\mathbf{l}(\text{met}) + 1) \bmod i)] \rangle.$$

We now define the remaining transitions for the execution of procedure GREATER-SUM. Let $\mathbf{q} \in Q'$ and $\mathbf{l} \in Q_L \setminus \{\mathbf{f}, \mathbf{t}\}$. Let us first establish some abbreviations.

Let:

$$Z \stackrel{\text{def}}{=} \begin{cases} X & \text{if } \mathbf{q}(\gamma) \in A, \\ Y & \text{if } \mathbf{q}(\gamma) \in B. \end{cases}$$

Further let $\mathbf{l}_{\text{incr}} \stackrel{\text{def}}{=} \mathbf{l}[\text{val}_Z := \mathbf{l}(\text{val}_Z) + \text{bin}(\mathbf{q}(\gamma))_{\text{pos}}]$. Intuitively, \mathbf{l}_{incr} represents the update to the leader state that results from the incrementation in line 6 of Procedure PROBE.

Whenever $\mathbf{q}(b) = 0$ and $\mathbf{l}(\text{reset}) = 0$, we add the following transitions:

$$\langle \mathbf{q}, \mathbf{l} \rangle \mapsto \langle \mathbf{q}[b := 1], \mathbf{l}' \rangle$$

where \mathbf{l}' is specified in Table 1. ◀

Remainder. Consider now a predicate $\varphi(\mathbf{x}) \stackrel{\text{def}}{=} (\alpha \cdot \mathbf{x} \equiv_m b)$ with $m \in \mathbb{N}$ and $0 \leq b \leq m$. We show that for every $i \in \mathbb{N}$, there exists a halting protocol with one leader and $\mathcal{O}(\text{poly}(|\varphi| + i))$ states that computes $(\varphi \mid i)$.

| Conditions satisfied by l | Value of l' | Corresponds to |
|--|--|--|
| $l(\text{met}) < i - 1$ | $l_{\text{incr}}[\text{met} := l(\text{met}) + 1]$ | Line 6 of Procedure PROBE. |
| $l(\text{met}) = i - 1$ $l(\text{pos}) < l(\text{tgt})$ | $l_{\text{incr}}[\text{val}_X := \text{val}_X \text{ div } 2;$ $\text{val}_Y := \text{val}_Y \text{ div } 2;$ $\text{pos} := l(\text{pos}) + 1;$ $\text{met} := 0;$ $\text{reset} := 1]$ | Continuation of for loop in line 4 of Procedure PROBE. |
| $l(\text{met}) = i - 1,$ $l(\text{pos}) = l(\text{tgt}),$ $l_{\text{incr}}(\text{val}_X) \neq l_{\text{incr}}(\text{val}_Y)$ | \mathbf{t} if $l_{\text{incr}}(\text{val}_X) > l_{\text{incr}}(\text{val}_Y),$ \mathbf{f} otherwise. | Return statement in line 6 of Procedure GREATER-SUM. |
| $l(\text{met}) = i - 1,$ $l(\text{pos}) = l(\text{tgt}) > 1,$ $l_{\text{incr}}(\text{val}_X) = l_{\text{incr}}(\text{val}_Y).$ | $l[\text{tgt} := l(\text{tgt}) - 1;$ $\text{pos} = 1;$ $\text{met} = 0;$ $\text{reset} = 1]$ | Continuation of the for loop in Procedure GREATER-SUM |
| Other | \mathbf{f} | Return statement in line 9 of Procedure GREATER-SUM. |

■ **Table 1** Transitions of the protocol implementing GREATER-SUM.

The protocol in which a leader interacts with every other agent, storing in its state the value of $\alpha \cdot v' \bmod m$, where v' is the vector of the agents it has already interacted with, does not work: For $m \in \Theta(|\varphi|)$, which can be the case, this requires $\mathcal{O}(2^{|\varphi|})$ states. So we proceed in a different way.

► **Theorem 36.** *Let $\varphi(x) \stackrel{\text{def}}{=} (\alpha \cdot x \equiv_m b)$ with $b, m \in \mathbb{N}$, $0 \leq b < m$ and $\alpha \in \mathbb{Z}^{|X|}$. For every $i \in \mathbb{N}$, there exists a halting protocol with one leader and $\mathcal{O}(|X| \cdot i^3(|\varphi| + \log i)^3)$ states that computes $(\varphi \mid i)$.*

Proof. Let $\alpha' \stackrel{\text{def}}{=} (\alpha(1) \bmod m, \dots, \alpha(|X|) \bmod m)$. Since $\alpha \cdot x \equiv_m \alpha' \cdot x$, setting $\varphi'(x) \stackrel{\text{def}}{=} (\alpha' \cdot x \equiv_m b)$ yields $\varphi(x) = \varphi'(x)$ for every input x . Consider an input x of size i . We have $\alpha' \cdot x \leq m \cdot i$, hence $\varphi'(x) = 1$ iff $\alpha' \cdot x \in \{b, m + b, \dots, (i - 1)m + b\}$, and consequently:

$$\varphi(x) \equiv \varphi'(x) \equiv \bigvee_{j=0}^{i-1} (\alpha' \cdot x = j \cdot m + b),$$

which is a disjunction of i threshold predicates $\varphi'_1, \dots, \varphi'_i$. For every $j \in [i]$ it holds:

$$\begin{aligned} |\varphi'_j| &\in \mathcal{O}(\log\|\varphi'_j\| + \text{len}(\varphi'_j) + |X|) \\ &\subseteq \mathcal{O}(\log(\|\varphi\| \cdot i) + \text{len}(\varphi) + |X|) \\ &= \mathcal{O}(\log i + \log\|\varphi\| + \text{len}(\varphi) + |X|) \\ &= \mathcal{O}(\log i + |\varphi|) \end{aligned}$$

By Lemma 11 there is a protocol \mathcal{P}_j computing $(\varphi'_j \mid i)$ with

$$\begin{aligned} &\mathcal{O}(i^2(|\varphi'_j| + \log i)^3) \\ &\in \mathcal{O}(i^2(|\varphi| + 2\log i)^3) \\ &\in \mathcal{O}(i^2(|\varphi| + \log i)^3) \end{aligned}$$

states. By Theorem 12, $(\varphi' \mid i)$ can be computed by a protocol with

$$\begin{aligned} &\mathcal{O}(|X| \cdot (i + i \cdot (i^2(|\varphi| + \log i)^3))) \\ &\in \mathcal{O}(|X| \cdot i^3(|\varphi| + \log i)^3) \end{aligned}$$

states. ◀

D

 Proof of Theorem 13

► **Theorem 13.** *For every polynomial p , every algorithm that accepts a formula φ of PA as input, and returns a population protocol computing φ , runs in time $2^{\omega(p(|\varphi|))}$.*

Proof. We show that if such an algorithm runs in time $2^{p(n)}$ for some polynomial p , then the validity problem for PA formulas is in EXPTIME, contradicting the fact that its complexity lies between 2-NEXP and 2-EXPSpace [7, 12]. Recall that the validity problem for PA formulas asks whether a given sentence, i.e., a formula without free variables, is true or false.

Let φ be a sentence of PA, and let $n \stackrel{\text{def}}{=} |\varphi|$. Consider the formula $\psi(x) \stackrel{\text{def}}{=} (x \geq 2) \wedge \varphi$ (notice that the smallest possible size of a population is 2). Clearly, φ is valid iff $\psi(2)$ holds. Assume there exists an algorithm that on input ψ executes at most $f(n)$ steps and outputs a population protocol \mathcal{P} that computes ψ . Clearly, \mathcal{P} has at most $\mathcal{O}(f(n))$ states, and φ is valid iff \mathcal{P} computes 1 for input $x = 2$.

Let C be the initial configuration of \mathcal{P} for input $x = 2$. A configuration of \mathcal{P} with two agents can be stored in space $\mathcal{O}(\log f(n))$, and so C , and every configuration reachable from it, can be stored using $\mathcal{O}(\log f(n))$ space. Protocol \mathcal{P} computes 1 from C iff there exists a configuration C' such that:

- (i) $C \xrightarrow{*} C'$,
- (ii) C' has output 1,
- (iii) for every configuration C'' , if $C' \xrightarrow{*} C''$, then $C'' \xrightarrow{*} C'$.

Observe that (i)-(iii) can be expressed in FO(TC), i.e. first-order logic with transitive-closure. By Immermann's theorem, deciding (i)-(iii) belongs to NSPACE($\log f(n)$) [14], and so it can be solved in $\mathcal{O}(f(n)^k)$ deterministic time for some $k \geq 1$. Consequently, if there exists a polynomial p such that $f \in \mathcal{O}(2^{p(n)})$, then the validity of φ can be decided in time $2^{\mathcal{O}(p(n))}$, and so the validity problem for PA is in EXPTIME. The latter is impossible by the time hierarchy theorem. ◀