Monadic Presburger Predicates Have Robust Population Protocols

Philipp Czerner
Technical University of Munich, Germany

Javier Esparza
Technical University of Munich, Germany

Vincent Fischer
Technical University of Munich, Germany

Roland Guttenberg
Technical University of Munich, Germany

Julian Pins
Technical University of Munich, Germany

Simon Reilich
Technical University of Munich, Germany

Abstract

Population protocols are a model of distributed computation in which a collection of indistinguishable finite-state agents interact randomly in pairs to decide a predicate of their initial configuration. The agents decide by achieving a stable consensus on whether the predicate holds or not. It is known that population protocols can decide exactly the predicates expressible in Presburger arithmetic.

Recently, Lossin et al. have introduced a notion of protocol robustness against adversarial crash failures. They show that all *atomic* Presburger predicates can be decided by robust protocols, and ask whether the same holds for *every* Presburger predicate. We make progress towards settling this question by proving that all predicates expressible in monadic Presburger arithmetic have robust protocols. In addition, we analyze the cost of robustness in terms of state complexity. We study the ratio between the number of states of the smallest robust protocol for a given predicate and the smallest protocol for it. We show that the cost of robustness is at least double exponential in the size of the predicate, and prove that the robust protocols by Lossin *et al.* for threshold predicates $x \geq k$ have optimal state complexity.

2012 ACM Subject Classification Theory of computation → Distributed computing models

Keywords and phrases Population protocols, fault-tolerance, state complexity

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2026.28

Related Version *Previous Version*: `https://arxiv.org/abs/2604.27767`

Funding This work was partially supported by project 551606821 "Verification and Synthesis of Dynamic Network Algorithms" of the German Research Council.
Vincent Fischer: Funded by the Research Training Group 2428 "ConVeY" of the German Research Council.

1 Introduction

Population protocols are a much studied model of distributed computation in which a collection of indistinguishable finite-state agents interact randomly in pairs with the goal of deciding a property of their initial configuration [4, 6, 2, 12]. The decision is taken by stable consensus: Eventually all agents agree on whether the property holds or not, and never change their mind again. We say that the protocol (stably) *converges* to "yes" or "no".

Since agents are indistinguishable, a configuration of a protocol can be modeled as a mapping assigning to each state q the number of agents currently in q. Given a protocol with initial states $q_1, \ldots, q_n$, a property of the protocol can be modeled as a predicate $P(x_1, \ldots, x_n)\colon \mathbb{N}^n \to \{0, 1\}$: The initial configurations satisfying the property are those mapped to 1 by P. It was shown in [5] that population protocols can decide exactly the Presburger predicates, that is, the predicates expressible in the theory $Th(\mathbb{N}, +, <, 0, 1)$.

Population protocols have been used to model ad hoc networks of tiny sensors, and chemical reaction networks, in which agents are molecules and interactions are chemical reactions. In both settings the number of agents can be very large, and so failures are bound to occur. This raises the question of which fragment of the Presburger predicates can be decided *robustly*, even in the presence of failures. In this paper we consider *adversarial crash failures* (called just *crashes* or *failures* in the following) that cause an agent to stop interacting with other agents. Adversarial means that the adversary can choose which agent crashes and at which moment during the execution of the protocol.

Robust decision under crash failures was first studied by Delporte-Gallet *et al.* [11]. Loosely speaking, they showed that all Presburger predicates can be decided robustly under the assumption that the number of failures is bounded by a constant, independent of the number of agents. In [17], Lossin *et al.* study if this assumption can be lifted. They propose a stronger notion of robustness. To give the intuition, consider the Presburger predicate $x \geq k$, corresponding to the property "the initial configuration contains at least k agents". A protocol deciding $x \geq k$ is robust if for every configuration with $k' > k$ agents the following holds: if at most $k' - k$ agents fail, the protocol converges to "yes". Observe that the number of failures a robust protocol must tolerate is no longer constant. Further, it is the best possible number. Indeed, assume that more than $k' - k$ agents fail. If the adversary makes them fail at the start of the protocol, before they interact with any other agent, the "survivors" do not even know that these agents exist, and so they cannot possibly converge to "yes".

Lossin *et al.* generalize this definition of robustness to arbitrary predicates. A protocol robustly decides a predicate $P(x_1, \ldots, x_n)$ if for every initial configuration $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$ and every initial configuration $b \leq a$ obtained by letting $a - b$ agents crash before the protocol starts, the following holds: if $P(a) = P(b)$, then the protocol converges to $P(a)$ even if the crash times of the agents of $a - b$ are chosen by the adversary. They ask the question whether every Presburger predicate can be robustly decided. They exhibit robust protocols for *threshold* and *modulo* predicates of the form $\sum_{i=1} a_i x_i \leq b$ for some $a_1, \ldots, a_n, b \in \mathbb{Z}$, and $\sum_{i=1} a_i x_i \leq b \pmod m$ for some $a_1, \ldots, a_n \in \mathbb{Z}$ and $b, m \in \mathbb{N}$ with $b < m$, respectively. However, whether every Presburger predicate can be robustly decided remains open.

Our first contribution shows that every *monadic* Presburger predicate can be robustly decided. A Presburger predicate is monadic if it is represented by a formula in which each threshold and modulo atomic formula contains at most one variable[1]. To our knowledge, this is the first fragment of Presburger arithmetic closed under logical operations for which robust decidability is proved.

In our second contribution we study the cost of robustness in terms of *state complexity*, defined as the number of states of the smallest protocol deciding a predicate. We obtain a lower bound for the state complexity of *asymptotically constant predicates* defined as the predicates $P(x_1, \ldots, x_n)$ for which there exists a k such that $P(x_1, \ldots, x_n)$ returns the same value for all inputs $(x_1, \ldots, x_n)$ satisfying $x_i \geq k$ for every $1 \leq i \leq n$. As a corollary, we show

[1] Notice that monadic formulas may have multiple variables; for example, $(x \leq 3 \vee y \geq 4) \wedge z \bmod 7 \leq 2$ is monadic.

that the smallest robust protocol for the predicate $x \geq k$ has exactly k states. It is known that $x \geq t$ is decidable by protocols with $\Theta(\log \log t)$ states for infinitely many $t \geq 0$, and every protocol needs $\Omega(\log \log t)$ [7, 10, 9, 16]. Therefore, the price of robustness in terms of state complexity is at least double exponential.

Related work. We have already discussed the work of [11] and [17] on crash failures. Our paper extends this line of work. There also exists work on other failure models. In [13], Guerraoui and Ruppert extend population protocols by assigning unique identities to agents and construct protocols for all predicates in NSPACE($\log n$) that are robust with respect to a constant number of Byzantine failures. Alistarh *et al.* have studied robustness of population protocols and chemical reaction networks with respect to failures in which agents can silently and probabilistically move to adversarially chosen states [1, 3]. Di Luna *et al.* study in [18, 19] omission faults in which an agent fails to read its partner's state during an interaction.

Structure of the paper. Section 2 introduces general notation and formally defines population protocols. Section 3 defines robustness of protocols and Section 4 presents robust protocols for monadic predicates. Section 5 then proves a lower bound on the state complexity of robust protocols. Section 6 contains conclusions.

2 Preliminaries

Natural Numbers. We denote the set of all natural numbers as $\mathbb{N} := \{0, 1, 2, \ldots\}$ and the set of all natural numbers modulo m as $\mathbb{Z}_m := \{0, 1, \ldots, m-1\}$. Additionally we define the set $[k] := \{1, 2, \ldots, k\}$.

Multisets. Let E be a finite set. A multiset over E is a mapping $E \to \mathbb{N}$, and $\mathbb{N}^E$ denotes the set of all multisets over E. We sometimes write multisets using set-like notation, e.g. $\wr a, 2 \cdot b \wr$ denotes the multiset v such that $v(a) = 1$, $v(b) = 2$ and $v(e) = 0$ for every $e \in E \setminus \{a, b\}$. The empty multiset $\wr \wr$ is also denoted $\emptyset$. For $E' \subseteq E$, $v(E') := \sum_{e \in E'} v(e)$ is the number of elements in v that are in E'. The *size* of $v \in \mathbb{N}^E$ is $|v| := v(E)$. The *support* of $v \in \mathbb{N}^E$ is the set $[\![v]\!] := \{e \in E \mid v(e) > 0\}$. Given $u, v \in \mathbb{N}^E$, we let $u \leq v$ denote that $u(e) \leq v(e)$ holds for every $e \in E$; further, we let $u + v$ and $u - v$ denote the multisets given by $(u + v)(e) := u(e) + v(e)$ and $(u - v)(e) := u(e) - v(e)$ for every $e \in E$. The latter is only defined if $u \geq v$.

Population Protocols. Let K be a finite set of *outputs*. A *population protocol* over K is a four-tuple $P = (Q, I, O, \delta)$, where Q is a finite set of *states*, $I \subseteq Q$ is the set of *initial states*, $O : Q \to K$ is the *output function*, and $\delta : \mathbb{N}_2^Q \to \mathbb{N}_2^Q$ is the *transition function*, where $\mathbb{N}_2^Q$ denotes the multisets over Q of size exactly 2.

If $\delta(\wr q_1, q_2 \wr) = \wr q_1', q_2' \wr$, then we write $q_1, q_2 \mapsto q_1', q_2'$ and call this expression a *transition*. Given a transition $t = q_1, q_2 \mapsto q_1', q_2'$, we let $pre(t) := \wr q_1, q_2 \wr$ and $post(t) := \wr q_1', q_2' \wr$. If $\delta(\wr q_1, q_2 \wr)$ is not explicitly defined, then we assume $\delta(\wr q_1, q_2 \wr) = \wr q_1, q_2 \wr$.

Configurations, steps, (fair) executions. A *configuration* of $P = (Q, I, O, \delta)$ is a multiset $C \in \mathbb{N}^Q$. $C(q)$ models the number of agents in state q, and $|C| := \sum_{q \in Q} C(q)$ denotes the total number of agents. C *populates* state q if $C(q) > 0$. The *support* of C, denoted $[\![C]\!]$, is the set of states populated by C. C is *initial* or an *input* if $[\![C]\!] \subseteq I$, that is, if it only

populates initial states. C can make a *step* to D, denoted $C \to D$, if $C = D$ or there exists a transition t such that $C \geq pre(t)$ and $D = C - pre(t) + post(t)$. We let $\xrightarrow{*}$ denote the reflexive transitive closure of $\to$. If $C \xrightarrow{*} D$, then we say that D is *reachable* from C.

An *execution* is an infinite sequence of configurations $\pi = C_0, C_1, C_2, \ldots$ such that $C_i \to C_{i+1}$ for all $i \in \mathbb{N}$. An execution π is *fair* if for every two configurations C, D, if C occurs infinitely often in π and $C \xrightarrow{*} D$ then D also occurs infinitely often in π.

Consensus, stable consensus, computed function. Let $r \in K$ be an output value. A configuration C is an *r-consensus* if $O(\llbracket C \rrbracket) = \{r\}$, that is, all agents have output r. Further, C is a *stable* r-consensus if every configuration reachable from C is also a r-consensus. The protocol P computes a function $f : \mathbb{N}^I \to K$ if for every input $C_0 \in \mathbb{N}^I$ every fair execution starting at $C_0 \in \mathbb{N}^I$ *converges* to $f(C_0)$, meaning that it eventually reaches a stable $f(C_0)$-consensus. If $K = \{0, 1\}$, then we also call f a *predicate* and say that the protocol *decides* f. Notice that not every protocol computes a function. Observe also that the arity of the function computed by a protocol, if any, is equal to its number of initial states.

▶ **Example 1.** We introduce Pebble, a simple protocol deciding the threshold predicate $x \geq k$ for some fixed $k \in \mathbb{N}$. Intuitively, each agent initially carries one pebble, and some agent eventually collects all pebbles from all agents. If the number of pebbles is at least k, then it tells other agents that the threshold has been reached.

Protocol Pebble**.**

States. $Q = \{0, 1, \ldots, k-1, k\}$. (Each agent may hold up to k pebbles.)
Input. $I = \{1\}$. (Each agent starts with one pebble.)
Transitions. For every $i, j \in Q$:
- $i, j \mapsto i + j, 0$ if $i + j < k$. (One agent gives its pebbles to the other.)
- $i, j \mapsto k, k$ if $i + j \geq k$. (State k models "I know the threshold is reached".)

Output. $O(q) =$ **if** $q = k$ **then** 1 **else** 0. (Agents in state k output 1, others 0.)

Computational power of population protocols. Presburger arithmetic is the first-order theory of addition, usually defined as the theory $\mathrm{Th}(\mathbb{N}, +, <, 0, 1)$. We consider here the equivalent theory whose atomic formulas are *threshold* formulas of the form $\sum_{i=1}^{n} a_i x_i \leq b$ for some $a_1, \ldots, a_n, b \in \mathbb{Z}$ and modulo formulas of the form $(\sum_{i=1}^{n} a_i x_i) \bmod m \leq b$ for some $a_1, \ldots, a_n \in \mathbb{Z}$ and $b, m \in \mathbb{N}$ with $b < m$. The theory is obtained by closing the atomic formulas under Boolean operations and existential quantification. The semantics of a formula $F(x_1, \ldots, x_n)$ with free variables $X = \{x_1, \ldots, x_n\}$ is the predicate $p_F : \mathbb{N}^X \to \{0, 1\}$ defined in the expected way, e.g. for $F(x, y) = 2x - 3y > 5$ we have $p_F(n_1, n_2) = 1$ iff $2n_1 - 3n_2 > 5$; the predicates of Presburger formulas are called Presburger predicates. This version of Presburger arithmetic has a quantifier elimination procedure, and so every Presburger formula is equivalent to a Boolean combination of threshold and modulo formulas (resp. predicates) [14].

Let K be a finite set. A function $f \colon \mathbb{N}^X \to K$ is a *Presburger function* if for every $i \in K$ there is a Presburger formula $F_i(x_1, \ldots, x_n)$ such that $f(a_1, \ldots, a_n) = i$ iff $p_{F_i}(a_1, \ldots, a_n) = 1$. Angluin *et al.* show in [5] that a function $f \colon \mathbb{N}^I \to K$ is computable by a protocol with I as set of initial states iff f is Presburger. In particular, a predicate $f \colon \mathbb{N}^I \to \{0, 1\}$ can be computed or *decided* by a population protocol iff f is a Presburger predicate.

3 Snipes and robustness

The above notions capture the failure-free behavior of a protocol. In practice, however, agents may crash in an adversarial way. We introduce the failure model of [17] and a notion of robustness. The notion is slightly stronger than the one of [17]. At the end of the section we explain the difference and why a new notion is needed.

In [17], failures are modeled as *snipes*. Intuitively, a snipe deletes an agent from a configuration.

▶ **Definition 2** (Snipes, executions with snipes). *Let C, D be configurations. If $D \leq C$ and $|C - D| = 1$ we write $C \hookrightarrow D$ and call this expression a* snipe step*, or just a* snipe*. An* execution with at most k snipes *is an infinite sequence of configurations, $\pi = C_0, C_1, C_2, \ldots$ such that $C_i \to C_{i+1}$ or $C_i \hookrightarrow C_{i+1}$ for all $i \in \mathbb{N}$ and the number of indices i where $C_i \hookrightarrow C_{i+1}$ is at most k. The sequence π is* fair *if for every two configurations C, D, if C occurs infinitely often in π and $C \xrightarrow{*} D$ then D also occurs infinitely often in π.*

We introduce a notion of robustness. Intuitively, a protocol computing a function $f : \mathbb{N}^I \to K$ is robust if for any input A and any number $j \geq 0$ of snipes taking place at adversarial times, the output of the protocol coincides with the output for some input $A' \leq A$ such that $|A' - A| \leq j$. For example, let $f(x) := \min(x, 6)$ and take the input $x := 7$. Then, for 1 snipe a robust protocol must output 6, for two snipes 6 or 5, for three snipes 6, 5, or 4, etc. This fault-tolerance is optimal. Consider for example the case of two snipes. No protocol can guarantee that the output will be only 6, because the adversary can snipe two agents *before the agents start to interact at all* and the five remaining agents, ignorant that two agents have crashed, can only reach consensus 5. Similarly, no protocol can guarantee that the output will be exactly 5, because the adversary can snipe one agent at the beginning, wait until the remaining agents reach stable consensus 6, and then snipe a second agent; since the consensus with 6 agents was stable, after sniping one agent the remaining configuration is still a stable consensus.

▶ **Definition 3** (Robustness). *A population protocol computing a function $f : \mathbb{N}^I \to K$ is* robust *if for every initial configuration $A \in \mathbb{N}^I$, every $j < |A|$, and every fair execution with j snipes starting at A, there exists an output value $r \in K$ such that*

- *the execution eventually reaches an r-stable consensus, and*
- *$f(A') = r$ for some initial configuration $A' \leq A$ satisfying $|A - A'| \leq j$.*

We say that r is a permissible output *for A and j.*

Remark. We limit j to be less than $|A|$, since if all agents are sniped, the output is no longer defined at all.

The following example is taken from [17].

▶ **Example 4.** The Pebble family of protocols from Example 1 is not robust. Consider the Pebble protocol for $x \geq 100$, and the initial configuration with 199 agents in state 1. If the protocol were robust, then it should always output 1 for any number of snipes between 0 and 99. However, one single snipe may suffice to change the output. Indeed, it is easy to see that the configuration in which two agents have 100 and 99 pebbles, respectively, and the rest have 0 pebbles, is reachable from the initial configuration. If the agent with 100 pebbles is now sniped, then only 99 pebbles remain, and the protocol outputs 0.

However, the Tower protocol below robustly computes $x \geq k$. Intuitively, at every moment in time each agent of the protocol is at one of the floors or *levels* of a tower of height k. Each agent wants to have a level just for itself, and so if two agents at the same level interact – and so discover they are not alone – then one of them climbs to the next level.

Protocol TOWER**.**

States. $Q = \{1, 2, \ldots, k\}$. (Each agent is at some level in a tower of height k.)

Input. $I = \{1\}$. (Every agent starts at level 1.)

Output. $O(i) =$ **if** $i < k$ **then** 0 **else** 1. (Agents at level k output 1, others 0.)

Transitions. For every $i < k$:

- $(i, i) \mapsto (i, i + 1)$. (If two agents meet at the same level (except the top one), one of them climbs to the next level.)
- $(k, i) \mapsto (k, k)$. (An agent at the top level "pulls up" any other agent it meets.)

To see that TOWER is robust, observe that from any configuration with at least $n \leq k$ agents, at least one agent will eventually reach level n or a higher level. Indeed, if no agent is at level n or higher, by the pigeonhole principle some level has at least two agents, allowing one of them to climb up. Thus, if the initial configuration has $k + s$ agents, then even if up to s agents are sniped adversarially, at least one of the remaining agents eventually reaches level k and then pulls up all other remaining agents to level k as well.

Remark. The definition of robustness of [17] is for protocols that compute functions $f : \mathbb{N}^I \to \{0, 1\}$ (i.e., protocols that decide predicates). We have generalized it to functions $f : \mathbb{N}^I \to K$ for any finite set K. However, the restriction of our definition to the case $K = \{0, 1\}$ is slightly stronger than the definition of [17]. To see why, consider the predicate $x \geq 7$ and the input 7. The definition of [17] does not put any constraint on the fair executions with one snipe starting at 7: some fair executions may not reach any consensus at all. On the contrary, Definition 3 requires that all fair executions converge to either 0 or 1. Since protocols that are robust w.r.t. Definition 3 are better behaved, and all protocols of [17] either are or can be easily made robust according to it, we consider Definition 3 the correct one[2].

4 Robust Protocols for Monadic Presburger Functions

As mentioned in the introduction, the question whether every Presburger predicate can be decided by a robust population protocol was left open in [17]. In this section we give a positive answer for *monadic* Presburger predicates.

▶ **Definition 5.** *A formula of Presburger arithmetic is* monadic *if every atomic formula appearing in it contains exactly one variable. A function* $f \colon \mathbb{N}^X \to K$ *is* monadic *if it is the semantics of some monadic Presburger formula.*

In other words, a formula is monadic if its atomic formulas are of the form $x_i \geq k$ or $x_i \bmod m \in M$. An example of a monadic formula that we use as running example is

$$F(x, y) = \big(x \geq 3 \lor y \bmod 7 \geq 2\big) \land \big(x \bmod 5 \geq 1 \lor y \bmod 2 \geq 1\big) \land \big(x \geq 4 \lor y \geq 17\big)$$

An example of a non-monadic formula is $x - y \leq 3$. Hague *et al.* have shown that deciding if a given Presburger formula has an equivalent monadic formula is co-NP-complete [15].

It is well known that every monadic Presburger formula is equivalent to a Boolean combination of monadic threshold and modulo formulas [14]. So it suffices to show that every such combination, like $F(x, y)$ above, has a robust protocol. For this, fix an arbitrary Boolean combination $F(x_1, \ldots, x_n)$ of monadic threshold predicates $x_i \geq t$ and monadic modulo predicates $(x_i \bmod m) \in M$. We construct a robust protocol P_F that decides F. We will proceed in three steps:

[2] The authors of [17], three of whom are also authors of this paper, agree (private communication).

The functions $f^1, \ldots, f^n$. For every variable x_i, define $t^i, m^i \in \mathbb{N}$ as follows:

- t^i is the maximum of all $t \in \mathbb{N}$ such that the atomic formula $x \geq t$ appears in F; if there are none, then $t^i = 0$.
- m^i is the least common multiple (lcm) of all $m \in \mathbb{N}$ such that the atomic formula $x \bmod m \in M$ appears in F for some $M \subseteq \mathbb{Z}_m$; if there are none, then $m^i = 1$.

We define the function $f^i \colon \mathbb{N} \to [0, t^i] \times \mathbb{Z}_{m^i}$ as $f^i(x_i) = (\min(x_i, t^i), x_i \bmod m^i)$. In our running example we have $f^x(x) = (\min(x, 4), x \bmod 5)$ and $f^y(y) = (\min(y, 17), y \bmod 14)$.

The robust protocols $P_F^1, \ldots, P_F^n$. For every variable x_i, we construct a robust protocol P_F^i computing $f^i(x_i)$. Observe that for every $k \in \mathbb{N}$ the output of $f^i(k)$ determines the values of *all* the threshold and modulo atomic formulas of F on x_i. For example we have $f^y(26) = (17, 12)$, and from the output $(17, 12)$ we can deduce that $y \bmod 7 \geq 2$ is true, because $y \equiv 12 \pmod{14}$ implies $y \equiv 5 \pmod 7$; that $y \bmod 2 \geq 1$ is false, because $y \equiv 12 \pmod{14}$ implies $y \equiv 0 \pmod 2$; and that $y \geq 17$ is true. (Notice, however, that the output does not determine the value of y.) In particular, for every $(k_1, \ldots, k_n) \in \mathbb{N}^n$ the outputs of $f^1(k_1), \ldots, f^n(k_n)$ determine the truth value of every atomic formula of F, and so the truth value of $F(k_1, \ldots, k_n)$. This leads to the next step.

The robust protocol P_F. Let $f(x_1, \ldots, x_n) \colon ([0, t^1] \times \mathbb{Z}_{m^1}) \times \cdots \times ([0, t^n] \times \mathbb{Z}_{m^n})$ be the function defined by $f(x_1, \ldots, x_n) := (f^1(x_1), \ldots, f^n(x_n))$. Given robust protocols $P_F^1, \ldots, P_F^n$ computing $f^1, \ldots f^n$, we construct a robust protocol P_F computing f. The output of this protocol determines the value of F and so, by adjusting the output mapping of the protocol, we obtain a robust protocol that decides F.

In our example, we have $f(x, y) = (\min(x, 4), x \bmod 5, \min(y, 17), y \bmod 14)$, and so for instance $f(14, 26) = (4, 4, 17, 12)$. From $(4, 4, 17, 12)$ we obtain that all atomic formulas of $F(14, 26)$ are true except $y \bmod 2 \geq 1$. So $F(14, 26)$ is true.

In the rest of the section we develop this proof outline. The protocols P_F^i are constructed in three steps: Sections 4.1 and 4.2 construct robust protocols computing the functions $\min(x_i, k)$ and $(x_i \bmod k)$, respectively. Section 4.3 gives a robust protocol for the function $(\min(x_i, k), x_i \bmod k)$. Section 4.4 constructs P_F given $P_F^1, \ldots, P_F^n$.

4.1 Computing $\min(n, k)$ robustly

We present a robust protocol computing $\min(n, k)$ for a given constant k. Recall that in the Tower protocol, which robustly decides if $n \geq k$, an agent who reaches the highest level of the tower pulls all other agents to the top. We modify Tower by adding a second component to the states: In addition to their own level in the tower, agents now also remember the highest level they have seen any agent occupy in all their interactions so far, or, in other words, the highest level they currently know has already been reached by some agent.

Protocol RobustMin**.**

States.	$Q = [k] \times [k]$. Models: (own-level, highest-level-I-know-has-been-reached).
Input.	$I = \{(1, 1)\}$
Output.	$O(i, h) = h$

Transitions. When agents with the same level meet, one of them climbs to the next level. Additionally, agents exchange and update their knowledge about the highest occupied level.

$$(i, h_1), (i, h_2) \mapsto (i, \max\{h_1, h_2, i+1\}), (i+1, \max\{h_1, h_2, i+1\}) \quad \text{if } i < k \quad \langle \textsc{Step} \rangle$$
$$(i_1, h_1), (i_2, h_2) \mapsto (i_1, \max\{h_1, h_2\}), (i_2, \max\{h_1, h_2\}) \quad \text{if } i_1 \neq i_2 \ \langle \textsc{Inform} \rangle$$

Note that we no longer need to pull up agents to the topmost level. This functionality is completely replaced by the second state component.

▶ **Theorem 6.** *RobustMin computes* $\min(n, k)$ *robustly.*

Proof. Let $\pi = C_0, C_1, \ldots$ be a fair execution with s snipes starting at the initial configuration of RobustMin that puts n agents in its input state. After the last snipe, all subsequent configurations have $n - s$ agents.

We first show that some agent eventually reaches level $\min(n-s, k)$ or above. As long as two agents share a level below k, by fairness they eventually interact and one climbs via transition $\langle \textsc{Step} \rangle$ to the next level. So eventually each level below k has at most one agent, and by the pigeonhole principle at least one agent is at level $\min(n-s, k)$ or above. Further, all transitions preserve the invariant that if level i is occupied then so are all levels $1, \ldots, i-1$. Hence no agent ever reaches a level above n.

Let h be the largest second component of the state of any agent after the last snipe. Some agent must have been at level h at some point, and so $h \leq n$. Further, all transitions preserve the invariant that the second component of the state of an agent is at least as large as the first. So $h \geq \min(n-s, k)$, which implies that h is a permissible output of the protocol. By fairness, transition $\langle \textsc{Inform} \rangle$ eventually propagates h to all agents. ◀

4.2 Computing $(n \bmod m)$ robustly

We now consider functions $f(n) = n \bmod m$ for some constant $m \geq 2$. In [17] Lossin *et al.* describe a protocol that robustly decides whether $f(n) \in A$ for some fixed set A. We construct a robust protocol computing $f(n)$. We start with some intuition about the construction.

The first insight we need is that if at least $m - 1$ agents are sniped, then every output is permissible for $(n \bmod m)$, so we mainly need to worry about the case of at most $m - 2$ snipes. Since this number is independent of n, we can apply redundancy: loosely speaking, we *replicate* a protocol for $(n \bmod m)$, and return the output produced by the largest number of replicas. If the number of replicas is large enough, then, even if some of them fail, the output of the surviving ones is still correct.

Our protocol extends RobustMin. An agent's level within the tower determines the replica he currently works for. Each agent carries one pouch for each level. The agents at level i count (modulo m) how many agents they have seen pass through level i: whenever two agents in level i meet, one passes the pebbles in its i-th pouch to the other agent and climbs to level $i+1$. There it starts counting anew by placing a single pebble in his $(i+1)$-th pouch. If no agents are sniped then the count in the i-th level will converge to $(n - i + 1) \bmod m$, as all agents are counted, except for the $i - 1$ agents occupying the lower levels. Hence $n \bmod m$ can be determined by simply adding $i - 1$ to this count. Sniping an agent in level i may have an arbitrary effect on the count in that replica; however, in other replicas, at most the sniped agent itself is not counted. If the number of replicas is large enough, this ensures that the most frequently occurring output among the replicas is permissible.

Agents in different levels inform each other about the counts in their respective replicas. Each then chooses the most frequently given result as final output. To ensure convergence even if no agent remains for a replica, if two agents disagree on the count for a replica neither of which work for, they both reset it to 0. Finally, if not enough agents are available to populate all levels, the output is determined from the highest occupied level.

Protocol RobustMod**.**

States. Let $c = m^2$.
$Q \coloneqq [c{+}1]$ current level of the agent
$\times\ [c{+}1]$ highest level the agent knows has been populated
$\times\ \mathbb{Z}_m^c$ agent's information on the counts in each replica

Input. $I \coloneqq \{(1, 1, (1, 0, \dots, 0))\}$

Output. Given $\vec{u} \in \mathbb{Z}_m^c$, let $\mathrm{pl}(\vec{u})$, the *plurality* of $\vec{u}$, be the value that occurs most frequently in the components of $\vec{u}$, or the smallest one in case of a tie.

$$O(i, t, \vec{u}) \coloneqq \begin{cases} t \bmod m & \text{if } t \leq c \\ \mathrm{pl}(\vec{u} + (0, 1, 2, \dots c-1)) & \text{otherwise} \end{cases}$$

Transitions. Intuitively, an agent that moves up in the tower transfers its current count to the agent remaining behind, and starts counting anew in the next level. Agents exchange information on the highest level they know has been populated, and on the counts of all replicas. Agents are an authority on the output of the replica they currently work for, which in an interaction is just copied by others. If two agents disagree on the output of a replica neither of them is an authority on, they both reset their values for it to 0. This ensures convergence in the case that an agent is sniped. For the formal description, let $\vec{u} \sqcap \vec{v}$ denote the *reconciliation* of two vectors, defined by $(\vec{u} \sqcap \vec{v})_j = u_j$ if $u_j = v_j$, and 0 otherwise. There are two sets of transitions, depending on whether the interacting agents occupy the same level, different from the top one, or not. The first one is:

$$(i, t_1, \vec{u}), (i, t_2, \vec{v}) \mapsto (i, t', \vec{w}), (i+1, t', \vec{w}) \qquad \text{if } i \leq c \qquad \langle \textsc{MoveUp} \rangle$$

where $\vec{w}$ is obtained from $\vec{u} \sqcap \vec{v}$ by replacing the i-th component with $(u_i{+}v_i) \bmod m$, and, if $i < c$, the $(i{+}1)$-th component with 1; and $t' \coloneqq \max\{t_1, t_2, i{+}1\}$. The second one is:

$$(i_1, t_1, \vec{u}), (i_2, t_2, \vec{v}) \mapsto (i_1, t', \vec{w}), (i_2, t', \vec{w}) \qquad \text{if } i_1 \neq i_2 \text{ or } i_1 = i_2 = c+1 \qquad \langle \textsc{Exchange} \rangle$$

where $\vec{w}$ is obtained from $\vec{u} \sqcap \vec{v}$ by replacing, if $i_1 \leq c$, the i_1-th component with u_{i_1}, and, if $i_2 \leq c$, the i_2-th component with v_{i_2}; and $t' \coloneqq \max\{t_1, t_2\}$.

We prove correctness and robustness of RobustMod.

▶ **Definition 7.** *Let $q = (i, t, \vec{u})$ be a state of* RobustMod*. The* level *of q is $\ell(q) \coloneqq i$. The* value vector *of q is $\vec{v}(q) \coloneqq \vec{u}$. The* output vector *of q is $\vec{\omega}(q) \coloneqq \vec{u} + (0, 1, \dots, c-1)$. The value vector $\overrightarrow{V}(C)$ and output vector $\overrightarrow{\Omega}(C)$ of a configuration C are given by:*

$$V_i(C) \coloneqq \sum_{(i,t,\vec{u}) \in Q} C(i, t, \vec{u}) \cdot (u_i \bmod m) \qquad \overrightarrow{\Omega}(C) \coloneqq \left(\overrightarrow{V}(C) + (0, 1, \dots, c-1)\right) \bmod m$$

Loosely speaking $V_i(C)$ counts (modulo m) how many agents have reached level i or higher in C; it does not count agents in levels below i. The output vector "compensates" for this.

Consider an execution $C_0, C_1, \dots$, possibly with snipes, of RobustMod. For the proof it is convenient to assign identities to agents, which we assume are natural numbers. Let q_j^t be the state of the j-th agent in C_t. We consider sniped agents to move to a special state $\perp$.

▶ **Lemma 8.** *For every t and $1 \leq i \leq c$, if until step t no agent in level i has been sniped, then $V_i(C_t) \equiv \left|\left\{j \;\middle|\; \exists t' \leq t.\, \ell\left(q_j^{t'}\right) = i\right\}\right| \pmod m$.*

Proof. We prove the claim by induction on t. The base case is $t = 0$. At this point $\ell(q_j^0) = 1$ and $\vec{v}(q_j^0) = (1, 0, \dots, 0)$ for all agents j, so $V_1(C_0) = n \bmod m$ and $V_i(C_0) = 0$ for all $i > 1$. For the induction step, consider the step from C_t to C_{t+1}. We have either $C_t \to C_{t+1}$ or

$C_t \hookrightarrow C_{t+1}$. In the first case, the step is caused by either ⟨Exchange⟩ or ⟨MoveUp⟩. Transition ⟨Exchange⟩ does not change the level of either agent, and for both involved agents $v_{\ell(q_j^t)}(q_j^t) = v_{\ell(q_j^{t+1})}(q_j^{t+1})$. So $\overrightarrow{V}(C_t) = \overrightarrow{V}(C_{t+1})$. Transition ⟨MoveUp⟩ moves one agent from level i to level $i+1$. Accordingly, as this agent sets the $i+1$-th component of the value vector to 1, $V_{i+1}(C_{t+1})$ increases by one compared to $V_{i+1}(C_t)$. Further, $V_i(C_{t+1})$ does not change, as the contribution of the agent moving up is added to the other agent. Finally, if $C_t \hookrightarrow C_{t+1}$ happens by sniping an agent in level i, then only $V_i(C_{t+1})$ can change compared to $V_i(C_t)$, and we explicitly exclude this case in the statement of the lemma. ◀

▶ **Theorem 9.** *RobustMod computes* $(n \bmod m)$ *robustly.*

Proof. Consider a fair execution $\pi = C_0, C_1, \ldots$ with s snipes. As in RobustMin, the protocol eventually reaches a configuration where each level $i \leq c$ has at most one agent, which we call the *leader* of level i, and all remaining agents are in level $c{+}1$. After that, only ⟨Exchange⟩ transitions occur, which propagate each leader's value to all agents. For levels whose leader has been sniped, disagreements are resolved to 0 by reconciliation, so all agents eventually agree on the same value vector. More specifically, let C be the configuration reached after convergence. Then, for levels i in which no snipe has occurred, $v_i(q) = V_i(C)$, since $V_i(C)$ is completely determined by the leader of level i.

This already shows that the protocol always converges to some result. We now show that that result is permissible. Since we compute modulo m, for $s \geq m-1$ any output is permissible by the definition of robustness. Thus we only need to consider $s \leq m-2$ snipes.

Let s_i be the number of agents that have been sniped in level i. Then there exist $s'_i \leq s_i$, such that the number of agents which have at some point been in level i is $n - i + 1 - \sum_{j=1}^{i-1} s'_j$. This is because all agents that have not been sniped in a lower level, or are a leader of a lower level, have at some point passed through level i. Hence, by Lemma 8, for any level i with $s_i = 0$, we have $V_i(C) \equiv n - i + 1 - \sum_{j=1}^{i-1} s'_j \pmod m$ and therefore $\Omega_i(C) \equiv n - \sum_{j=1}^{i-1} s'_j \pmod m$. Since there are s snipes total, at most s levels can have $s_i > 0$. Thus at least $c - s$ levels remain unaffected. The output in each of these levels is permissible, as it is of the form $n - k \bmod m$ for some $k \leq s$. Since we assumed $s \leq m-2$ there are at most $m-1$ permissible outputs. By the pigeonhole principle, one of these must occur in at least $\frac{c-s}{m-1}$ replicas, which is more than the maximum number of replicas affected by snipes:

$$\frac{c-s}{m-1} \geq \frac{m^2 - m + 2}{m-1} = m + \frac{2}{m-1} > m > s$$

Thus among the outputs of all replicas there is a plurality for a permissible output. ◀

4.3 Computing $(\min(n,k), n \bmod m)$ robustly

We show that $\min(n,k)$ and $(n \bmod m)$ can be computed *simultaneously* and robustly by a slight modification of RobustMod. Recall that the first component of the states of RobustMod model a tower of height c. It suffices to extend the height to $\max(c{+}1, k{+}m)$. The resulting protocol is:

Protocol RobustMinMod**.**

States. Let $c \coloneqq m^2$ and $T \coloneqq \max(c{+}1, k{+}m)$.
$Q \coloneqq [T]$ agent's own level
$\times\ [T]$ highest level the agent knows of
$\times\ \mathbb{Z}_m^c$ values computed in each mod replica

Input. $I \coloneqq \{(1, 1, (1, 0, \ldots, 0))\}$

Output. $O(i,t,\vec{u}) := \begin{cases} (\min(t,k),\ t \bmod m) & \text{if } t < T \\ (\min(t,k),\ \mathrm{pl}(\vec{u}+(0,1,2,\dots c-1))) & \text{otherwise} \end{cases}$

Transitions. As in RobustMod, with c replaced by T.

Theorems 6 and 9 already show that the individual components of the output given by RobustMinMod are permissible under robustness. However, we need to show that the combination of both is also permissible. Consider for example the case where $k = 5$, $m = 3$, $n = 6$. In a run with 2 snipes, 4 would be a permissible output for the first component $(\min(n,k))$ and 2 would be a permissible output for the second component $(n \bmod m)$. However, achieving either requires removing different amounts of agents initially, and hence the combination of both is not permissible. We show that such a situation cannot occur.

We define the level ℓ, the value vector v and output vector ω of a state as well as the value vector V and output vector Ω of a configuration the same as for the RobustMod protocol in Definition 7.

▶ **Theorem 10.** *RobustMinMod computes* $(\min(n,k), n \bmod m)$ *robustly.*

Proof. Theorems 6 and 9 already show that all the agents will come to a consensus on both the tower height h and the value vector v for the mod-replicas. Hence it only remains to show that the combination is permissible. Let C be the final configuration reached. We distinguish two cases:

Case 1. $h < T$. In this case the two output components are $\min(h,k)$ and $h \bmod m$. These are obviously compatible.

Case 2. $h = T$. In this case the output components are $\min(h,k) = k$ and $\mathrm{pl}(\vec{\omega}_i)$. Let s be the total number of snipes, and let s_i be the number of snipes on level i. If $s \geq m-1$, then any mod-output is permissible. Since $n \geq T$ and $T - k \geq m$, for any $r \in \mathbb{Z}_m$ we can remove $(n-r) \bmod m$ agents to achieve mod-output r while still having at least k agents remaining, so (k,r) is permissible.

In the following we assume that $s < m-1$. Recall from the proof of Theorem 9, that there exist $s'_i \leq s_i$, such that the number of agents which have at some point been on level i is $n-i+1-\sum_{j=1}^{i-1} s'_j$, and then by Lemma 8, for all levels i without snipes $\Omega_i(C) \equiv n - \sum_{j=1}^{i-1} s'_j \pmod m$. On the other hand, for $h = T$ to be possible, at least one agent must have reached level T, which means that at least $T-i+1$ must have reached level i at some point:

$$\left(n - i + 1 - \textstyle\sum_{j=1}^{i-1} s'_j \geq T - i + 1\right) \quad \text{iff} \quad \left(n - \textstyle\sum_{j=1}^{i-1} s'_j \geq T\right)$$

For $\Omega_i(C)$, where $s_i = 0$, this implies $\Omega_i(C) \equiv n' \pmod m$ for some $T \leq n' \leq n$. And then, since $k < T$, for each level i without snipes, $(k, \Omega_i(C))$ is a permissible combination of output components. Since $s < m-1$, the plurality of the outputs corresponds to Ω_i for some level i without snipes by the same argument as in the proof of Theorem 9. Hence this protocol is also robust. ◀

4.4 Monadic Predicates

Given functions $f^1 \colon \mathbb{N} \to K_1, \dots, f^n \colon \mathbb{N} \to K_n$ of arity 1 and robust protocols $P^1, \dots, P^n$ computing $f^1, \dots f^n$, we show how to construct a robust protocol P computing the function $f \colon \mathbb{N}^n \to K_1 \times \cdots \times K_n$ of arity n given by $f(x_1, \dots, x_n) := (f^1(x_1), \dots, f^n(x_n))$.

Remark. Observe that f is again a function $\mathbb{N}^n \to K$ for $K := K_1 \times \cdots \times K_n$. In other words, if $f^1, \dots, f^n$ are functions with a finite range, then so is f. However, if $f^1, \dots, f^n$ are predicates, i.e., $K_1 = \cdots = K_n = \{0,1\}$, then f is not a predicate. This explains why we defined robustness for functions, not just predicates.

We present a construction that, given robust protocols P_i for $f^i(x_i)$, delivers a robust protocol P for $f(x_1, \ldots, x_n)$. Using the construction it is easy to prove the existence of robust protocols for all monadic Presburger predicates.

A subtlety here is that, although a population protocol has no defined output on the empty configuration, any individual variable may be 0 initially or be reduced to 0 by snipes; our construction gives permissible outputs in either case.

▶ **Definition 11.** *Let $f\colon F \to F'$ and $g\colon G \to G'$. The* parallel composition *of f and g is the function $f \times g\colon F \times G \to F' \times G'$ with $(a, b) \mapsto (f(a), g(b))$*

Let $P_f = (Q_f, I_f, O_f, \delta_f)$ and $P_g = (Q_g, I_g, O_g, \delta_g)$ be population protocols computing functions $f\colon \mathbb{N}^{I_f} \to F'$ and $g\colon \mathbb{N}^{I_g} \to G'$ respectively. We define the parallel composition of these two protocols:

Protocol $P_f \times P_g$.

States. $Q \coloneqq (\{f\} \times Q_f \times G') \cup (\{g\} \times Q_g \times F')$
Each agent computes either f or g and stores the current output of the other function.

Input. $I \coloneqq \{f\} \times I_f \times \{g(\mathbf{0})\} \cup \{g\} \times I_g \times \{f(\mathbf{0})\}$
where $\mathbf{0}$ denotes the zero input.

Output. $O(f, q_f, o_g) \coloneqq (O_f(q_f), o_g) \qquad O(g, q_g, o_f) \coloneqq (o_f, O_g(q_g))$

Transitions. If $h \in \{f, g\}$ and $(q_1, q_2) \to (q_1', q_2') \in \delta_h$:

$$(h, q_1, o), (h, q_2, o) \mapsto (h, q_1', o), (h, q_2', o) \qquad \langle \textsc{Execution} \rangle$$

Agents computing the same function execute their protocol if they agree on the other function's output. If they disagree, they reset to the zero output:

$$\begin{aligned} &(f, q_1, o_1), (f, q_2, o_2) \mapsto (f, q_1', g(\mathbf{0})), (f, q_2', g(\mathbf{0})) \\ &(g, q_1, o_1), (g, q_2, o_2) \mapsto (g, q_1', f(\mathbf{0})), (g, q_2', f(\mathbf{0})) \end{aligned} \qquad \langle \textsc{Reset} \rangle$$

where $(q_1, q_2) \to (q_1', q_2') \in \delta_f$ or δ_g respectively, and $o_1 \neq o_2$.
Agents computing different functions exchange information:

$$(f, q_f, \_), (g, q_g, \_) \mapsto (f, q_f, O_g(q_g)), (g, q_g, O_f(q_f)) \qquad \langle \textsc{Exchange} \rangle$$

▶ **Proposition 12.** *If P_f and P_g are robust, then $P_f \times P_g$ is also robust.*

Proof. Let the initial configuration be $C = \{f\} \times C_f + \{g\} \times C_g$ with $C_f \in \mathbb{N}^{I_f}$ and $C_g \in \mathbb{N}^{I_g}$. Consider a fair execution π with s snipes, of which s_f affect agents computing f and s_g affect agents computing g.

If $C_g = \emptyset$, then $P_f \times P_g$ merely simulates the robust protocol P_f on C_f, which gives a permissible output for f. Since the initial (and correct) value of $g(\mathbf{0})$ cannot change, the overall output is permissible. The symmetric case $C_f = \emptyset$ is analogous. In the remainder of the proof we assume $C_f, C_g \neq \emptyset$.

Let π_f, π_g be the projections of π onto the P_f and P_g components. Observe that π_f, π_g are also fair. Let D_f be a configuration of P_f occurring infinitely often in π_f, and let E_f be reachable from D_f. Since there are only finitely many configurations of P_g reachable from C_g, there must be at least one configuration D_g such that $\{f\} \times D_f + \{g\} \times D_g$ occurs infinitely often in π. Since the configuration $\{f\} \times E_f + \{g\} \times D_g$ is reachable and we assumed π to be fair, it must also occur infinitely often. Hence E_f occurs infinitely often in π_f. The same argument holds for π_g.

By robustness of P_f and P_g, the agents within each group eventually reach a stable consensus on permissible outputs $b \in F'$ and $d \in G'$, respectively. If at least one agent remains in both groups, transition $\langle \textsc{Exchange} \rangle$ propagates b and d to all agents, and a stable consensus is reached.

If all agents computing g have been sniped, then the surviving f-agents either disagree on their stored g-output, or they all agree on some $d' \in G'$. In the first case, transition $\langle \textsc{Reset} \rangle$ resets them to $g(\mathbf{0})$, which is permissible since it corresponds to sniping all g-agents in C_g before any interaction. In the second case, d' was copied from a g-agent before it was sniped, so there exists a configuration $D_g \in \mathbb{N}^{Q_g}$ of P_g such that $C_g \xrightarrow{*} D_g$ and D_g populates some state q with $O_g(q) = d'$. Sniping all but this one agent in D_g leaves the singleton $\wr q \wr$, which is a stable d'-consensus, so by robustness of P_g the output d' is permissible for $|C_g| - 1$ snipes, and hence also for $|C_g|$ snipes. The symmetric case where all agents computing f have been sniped is analogous. ◀

▶ **Theorem 13.** *Robust population protocols exist for every monadic Presburger predicate.*

Proof. Let $\varphi\colon \mathbb{N}^k \to \{0,1\}$ be a monadic Presburger predicate. Each atomic predicate occurring in φ is either a threshold predicate $\nu_{i,j}(x_i) \Leftrightarrow x_i \geq t_{i,j}$ for $t_j \in \mathbb{N}$, or a modulo predicate $\mu_{i,j}(x_i) \Leftrightarrow (x_i \bmod m_{i,j}) \in M_{i,j}$ for some $m_{i,j} \in \mathbb{N}$ and $M_{i,j} \subseteq \mathbb{Z}_{m_{i,j}}$. For every atomic predicate for variable x_i define $t_i := \max_j t_{i,j}$ and $m_i := \operatorname{lcm}_j m_{i,j}$. Then we have $\nu_{i,j}(x_i)$ iff $\nu_{i,j}(\min\{x_i, t_i\})$ and $\mu_{i,j}(x_i)$ iff $\mu_{i,j}(x_i \bmod m_i)$. By Theorem 10 there exist robust protocols computing $f^i(x_i) := (\min(x_i, t_i), x_i \bmod m_i)$ for every variable x_i and by repeated application of Proposition 12 we can merge these into a single robust protocol computing $f(x_1, \ldots, x_k) := (f^1(x_1), \ldots, f^k(x_k))$. Since all atomic predicates of φ are determined by the output of f, adjusting the output function yields a robust protocol deciding φ. ◀

5 State Complexity of Robust Computation

Consider the family $\{x \geq k\}_{k \geq 0}$ of predicates. The family can be decided by protocols with $O(\log k)$ states [7, 10]. Further, an infinite subfamily can be decided by protocols with only $\Theta(\log \log k)$ states, and no infinite subfamily can be decided by protocols with $O(\log \log \log k)$ states [7, 8, 9, 10, 16]. However, a simple inspection shows that none of these protocols are robust[3]. This raises the question of how succinctly (i.e., with the least possible number of states) can a family of protocols robustly decide $\{x \geq k\}_{k \geq 0}$. The Tower protocols of Example 4 decide $x \geq k$ with k states. We prove that they are optimal: every protocol robustly deciding $x \geq k$ has at least k states. We derive this result from a more general theorem giving a lower bound on the number of states of protocols that robustly decide *asymptotically constant* predicates, defined next.

▶ **Definition 14.** *Let $\varphi : \mathbb{N}^I \to \{0,1\}$ be a predicate. An input $A \in \mathbb{N}^I$ of φ is* upward invariant *if $\varphi(A') = \varphi(A)$ for every $A' \in \mathbb{N}^I$ such that $A \leq A'$. The set of upward-invariant inputs of φ is denoted $\mathcal{U}_\varphi$.*

Upward-invariant inputs have the following property:

▶ **Lemma 15.** *All upward-invariant inputs of a predicate have the same output.*

Proof. Let $\varphi : \mathbb{N}^I \to \{0,1\}$ be a predicate and let $A_1, A_2 \in \mathcal{U}_\varphi$. Let $A \in \mathbb{N}^I$ be any input such that $A \geq A_1$ and $A \geq A_2$. Since A_1 and A_2 are upward invariant, we have $\varphi(A) = \varphi(A_1)$ and $\varphi(A) = \varphi(A_2)$, and so $\varphi(A_1) = \varphi(A_2)$. ◀

▶ **Definition 16.** *A predicate φ is* asymptotically constant*, or an* ac-predicate*, if $\mathcal{U}_\varphi$ is nonempty. An ac-predicate φ is*

- asymptotically zero*, or an a0-predicate, if $\varphi(A) = 0$ for every $A \in \mathcal{U}_\varphi$; and*
- asymptotically one*, or an a1-predicate, if $\varphi(A) = 1$ for every $A \in \mathcal{U}_\varphi$.*

[3] We do not further elaborate on this, because it is also a consequence of the results of this section.

▶ **Example 17.** All predicates $x \geq k$ are a1-predicates with $\mathcal{U}_{x\geq k} = \{k' \in \mathbb{N} \mid k' \geq k\}$. The predicate $x + y \leq 4$ is an example of an a0-predicate. Predicates like $x - y \geq 0$ or $x \bmod m \geq k$ are not asymptotically constant.
The case distinction of Definition 16 is exhaustive by Lemma 15. Our goal is to prove:

▶ **Theorem 18.** *Every robust population protocol computing an ac-predicate φ has at least $\min_{A\in\mathcal{U}_\varphi} \max_{x\in I} A(x)$ states.*

This bound implies:

▶ **Corollary 19.** *Every robust protocol deciding $x \geq k$ has at least k states. In particular, the* Tower *protocol deciding $x \geq k$ has a minimal number of states.*

Proof. Since $x \geq k$ is an ac-predicate, Theorem 18 applies. Let $P = (Q, I, O, \delta)$ be an arbitrary protocol deciding $x \geq k$. P has one single initial state, which we can also call x. So $I = \{x\}$ and the bound of Theorem 18 becomes $\min_{A\in\mathcal{U}_{x\geq k}} A(x)$. Since $\mathcal{U}_{x\geq k} = \{k' \in \mathbb{N} \mid k' \geq k\}$, we get $|Q| \geq \min\{k' \in \mathbb{N} \mid k' \geq k\} = k$. ◀

Proof of Theorem 18

We only prove Theorem 18 for a1-predicates, the proof for a0-predicates being completely analogous. We proceed in two steps. First, we introduce the *critical input* of a protocol deciding a given a1-predicate, and show that if this input is upward invariant, then the protocol has at least the number of states of Theorem 18 (Proposition 21). Then we prove that the critical input is indeed upward invariant (Proposition 26).

Notation. We fix an a1-predicate φ and a robust protocol $R = (Q, I, O, \delta)$ deciding φ. We let Q_0 and Q_1 denote the sets of states $q \in Q$ with output 0 and 1, and call them the sets of *rejecting* and *accepting* states of R. Observe that $Q = Q_0 \cup Q_1$.

The critical input. Loosely speaking, the critical input of R puts in each initial state as many agents as the number of rejecting states of R plus 1.

▶ **Definition 20.** *The* critical input $A_R \in \mathbb{N}^I$ *of R is the input A_R given by $A_R(x) := |Q_0|+1$ for all $x \in I$ (recall that Q_0 and Q_1 are the rejecting and accepting states of R).*

▶ **Proposition 21.** *If A_R is upward invariant, then R has at least $\min\limits_{A\in\mathcal{U}_\varphi} \max\limits_{x\in I} A(x)$ states.*

Proof. Since R decides an a1-predicate, it has at least one accepting state, and so $|Q| \geq |Q_0|+1$. By definition, $|Q_0|+1 = A_R(x)$ for every $x \in I$, and so $|Q_0|+1 = \max_{x\in I} A_R(x)$. Since A_R is upward invariant, we have $A_R \in \mathcal{U}_\varphi$ and so $\max_{x\in I} A_R(x) \geq \min_{A\in\mathcal{U}_\varphi} \max_{x\in I} A(x)$. ◀

The critical input is upward invariant. We first characterize the set of inputs of R with output 1 (Lemma 23).

▶ **Definition 22.** *Let $Q' \subseteq Q$ be a set of states and let C be a configuration.*
- *C is* confined to Q' *if $[\![D]\!] \subseteq Q'$ for every configuration D reachable from C.*
- *C is* confinable to Q' *if some configuration reachable from C is confined to Q'.*

▶ **Lemma 23.** *An input A of R has output 1 iff A is* not *confinable to Q_0.*

Proof. Since R decides φ, for every input A of R and every $b \in \{0, 1\}$ we have: (1) if A is confinable to Q_b, then $\varphi(A) = b$, and (2) A is either confinable to Q_0 or confinable to Q_1. The result follows. ◀

Lemma 23 allows us to redefine our goal: instead of proving that the critical input is upward invariant, i.e., that $A \geq A_R$ implies $\varphi(A) = \varphi(A_R) = 1$, we equivalently prove that $A \geq A_R$ implies that A is not confinable to Q_0. We do this in Proposition 26. First, we obtain a sufficient condition for "non-confinedness" in Lemma 25.

▶ **Definition 24.** *Let $Q' \subseteq Q$ be a set of states. A transition t of R is an* escape transition *of Q' if $[\![\mathrm{pre}(t)]\!] \subseteq Q'$ and $[\![\mathrm{post}(t)]\!] \not\subseteq Q'$.*

▶ **Lemma 25.** *Let $Q' \subseteq Q$ and let $n := |Q'|$. Let D be a configuration such that (1) $|D| > n$, and (2) for every configuration C satisfying $D \overset{*}{\hookrightarrow} nC$, every set of states S with $[\![C]\!] \subseteq S \subseteq Q'$ has an escape transition. Then D is not confined to Q'.*

Proof. Given $k \geq 0$, we say that D is *k-snipe-confinable to Q'* if some configuration C such that $D \overset{*}{\hookrightarrow} kC$ is confined to Q'. Observe that D is confined to Q' iff D is 0-snipe-confinable to Q'. Let D be a configuration satisfying the conditions of the lemma. We claim:

For all $i \leq n$ and all $S \subseteq Q'$ of size at most i: D is not $(n-i)$-snipe-confinable to S.

Once the claim is proved, the lemma follows by taking $i := n$. Indeed, in this case Q' itself has size at most i, and so we can set $S := Q'$. This yields that D is not 0-snipe-confinable to Q', and so that D is not confined to S. We prove the claim by induction on i:

Base $i = 0$. In this case $S = \emptyset$. Since $|D| > n$ by assumption (1), every configuration C such that $D \overset{*}{\hookrightarrow} nC$ satisfies $|C| > 0$, and so C is not confined to the empty set.

Step $i \to i+1$. Let S be an arbitrary set of states of size at most $i+1$, and let C be a configuration with $D \overset{*}{\hookrightarrow} n-(i+1)C$. We prove that C is not confined to S, i.e., that some configuration B reachable from C satisfies $B(q) > 0$ for some $q \notin S$. From now on, we say that B *populates* q, and that C *can populate* q; in particular, C is not confined to S iff it can populate some state outside S.

If $[\![C]\!] \not\subseteq S$, then C already populates a state outside S. Otherwise $[\![C]\!] \subseteq S \subseteq Q'$, and so by our assumption S has an escape transition t. There are two cases: $|[\![\mathrm{pre}(t)]\!]| = 2$ and $|[\![\mathrm{pre}(t)]\!]| = 1$. We only consider the first one, the second being analogous. If $|[\![\mathrm{pre}(t)]\!]| = 2$, then $\mathrm{pre}(t) = \{\!|q_1, q_2|\!\}$ with $q_1 \neq q_2$. There are three possible cases:

1. If $C(q_1) = C(q_2) = 0$, define $S' := S \setminus \{q_1\}$ and let C' be any configuration obtained by sniping one agent from C. Then $D \overset{*}{\hookrightarrow} n-(i+1)C \overset{*}{\hookrightarrow} 1C'$ and so $D \overset{*}{\hookrightarrow} n - iC'$. By induction hypothesis applied to C' and S', the configuration C' can populate a state in $Q \setminus S'$. Since $C \geq C'$, this state can also be populated from C. If the state is q_1, we proceed with (2) or (3); otherwise we have already populated a state outside S.
2. If C populates either q_1 or q_2 (w.l.o.g. q_1), define $S' := S \setminus \{q_2\}$ and let $C' := C - \{\!|q_1|\!\}$. As in the previous case we have $D \overset{*}{\hookrightarrow} n - iC'$, and so by induction hypothesis applied to C' and S' some configuration C'' reachable from C' populates some state outside S', which is either q_2 or some state outside S. It follows $C = C' + \{\!|q_1|\!\} \to^\star C'' + \{\!|q_1|\!\}$, and so C can either populate a state outside S, in which case we are done, or populate q_2 in addition to the agent already in q_1, in which case we proceed with (3).
3. If C populates both q_1 and q_2 then the escape transition t is enabled, and firing t leads to a configuration populating a state outside S. ◀

▶ **Proposition 26.** *The critical input A_R of R is upward invariant.*

Proof. Let $A'_R \geq A_R$ arbitrary. We prove $\varphi(A'_R) = \varphi(A_R) = 1$, which shows that A_R is upward invariant.

Claim I. Every configuration D reachable from A'_R (without snipes, i.e., $A'_R \xrightarrow{*} D$) satisfies premises (1) and (2) of Lemma 25 for $Q' := Q_0$.
The first premise is $|D| > |Q_0|$. It follows from $|A'_R| \geq |A_R| > |Q_0|$, which holds by the definition of the critical input, and $|A'_R| = |D|$, which holds because $A'_R \xrightarrow{*} D$. For the second premise, let C be any configuration satisfying $D \overset{*}{\hookrightarrow} nC$, and let S be any set of states such that $[\![C]\!] \subseteq S \subseteq Q_0$. We prove that S has an escape transition. Let A be any upward invariant input of φ (which exists because φ is an ac-predicate by assumption) and let $m := \max_{x \in I} A(x)$. Since $A'_R \to^* D \overset{*}{\hookrightarrow} nC$, we have $m \cdot A'_R \to^* m \cdot D \overset{*}{\hookrightarrow} mnm \cdot C$.

Claim II. (subclaim of claim I): The configuration $m \cdot C$ stabilizes to 1.
For the proof, remember that $n := |Q_0|$. We have $(m \cdot A'_R)(x) \geq m(|Q_0| + 1) = mn + m$ for all $x \in I$. Therefore, any configuration E obtained by removing mn agents from $m \cdot A'_R$ has at least m agents in each input state and is therefore greater than or equal to A. Since A is upward invariant, we have $\varphi(E) = 1$. By robustness, any configuration reached from $m \cdot A'_R$ with at most mn snipes stabilizes to 1. In particular, since $m \cdot A'_R \to^* m \cdot D \overset{*}{\hookrightarrow} mnm \cdot C$, the configuration $m \cdot C$ stabilizes to 1, and claim II is proved.

Since $[\![m \cdot C]\!] = [\![C]\!]$, we have $[\![m \cdot C]\!] \subseteq S \subseteq Q_0$, i.e., $m \cdot C$ only populates states of Q_0. But then S must have an escape transition: Otherwise $m \cdot C$ would be confined to $S \subseteq Q_0$, which would imply that $m \cdot C$ stabilizes to 0, contradicting claim II. Hence claim I is proved.

Since D satisfies the two premises of Lemma 25 for $Q' := Q_0$, we can apply the lemma, yielding that D is not confined to Q_0. Since D is reachable from A'_R, it follows that A'_R is not confinable to Q_0, and so that it cannot stabilize to output 0. Since R decides a predicate, A'_R must therefore stabilize to 1, and so $\varphi(A'_R) = 1$, and we are done. ◀

Proof of Theorem 18. Let R be a robust protocol deciding φ. By Lemma 15, all upward-invariant inputs of φ have the same output. W.l.o.g, let this output be 1. By Proposition 26, the critical input of R is upward invariant, and the result follows by Proposition 21. ◀

6 Conclusions

We have proved that every monadic Presburger predicate can be computed by a robust population protocol exhibiting optimal fault-tolerance against crash failures. Further, we have shown that achieving robustness has at least double exponential cost in terms of state complexity. Finally, we have given optimal robust protocols for the predicates $x \geq k$. In future work we expect to decide whether all Presburger predicates admit a robust protocol, and determine the exact cost of robustness.

References

1 Dan Alistarh, Bartlomiej Dudek, Adrian Kosowski, David Soloveichik, and Przemyslaw Uznanski. Robust detection in leak-prone population protocols. *CoRR*, abs/1706.09937, 2017. arXiv:1706.09937.

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 Dan Alistarh, Martin Töpfer, and Przemyslaw Uznanski. Comparison dynamics in population protocols. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 55–65. ACM, 2021. doi:10.1145/3465084.3467915.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Comput.*, 18(4):235–253, 2006. doi:10.1007/S00446-005-0138-3.

5 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Comput.*, 20(4):279–304, 2007. doi:10.1007/S00446-007-0040-2.

6 James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009. doi:10.1007/978-3-540-89707-1_5.

7 Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: on the minimal size of population protocols. In *STACS*, volume 96 of *LIPIcs*, pages 16:1–16:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.16.

8 Philipp Czerner. Breaking through the Ω(n)-space barrier: Population protocols decide double-exponential thresholds. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*, volume 319 of *LIPIcs*, pages 17:1–17:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.DISC.2024.17.

9 Philipp Czerner, Javier Esparza, and Jérôme Leroux. Lower bounds on the state complexity of population protocols. *Distributed Comput.*, 36(3):209–218, 2023. doi:10.1007/S00446-023-00450-4.

10 Philipp Czerner, Roland Guttenberg, Martin Helfrich, and Javier Esparza. Fast and succinct population protocols for presburger arithmetic. *J. Comput. Syst. Sci.*, 140:103481, 2024. doi:10.1016/J.JCSS.2023.103481.

11 Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Eric Ruppert. When birds die: Making population protocols fault-tolerant. In Phillip B. Gibbons, Tarek F. Abdelzaher, James Aspnes, and Ramesh R. Rao, editors, *Distributed Computing in Sensor Systems, Second IEEE International Conference, DCOSS 2006, San Francisco, CA, USA, June 18-20, 2006, Proceedings*, volume 4026 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2006. doi:10.1007/11776178_4.

12 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bull. EATCS*, 126, 2018. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/549/546.

13 Rachid Guerraoui and Eric Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 484–495. Springer, 2009. doi:10.1007/978-3-642-02930-1_40.

14 Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018. doi:10.1145/3242953.3242964.

15 Matthew Hague, Anthony W. Lin, Philipp Rümmer, and Zhilin Wu. Monadic decomposition in integer linear arithmetic. In *IJCAR (1)*, Lecture Notes in Computer Science, pages 122–140. Springer, 2020. doi:10.1007/978-3-030-51074-9_8.

16 Jérôme Leroux. State complexity of protocols with leaders. In *PODC*, pages 257–264. ACM, 2022. doi:10.1145/3519270.3538421.

17 Benno Lossin, Philipp Czerner, Javier Esparza, Roland Guttenberg, and Tobias Prehn. The black ninjas and the sniper: On robust population protocols. In *Principles of Verification (3)*, volume 15262 of *Lecture Notes in Computer Science*, pages 206–233. Springer, 2024. doi:10.1007/978-3-031-75778-5_10.

18 Giuseppe Antonio Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Population protocols with faulty interactions: The impact of a leader. *Theor. Comput. Sci.*, 754:35–49, 2019. doi:10.1016/J.TCS.2018.09.005.

19 Giuseppe Antonio Di Luna, Paola Flocchini, Taisuke Izumi, Tomoko Izumi, Nicola Santoro, and Giovanni Viglietta. Fault-tolerant simulation of population protocols. *Distributed Comput.*, 33(6):561–578, 2020. doi:10.1007/S00446-020-00377-0.