Distrib. Comput. (2007) 20:279–304
DOI 10.1007/s00446-007-0040-2

The computational power of population protocols

Dana Angluin · James Aspnes · David Eisenstat · Eric Ruppert

Received: 14 August 2006 / Accepted: 24 July 2007 / Published online: 23 August 2007

Abstract We consider the model of population protocols introduced by Angluin et al. (Computation in networks of passively mobile finite-state sensors, pp. 290–299. ACM, New York, 2004), in which anonymous finite-state agents stably compute a predicate of the multiset of their inputs via two-way interactions in the family of all-pairs communication networks. We prove that all predicates stably computable in this model (and certain generalizations of it) are semilinear, answering a central open question about the power of the model. Removing the assumption of two-way interaction, we also consider several variants of the model in which agents communicate by anonymous message-passing where the recipient of each message is chosen by an adversary and the sender is not identified to the recipient. These one-way models are distinguished by whether messages are delivered immediately or after a delay, whether a sender can record that it has sent a message, and whether a recipient can queue incoming messages, refusing to accept new messages until it has had a chance to send out messages of its own. We characterize the classes of predicates stably computable in each of these one-way models using natural subclasses of the semilinear predicates.

James Aspnes was supported in part by NSF grants CNS-0305258 and CNS-0435201.
David Eisenstat was supported in part by a National Defense Science and Engineering Graduate Fellowship and by a Gordon Y. S. Wu Graduate Fellowship.
Eric Ruppert was supported in part by the Natural Sciences and Engineering Research Council of Canada.

D. Angluin (✉) · J. Aspnes
Yale University, New Haven, CT, USA
e-mail: dana.angluin@yale.edu

J. Aspnes
e-mail: james.aspnes@yale.edu

D. Eisenstat
Princeton University, Princeton, NJ, USA
e-mail: deisenst@cs.princeton.edu

E. Ruppert
York University, Toronto, ON, Canada
e-mail: ruppert@cs.yorku.ca

1 Introduction

In 2004, Angluin et al. [3] proposed a new model of distributed computation by very limited agents called a population protocol. In this model, finite-state agents interact in pairs chosen by an adversary, with both agents updating their state according to a joint transition function. For each such transition function, the resulting population protocol is said to stably compute a predicate on the initial states of the agents if, after sufficiently many interactions in a fair execution, all agents converge to having the correct value of the predicate. Motivating scenarios include models of the propagation of trust in populations of agents [17] and interactions of passively mobile sensors [3,4]. Similar models of pairwise interaction have been used to study phenomena in other fields, for example, the propagation of diseases [13] and rumors [20] in human populations. In chemistry, a model consisting of a finite population of molecules of a fixed number of different kinds with stochastic rules to select pairs of molecules to interact and to determine the products of their interaction has been used to justify the Chemical Master Equation and to simulate specific, biologically relevant systems of molecules [24,25]. These results suggest that the model of population protocols may be fundamental in several fields of study.

Because the agents in a population protocol have only a constant number of states, independent of the population size, it is impossible for them to adopt distinct identities, making them effectively anonymous. An agent encountering another

agent cannot tell in general whether it has interacted with that agent before. Despite these limitations, populations of such agents can compute surprisingly powerful predicates on their initial states under a reasonable global fairness condition. When each agent may interact with every other agent, any predicate over the counts of initial states definable in Presburger arithmetic is computable [3,4]. When each agent has only a bounded set of neighbors with which it can interact, linear-space computable predicates are computable [1].

In this paper we give exact characterizations of the class of stably computable predicates in the family of all-pairs interaction graphs for the original population protocol model and certain natural variants of it that we introduce to model the limitations of one-way communication. Most of the results in this paper have appeared in extended abstract form in the two conference papers [6,7]. (In Sect. 7 we also correct an erroneous claim in [7].)

1.1 Stably computable predicates are semilinear

Angluin et al. [3,4] showed that various common predicates such as parity of the number of agents, whether agents in some initial state a outnumber agents in another initial state b, and so forth, could be stably computed by simple population protocols. They further showed that population protocols could in fact compute any semilinear predicate, which are precisely those predicates definable in first-order Presburger arithmetic [38]. But it was not known whether there were other, stronger predicates that could also be computed by a population protocol, at least in the simplest case where every agent was allowed to interact with every other agent. We show that this is not the case: that the semilinear predicates are precisely the predicates that can be computed by a population protocol if there is no restriction on which pairs of agents can interact with each other. This gives an exact characterization of the predicates stably computable by population protocols, answering the major open question of [3,4].

These results also answer an open question in [1]: whether requiring population protocols to deal with stabilizing inputs (rather than assuming all inputs are available at the start of computation) reduces their computational power. We show that the answer is no; in both cases, the stably computable predicates are precisely the semilinear predicates.

The semilinearity theorem is proved for a generalization of the model of population protocols, and applies to stable computation in other models that have the property that a partitioned subpopulation can still run on its own (in the sense that agents do not have any mechanism to detect if additional agents are present but not interacting). Other models with this property include vector addition systems [28], some forms of Petri nets, and 1-cell catalytic P-systems.

Semilinearity is strongly tied to the notion of stable computation, in which a correct and stable output configuration must always be reachable at any step of the computation. Mere reachability of a desired final configuration is not enough: the reachability sets of population protocols are not in general semilinear. An example of this phenomenon may be derived from the construction given by Hopcroft and Pansiot of a non-semilinear reachability set for a six-dimensional vector addition system [28]. Ibarra et al. [29] study 1-cell catalytic P-systems, in which all state-changing interactions occur between an unmodified catalyst and at most one non-catalyst agent. In contrast to the general case, they show that the reachability sets of 1-cell catalytic P-systems are semilinear.

1.2 One-way communication

We also introduce variants of the population protocol model that use forms of one-way communication analogous to traditional asynchronous message-passing models, and exactly characterize their computational power in the family of all-pairs communication graphs in terms of natural subclasses of the semilinear predicates. In the one-way models, pairwise interactions are split into separate send and receive events that each affect at most a single agent. These models may better reflect communication in the context of sensor networks, where radio communication may not be bidirectional, even between nearby sensors. Moreover, one-way message-passing primitives may be easier to implement in practice.

For the one-way models, we consider the following attributes. The sender may be allowed to change its state as a result of sending a message (transmission models), or not (observation models). The send and receive events for a message may occur simultaneously (immediate delivery models) or may be subject to a variable delay (delayed delivery and queued delivery models). In the queued delivery model, a receiver may choose to postpone incoming messages until it has had a chance to send a message of its own; in the simpler delayed delivery models, the receiver does not have this option. Thus, we consider five one-way models: immediate and delayed observation, immediate and delayed transmission, and queued transmission.

The immediate models are perhaps best understood in terms of interactions between agents, while the delayed and queued models are better thought of in terms of messages sent from one agent to another. In the transmission models, the sender is aware that it has sent a message, and may update its state accordingly. In the observation models, the sender is unaware of being observed by the receiver, and therefore does not update its state. For example, consider a passive radio frequency tag that does not update its state in response to being read; this is an example of immediate observation, in which the message (tag data) is immediately delivered (read), but the state of the sender is unchanged. A web page that updates a counter in response to a visit can be thought of as an example

of immediate transmission: the message (web page contents) is immediately delivered (viewed), and the web page is aware of the transaction and updates its state accordingly.

We give exact characterizations of the classes of predicates that can be stably computed in each of the one-way models we consider, which are summarized in Fig. 2. We show that the model of queued transmission stably computes exactly the semilinear predicates and is equivalent in power to the standard population protocol model with two-way interactions. The other models are strictly weaker, and can be characterized in terms of natural subclasses of the semilinear predicates. Weakest of all is the delayed observation model, which allows only the detection of the presence or absence of each possible input symbol. Next is the immediate observation model, which allows counting of input symbols up to an arbitrary constant limit. The immediate and delayed transmission models are equivalent to each other in power, and add the ability to count input symbols modulo an arbitrary constant. These results give us a precise and detailed understanding of the capabilities of the one-way models.

2 Related work

2.1 Population protocols and related models

Stable computation by population protocols was introduced in [3,4]. It was shown that all semilinear predicates are stably computable by population protocols in the family of all-pairs interaction graphs. It was also shown that the all-pairs interaction graph has the least computational power, in the sense that it may be simulated in any other connected interaction graph with the same number of agents. This work was inspired by models of the propagation of trust studied by Diamadi and Fischer [17]. Related models of automata with a central finite control and storage consisting of an unordered multiset of tokens was studied in [2].

Angluin et al. [3,4] also introduced probabilistic population protocols, in which a uniform random choice of pairs to interact replaces the fairness condition; this enables quantification of the probability of error and the number of interactions to convergence as a function of n, the number of agents in the population. It was shown that each semilinear predicate can be computed in an expected $O(n^2 \log n)$ interactions, and that a register machine with a constant number of registers with $O(\log n)$ bits per register could be simulated with inverse polynomial error probability and polynomial slowdown. Recent work [5] has given faster protocols for these problems under the assumption that a unique leader is present in the population: each semilinear predicate can be computed in an expected $O(n \log^4 n)$ interactions, and an improved register machine simulation can be done with inverse polynomial error in $O(n \operatorname{polylog}(n))$ interactions per step.

The question of what properties of the interaction graph are stably computable by population protocols was studied in [1]. It was shown that for any d, there is a population protocol that organizes any connected interaction graph of maximum degree at most d into a linear memory of $\Theta(n)$ bits, which is asymptotically optimal. In the same paper, the population protocol model was extended to stabilizing inputs, in which each agent has an input that may change finitely many times over the course of the computation before stabilizing to a final value; this permits composition of protocols, in which the stabilizing outputs of one protocol become the stabilizing inputs of another protocol. It was shown that all the semilinear predicates can be computed with stabilizing inputs in the family of all-pairs interaction graphs. Thus, the present paper shows that precisely the same predicates are stably computable with or without stabilizing inputs in the family of all-pairs interaction graphs.

Self-stabilizing population protocols were studied in [8], which gives self-stabilizing protocols for token circulation in a directed ring, directing an undirected ring, local addressing in a degree bounded interaction graph, and leader election in a ring with a constant bound k on the smallest nondivisor of the ring size. Self-stabilizing leader election assuming an oracle for eventual leader detection was studied in [22], which gives self-stabilizing leader election protocols for complete graphs and rings. The question of resilience to crash faults and transient faults was considered by Delporte-Gallet et al. [18], who give a general method to transform a population protocol into one that can withstand up to c crash faults and t transient faults. The problem of stabilizing consensus was defined and studied in [9].

2.2 Comparison with asynchronous message-passing

In an asynchronous message-passing model, agents communicate by sending messages. An agent may spontaneously send a message at any time, which is delivered to a recipient at some later time. The recipient may respond to the message by updating its state and possibly sending one or more messages. In the standard asynchronous model, senders can choose the recipients of their messages, and recipients are aware of the identities of the senders of messages they receive; however, in the population protocol models we consider, these assumptions are dropped.

Agents in the population protocol models are assumed to be finite-state. Moreover, algorithms in this model are uniform: the description of the protocol cannot depend on the number of agents in the system. Together with a transition rule that depends only on the states of the two interacting agents, these assumptions naturally yield a model in which agents are effectively anonymous. In this respect, the population protocol models are weaker than a typical message-passing model, where processes have identities. In addition,

not only does a receiver not learn the identity of the sender, but a sender cannot direct its message to a particular receiver. This is unusual even in anonymous message-passing models, which typically assume that a process can use some sort of local addressing to direct messages to specific neighbors.

The question of what computations can be performed in anonymous systems, where processes start with the same state and the same programming, has a long history in theoretical distributed computing. Many early impossibility results such as [11] assume both anonymity and symmetry in the communication model, which limits what can be done without some mechanism for symmetry-breaking. See [23] for a survey of many such impossibility results. More recent work targeted specifically at anonymity has studied what problems are solvable in message-passing systems under various assumptions about the initial knowledge of the processes [15,16,39], or in anonymous shared-memory systems where the properties of the supplied shared objects can often (but not always, depending on the details of the model) be used to break symmetry and assign identities [10,12,14,21,26,31,32,35,37,40]. This work has typically assumed few limits on the power of the processes in the system other than the symmetry imposed by the model.

Asynchronous message-passing systems may be vulnerable to a variety of failures, including failures at processes such as crashes or Byzantine faults, and failures in the message delivery system such as dropped or duplicated messages. In this paper we assume fault-free executions; however, Delporte-Gallet et al. [18] characterize the power of population protocols in the presence of crash faults and transient faults.

Because message delivery is asynchronous, making any sort of progress requires adopting some kind of fairness condition to exclude executions in which indefinitely-postponed delivery becomes equivalent to no delivery. A minimal fairness condition might be that if some process sends a particular message m infinitely often, then each other process receives the same message m infinitely often. In Sect. 15, we show that even with unbounded states and message lengths, this minimal fairness condition provides only enough power to detect the presence or absence of each possible input because of the very strong anonymity properties of the model. Instead we adopt a stronger global fairness condition derived from that used in [3,4], which is defined and further discussed in Sect. 4.

Considering the communication capabilities of the population protocol models, the two-way population protocol model is stronger than a typical message-passing model: communication between two interacting agents is instantaneous and bidirectional. Instantaneous communication is also a feature of the immediate transmission and immediate observation models. The observation models are weaker than message-passing in that the sender is not aware of sending a message. In the immediate and delayed transmission models, an agent must be prepared to receive and react to a message at all times. In the queued transmission model, which most closely approximates asynchronous message-passing, an agent may postpone receiving a message until it has sent messages of its own, and we show that this capability is essential to achieve the full power of the model.

3 Preliminaries

In this section we introduce a vector notation that will be used to represent the multiset of states of agents in a given configuration of the population: indices are states, and the value indexed is the number of agents in that state.

Let $\mathbb{N}$ denote the set of natural numbers, $0, 1, 2, \ldots$. The set of all functions from a set X to a set Y is denoted Y^X. Let E be a finite nonempty set. For all $f, g \in \mathbb{R}^E$, we define the usual vector space operations

$$\begin{aligned}
(f+g)(e) &:= f(e) + g(e) \quad \forall e \in E \\
(f-g)(e) &:= f(e) - g(e) \quad \forall e \in E \\
(cf)(e) &:= cf(e) \quad \forall c \in \mathbb{R}, e \in E \\
f \cdot g &:= \sum_{e \in E} f(e)g(e).
\end{aligned}$$

Abusing notation, we define a 0 vector and standard basis vectors

$$\begin{aligned}
0(e) &:= 0 \quad \forall e \in E \\
e(e') &:= [e = e'] \quad \forall e, e' \in E,
\end{aligned}$$

where $[condition]$ is 1 if $condition$ is true and 0 otherwise. We define a natural partial order on $\mathbb{R}^E$ componentwise:

$$f \leq g \Leftrightarrow (\forall e \in E) f(e) \leq g(e).$$

Next we define the set of **populations** on E:

$$\mathrm{Pop}(E) := \mathbb{N}^E \setminus \{0\}.$$

These may be interpreted as the nonempty multisets on E: for any $f \in \mathrm{Pop}(E)$ and $e \in E$, $f(e)$ represents the multiplicity of the element e in the multiset represented by f. Then, the partial order $\leq$ corresponds to the subset order on multisets. The **cardinality** of a population is the sum of the multiplicities of its elements.

4 A unified framework

In each of the models we consider subsequently, a **protocol** can be considered as determining five components: a countable set $\mathcal{C}$ of **configurations**; a set of **input symbols** Σ; a binary relation $\rightarrow$ on $\mathcal{C}$ that captures when the first configuration **can reach** the second in one step; a function $I : \mathrm{Pop}(\Sigma) \rightarrow \mathcal{C}$ that takes inputs to **initial** configurations; and a partial function $O : \mathcal{C} \rightarrow \{0, 1\}$ that gives the **output** of

each configuration on which it is defined. We take $\xrightarrow{*}$ to be the reflexive-transitive closure of $\rightarrow$. We say c' is **reachable** from c if $c \xrightarrow{*} c'$. In this unified framework, we make the following definitions.

A configuration c is **output stable with output** b if $O(c) = b$ and for every d such that $c \xrightarrow{*} d$, $O(d) = b$. We define $\mathcal{S}_b$ to be the set of configurations that are output stable with output b, for $b \in \{0, 1\}$, and define $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$ to be the set of all **output stable** configurations. Thus a configuration is output stable if and only if its output is defined, and every configuration reachable from it has the same defined output.

An **execution** is a (finite or infinite) sequence of configurations $c_0, c_1, \ldots$ such that for all j, we have $c_j \rightarrow c_{j+1}$. An execution $c_0, c_1, \ldots$ is **fair** if for all $c \in \mathcal{C}$, either there exist infinitely many j such that $c_j = c$ or there exists j such that $c_j \not\xrightarrow{*} c$. Consequently, any global configuration that is infinitely often reachable in a fair execution must occur infinitely often in that execution. Another straightforward consequence is that if D is a finite set of configurations such that for each j there is some $d \in D$ such that $c_j \xrightarrow{*} d$, then there exists some $d \in D$ such that $c_j = d$ for infinitely many j. This condition may be viewed as an attempt to capture useful probability 1 properties in a probability-free model.

This fairness condition is generalized from the original one defined in [3,4]. That original condition specifies that an infinite execution is fair if whenever $c \rightarrow c'$ and $c = c_i$ for infinitely many i, then also $c' = c_j$ for infinitely many j.

An infinite execution that is fair with respect to the new definition is fair with respect to the original definition, because one-step reachability is a special case of reachability. The converse implication does not hold in general, but does hold when (as in [3,4]), the configurations in an execution are drawn from a finite subset of $\mathcal{C}$. To see this, suppose $c_0, c_1, c_2, \ldots$ is an infinite execution that is fair with respect to the original definition and all c_i are from some finite subset D of $\mathcal{C}$. Because D is finite, there must be some $d \in D$ such that $d = c_j$ for infinitely many j. For any $c \in \mathcal{C}$, either for some c_j, $c_j \not\xrightarrow{*} c$, or for all c_j, $c_j \xrightarrow{*} c$ and therefore $d \xrightarrow{*} c$. In the latter case, choose some finite execution $d = d_0, d_1, \ldots, d_r = c$ reaching c from d; then, by finite induction from 0 to r and the assumption that $c_0, c_1, c_2, \ldots$ is fair with respect to the original definition, each d_k occurs infinitely often in $c_0, c_1, c_2, \ldots$, and therefore, c occurs infinitely often in the execution.

To see that the converse implication does not hold in general, consider a system in which agents can generate an unbounded number of delayed messages. Let c_i be a configuration with i undelivered messages, so that the one-step reachability relation is $c_i \rightarrow c_{i+1}$ and $c_{i+1} \rightarrow c_i$ for all natural numbers i. Then the execution $c_1, c_2, c_3, \ldots$ is fair under the original definition (because no configuration occurs infinitely often), but not with respect to the new definition (because c_0 is infinitely often reachable but never reached).

Our new stronger definition of fairness is intended to deal with the extension of the definition of configuration, which in [3,4] includes only the states of the agents in the population, to allow also an arbitrary finite number of messages in transit. However, this extension also allows us to deal with interaction rules that may create or delete agents in the population.[1]

Because $\mathcal{C}$ is countable, any finite execution is a prefix of a fair execution, which can be constructed as follows. Fix an enumeration of $\mathcal{C}$ where each configuration appears infinitely often. Then, starting with the finite execution, repeatedly extend it with a sequence of configurations that reaches the next configuration in the enumeration that is reachable from the last configuration of the execution constructed in the previous step.

In Sect. 15 we show that a weaker, but plausible, fairness condition severely limits the power of the model. Comparisons of several different fairness conditions may be found in Hong Jiang's Ph. D. thesis [30].

A fair execution $c_0, c_1, \ldots$ **converges with output** b if there exists an m such that for all $j \geq m$, the function O is defined on c_j and $O(c_j) = b$. In particular, a fair execution converges with output b if and only if it reaches an output stable configuration with output b. A protocol is **well-specified** if, for all initial configurations $c_0 \in I(\mathrm{Pop}(\Sigma))$, there exists a b such that all fair executions $c_0, c_1, \ldots$ converge with output b. In a well-specified protocol, every fair execution starting from an input configuration converges with an output that is determined by that input configuration.

A well-specified protocol induces a predicate $\psi : \mathrm{Pop}(\Sigma) \rightarrow \{0, 1\}$. We say that this protocol **stably computes** ψ. We remark that by interchanging 0 and 1 in the range of the output function O, we obtain a protocol that stably computes the complement of ψ. The results in this paper characterize the predicates ψ stably computable in various models of finite local state distributed computing.

5 Definitions of models

In this section we define the standard model of two-way population protocols in the family of all-pairs interaction graphs and certain one-way variants of it, as well as a new model of message transmission with queuing and certain restricted variants of it, and finally the Abstract Model, which subsumes all of these. For each model, we specify the configurations,

[1] We remark that our proof of semilinearity goes through as long as the definition of fairness guarantees that for any input x there exists an output value b such that any configuration c reachable from $I(x)$ can reach an output-stable configuration with output b.

input alphabet, one-step reachability relation, input map and output map required by the unified framework above.

5.1 Two-way

In the two-way model there is a finite population of agents that may interact in ordered pairs, in which one agent is the initiator and the other is the responder. In this interaction, each agent learns its role in the interaction (whether initiator or responder) and the state of the other agent, and updates its own state accordingly. In this paper we consider the case in which every ordered pair of agents may interact (although the general definition [3,4] permits any weakly connected interaction graph). The states of the agents come from a finite set which is independent of the number of agents in the population. There is a finite set of input symbols; we assume that the input to a computation is an assignment of one input symbol to each agent, indicated by the initial state of the agent.

We study what predicates of the multiset of input symbols can be stably computed in this model. If the input symbols are $\{a, b\}$, examples of such predicates are "at least one a occurs" or "an odd number of as occur" or "there are twice as many as as bs" or "the number of as is the square of the number of bs." It has been shown [3,4] that the first three of these predicates are stably computable in this model; one consequence of the current paper is that the fourth one is not.

To indicate an output, the states are mapped to the output symbols 0 and 1. To stably compute the correct value (0 or 1) of the predicate of the multiset of input states, we require that after some finite execution, every agent remains in some state with the correct output in every possible continuation of the execution. This does not require the state of an agent to stop changing, only its output value. Nor does it mean that any particular agent will be able to tell when convergence to the correct output has occurred. The formal definition follows.

A **two-way** protocol is specified by five components: Q, a finite set of states; Σ, a finite set of input symbols; $\delta : Q \times Q \to Q \times Q$, a joint transition function; $\iota : \Sigma \to Q$, the initial state mapping; and $o : Q \to \{0, 1\}$, the individual output function.

We define

$$\mathcal{C} := \mathrm{Pop}(Q)$$
$$I(x) := \sum_{\sigma \in \Sigma} x(\sigma)\iota(\sigma)$$
$$O(c) := b \text{ if for all } q \in Q, \quad c(q) \geq 1 \Rightarrow o(q) = b$$

We define $c \to c'$ if $q_1 + q_2 \leq c$, $c' = c - q_1 - q_2 + q_1' + q_2'$ and $\delta(q_1, q_2) = (q_1', q_2')$.

A configuration in this model is a multiset that gives the states of all the agents. Because agents do not have identifiers and we are considering the all-pairs communication graph, agents in the same state are interchangeable; thus, the multiset of their states completely specifies the global state of the population. An initial configuration sets the state of each agent according to the input symbol assigned to it. A step is an interaction between two agents and simultaneously updates both of their states according to the value of the joint transition function of their current states. If the transition used is $\delta(q_1, q_2)$, we refer to the agent in state q_1 as the **initiator** and the other agent as the **responder**.

Each agent has an individual output of 0 or 1 determined by its current state via the function o. The configuration output function is 0 (respectively, 1) if all the individual outputs are 0 (respectively, 1). If the individual outputs are mixed 0s and 1s, then the configuration output function is undefined. Note that interchanging 0 and 1 in the range of the individual output function o interchanges 0 and 1 in the configuration output map O because mixed configurations remain mixed. Thus, the complements of stably computable predicates are stably computable in this model.

In fact, all boolean combinations of stably computable predicates are stably computable in this model [3,4]. We give the construction and verify that it works correctly with our fairness condition.

Lemma 1 *The class of predicates stably computable by two-way protocols is closed under boolean combinations.*

Proof Given two protocols stably computing ψ_1 and ψ_2 over a common input alphabet and a binary boolean function f, we take a direct product of the two protocols: the state space is the Cartesian product of the two state spaces, the combined transition function is $\delta((q_1, q_2), (q_1', q_2')) := (\delta_1(q_1, q_1'), \delta_2(q_2, q_2'))$, the input map is $\iota(\sigma) := (\iota_1(\sigma), \iota_2(\sigma))$, and the output map is $o(q_1, q_2) := f(o_1(q_1), o_2(q_2))$.

We show that this construction correctly computes $f(\psi_1, \psi_2)$. The key claim is that a fair execution of the direct product protocol projects down to fair executions of the original protocols. This guarantees that they individually reach an output stable configuration with the correct output, and thus that the direct product protocol does the same. To see the claim, fix a fair execution $c_0, c_1, c_2, \ldots$ of the direct product protocol, and let $c_0', c_1', c_2', \ldots$ be the projection onto its first component; this is an execution of the first protocol. Let e' be any configuration of the first protocol. Either there exists some j such that $c_j' \not\xrightarrow{*} e'$ or there exist infinitely many j such that $c_j' \xrightarrow{*} e'$. In the second case, because there are only finitely many possibilities for configurations c_j, there exists a configuration d such that there are infinitely many j for which $c_j = d$ and $d' \xrightarrow{*} e'$, where d' is the projection of d. Thus there exists some configuration e such that $d \xrightarrow{*} e$ and e projects to e'; e can be obtained from d by applying the steps that produce e' from d' in the first protocol, and arbitrarily chosen steps in the second protocol. By the fairness of $c_0, c_1, c_2, \ldots, e$ must occur infinitely often in this execution, and therefore e' occurs infinitely often in $c_0', c_1', c_2', \ldots$. □

5.2 Transmission with queuing

We define a related new model of message transmission with queuing. Once again there is a finite population of agents, but we consider that they communicate by sending messages rather than by direct interaction with each other. In this model, when a sender sends a message it does not learn the state of the eventual recipient of the message. The set of possible states of the agents and the set of possible messages are both finite, independent of the size of the population of agents; thus, the identity of the sender cannot necessarily be deduced from a message. Also, rather than direct a message to a particular recipient, a sender "launches" a message, which remains in the multiset of messages in transit until it is received by some agent enabled to receive it. Instead of a joint transition function, there are separate transition functions for sending a message and for receiving a message. The transition function for receiving messages may be partial; that is, in some states an agent may refuse delivery of some or all messages. As we show, this ability is necessary for the full power of this model.

We assume that inputs from a finite alphabet are assigned to the agents at the start of the computation, and ask what predicates of the multiset of inputs can be stably computed in this model. A configuration in this model specifies the multiset of the states of the agents, and the multiset of messages in transit, that is, messages that have been sent but not received. The formal definition follows.

A **queued transmission** protocol is specified by seven components: Q, a finite set of states; M, a finite set of messages that is disjoint from Q; Σ, a finite set of input symbols; $\delta_s : Q \to M \times Q$, a transition function for sent messages; $\delta_r : Q \times M \to Q$, a partial transition function for received messages; $\iota : \Sigma \to Q$, the initial state function; and $o : Q \to \{0, 1\}$, the individual output function. We define

$$\mathcal{C} := \{c \in \mathbb{N}^{Q \cup M} : c(q) > 0 \text{ for some } q \in Q\}$$
$$I(x) := \sum_{\sigma \in \Sigma} x(\sigma)\iota(\sigma)$$
$$O(c) := b \text{ if for all } q \in Q, \quad c(q) \geq 1 \Rightarrow o(q) = b.$$

We have $c \to c'$ if for some state $q \in c$ and some message m, $c' = c - q + q' + m$ where $(m, q') = \delta_s(q)$; or if there exists a message $m \in c$ and a state $q \in c$ such that $c' = c - q - m + \delta_r(q, m)$. In the latter case, $\delta_r(q, m)$ must be defined.

A configuration in this model is the multiset of all agents' states and all messages in transit. (They cannot be confused, because Q and M are disjoint.) An input configuration has no messages in transit and sets the state of each agent according to the input symbol assigned to it. A step is either a send event, in which an agent in state q adds a message m to the multiset of messages in transit and goes to state q' (according to $\delta_s(q) = (m, q')$,) or a receive event, in which a message m is removed from the multiset of messages in transit and is delivered to an agent in state q, which updates its state to q' (according to $\delta_r(q, m) = q'$). If $\delta_r(q, m)$ is not defined, then message m cannot be delivered to an agent in state q; this permits agents to refuse to receive messages temporarily.

The class of predicates stably computable in this model is closed under complement, by interchanging 0 and 1 in the range of o. Because we ultimately characterize this class as the class of semilinear predicates, it is closed under general boolean operations. However, we note that the direct product construction in the proof of Lemma 1 cannot be used directly to show closure under general boolean operations in this model. This is because an infinite number of different configurations may occur in an execution (as there is no bound on the size of the multiset of messages in transit). This property means that extra care is required in the use of the fairness condition in this model.

5.3 Immediate transmission and observation

We consider two progressively more restricted versions of the two-way model in which the initiator does not learn the state of the responder. This one-way communication may more accurately model certain situations, and may be easier to implement in practice. However, as we show, these restrictions limit the computational power of the models.

Immediate transmission is a special case of the two-way model in which the state of the initiator is updated independent of the state of the responder. That is, there exist functions $\delta_1 : Q \to Q$ and $\delta_2 : Q \times Q \to Q$ such that for all $q_1, q_2 \in Q$, we have $\delta(q_1, q_2) = (\delta_1(q_1), \delta_2(q_1, q_2))$. Thus, the initiator (or sender) is aware of the fact that an interaction has taken place (or of sending a message), and may update its state accordingly, but it is not aware of the state of the responder (or receiver).

Immediate observation is a special case of immediate transmission in which δ_1 is the identity function. This is the situation in which the initiator (or sender) is not aware of being "observed" by the responder (or recipient) and therefore does not have an opportunity to update its state.

Note that when the direct product construction in the proof of Lemma 1 is applied to two immediate transmission protocols, it yields an immediate transmission protocol. Therefore, the class of predicates stably computable by immediate transmission protocols is closed under general boolean operations. The same holds of immediate observation protocols.

5.4 Delayed transmission and observation

We also consider two progressively more restricted versions of the queued transmission model, which help us understand which features of the model affect its computational power.

Delayed transmission is a special case of queued transmission, with the requirement that δ_r be a total function. In this case, the recipient does not have the option of temporarily refusing to receive messages, which means that it is in danger of being "overwhelmed" by incoming messages. Our characterization results show that this indeed limits the power of protocols in this model.

Delayed observation is a special case of delayed transmission in which for all $q \in Q$, we have $\delta_s(q) = (q, m)$ for some $m \in M$. In this model, the weakest of those we consider, an agent can neither refuse incoming messages nor update its state when it has sent a message.

We note that because an agent in an observation model does not change state upon being observed, there would be no way for it to leave a non-receive enabled state; thus we do not consider a **queued observation model**.

When we remove the possibility of refusing incoming messages, we ensure that it is always possible to reach a configuration in which all the messages have been delivered, that is, the multiset of undelivered messages is empty. Combining that fact with fairness, we have the following.

Lemma 2 *Let $c_0, c_1, c_2, \ldots$ be a fair execution of a delayed transmission protocol. Then there is a configuration d with no undelivered messages such that $d = c_j$ for infinitely many j.*

Proof Let D be the set of configurations d with no messages in transit such that for some j, $c_j \xrightarrow{*} d$. Then D is finite and for every j there exists $d \in D$ such that $c_j \xrightarrow{*} d$, by delivering all the messages. Because of fairness, this implies that there is some $d \in D$ such that $c_j = d$ for infinitely many j. □

This in turn means that a variant of the direct product construction of Lemma 1 can be successfully applied to delayed transmission protocols.

Lemma 3 *The class of predicates stably computable by delayed transmission protocols is closed under boolean operations.*

Proof Given two delayed transmission protocols stably computing ψ_1 and ψ_2 and a binary boolean function f, we define a direct product of the two protocols that stably computes $f(\psi_1, \psi_2)$. The state space is the Cartesian product $\{1, 2\} \times Q_1 \times Q_2$, where the first component alternates between 1 and 2 and indicates which protocol will next send a message. The message space is the disjoint union of the two message spaces M_1 and M_2, so that messages can be uniquely associated with their respective protocols. The input map is $\iota(\sigma) := (1, \iota_1(\sigma), \iota_2(\sigma))$ and the output map is $o(t, q_1, q_2) := f(o_1(q_1), o_2(q_2))$, where f is the given boolean function. The transition function for sent messages is defined for (t, q_1, q_2) by using t to determine which protocol is to send a message, adding the appropriate message for that protocol to the multiset of messages in transit, and updating the state of that protocol accordingly (leaving the state of the other protocol unchanged), and then flipping t between 1 and 2. The transition function for received messages is defined for a message m and a state (t, q_1, q_2) by delivering the message to whichever protocol is appropriate (according to whether m belongs to M_1 or M_2) and updating the state of that protocol accordingly, leaving the turn indicator t and the state of the other protocol unchanged. Note that the resulting protocol is a delayed transmission protocol, but is not in general a delayed observation protocol because of the need to flip between $t = 1$ and 2.

To see that this direct product protocol correctly computes $f(\psi_1, \psi_2)$ we argue as in the proof of Lemma 1 that suitably projecting a fair execution of the direct product protocol yields a fair execution of the first protocol, and similarly for the second protocol. Thus, both must reach correct output stable configurations, which means the direct product protocol reaches a correct output stable configuration.

Suppose $c_0, c_1, c_2, \ldots$ is a fair execution of the direct product protocol. By Lemma 2, there is some configuration d with no undelivered messages such that $c_j = d$ for infinitely many j. We construct a sequence $c'_0, c'_1, c'_2, \ldots$ by projecting $c_0, c_1, c_2, \ldots$ onto the first protocol's component, and deleting the steps in which the second protocol sends or receives a message (which leaves the configuration of the first protocol unchanged). For an arbitrary configuration e' of the first protocol, either $c'_j \not\xrightarrow{*} e'$ for some j, or $c'_j \xrightarrow{*} e'$ for infinitely many j. In the second case, e' must be reachable from d', where d' is the projection of d onto the first protocol's component. Thus, there must be a configuration e reachable from d in which the first protocol's component is e'; e may be obtained from d by using the same sequence of sends and receives for the first protocol that reaches e' from d', with arbitrarily chosen sends for the second protocol. By the fairness of $c_0, c_1, c_2, \ldots$, because e is reachable from d and d occurs infinitely often, $c_i = e$ for infinitely many i, and therefore $c'_j = e'$ for infinitely many j, establishing the fairness of $c'_0, c'_1, c'_2, \ldots$. □

5.5 The abstract model

In order to present our semilinearity characterization in full generality, we introduce another model. All of the preceding single step rules can be described in terms of replacing one collection of elements (states or messages) with another without regard to the other elements in the configuration. In this model it is also convenient to identify input symbols with the states they map to, dispensing with the need for an initial state map ι.

E is a set of elements, which may be either states or messages; $\Sigma \subseteq E$ is a set of input symbols; $\rightarrow$ is a relation on $\mathrm{Pop}(E)$ such that if $c \rightarrow c'$, then for all $d \in \mathrm{Pop}(E)$, we have $c+d \rightarrow c'+d$; $o : E \rightarrow \{0, 1\}$ is the individual output map. Then,

$\mathcal{C} := \mathrm{Pop}(E)$
$I(x) := x$, and
$O(c) := b$ if for all $e \in E$, $c(e) \geq 1 \Rightarrow o(e) = b$.

We note that the class of predicates stably computable by a protocol in the Abstract Model is closed under complement, because we may exchange 0 and 1 in the range of o. To see that this model generalizes the two-way model, we take E to be the disjoint union of the input symbols and states of the two-way model, and treat the input symbol σ as equivalent to the state $\iota(\sigma)$, extending the individual output map o to Σ by $o(\sigma) = o(\iota(\sigma))$. To see that the Abstract Model also generalizes the queued transmission model, we define E to be the disjoint union of the input symbols, states, and two copies of the messages of the queued transmission model. Having two copies of each message m allows us to designate one copy as having output 0 and the other as having output 1. Again we treat input symbol σ as equivalent to the state $\iota(\sigma)$, and extend the individual output map o to Σ by $o(\sigma) = o(\iota(\sigma))$. Also, we extend o to the two copies of each message by defining it to be the designated output of that copy. To ensure that outputs propagate from agent states to messages, we add rules that take a state of an agent and a copy of a message, and change (if necessary) the designated output of the message to be the same as the output of the state. Each send rule is modified to send a message with the same output value as the sender, and the receive rules ignore the output values of the messages. These changes guarantee that when the outputs of the agents stabilize, the designated outputs of the messages stabilize to the same thing. Thus, every predicate stably computable in either the two-way or the queued transmission model is stably computable in the Abstract Model.

5.6 Mirrors and messages to self

Separating message transmission and receipt creates the possibility that an agent may receive its own message. Because senders are not identified, such an agent will in general not be able to recognize the message as its own. This can be thought of as including **mirrors**, or self-loops in the interaction graph controlling which agents can communicate, which we otherwise take to consist of all ordered pairs of agents. In general, we assume that this does not occur in the two-way and immediate delivery models, which are perhaps best thought of as interaction models, but may occur in the delayed and queued delivery models, on the principle that once an anonymous message is sent it may be delivered to anyone. This has at most a minor effect on the computational power of the models we consider, which we note below as appropriate.

6 Predicate classes

We now define the classes of predicates used in our characterizations. Although stably computable predicates are defined only on $\mathrm{Pop}(\Sigma)$, our proofs are facilitated by defining predicate classes on $\mathbb{Z}^\Sigma$. The **support** of a predicate is the set of all inputs that make it true.

6.1 Semilinear predicates

The most important class of predicates we consider is the class of **semilinear** predicates, which we write **SLIN**. This class can be defined in several equivalent ways.

A **linear set** is a set of the form $\{b + k_1 p_1 + k_2 p_2 + \cdots + k_n p_n \mid k_1, k_2, \ldots, k_n \geq 0\}$, where $b, p_1, p_2, \ldots, p_n$ are vectors. The vector b is the **base** of the linear set, and the vectors p_i are the **period** vectors. A **semilinear set** is a finite union of linear sets. A **(semi)linear predicate** is a predicate whose support is (semi)linear predicate.

Semilinear sets are also precisely the sets definable by first-order formulas in **Presburger arithmetic** [38] which are formulas in arithmetic that use only $<$, $+$, 0, 1, and the standard logical quantifiers and connectives. Here the set consists of all satisfying assignments of the free variables; for example, the semilinear set $S = \{(1, 0) + k_1(0, 1) + k_2(2, 1)\} \cup \{(0, 2) + k_3(2, 0)\}$, depicted in Fig. 1, consists precisely of the satisfying assignments (x, y) of the formula

$$(\exists z : (z \geq 0) \wedge (x = z + z + 1) \wedge (y \geq z)) \\ \vee (\exists z : (z \geq 0) \wedge (x = z + z) \wedge (y = 1 + 1)), \qquad (1)$$

where $x = y$ abbreviates $\neg((x < y) \vee (y < x))$ and $x \geq y$ abbreviates $\neg(y < x)$. It follows immediately from the correspondence between semilinear sets and Presburger formulas that the semilinear sets are closed under complement, finite intersection and finite union. Thus a predicate is semilinear if and only if its complement is semilinear.

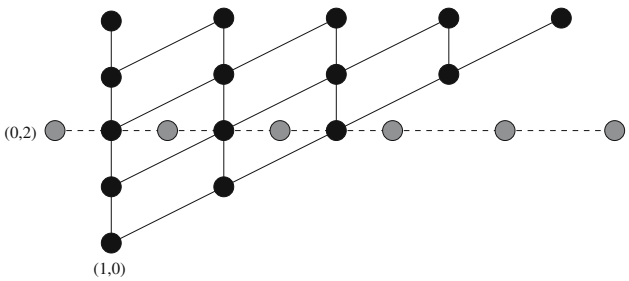

Fig. 1 A semilinear set S, equal to the union of the linear set of all points $\{(1, 0) + k_1(0, 1) + k_2(2, 1)\}$ (dark circles) and the linear set $\{(0, 2) + k_3(2, 0)\}$ (shaded circles)

A curious and useful property of Presburger formulas is that all quantifiers (and their bound variables) can be eliminated by the addition of binary relations $\equiv_m$ that test for equality modulo m for any nonnegative integer m [38]. For example, the formula (1) defining S can be rewritten without quantifiers as

$$((x \geq 0) \wedge (x \equiv_2 1) \wedge (x \leq y + y + 1)) \vee ((x \geq 0) \wedge (x \equiv_2 0) \wedge (y = 1 + 1)).$$

This yields another characterization: a semilinear predicate is a boolean combination of threshold predicates, whose support takes the form $\{x \mid x \cdot v \geq r\}$ for some $v \in \mathbb{Z}^\Sigma$ and $r \in \mathbb{Z}$, and modulo predicates, whose support takes the form $\{x \mid x \cdot v \equiv r \pmod{m}\}$ for some $v \in \mathbb{Z}^\Sigma$ and $r, m \in \mathbb{Z}$ with $m > 0$. Viewed geometrically, sets of the first type consist of points on one side of a hyperplane, and sets of the second type are lattices. As an example of a predicate of the first type, consider **comparison**, which is true if the number of as in the input exceeds the number of bs in the input. As an example of a predicate of the second type, consider **parity**, which is true if the number of as in the input is odd.

In yet another characterization, Parikh's Theorem [36] shows that a subset S of $\mathbb{N}^d$ is semilinear if and only if there is a context-free language L over an alphabet of d symbols such that S consists of the vectors of multiplicities of alphabet symbols of strings in L. Moreover, the same statement holds with regular languages in place of context-free languages. Using this characterization, it is not difficult to see that the following predicates on the number of as and bs in the input are not semilinear: the number of as is a prime, the number of as is the square of the number of bs, the number of as is a power of 2, and the number of as is bounded above by $\sqrt{2}$ times the number of bs.

6.2 The classes **MOD** and **coreMOD**

We write **MOD** for the class of boolean combinations of modulo predicates only. This class includes the predicates true and false, as well as such predicates as the number of as is congruent to 3 or 4 modulo 5 and the number of bs is not congruent to 1 modulo 17.

We next need a technical notion of similarity of two predicates with respect to subalphabets of their input alphabet. Let $\Sigma' \subseteq \Sigma$ be any nonempty subset of the input alphabet. Then $x \in \mathbb{Z}^\Sigma$ is a **k-rich profile with respect to Σ'** if $x(\sigma') \geq k$ for all $\sigma' \in \Sigma'$ and $x(\sigma) = 0$ for all $\sigma \in \Sigma \setminus \Sigma'$. Let ψ and ψ' be predicates on $\mathbb{Z}^\Sigma$. If $\psi(x) = \psi'(x)$ for every x that is a k-rich profile with respect to Σ', then we say that ψ and ψ' are **k-similar with respect to Σ'**.

The class **coreMOD** is the class of predicates ψ such that for every nonempty $\Sigma' \subseteq \Sigma$, there exists a $\psi' \in$ **MOD** and $k \geq 0$ such that ψ is k-similar to ψ' with respect to Σ'. Clearly **MOD** $\subseteq$ **coreMOD**, so the parity predicate is in **coreMOD**. The comparison predicate is not in **coreMOD** because it is not k-similar to any modulo predicate with respect to $\{a, b\}$ for any $k \geq 0$. However, if we define the **exactly-one c comparison predicate** ψ over the alphabet $\{a, b, c\}$ to be true when there is *exactly one* c and the number of as exceeds the number of bs, then ψ is in **coreMOD**. To see this, note that ψ is false when the number of cs is 0 or at least 2. Thus, if Σ' is any nonempty subalphabet of $\{a, b, c\}$, ψ is 2-similar to the false predicate with respect to Σ'.

6.3 Simple threshold predicates

We define a **simple threshold predicate** to be a threshold predicate with support $\{x \mid x \cdot v \geq r\}$ in which $v = \sigma$ for some input symbol σ. An example of a simple threshold predicate is one that is true when the number of as is at least 5. Then we define $\mathbf{COUNT}_k$ to be the class of boolean combinations of simple threshold predicates in which the threshold value $r \leq k$. A predicate in $\mathbf{COUNT}_k$ is completely determined by the counts of the input symbols truncated at k. For example, when $k = 1$, such a predicate depends only on the presence or absence of each input symbol. Finally, $\mathbf{COUNT}_*$ is the union of the classes $\mathbf{COUNT}_k$ for $k = 1, 2, 3, \ldots$. The comparison predicate is an example of a threshold predicate that is not in $\mathbf{COUNT}_*$.

7 Summary of characterizations

The computational power of the population protocol models we consider is summarized in Fig. 2. For each model we give the class of predicates on $\mathrm{Pop}(\Sigma)$ that can be stably computed by protocols in the model. Protocols in the Abstract Model, which subsumes the two-way and queued transmission models, stably compute exactly the semilinear predicates. The immediate and delayed transmission models are equal in power, and stably compute exactly those semilinear predicates that are in **coreMOD**; they cannot stably compute the comparison predicate. Immediate observation

Model	Power
Abstract	**SLIN**
Two-way	**SLIN**
Queued Transmission	**SLIN**
Immediate Transmission	**SLIN** $\cap$ **coreMOD**
Delayed Transmission	**SLIN** $\cap$ **coreMOD**
Immediate Observation	$\mathbf{COUNT}_*$
Delayed Observation	$\mathbf{COUNT}_1$

Fig. 2 The power of population protocols

protocols stably compute exactly those predicates determined by the counts of the input symbols truncated at k for some k; they cannot stably compute the comparison predicate or the parity predicate. Delayed observation protocols stably compute exactly those predicates determined by the presence or absence of each input symbol; they cannot stably compute the comparison predicate, the parity predicate or simple threshold predicates for thresholds greater than 1.

The results in [7] incorrectly claimed that any predicate ψ that is k-similar with respect to Σ to a predicate in **MOD** is stably computable in the immediate and delayed transmission models. To see that this is false, consider the **at-most-one** c **comparison predicate** ψ over the alphabet $\{a, b, c\}$ defined to be true if there is *at most one* c in the input and the number of as exceeds the number of bs. Then ψ is 2-similar to the constant false predicate with respect to $\{a, b, c\}$, but the characterization results we shall prove show that it is not stably computable in the immediate and delayed transmission models. In particular, ψ is not in the class **coreMOD**, because when we consider the subalphabet $\{a, b\}$, ψ is not k-similar to any predicate in **MOD** with respect to $\{a, b\}$ for any $k \geq 0$. This is the reason for the extra quantification over all the non-empty subalphabets of Σ in the definition of the class **coreMOD**.

We note the contrast between the exactly-one c comparison predicate (defined in Sect. 6.2), which is in **coreMOD** $\cap$ **SLIN**, and the at-most-one c comparison predicate, which is not in **coreMOD** $\cap$ **SLIN**. As we shall see, the ability to assume that there is exactly one (or any fixed number) of some symbol(s) in the input can be exploited computationally by the immediate and delayed transmission models.

8 Protocols

We first give protocols for each model in Fig. 2 to establish that each model is at least as powerful as claimed.

8.1 The two-way model

From [3,4] we have the following.

Theorem 4 *Every semilinear predicate on* $\mathrm{Pop}(\Sigma)$ *is stably computable by a two-way population protocol.*

To help us describe protocols in the one-way models, we here give specific two-way protocols for simple threshold, modulo and general threshold predicates. Because the class of predicates stably computable by two-way protocols is closed under boolean operations, it is sufficient to give algorithms for these base predicates.

Simple threshold predicates. The following protocol computes the simple threshold predicate $\psi(x) := [x \cdot \sigma \geq k]$, which is true when there are at least k occurrences of the input symbol σ in the input x.

$$
\begin{aligned}
Q &:= \{0, 1, \ldots, k\} \\
\delta(q_1, q_2) &:= \begin{cases} (q_1, k) & \text{if } q_1 = k \\ (q_1, q_2 + 1) & \text{if } 1 \leq q_1 < k \text{ and } q_1 = q_2 \\ (q_1, q_2) & \text{otherwise} \end{cases} \\
\iota(\sigma') &:= [\sigma' = \sigma] \\
o(q) &:= [q = k]
\end{aligned}
$$

Initially, agents with input σ are in state 1 and all other agents are in state 0. As this protocol runs, the states of the agents make a "tower". All but one of the agents in state i advance to state $i + 1$ in each case. This tower will extend to k if and only if there are initially at least k agents in nonzero states. We note that this is an immediate observation protocol because the state of the initiator remains unchanged by each interaction.

Active and passive agents. In the protocols for the modulo predicate $\psi(x) = [x \cdot v \equiv r \pmod m]$ and the threshold predicate $\psi(x) = [x \cdot v \geq r]$ agents fall into one of two categories: active or passive. Every agent is initially active, and there is at least one active agent at all times. Each active agent also has a data value, initially $\sigma \cdot v$, where σ is the input symbol for this agent. When a passive responder meets an active initiator, it copies the output of the active initiator. When two passive agents meet, nothing happens. When two active agents meet, they attempt to combine their data values, and one of them may become passive.

Modulo predicates. For a modulo predicate, $\psi(x) = [x \cdot v \equiv r \pmod m]$, the data values of active agents combine by sum modulo m. The initiator becomes passive while the responder remains active and keeps the combined data value. Eventually, exactly one agent is active and has the correct sum modulo m, and distributes the correct output to all of the other agents. We note that this is an immediate transmission protocol because the state of the initiator is updated uniformly, independent of the state of the responder.

As a concrete example, we specify a protocol over the alphabet $\{a, b, c\}$ that computes whether the number of occurrences of a in the input is congruent to 2 modulo 3. The states of the agents are of the form (x, y), where $x \in \{A, P\}$ indicates whether the agent is active or passive, and $y \in \{0, 1, 2\}$ is a data value modulo 3. The input map ι maps the input symbol a to the state $(A, 1)$ and the input symbols b and c to the state $(A, 0)$. The output map o maps the state (x, y) to 1 if $y = 2$ and to 0 otherwise. The transition function is defined as follows, where $+$ denotes addition modulo 3. Note that the initiator's state is updated independently of the responder's state.

$$\delta((x_1, y_1), (x_2, y_2)) := \begin{cases} ((P, y_1), (A, y_1 + y_2)) & \text{if } x_1 = A, \\ & \quad x_2 = A \\ ((P, y_1), (A, y_1)) & \text{if } x_1 = A, \\ & \quad x_2 = P \\ ((P, y_1), (x_2, y_2)) & \text{otherwise} \end{cases}$$

Threshold predicates. The situation for a threshold predicate, $\psi(x) = [x \cdot v \geq r]$, is more complicated. In the end there may be multiple active agents, but they will all agree on the output. We assume without loss of generality that $r \geq 0$. The active states are a range of values that include all possible initial data values and $2r - 1$. The output is 1 if and only if the data value is at least r. Suppose y_1 and y_2 are the data values when two active agents meet. If $y_1 + y_2$ is representable by an active state, one agent remains active with this data value; the other becomes passive. Otherwise, both agents remain active and they average their data values, that is, one becomes $\lceil \frac{y_1+y_2}{2} \rceil$ and the other becomes $\lfloor \frac{y_1+y_2}{2} \rfloor$. These transitions maintain the invariant that the sum of all the active agents' data values is $x \cdot v$. Note that this protocol involves updates of both states depending on both states; it is not an immediate transmission protocol.

To show correctness, we distinguish three cases based on the sum of all the initial data values. In the first case, the sum is negative. Eventually, no active agent will have a positive data value. In the second case, the sum is between 0 and $r-1$. There will eventually be exactly one active agent with this data value. In the third case, the sum is at least r. The number of active agents with data value less than r never increases. The only way it can increase is after an interaction involving an active agent with value at least r where both agents remain active. This will happen, however, only in the case when the two agents average their values, which never happens with an agent with value at least r unless the result is both agents having value at least r. After there are no more agents with negative values, any interaction with an agent whose value is less than r will decrease the number of such agents.

As a concrete example, we specify a protocol over the alphabet $\{a, b, c\}$ to determine whether the number of occurrences of a exceeds three times the number of occurrences of b by at least 2. Thus v is the vector $(1, -3, 0)$ and r is the value 2. Hence the required range of data values is $D = \{-3, -2, -1, 0, 1, 2, 3\}$, because the initial data values may be 1, -3 or 0, and the range must include $2r - 1 = 3$. States consist of pairs (x, y) where $x \in \{A, P\}$ indicates whether the agent is active or passive, and $y \in D$. Then ι maps a to $(A, 1)$, b to $(A, -3)$ and c to $(A, 0)$, and o maps (x, y) to 1 if y is at least 2, and 0 otherwise. The transition function is defined as follows. Note that the case resulting in averaging means that the update to the initiator depends on the state of the responder.

$$\delta((x_1, y_1), (x_2, y_2)) := \begin{cases} ((P, y_1), (A, y_1 + y_2)) & \text{if } x_1 = A, x_2 = A \\ & \quad \text{and } y_1 + y_2 \in D \\ ((A, \lceil \frac{y_1+y_2}{2} \rceil), (A, \lfloor \frac{y_1+y_2}{2} \rfloor)) & \text{if } x_1 = A, x_2 = A \\ & \quad \text{and } y_1 + y_2 \notin D \\ ((P, y_1), (A, y_1)) & \text{if } x_1 = A, x_2 = P \\ ((x_1, y_1), (x_2, y_2)) & \text{otherwise} \end{cases}$$

8.2 Queued transmission

When combined with Theorem 4, the following theorem implies that every semilinear predicate on $\mathrm{Pop}(\Sigma)$ is stably computable in the queued transmission model.

Theorem 5 *Every predicate on* $\mathrm{Pop}(\Sigma)$ *stably computable in the two-way model is stably computable in the queued transmission model.*

Proof Given any protocol in the two-way model, we describe a simulation of it by a protocol in the queued transmission model. Each agent can store up to two states from the two-way protocol. Initially, each agent has one simulated state, which is determined from the input symbol for that agent by the initial state map of the simulated protocol. Agents transfer states by sending a message. An agent called upon to send will send a message containing the state it has been holding longest. If it is not currently holding any states, it sends a null message. Whenever an agent receives a null message ignores it, that is, does not change its state. Any agent with free space is eligible to receive a non-null message. Whenever an agent receives its second state, it uses the transition function from the two-way protocol to have the two interact, with the state it already has acting as the initiator.

The simulated configuration cannot make any steps that were impossible under the two-way model. Conversely, any sequence of interactions made under the two-way model can be achieved by first having every agent launch all states it holds, and then by repeating the following actions: deliver two states to a particular agent, and then have the agent release both in messages. Therefore, the simulation is faithful.

To see that every fair execution of the simulating protocol is a fair execution of the simulated protocol, we consider a fair execution $c_0, c_1, c_2, \ldots$ of the simulating protocol. Let D be the set of all configurations d with no null messages in transit such that $c_0 \xrightarrow{*} d$; then D is finite and for each j, some element of D is reachable from c_j (by delivering all the null messages). Thus by fairness, there exists some $d \in D$ such that $c_j = d$ for infinitely many j.

Now consider $c'_0, c'_1, c'_2, \ldots$, obtained by projecting out the simulated states (whether held by simulating agents or in the multiset of undelivered messages) and deleting steps that do not correspond to steps in the simulated protocol. Let d' be the projection of d; then $c'_j = d'$ for infinitely many j. For any configuration e' of the simulated protocol either $c'_j \stackrel{*}{\not\to} e'$ for some j, or $c'_j \stackrel{*}{\to} e'$ for infinitely many j. In the second case, $d' \stackrel{*}{\to} e'$ via some sequence of steps. Simulating that sequence of steps starting with d reaches some configuration e of the simulating protocol. By the fairness of $c_0, c_1, c_2, \ldots$, we must have $c_j = e$ for infinitely many j, and therefore $c'_i = e'$ for infinitely many i, establishing the fairness of the projected execution $c'_0, c'_1, c'_2, \ldots$.

Thus, if the two-way protocol stably computes the predicate ψ, then for every input x and every fair execution of the simulating protocol from the input configuration, the simulated protocol experiences a fair execution and must reach a correct output stable configuration, which is also a correct output stable configuration for the simulating protocol for ψ on input x. □

8.3 Immediate and delayed transmission

Theorem 6 *Every predicate on* $\mathrm{Pop}(\Sigma)$ *in the class* **SLIN** ∩ **coreMOD** *is stably computable in the immediate transmission model and also in the delayed transmission model.*

Proof We first show that in both models, the class of stably computable predicates is closed under boolean operations and includes all the simple threshold predicates and predicates in **MOD**. For immediate transmission, as we previously noted, the construction in Lemma 1 implies closure under boolean operations. Also, the two-way protocols given in Sect. 8.1 for simple threshold and modulo predicates are immediate transmission protocols.

For the delayed transmission model, Lemma 3 shows closure under boolean operations. To see that simple threshold and modulo predicates are computable in this model, we describe protocols as follows. Every agent state is active or passive; all are initially active. A sender sends its state and becomes passive, preserving its previous output. Passive messages are received and ignored by all agents. An active receiver combines its data with the data from an active message, remaining active. (The data values u and v are combined as $(u+v) \bmod m$ for a modulo predicate or $\min(k, u+v)$ for a simple threshold predicate.) A passive receiver sets its state equal to an active message, becoming active again. Consider a fair execution $c_0, c_1, c_2, \ldots$ from the input configuration $I(x)$ and the set D of configurations of this population in which there is exactly one active agent and no messages in transit and every agent has the same output. The set D is finite and some element of D is reachable from every c_j, so by fairness there is some $d \in D$ such that $c_j = d$ for infinitely many d. Once d is reached, the agents all compute the correct output value for the input x, and the configuration is output stable.

Suppose $\psi \in$ **SLIN**∩**coreMOD**. For each nonempty subset Σ' of the input alphabet Σ, we describe a protocol that computes ψ assuming that Σ' is exactly the set of symbols present in the input x. We can "and" this protocol with one that verifies this assumption (it is a boolean combination of simple threshold predicates, and therefore stably computable in both models), and "or" together the resulting protocols for each choice of Σ', and the result will stably compute ψ.

Because $\psi \in$ **coreMOD**, there is a predicate $\psi' \in$ **MOD** and an integer $k \geq 0$ such that ψ is k-similar to ψ' with respect to Σ'. Then ψ' is stably computable by an immediate (or delayed) transmission protocol, as is the predicate that is true if x is k-rich with respect to Σ', so the conjunction of these predicates correctly computes ψ when there are at least k occurrences of each input symbol from Σ' and no occurrences of any other symbol.

In the remaining cases, some $\sigma \in \Sigma'$ occurs between 1 and $k-1$ times. Because $\psi \in$ **SLIN**, it is stably computable by a two-way protocol. For each input symbol $\sigma \in \Sigma'$ we describe below how to adapt the simulation of the two-way protocol in the proof of Theorem 5 to show that the predicate that is the conjunction of ψ and the predicate that is true when there are between 1 and $k-1$ occurrences of σ in the input is stably computable in the immediate (or delayed) transmission models. Then we "or" these protocols together for all choices of $\sigma \in \Sigma'$. The simulation of a two-way protocol by a queued transmission protocol requires queuing in order to prevent agents' storage from being overwhelmed by messages. Suppose it may be assumed, however, that at least 1 and at most $k-1$ copies of σ can appear in the input. The agents distinguished by receiving input σ then allow a protocol to simulate flow control.

Each agent keeps a queue of simulated states, containing initially the simulated state determined by that agent's input and the input map from the simulated protocol. Agents also keep an additional 1-bit value that represents the minimum number of simulated states that they must keep in their queues; agents with input σ have value 0, and all others have value 1. No interactions cause agents to change these 1-bit values. When an agent transmits, if it can dequeue a simulated state without violating the above invariant, it does so and sends the state out. Otherwise, it sends a "trigger" message. On the other end, when an agent receives a state, it enqueues it. When an agent receives a trigger message, if it has at least two states in its queue, it simulates an interaction initiated by the state at the head of the queue with the state just behind it; otherwise, it does nothing.

We must check several properties of this simulation. First, no agent can ever have more than k states in its queue, so the simulation can be carried out by finite-state agents. In a

population of n agents, at least $n - (k - 1)$ agents, namely those whose input was not σ, have at least one state in their queue by the invariant. There are at most $k - 1$ remaining states, which may be distributed arbitrarily, but it is not possible to exceed the maximum of k for any agent. Second, at every step the simulated configuration can always be reached from the input under the protocol being simulated, because the simulation ensures that the number of simulated agents remains constant, and the only changes to the simulated population are made in accordance with the transition rules of the protocol. Third, the fairness condition holds with respect to the simulated populations, which, together with the first two propositions, ensures that the simulation is faithful.

In the case of immediate transmission, the set of configurations reachable from a particular input is finite, and it suffices to show that any configuration reachable by the simulated population can be reached by the simulation. Assume without loss of generality that the population has at least two agents, and induct on the number of steps needed to reach the desired configuration. Choosing a simulated initiator and responder, the first objective is for the initiator to be alone in some agent's queue. If there are simulated agents ahead of the initiator, send them to another agent. If the initiator is at the head of the queue with one or more simulated agents behind it, some agent's queue is empty by an averaging argument, and the initiator can be sent to that agent.

The second objective is to move the responder behind the initiator. In the event that the initiator occupies the only agent with value 0, the initiator should be sent to an agent with value 1, and all of the simulated agents now ahead of it should be sent elsewhere. If there are only two agents, the responder now resides in the queue of an agent with value 0 and can be sent to the initiator's queue. Otherwise, all simulated agents ahead of the responder should be sent to a third agent. Now, either the responder can be sent to the initiator's queue, or it is alone in the queue of an agent with value 1. In the latter case, some other agent besides the one with the initiator has an extra agent that can be sent to the responder's queue to release the responder. Finally, once the initiator and responder share a queue in the right order, some agent has no simulated agents and can trigger an interaction.

For the case of delayed transmission, we note that every configuration reachable in the simulated population is reachable in the simulation by the same inductive argument as for immediate transmission, because each immediate transmission step can be accomplished by a send immediately followed by a receive in the delayed transmission model. As for the fairness condition, it is always the case by an averaging argument that some agent has zero or one simulated agents, and thus it is possible to deliver all trigger messages in flight without causing an interaction. We finish along the lines of Lemma 3. □

Note in particular that immediate and delayed transmission protocols can stably compute the exactly-one c comparison predicate over the alphabet $\{a, b, c\}$, which is true if there is exactly one c and the number of as exceeds the number of bs. The characterization in Sect. 14.1 shows that the standard comparison predicate over $\{a, b\}$, which is true if the number of as exceeds the number of bs, is not stably computable by immediate or delayed transmission protocols, essentially because no such control of incoming messages is possible in this case.

8.4 Immediate observation

Theorem 7 *Every predicate on* $\mathrm{Pop}(\Sigma)$ *in* $\mathbf{COUNT}_*$ *is stably computable in the immediate observation model.*

Proof Every predicate in $\mathbf{COUNT}_*$ can be expressed as a boolean combination of simple threshold predicates, each of which depends only on the count of one input symbol. The two-way protocol in Sect. 8.1 for the simple threshold predicate $[x \cdot \sigma \geq k]$ is in fact an immediate observation protocol because only the responder's state is updated in each case. Also, the class of predicates stably computable by immediate observation protocols is closed under boolean operations. □

The protocol from Sect. 8.1 uses $k + 1$ states, which can be reduced to k (by removing state 0) if the input alphabet is unary. It can be shown that $k - 1$ states are not sufficient to stably compute this predicate in the immediate observation model in the case of a unary alphabet; the proof is rather involved and will appear elsewhere.

8.5 Delayed observation

Theorem 8 *Every predicate on* $\mathrm{Pop}(\Sigma)$ *in* $\mathbf{COUNT}_1$ *is stably computable in the delayed observation model.*

Proof The value of a predicate ψ in $\mathbf{COUNT}_1$ is completely determined by the set of symbols that occur in the input. We define a protocol in which the states are subsets of the set of input symbols, the initial state of an agent is $\{\sigma\}$ if σ is the input symbol assigned to that agent and the output is determined by the value of ψ for the agent's current state. When called upon to send, an agent sends its current state and does not update it, consistent with the observation restriction. When an agent receives a message, it updates its state to be the union of its current state and the message.

Consider any fair execution of this protocol, $c_0, c_1, c_2, \ldots$, starting with the input configuration $I(x)$, and let d be the configuration of this population in which there are no undelivered messages and every agent's state is equal to the set of symbols that occur in the input x. Then d is reachable from every c_j, and by fairness, must occur infinitely often.

Because d is output stable and has the correct output, this protocol stably computes ψ. □

Because an agent may receive its own message, it cannot tell whether an input symbol occurs with multiplicity greater than one. However, if agents do not receive messages from themselves, then predicates in $\mathbf{COUNT}_2$ are stably computable in the delayed observation model.

9 Characterization of the power of the abstract model

To complete the characterizations in Fig. 2, we have to demonstrate that each model is limited to the indicated power. In Sect. 14, we consider the one-way models. Here we consider the Abstract Model, which includes the two-way model and the queued transmission model as special cases, and prove the following semilinearity theorem.

Theorem 9 *Every predicate on* $\mathrm{Pop}(\Sigma)$ *that is stably computable in the Abstract Model is semilinear.*

Because the two-way and queued transmission models are special cases of the Abstract Model, and both models can stably compute all the semilinear predicates over $\mathrm{Pop}(\Sigma)$, we have the following characterizations.

Corollary 10 *The predicates stably computable in* (1) *the two-way model of population protocols, or* (2) *the queued transmission model of population protocols are exactly the semilinear predicates on* $\mathrm{Pop}(\Sigma)$.

Stable computation of a predicate by a two-way population protocol with stabilizing inputs was defined in [1]. We do not repeat the definition here, but the idea is that each agent has an input register that may change finitely many times over the course of the execution before stabilizing to its final value, and the goal is to compute a predicate on the multiset of all the agents' final input values. Because this is a more restrictive definition, every predicate stably computable by a two-way protocol with stabilizing inputs is stably computable by a two-way protocol with fixed inputs, but whether the converse held was left as an open problem. However, in [1] it was shown that every semilinear predicate on $\mathrm{Pop}(\Sigma)$ is stably computable by a two-way protocol with stabilizing inputs. Thus we get the following corollary showing that stabilizing inputs do not reduce the power of the standard two-way model.

Corollary 11 *The predicates stably computable by two-way population protocols with stabilizing inputs are exactly the semilinear predicates on* $\mathrm{Pop}(\Sigma)$.

To prove Theorem 9, we must show that the support of any stably-computable predicate is a semilinear set: in particular, that there is a finite set of base points each attached to a finitely-generated cone such that the support is precisely the elements of these cones. The first step, which requires the development of the machinery of Sect. 10 and is completed in Sect. 11, is to show that the support can be decomposed into a finite collection of monoid cosets that are *not necessarily finitely generated*. We then proceed, in Sects. 12 and 13, to show that any such decomposition can be further decomposed into a finite covering by cosets of finitely generated monoids, which gives us the full result.

10 Groundwork

We assume that ψ is a predicate stably computed by a protocol $(E, \to, \Sigma, o)$ in the Abstract Model and establish some basic results. These will lead up to the Pumping Lemma of Sect. 11.

10.1 Monoids, groups, and semilinearity

A subset M of $\mathbb{Z}^d$ is a **monoid** if it contains the zero and is closed under addition; if it is also closed under subtraction, M is a **group**. A monoid $M \subseteq \mathbb{Z}^d$ is **finitely generated** if there exists a finite subset $A \subseteq M$ such that every element of M is a sum of elements from A. It is a classic result in abstract algebra that every subgroup of $\mathbb{Z}^d$ is finitely generated [33], but submonoids are not always finitely generated. For example, the following monoid is not finitely generated.

$$M_{\sqrt{2}} = \{(i, j) \in \mathbb{N}^2 : i \leq \sqrt{2}j\}.$$

A subset H of $\mathbb{Z}^d$ is a **group coset** (resp., **monoid coset**) if there exists an element $v \in \mathbb{Z}^d$ such that $H = v + G$ and G is a group (resp., monoid).

Then from the definitions in Sect. 6.1, we have yet another characterization of linear and semilinear sets. A subset L of $\mathbb{Z}^d$ is **linear** if it is a coset of a finitely generated monoid in $\mathbb{Z}^d$, and is **semilinear** if it is a finite union of linear sets.

10.2 Higman's lemma

We shall make extensive use of some corollaries to Higman's Lemma [27], a fundamental tool in well-quasi-order theory.

Lemma 12 *Every subset of* $\mathbb{N}^d$ *has finitely many minimal elements under* $\leq$.

Lemma 13 *Every infinite subset of* $\mathbb{N}^d$ *contains an infinite chain* (*i.e., an infinite totally ordered sequence*) *under* $\leq$.

These both follow from the fact that Higman's Lemma implies that $\mathbb{N}^d$ ordered by $\leq$ is a **well-quasi-order**, that is, a set in which any infinite sequence $a_1, a_2, \ldots$ contains elements a_i, a_j with $i < j$ and $a_i \leq a_j$. The special case of

Higman's Lemma given in Lemma 12 was proved earlier by Dickson [19].

10.3 Truncation maps and their properties

Recall from the description of the Abstract Model in Sect. 5.5 that E is the set of elements (states or messages) of a population protocol, and that $\mathcal{C}$ denotes the set of configurations, that is, the set of nonempty multisets of elements of E, which we also think of vectors of natural numbers indexed by E.

For each $k \geq 1$, we define a map τ_k from $\mathcal{C}$ to $\mathcal{C}$ by

$$\tau_k(c)(e) := \min(k, c(e)) \quad \text{for all } e \in E.$$

This map truncates each component of its input to be at most k; clearly $\tau_k(c) \leq c$ for all $c \in \mathcal{C}$. Two useful properties of τ_k are that it respects both the inclusion ordering and addition.

Lemma 14 *For all $c, d \in \mathcal{C}$ and $k \geq 1$, if $c \leq d$ then $\tau_k(c) \leq \tau_k(d)$.*

Proof For each $e \in E$, we have $c(e) \leq d(e)$, so $\min(k, c(e)) \leq \min(k, d(e))$. Thus $\tau_k(c) \leq \tau_k(d)$. □

Lemma 15 *For all $c, c', d \in \mathcal{C}$ and $k \geq 1$, if $\tau_k(c) = \tau_k(c')$, then $\tau_k(c + d) = \tau_k(c' + d)$.*

Proof For each $e \in E$, either $c(e) = c'(e)$ or both are at least k. In either case, $\min(k, c(e)+d(e)) = \min(k, c'(e)+d(e))$, so $\tau_k(c + d) = \tau_k(c' + d)$. □

10.4 Truncation and stability

Truncation is important because membership of a configuration c in the set of output stable configurations $\mathcal{S}$ can be determined from a truncate of fixed size. Let $\mathcal{U} := \mathcal{C} \setminus \mathcal{S}$, the set of **output unstable** configurations.

Lemma 16 *For all $c \leq d$, if $c \in \mathcal{U}$, then $d \in \mathcal{U}$ ($\mathcal{U}$ is closed upward under inclusion).*

Proof Suppose $c \in \mathcal{U}$ and $c \leq d$. Then either (1) $O(c)$ is undefined, or (2) $O(c) = b$, but for some configuration c' such that $c \xrightarrow{*} c'$ either $O(c')$ is undefined or $O(c') \neq b$. In case (1), $O(d)$ is undefined and $d \in \mathcal{U}$. In case (2), $d = d - c + c \xrightarrow{*} d - c + c'$. If $O(d)$ is not defined, then $d \in \mathcal{U}$. If $O(c')$ is not defined, then $O(d - c + c')$ is not defined, and $d \in \mathcal{U}$. If both $O(d)$ and $O(c')$ are defined, we have that $O(d) = O(c) \neq O(c')$, and $O(d-c+c')$ is either undefined or equal to $O(c')$, so $d \in \mathcal{U}$. Thus $\mathcal{U}$ is closed upwards under inclusion. □

Lemma 17 *There exists $k \geq 1$ such that $c \in \mathcal{U}$ if and only if $\tau_k(c) \in \mathcal{U}$.*

Proof By Higman's Lemma, only finitely many elements $u_1, \ldots, u_n$ are minimal in $\mathcal{U}$, and because $\mathcal{U}$ is upwards closed, $c \in \mathcal{U}$ if and only if $u_i \leq c$ for some i. Let k be the maximum value $u_i(e)$ for all $i \in \{1, \ldots, n\}$ and all $e \in E$. Then $\tau_k(u_i) = u_i$ for each i.

Suppose $c \in \mathcal{U}$. Then $u_i \leq c$ for some i, so $u_i = \tau_k(u_i) \leq \tau_k(c)$ by Lemma 14, and thus $\tau_k(c) \in \mathcal{U}$. Conversely, if $\tau_k(c) \in \mathcal{U}$, then $u_i \leq \tau_k(c) \leq c$ for some i, and therefore $c \in \mathcal{U}$. □

Lemma 18 *There exists $k \geq 1$ such that for all $c \in \mathcal{C}$ and $b \in \{0, 1\}$, we have $c \in \mathcal{S}_b$ if and only if $\tau_k(c) \in \mathcal{S}_b$.*

Proof By Lemma 17, there exists $k \geq 1$ such that for all $c \in \mathcal{C}$, we have $c \in \mathcal{U}$ if and only if $\tau_k(c) \in \mathcal{U}$. Taking the contrapositive, we have $c \in \mathcal{S}$ if and only if $\tau_k(c) \in \mathcal{S}$. Since truncation does not affect output, the conclusion follows. □

10.5 Extensions

We define a map X from $\mathcal{C}$ to subsets of $\mathbb{N}^\Sigma$ as follows.

$$X(c) := \{x \in \mathbb{N}^\Sigma \mid \text{there exists } d \geq c \text{ such that } c + x \xrightarrow{*} d \text{ and } \tau_k(c) = \tau_k(d)\},$$

where k is the constant from the conclusion of Lemma 18. If $c \in \mathcal{S}$, then $X(c)$ is the set of inputs by which c can be pumped. We call such inputs the **extensions** of c. We first prove that pumping does not affect stable output.

Lemma 19 *If $x \in \mathrm{Pop}(\Sigma)$ and $c \in \mathcal{S}$ and $x \xrightarrow{*} c$, then ψ is constant on $x + X(c)$.*

Proof If $y \in X(c)$, then there exists $d \in \mathcal{C}$ such that $c + y \xrightarrow{*} d$ and $\tau_k(c) = \tau_k(d)$. Since $c \in \mathcal{S}$, by Lemma 18 we have $d \in \mathcal{S}$, and $O(c) = O(d)$. Thus $\psi(x) = \psi(x + y)$. □

We now prove that pumping operations can be composed, i.e., that $X(c)$ is a monoid.

Lemma 20 *$X(c)$ is a monoid for all $c \in \mathcal{C}$.*

Proof We have $0 \in X(c)$, with $d = c$ as a witness. If $x_1, x_2 \in X(c)$, then there exist d_1, d_2 such that $c \leq d_1$ and $c \leq d_2$ and $\tau_k(c) = \tau_k(d_1) = \tau_k(d_2)$ and $c + x_1 \xrightarrow{*} d_1$ and $c + x_2 \xrightarrow{*} d_2$. Thus

$$c + x_1 + x_2 \xrightarrow{*} d_1 + x_2 = (d_1 - c) + c + x_2 \xrightarrow{*} (d_1 - c) + d_2.$$

Taking $d := d_1 + d_2 - c$, we have $c \leq d$ and $c + x_1 + x_2 \xrightarrow{*} d$ and $\tau_k(c) = \tau_k(d_2) = \tau_k(c + d_2 - c)$ and by Lemma 15, $\tau_k(c + d_2 - c) = \tau_k(d_1 + d_2 - c) = \tau_k(d)$, since $\tau_k(c) = \tau_k(d_1)$. We conclude that $x_1 + x_2 \in X(c)$. □

11 A pumping lemma for stably computable predicates

Given a set of inputs $Y \subseteq \mathrm{Pop}(\Sigma)$, a **monoid-coset covering of Y with respect to** ψ is a set $\{(x_i, M_i)\}_{i\in I}$ of pairs of inputs and submonoids of $\mathbb{N}^\Sigma$ such that $Y \subseteq \bigcup_{i\in I}(x_i + M_i)$ and for all $i \in I$, we have $x_i \in Y$ and $\psi(x_i + M_i) = \{\psi(x_i)\}$. We say ψ **admits finite coset coverings** if for all Y there exists a finite monoid-coset covering of Y with respect to ψ. The following Pumping Lemma states that every stably computable predicate admits finite coset coverings. We show later (Theorem 24) that any predicate such that the predicate and its complement both admit finite coset coverings is semilinear.

Lemma 21 *Every stably computable predicate ψ admits finite coset coverings.*

Proof Consider any $Y \subseteq \mathrm{Pop}(\Sigma)$. If Y is a finite set $\{x_1, \ldots, x_n\}$ then it has a trivial finite covering $\{(x_i, \emptyset)\}_{1\le i\le n}$. So assume Y is infinite. Let $y_1, y_2, \ldots$ be any enumeration of Y such that $y_i \le y_j$ implies $i \le j$. (For example, fix an ordering of Σ. Then enumerate the elements of Y ordered by cardinality and then lexicographic order.) We define a family of sets $B_i \subseteq \mathrm{Pop}(\Sigma) \times \mathcal{S}$ inductively as follows.

$B_0 := \emptyset$.

$$B_i := \begin{cases} B_{i-1} \text{ if there exists } (x, c) \in B_{i-1} \text{ such that} \\ \quad y_i \in x + X(c) \\ B_{i-1} \cup \{(y_i, s(y_i))\} \cup \{(y_i, s(c + y_i - x)) \mid (x, c) \\ \quad \in B_{i-1} \text{ and } x \le y_i\} \text{ otherwise,} \end{cases}$$

where $s(d) \in \mathcal{S}$ is any stable configuration reachable from d. It is easy to see by induction on i that, for all $(x, c) \in B_i$, $x \xrightarrow{*} c$, and in the construction, $s(d)$ is applied only to reachable configurations d, so the existence of $s(d)$ is guaranteed by the requirements of stably computing a predicate. So, Lemma 19 implies that ψ is constant on $x + X(c)$. Let $B := \bigcup_{i\ge 1} B_i$. It follows from Lemma 20 that $\{(x, X(c)) \mid (x, c) \in B\}$ is a monoid-coset covering of Y with respect to ψ. We now show that B is finite to prove that this is a finite covering.

Assuming to the contrary that B is infinite, infinitely many different elements of Y appear as first components of elements of B, since each B_i is clearly finite. By Higman's Lemma, there exists an infinite chain $z_1 < z_2 < \cdots$ of such elements. Our construction guarantees the existence of associated configurations $\{d_i\}_{i\ge 1}$ such that $(z_i, d_i) \in B$ and $d_i + (z_{i+1} - z_i) \xrightarrow{*} d_{i+1}$ for all $i \ge 1$.

Consider the sequence $d_1, d_2, d_3, \ldots$; either only finitely many values occur (recall we are considering the Abstract Model, not just the two-way model) or infinitely many values occur and we can apply Higman's Lemma again. In either case, there exists an increasing function f such that $d_{f(1)} \le d_{f(2)} \le d_{f(3)} \le \cdots$. Thus the sequence $\tau_k(d_{f(i)})$ reaches a maximum and becomes constant at some index $i = j$. Consequently, we have $z_{f(j+1)} - z_{f(j)} \in X(d_{f(j)})$, which contradicts the membership of $(z_{f(j+1)}, d_{f(j+1)})$ in B. □

Applying this lemma with $Y = \psi^{-1}(1)$ (the support of ψ) we obtain a finite family of monoid cosets $x_i + M_i$ such that

$$\psi^{-1}(1) = \bigcup_i (x_i + M_i).$$

This does not prove semilinearity by itself, since some monoid M_i might not be finitely generated. However, every submonoid of $\mathbb{Z}^1$ is finitely generated, so in the special case of a *unary* alphabet, we have already that ψ is semilinear, and therefore regular.

Corollary 22 *Every stably computable predicate over a unary alphabet is semilinear. Thus, over a unary alphabet, the stably computable predicates are exactly the semilinear (in fact, regular) predicates.*

For example, the unary predicate that is true if the number of input symbols is a power of 2 (or a prime, or any other nonregular predicate) is not stably computable. Another easy corollary suffices to show certain other predicates over nonunary alphabets are not stably computable.

Corollary 23 *Suppose ψ is a stably computable predicate such that $L = \psi^{-1}(1)$ is infinite. Then L contains an infinite linear subset.*

Proof By Lemma 21 there is a finite monoid-coset covering of L. If L is infinite, some $(x + M) \subseteq L$ in the covering must be infinite. □

As an application, consider the set of inputs over the alphabet $\{a, b\}$ such that the number bs is the square of the number of as. This is an infinite set with no infinite linear subset, and is therefore not stably computable. Using closure results for stably computable predicates we can then show that the set of all inputs over alphabet $\{a, b, c\}$ such that the number of cs is the product of the number of as and the number of bs is not stably computable. The existence of a pumping lemma and the negative results for these particular predicates were conjectured in [3,4].

12 Proof of the semilinearity theorem: outline

We are now in a position to give an overview of how we go from the pumping lemma (Lemma 21) of Sect. 11 to the semilinearity theorem (Theorem 9), via Theorem 24.

Recall the following set of points in $\mathbb{N}^2$.

$$M_{\sqrt{2}} = \{(i, j) : i \le \sqrt{2}j\}.$$

This is a monoid but is not stably computable. To see this, suppose the contrary. By the Pumping Lemma (Lemma 21), there exist monoid cosets $x_i + M_i$ for $1 \leq i \leq m$ such that

$$M_{\sqrt{2}} = \bigcup_{1 \leq i \leq m} (x_i + M_i).$$

Let $v = (-1, \sqrt{2})$; then $x \cdot v > 0$ for all $x \in M_{\sqrt{2}}$. Let $\epsilon = \min_i \{x_i \cdot v\}$. Choose some $y \in M_{\sqrt{2}}$ such that $0 < y \cdot v < \epsilon$. Then for some i, $y \in x_i + M_i$, and $(y - x_i) \cdot v = y \cdot v - x_i \cdot v < \epsilon - \epsilon = 0$. Thus for a sufficiently large $n \in \mathbb{N}$, $(x_i + n(y - x_i)) \cdot v < 0$, which contradicts the fact that $x_i + M_i$ is a subset of $M_{\sqrt{2}}$. The issue here is that the line dividing the positive and negative inputs cannot have an irrational slope if the predicate is stably computable. One ingredient of our proof is a generalization of this idea to separating hyperplanes.

However, to be able to use separating hyperplanes, we first must deal with separating "intermixed" positive and negative inputs using their images in a finite group. For example, consider the following set of points in $\mathbb{N}^2$, which is stably computable.

$$L = \{(i, j) : i < j, (i + j) \text{ is odd}\}.$$

By first separating the points in $\mathbb{N}^2$ by their images (i mod 2, j mod 2) in the group $\mathbb{Z}_2 \times \mathbb{Z}_2$, we get four subproblems in which lines suffice to separate the positive and negative points.

A further issue is that because of the relationship between the monoids $X(c)$ and their cosets $x + X(c)$, instead of a single separating hyperplane we have to consider finite sets of parallel hyperplanes, which themselves may contain points from the support of ψ. These points can be handled by induction on dimension, but this requires that the main theorem be generalized to consider the intersection of the support of ψ and an arbitrary group coset (where we view each hyperplane as a group coset). We shall prove the following theorem by induction on the dimension of the group G; it clearly holds when G has dimension 0, i.e., when G is the trivial group.

Theorem 24 *If ψ and its complement admit finite coset coverings then for any group coset $H = x_0 + G \subseteq \mathbb{Z}^\Sigma$, we have $\psi^{-1}(1) \cap H$ is semilinear.*

This theorem, together with Lemma 21 can now be used to prove Theorem 9. If ψ is stably computable in the Abstract Model, then so is its complement (since the output function of the protocol can be negated). Thus, by Lemma 21, both ψ and its complement admit finite coset coverings. Applying Theorem 24 with $H = \mathbb{Z}^\Sigma$ establishes Theorem 9.

The details of the proof of Theorem 24 are quite involved and are in Sect. 13. To summarize briefly, the overall strategy is:

1. By dividing the space into residue classes with respect to appropriately chosen moduli, we can arrange for the vectors from the monoids associated with the cover to appear in all-positive and all-negative regions separated by hyperplanes. In this part of the proof we extend $\mathbb{N}^\Sigma$ to $\mathbb{Z}^\Sigma$ to make use of the fact that all subgroups of $\mathbb{Z}^\Sigma$ are finitely generated. (Sect. 13.1.)
2. We then further map the problem from $\mathbb{Z}^\Sigma$ to $\mathbb{R}^\Sigma$, and use techniques from convex geometry to show that appropriate hyperplanes separating the monoids indeed exist. (Sect. 13.2.) We obtain a (looser) separation by **slabs** (the space between two parallel hyperplanes) on the inputs by observing that each input is displaced a uniformly bounded amount from its corresponding extension vector.
3. Moving to $\mathbb{R}^\Sigma$ allows for the possibility of separating hyperplanes with irrational coefficients. Applying the Pumping Lemma as described above, we show that the resulting separating hyperplanes are normal to a vector with rational coordinates and thus correspond to cosets of subgroups of $\mathbb{Z}^\Sigma$. (Sect. 13.3.)
4. At this stage, we have shown that the positive inputs to the predicate consist of (a) those inputs in the interior of the separated regions, which can by identified first by computing the residue classes of their coordinates and then by identifying which of a finite number of polytopes (given by intersections of half-spaces with rational coordinates) they appear within, and (b) those inputs that lie in some slab. The first class is semilinear as identification of a residue class and identification of membership in a particular polytope are both expressible in Presburger arithmetic. The second class is then shown to be semilinear by induction on dimension, with the base case of dimension 0 being the trivially semilinear case of a single point. (Sect. 13.4.)

13 Proof of the semilinearity theorem: details

The proof of Theorem 24 is delayed to Sect. 13.4. First we describe the process of separating inputs in more detail.

13.1 Separating intermixed inputs

Let $\psi : \text{Pop}(\Sigma) \to \{0, 1\}$ be a predicate that admits finite coset coverings. Let G be a subgroup of $\mathbb{Z}^\Sigma$ and let $H = h + G$ be a coset of G. Take $\{(a_i, M_i)\}_{i \in I}$ and $\{(b_j, N_j)\}_{j \in J}$ to be finite monoid-coset covers of $\psi^{-1}(0) \cap H$ and $\psi^{-1}(1) \cap H$ respectively. Assume without loss of generality that $M_i, N_j \subseteq G$ by intersecting with G if necessary. Define $K(i, j) := \mathbb{Z}(M_i \cap N_j)$, the group generated by the intersection of M_i and N_j.

Lemma 25 *For all $i \in I$ and $j \in J$, no coset of $K(i, j)$ intersects both $a_i + M_i$ and $b_j + N_j$.*

Proof If we suppose the lemma is false, then there exist $x \in a_i + M_i$ and $x' \in b_j + N_j$ such that $(x - x') \in K(i, j)$. Because $M_i \cap N_j$ is a monoid, we can rewrite $x - x'$ as a difference $y' - y$ where $y \in M_i$ and $y' \in N_j$. Thus $x + y = x' + y'$. The former is in $a_i + M_i$, whereas the latter is in $b_j + N_j$. This contradicts the fact that $a_i + M_i$ and $b_j + N_j$ are disjoint (since they have different predicate values). □

Define

$$K := \bigcap_{K(i,j) \text{ has finitely many cosets in } G} K(i, j).$$

In addition to having finitely many cosets in G, the group K inherits the relevant properties of the groups $K(i, j)$ included in its defining intersection.

Lemma 26 *For all $i \in I$ and $j \in J$ such that $K(i, j)$ has finitely many cosets in G, no coset of K intersects both $a_i + M_i$ and $b_j + N_j$.*

Proof Since $K(i, j) \supseteq K$, each coset of K is contained in a coset of $K(i, j)$. The result follows from Lemma 22. □

Since all group cosets are semilinear, we can use membership in the cosets of K to separate those pairs of monoid cosets that are full rank. For the others, we have to take another approach.

13.2 Separating with hyperplanes

Let $g_1 + K, \ldots, g_n + K \subseteq G$ be the cosets of K in G. For each $\ell \in \{1, \ldots, n\}$ define $H_\ell := h + g_\ell + K$. Note that $\cup_\ell H_\ell = H$. If two monoid cosets $a_i + M_i$ and $b_j + N_j$ both meet some coset $h + (g_\ell + K) \subseteq H$, it is possible to separate $H_\ell \cap (a_i + M_i)$ from $H_\ell \cap (b_j + N_j)$ with a hyperplane (except for a set of lower dimension). Define $I_\ell := \{i \in I \mid (a_i + M_i) \cap H_\ell \neq \emptyset\}$ and $J_\ell := \{j \in J \mid (b_j + N_j) \cap H_\ell \neq \emptyset\}$. If $i \in I_\ell$ and $j \in J_\ell$, then H_ℓ, which is a coset of K, intersects both $a_i + M_i$ and $b_j + N_j$, so $K(i, j)$ has infinitely many cosets in G, by Lemma 26.

At this point we turn to the methods of geometry. We pass from groups to vector spaces by working in the vector space closure $\mathbb{R}G$ of G in $\mathbb{R}^\Sigma$. Instead of monoids, we consider the convex cones they generate: the set of nonnegative linear combinations of monoid elements. We connect the geometry to the algebra by observing that $K(i, j)$ has finitely many cosets in G if and only if $\mathbb{R}K(i, j)$, the vector space closure of $K(i, j)$, is all of $\mathbb{R}G$. Thus if two monoid cosets are not intermixed, their intersection of their associated monoids has strictly smaller dimension than G. This allows us to separate these monoids with a hyperplane.

Formally, given sets of vectors U and U', a set of nonzero vectors V **distinguishes** U **and** U' if for all $u \in U$ and $u' \in U'$, there exists $v \in V$ such that $(u \cdot v)(u' \cdot v) \leq 0$; that is, either one dot product is zero or one is negative and the other is positive. Define $\widehat{M}_\ell := \bigcup_{i \in I_\ell} M_i$ and $\widehat{N}_\ell := \bigcup_{j \in J_\ell} N_j$. The goal of this subsection is to show the existence of a finite set of vectors V that distinguishes $\widehat{M}_\ell$ from $\widehat{N}_\ell$. The main tool we use is the Separating Hyperplane Theorem from convex geometry. Note that int U denotes the interior of U.

Theorem 27 (Separating Hyperplane Theorem [34]) *If U and U' are convex subsets of $\mathbb{R}^d$ with nonempty interiors such that* int $U \cap$ int $U' = \emptyset$*, then there exists $v \in \mathbb{R}^d$ such that $u \cdot v \leq 0$ for all $u \in U$ and $u' \cdot v \geq 0$ for all $u' \in U'$.*

Unfortunately, M_i and N_j are not convex. Thus we are forced to consider the convex cones that they generate. We need Carathéodory's theorem, another result from convex geometry, to verify the seemingly trivial fact that the intersection of these cones lies in a proper vector subspace of $\mathbb{R}G$.

Theorem 28 (Carathéodory's Theorem [34]) *For any set of vectors $Y \subseteq \mathbb{R}^d$, if $x \in \mathbb{R}_+ Y$, then there exists a linearly independent subset $Y' \subseteq Y$ such that $x \in \mathbb{R}_+ Y'$.*

Lemma 29 *For all $i \in I_\ell$ and $j \in J_\ell$, the set $Z = \mathbb{R}_+ M_i \cap \mathbb{R}_+ N_j$ is contained in a proper vector subspace of $\mathbb{R}G$.*

Proof Suppose to the contrary. Then the vector subspace $\mathbb{R}Z$ is all of $\mathbb{R}G$ and has a basis $\{z_1, \ldots, z_d\} \subseteq Z$. By perturbing this basis to have rational coordinates, we can assume without loss of generality that $z_s \in \mathbb{Q}G$ as well for all $s \in \{1, \ldots, d\}$.

By Theorem 28, for each s, there is a linearly independent set $Y_s \subseteq M_i$ such that $z_s \in \mathbb{R}_+ Y_s$. When we write each z_s as its unique linear combination of elements in Y_s, we see by linear algebra over $\mathbb{Q}G$ that in fact $z_s \in \mathbb{Q}_+ M_i$. Repeating this argument with N_j yields that $z_s \in \mathbb{Q}_+ M_i \cap \mathbb{Q}_+ N_j$. We clear denominators to find numbers m_s such that $m_s z_s \in M_i \cap N_j$. The set $\{m_s z_s\}$, however, is a basis for $\mathbb{R}G$, which contradicts the fact that $\mathbb{R}K(i, j)$ is not all of $\mathbb{R}G$. □

Lemma 30 *For all $i \in I_\ell$ and $j \in J_\ell$, there exists a nonzero vector v such that $\{v\}$ distinguishes M_i and N_j.*

Proof If either M_i or N_j is contained in a proper vector subspace of $\mathbb{R}G$, then take v to be normal to that subspace.

Otherwise, consider the sets $U := \mathbb{R}_+ M_i$ and $U' := \mathbb{R}_+ N_j$. The intersection of their interiors is both open and contained in a proper vector subspace of $\mathbb{R}G$. Thus U and U' have no interior point in common. Therefore, by Theorem 27 there exists $v \in \mathbb{R}G$ such that for all $u \in U$, we have $u \cdot v \leq 0$; and for all $u' \in U'$, we have $u' \cdot v \geq 0$. It follows that $\{v\}$ distinguishes M_i and N_j. □

To obtain a set of vectors that distinguish $\widehat{M}_\ell$ and $\widehat{N}_\ell$, we put together all of the individual distinguishing vectors.

Lemma 31 *For all $1 \leq \ell \leq n$, there exists a finite set of vectors V that distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$.*

Proof The set $V := \{v(i,j) \mid i \in I_\ell \text{ and } j \in J_\ell\}$ distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$, where $v(i,j)$ is the vector from Lemma 30 such that $\{v(i,j)\}$ distinguishes M_i and N_j. □

Combining results in this section and the previous one, we have some powerful criteria for determining the predicate value of an input.

Lemma 32 *Let V be a set of vectors that distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$. Suppose $x \in \widehat{M}_\ell$ and $y \in H_\ell$ are such that for all $j \in J_\ell$ and $v \in V$, we have $(x \cdot v)((y - b_i) \cdot v) > 0$. Then $\psi(y) = 0$.*

Proof Suppose to the contrary that $\psi(y) = 1$. Then there exists $j \in J_\ell$ such that $(y - b_j) \in N_j$. The set V, however, fails to distinguish x and $y - b_j$, which is a contradiction. □

Clearly, this lemma has an analogous counterpart that establishes sufficient conditions for $\psi(y) = 1$.

13.3 Achieving rationality

The chief obstacle yet to be overcome is that a distinguishing set of vectors might include vectors with irrational coordinates. In order to rule out predicates like $\psi(r,s) := [r < (\sqrt{2})s]$, we need to show that we can always distinguish $\widehat{M}_\ell$ and $\widehat{N}_\ell$ by vectors with integral coordinates. We must use for a second time the fact that ψ admits finite coset covers.

Lemma 33 *For all ℓ there exists a finite set of vectors from $\mathbb{Z}^\Sigma$ that distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$.*

Proof Since we can clear denominators, it is enough to find a set of vectors in $\mathbb{Q}^\Sigma$ that distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$. By Lemma 31 there exists a finite set of vectors V that distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$. Assume that $V \setminus \mathbb{Q}^\Sigma$ has minimum cardinality, that is, as few vectors in V have irrational components as possible. To avoid a special case later, we assume without loss of generality that V contains the standard basis Σ of $\mathbb{R}^\Sigma$. We show that this set V contains only vectors in $\mathbb{Q}^\Sigma$.

Suppose to the contrary that there exists $v \in V \setminus \mathbb{Q}^\Sigma$. By assumption, the set $V' := V \setminus \{v\}$ cannot distinguish $\widehat{M}_\ell$ and $\widehat{N}_\ell$. Thus there exist $i \in I_\ell$ and $j \in J_\ell$ and $x \in M_i$ and $y \in N_j$ such that for all $v' \in V'$ we have $(x \cdot v')(y \cdot v') > 0$.

We consider two cases. In the first case, for all choices of x and y we have $(x \cdot v)(y \cdot v) = 0$. Then at least one vector in each problem pair is normal to v. By linear algebra over $\mathbb{Q}^\Sigma$, there exists a vector $v' \in \mathbb{Q}^\Sigma$ such that if $w \in G$ is normal to v, then w is normal to v'. Thus $V' \cup \{v'\}$ distinguishes $\widehat{M}_\ell$ and $\widehat{N}_\ell$, which is a contradiction, since $(V' \cup \{v'\}) \setminus \mathbb{Q}^\Sigma$ has fewer elements than $V \setminus \mathbb{Q}^\Sigma$.

In the second case, we have x and y such that $(x \cdot v')(y \cdot v') > 0$ for all $v' \in V'$ and $(x \cdot v)(y \cdot v) < 0$. Assume without loss of generality that $x \cdot v < 0 < y \cdot v$ by taking $-v$ instead of v if necessary. Let $\Omega := \{w \in \mathbb{R}G \mid v' \cdot w > 0 \text{ for all } v' \in V \text{ and } w(q) > 0 \text{ for all } q \in \Sigma\}$. Clearly Ω is an open set. Also, $y \in \Omega$, since by the assumption that $\Sigma \subseteq V$ we have $y(q) > 0$ for all $q \in \Sigma$. Given that Ω is a nonempty open set, we can extend x, y to a basis $w_1 = x, w_2 = y, \ldots, w_m$ such that $w_s \in \Omega$ for all $s \geq 2$. By perturbing each w_s slightly to have rational coordinates and clearing denominators, we assume without loss of generality that $w_s \in \Omega \cap K$. There exists s such that $(w_s \cdot v)/(w_1 \cdot v)$ is irrational, since otherwise some nonzero real scalar multiple of v belongs to $\mathbb{Q}^\Sigma$.

In consequence $\mathbb{N}(x \cdot v) + \mathbb{N}(w_s \cdot v)$ is dense in $\mathbb{R}$, so we can find sequences of positive integers m_t and m'_t such that

$$m_t(x \cdot v) + m'_t(w_s \cdot v)$$

is a negative monotone increasing sequence of real numbers approaching 0 as t approaches infinity. For all $v' \in V$, the points $(m_t x + m'_t w_s)_{t \geq 1}$ lie on the same side of the hyperplane normal to v, and as a sequence they approach the hyperplane normal to v arbitrarily closely. By another application of Higman's Lemma, there is an increasing function f such that $(m_{f(t)}, m'_{f(t)})$ is an increasing sequence.

Let $z \in (a_i + M_i) \cap H_\ell$. There exists some constant $c \geq 0$ such that for each r, the points $((z + (c + m_{f(t)})x + m'_{f(t)} w_s) - b_r)_{t \geq 1}$ all lie on the same side of each hyperplane as x. It is easily verified that each of these points belongs to H_ℓ. Thus by Lemma 32, $\psi(z + (c + m_{f(t)})x + m'_{f(t)} w_s)$ is constantly 0. Apply the pumping lemma (Lemma 21) again to these inputs to obtain a finite cover. Then there exist $t_1 < t_2$ such that $(m_{t_2} - m_{t_1})x + (m'_{t_2} - m'_{t_1})w_s \in P$, where the monoid coset $(z + m_{t_1} x + m_{t_2} w_s) + P$ belongs to the cover. Pumping $z + (c + m_{t_1})x + m'_{t_1} w_s$ by a sufficiently large multiple of $(m_{t_2} - m_{t_1})x + (m'_{t_2} - m'_{t_1})w_s$ yields an element z' such that $\psi(z') = 0$, but for each r, we have that $z' - a_r$ is on the same side of each hyperplane as w_s, which contradicts Lemma 32. □

13.4 Proof of Theorem 24

We can now prove Theorem 24 by induction on the dimension of G (the cardinality of the largest linearly independent subset of G).

Proof If the dimension of G is zero, then H is a single point and the result holds. If the dimension of G is greater than zero, then by Lemmas 26 and 33, there exist a group K and finite distinguishers $V_\ell \subseteq \mathbb{Z}^\Sigma$ for all cosets H_ℓ of K in H.

For each distinguishing vector $v \in V_\ell$, consider the set of points $x \in H_\ell$ such that it is not that the case that the following numbers are either all negative or all positive: $(x - a_i) \cdot v$ for $i \in I_\ell$ and $(x - b_j) \cdot v$ for $j \in J_\ell$. This set has finite width

in the direction of v. Thus it can be written as the union of finitely many cosets of a group of smaller dimension. The key to the induction is that any point not in the union of these sets over the different x satisfies the hypotheses of one of the variants of Lemma 32.

Define the sets

$$B := \bigcup_{\ell} \{\, x \in H_\ell \mid \exists y \in \widehat{N}_\ell \text{ such that } ((x - b_j) \cdot v)(y \cdot v) > 0 \text{ for all } j \in J_\ell \text{ and } v \in V_\ell \}$$

$$B' := \bigcup_{\ell} \{\, x \in H_\ell \mid \nexists y \in \widehat{M}_\ell \text{ such that } ((x - a_i) \cdot v)(y \cdot v) > 0 \text{ for all } i \in I_\ell \text{ and } v \in V_\ell \}.$$

By Lemma 32 we have $B \subseteq \psi^{-1}(1) \subseteq B'$. It is not difficult to verify that B and B' are semilinear. Moreover, $B' \setminus B$ is a union of finitely many group cosets of dimension less than G. It follows by inductive hypothesis that $\psi^{-1}(1) \cap (B' \setminus B)$ is semilinear, and thus that ψ itself is semilinear. □

14 Characterizations of the one-way models

In this section, we complete the characterizations of the one-way models in Fig. 2 by showing that they are limited to the indicated power.

14.1 Immediate and delayed transmission

We first consider the immediate and delayed transmission models to prove the following converse to Theorem 6.

Theorem 34 *Let ψ be a predicate that is stably computable by an immediate or delayed transmission protocol. Then ψ is in* **SLIN** $\cap$ **coreMOD**.

Thus, for example, the comparison predicate, true if there are more as than bs in the input, is not stably computable in the immediate or delayed transmission model, because it is not in **coreMOD**. Intuitively, the weakness of delayed transmission compared with queued transmission is in the inability of a receiver to refuse messages temporarily; in effect, by delivering a number of copies of some message to a receiver and and forcing it back into the same state, we cause the whole collection of messages to "disappear" from the computation.

Because a delayed transmission protocol is a special case of a queued transmission protocol and an immediate transmission protocol is a special case of a two-way protocol, Corollary 10 shows that ψ is in **SLIN**. To see that ψ is also in **coreMOD** requires some preliminary lemmas. Consider any delayed or immediate transmission protocol that stably computes ψ, and consider any nonempty subset Σ' of Σ.

To unify the treatment of the two kinds of protocols, we assume that in a delayed transmission protocol, the sender simply sends its current state; thus, we need not distinguish between states and messages in our description. Formally, in this case the set of messages consists of a disjoint copy of the set of states, distinguished by the phrases "the state q_i" and "the message q_i." In the case of immediate transmission, an interaction $(p, q) \mapsto (\delta_1(p), \delta_2(p, q))$ is described as the sender sending message p and going to state $\delta_1(p)$, and the receiver receiving message p and going to state $\delta_2(p, q)$.

We let Q' denote the set of states that appear in configurations reachable from inputs that contain only symbols from Σ'. For states $q, q' \in Q'$ of the protocol, we say that q **can reach** q' if there is some finite sequence of messages (from Q') sent and received by an agent in state q that enable it to enter state q'. Formally, q can reach q' if there is a sequence of states $q = q_1, q_2, \ldots, q_k = q'$ such that, for $2 \leq i \leq k$, either $q_i = \delta_s(q_{i-1})$ or $q_i = \delta_r(p, q_{i-1})$ for some message $p \in Q'$.

The following lemma shows that, for any message p, a sufficient number of copies of p can be "absorbed" by an agent in some state q, returning that agent to state q.

Lemma 35 *Fix any message $p \in Q'$. Define $f_p(q) = \delta_r(p, q)$. There exists a state $q \in Q'$ and a positive integer n such that $f_p^{(n)}(q) = q$.*

Proof For any state $r \in Q'$, the sequence of states r, $f_p(r)$, $f_p^{(2)}(r), \ldots$ in Q' must be eventually periodic of some period n. Choosing q to be any state in the periodic part, we see that $f_p^{(n)}(q) = q$. □

If n is a positive integer and $q_1, q_2 \in Q'$ are states, we say that q_1 and q_2 are n-**substitutable** if there exist an input x containing only symbols from Σ' and a configuration b of states, such that the configurations $c = nq_1 + b$ and $d = nq_2 + b$ are reachable from $I(x)$. We note that if q_1 and q_2 are n-substitutable, then they are mn-substitutable for any positive integer m. We define two states $q_1, q_2 \in Q'$ to be **substitutable** if there is some positive integer n such that they are n-substitutable. Substitutability is reflexive and symmetric by definition; we now show it is transitive.

Lemma 36 *Let $q_1, q_2, q_3 \in Q'$ be states and assume that q_1 and q_2 are substitutable and q_2 and q_3 are substitutable. Then q_1 and q_3 are substitutable.*

Proof For some input x containing only symbols from Σ', configuration b of states and integer n, the configurations $c = nq_1 + b$ and $d = nq_2 + b$ are reachable from $I(x)$. Similarly, for some input y containing only symbols from Σ', configuration b' of states and positive integer m, the configurations $c' = mq_2 + b'$ and $d' = mq_3 + b'$ are reachable from $I(y)$.

Consider the input $z = mx + ny$, which contains only input symbols from Σ'. From $I(z)$ there is a computation reaching

$mc + nc'$, which is equal to $mnq_1 + (mnq_2 + mb + nb')$, and also a computation reaching $md + nd'$, which is equal to $mnq_3 + (mnq_2 + mb + nb')$. Thus q_1 and q_3 are mn-substitutable. □

Now we show that reachability implies substitutability.

Lemma 37 *If $q_1, q_2 \in Q'$ are states such that q_1 can reach q_2 then q_1 and q_2 are substitutable.*

Proof Because substitutability is transitive, it suffices to consider the case in which q_2 is reachable in one step from q_1. For every state $q \in Q'$, we define c_q to be any configuration of states containing q that is reachable from an input containing only symbols from Σ'. We consider two cases.

If q_1 reaches q_2 by a send step, then by Lemma 35 we may choose a state $q_3 \in Q'$ and a positive integer n such that $f_{q_1}^{(n)}(q_3) = q_3$. (Intuitively, this means an agent in state q_3 will be back in q_3 after receiving n copies of message q_1.) We choose $b = n(c_{q_1} - q_1) + c_{q_3}$, and let $c = nq_1 + b$, so that $c = nc_{q_1} + c_{q_3}$. Then c is reachable from $I(x)$ for some input x containing only symbols from Σ'.

From c we may have each of n agents in state q_1 transmit a message (and reach state q_2) to a single agent in state q_3, which will again be in state q_3 after receiving the n copies of q_1. Formally, it is easy to see by induction on j that $c \xrightarrow{*} (n-j)q_1 + jq_2 + b - q_3 + f_{q_1}^{(j)}(q_3)$ for $0 \le j \le n$. Taking $j = n$, we have that $d = nq_2 + b$ is reachable from c, and therefore from $I(x)$, so q_1 and q_2 are substitutable.

Suppose q_1 reaches q_2 by a receive step, say of message $q_3 \in Q'$. Let q_4 be the state reached from state q_3 after a send. Then by Lemma 35, there is a state $q_5 \in Q'$ and a positive integer n such that $f_{q_3}^{(n)}(q_5) = q_5$. (This means that after receiving n copies of message q_3 an agent that starts in state q_5 will be back in state q_5.) We choose

$$b = n(c_{q_1} - q_1) + n(c_{q_3} - q_3) + nq_4 + c_{q_5},$$

and let

$$c = nq_1 + b = nc_{q_1} + n(c_{q_3} - q_3) + nq_4 + c_{q_5}.$$

Now $c' = nc_{q_1} + nc_{q_3} + c_{q_5}$ is reachable from $I(x)$ for some input x containing only symbols from Σ'. It is easy to see by induction on j that, for $0 \le j \le n$, $c' \xrightarrow{*} nc_{q_1} + nc_{q_3} - jq_3 + jq_4 + c_{q_5} - q_5 + f_{q_3}^{(j)}(q_5)$ by a sequence of steps in which j agents in state q_3 send a message (and go to state q_4) to a single agent that started in state q_5. Taking $j = n$, we see that $c' \xrightarrow{*} nc_{q_1} + nc_{q_3} - nq_3 + nq_4 + c_{q_5} = c$. Also, $d = nq_2 + b = n(c_{q_1} - q_1) + nq_2 + n(c_{q_3} - q_3) + nq_4 + c_{q_5}$ is reachable from c' by a computation in which we pair up n agents in state q_3 with n agents in state q_1 and in each of the n pairs have the agent in state q_3 send a message to the agent in state q_1, transforming the sender into state q_4 and the receiver into state q_2. Thus, in this case also, q_1 and q_2 are substitutable. □

Thus we have the following pumping lemma for predicates stably computable by immediate and delayed transmission protocols.

Lemma 38 *For every input symbol $\sigma \in \Sigma'$ there exist positive integers k and n such that for every input $y \in \mathrm{Pop}(\Sigma)$ that is k-rich with respect to Σ', y and $y + n\sigma$ are both accepted or both rejected.*

Proof Let k_0 be the constant in Lemma 18 such that k_0-truncates are sufficient to determine membership in the set $\mathcal{S}$ of output stable configurations for the protocol. We choose k_1 to be $|Q'|(k_0 - 1) + 1$.

Let σ be any input symbol from Σ' and let $q = \iota(\sigma)$, the initial state associated with σ. Let $R \subseteq Q'$ be the set of states that q can reach, and consider any $q' \in R$. By Lemma 37, q and q' are $n_{q'}$-substitutable for some positive integer $n_{q'}$. Let n be the least common multiple of the $n_{q'}$ for all $q' \in R$; q and q' are n-substitutable. That is, there exists an input $x_{q'}$ containing only symbols from Σ' and a configuration of states $b_{q'}$ such that the configurations $c_{q'} = nq + b_{q'}$ and $d_{q'} = nq' + b_{q'}$ are reachable from $I(x_{q'})$.

Let x be the sum of all $x_{q'}$ over $q' \in R$. We choose k_2 to be the maximum multiplicity of any input symbol in x. Let $k = \max(k_1, k_2)$ and let y be any input that is k-rich with respect to Σ'. Then $y \ge x$ and y contains only symbols from Σ'.

Consider a computation from $I(y)$ that first takes $I(x)$ to the sum $s = \sum_{q' \in R} c_{q'}$ and then proceeds to an output stable configuration e. Since y is k-rich with respect to Σ', there are at least $k \ge k_1$ agents with input symbol σ. By the pigeonhole principle, some state $q' \in R$ appears with multiplicity at least k_0 in e.

Now from $I(y + n\sigma) = I(y) + nq$ we consider a computation that takes $I(x - x_{q'})$ to $s - c_{q'}$ and $I(x_{q'})$ to $d_{q'}$, giving

$$\begin{aligned} &I(y - x) + nq + (s - (nq + b_{q'})) + nq' + b_{q'} \\ &\quad = I(y - x) + s + nq'. \end{aligned}$$

Now we continue by running $I(y-x)+s$ to e, giving $e+nq'$, which is output stable because $\tau_{k_0}(e) = \tau_{k_0}(e + nq')$. Thus y and $y + n\sigma$ are both accepted or both rejected. □

Now we can conclude the proof of Theorem 34.

Proof For each alphabet symbol $\sigma \in \Sigma'$, by Lemma 38 there exist integers k_σ and n_σ such that for all inputs y that are k_σ-rich with respect to Σ', $y + n_\sigma\sigma$ is accepted if and only if y is accepted. Let k be the maximum of the k_σs over $\sigma \in \Sigma'$. Then the acceptance or rejection of an input y that is k-rich with respect to Σ' depends only on the values of the number of occurrences of σ modulo n_σ, which implies that ψ is k-similar with respect to Σ' to a predicate in in **MOD**. Because Σ' was an arbitrary nonempty subset of Σ, this shows that $\psi \in$ **coreMOD**. □

14.2 Immediate observation

In the immediate observation model, transitions are of the form $(p, q) \mapsto (p, q')$ and there is no multiset of undelivered messages. We have shown in Theorem 7 that for any constant k, an immediate observation protocol can count the number of copies of each input symbol up to k, However, this is also the extent of its power.

Theorem 39 *Let ψ be a predicate that is stably computable by an immediate observation protocol. Then $\psi \in \mathbf{COUNT}_*$.*

Proof Let k be the constant in Lemma 18 such that k-truncates are sufficient to determine membership in the set $\mathcal{S}$ of output stable configurations. Assume that x is an input such that some input symbol σ occurs with multiplicity at least $k' = |Q|(k-1)+1$ in x. Let $c_0, c_1, c_2, \ldots, c_m$ be an execution from the initial configuration $c_0 = I(x)$ to an output stable configuration c_m. Then, $c_i = c_{i-1} - p_i + p_i'$ for some $p_i, p_i' \in Q$. We define $s_i(j)$ inductively to represent the state of the jth agent with input σ in c_i, as follows. Let $s_0(j) = \iota(\sigma)$. If the multiset $\{s_{i-1}(j) : 1 \leq j \leq k'\}$ is contained in c_i, then $s_i = s_{i-1}$. Otherwise, there is some $\hat{j}$ such that $s_{i-1}(\hat{j}) = p_i$, and we define $s_i(j) = \begin{cases} p_i' & \text{if } j = \hat{j} \\ s_{i-1}(j) & \text{otherwise.} \end{cases}$

By the pigeonhole principle, the multiset $\{s_m(j) \mid 1 \leq j \leq k'\}$ contains some state q' with multiplicity at least k. Choose j' such that $s_m(j') = q'$. We introduce a "clone" of this agent into the execution. Formally, we show by induction that $I(x+\sigma) \xrightarrow{*} c_i + s_i(j')$. When $i = 0$, $I(x+\sigma) = c_0 + s_0(j')$. Assume $I(x+\sigma) \xrightarrow{*} c_{i-1} + s_{i-1}(j')$. If $q_i(j') = s_{i-1}(j')$, then the claim is clearly true. Otherwise, $c_i + s_i(j') = c_{i-1} - p_i + 2p_i'$, which is reachable from $c_{i-1} + s_{i-1}(j') = c_{i-1} + p_i$ by having two agents in state p_i change to state p_i' by observing some state in $c_{i-1} - p_i$.

Thus, $I(x+\sigma) \xrightarrow{*} c_m + q'$. Because the multiplicity of q' in c_m is at least k, $\tau_k(c_m) = \tau_k(c_m + q')$. Therefore, since c_m is output stable, so is $c_m + q'$. Moreover, $c_m + q'$ has the same output as c_m, so $\psi(x) = \psi(x+\sigma)$. Because this holds of any input symbol σ of multiplicity at least k' in x, we have $\psi \in \mathbf{COUNT}_{k'} \subset \mathbf{COUNT}_*$. □

We may also consider a variant of the immediate observation model in which an agent may interact with itself, that is, a rule $(p, p) \mapsto (p, q)$ can be applied to a single agent in state p to change it to state q. This is the model **with mirrors**. The two versions of the model are equal in power, as they can simulate each other, using bit-flipping protocols to avoid or permit self-interactions.

Theorem 40 *The same predicates are stably computable by immediate observation protocols in the models with and without mirrors.*

Proof We may assume that n, the population size, is at least 3, since any predicate on 3 or fewer agents is computable in either model; this case can be detected and handled separately in either kind of protocol.

Given an immediate observation protocol that stably computes a predicate ψ in the model without mirrors, we can derive another protocol that stably computes ψ in the model with mirrors. For every old state q we introduce a new state q' and for every old transition $(p, q) \mapsto (p, r)$, where $p \neq q$, we introduce variants of the transition with all combinations of primed and unprimed states p, q, and r. We replace every old transition $(p, p) \mapsto (p, q)$ with four new transitions: $(p, p) \mapsto (p, p')$ and $(p', p') \mapsto (p', p)$, as well as $(p, p') \mapsto (p, q)$ and $(p', p) \mapsto (p', q)$. In the mirrored model, this allows an agent to flip back and forth between p and p' by itself, but to leave these two states, it must interact with someone other than itself (witnessed by having different "primation").

For the other direction, given an immediate observation protocol in the model with mirrors that stably computes a predicate ψ, we can again introduce primed and unprimed versions of each state. For each ordered pair of states p and q, we have rules $(p, q) \mapsto (p, q')$ and $(p, q') \mapsto (p, q)$, which allow a state to flip between primed and unprimed as long as there is at least one unprimed state. Because the input states are unprimed, these rules cannot eliminate the last unprimed state. In addition, for every rule $(p, q) \mapsto (p, r)$ with $p \neq q$ in the protocol with mirrors, there is a rule $(p', q) \mapsto (p', r)$ in the protocol without mirrors; this allows steps involving two agents in different states to be simulated by priming the initiator, unpriming the responder, and taking the step. For every rule $(p, p) \mapsto (p, r)$ in the protocol with mirrors, there are rules $(q', p') \mapsto (q', r')$ for every original state q. If a step involves two agents in state p, it can be simulated by priming both agents and taking the step. If a step involves one agent in state p, it can be simulated by priming that agent (to get p') and any other agent (to get q') and taking the step. Note that this simulation can be carried out if there are at least three agents in the population because there is at least one unprimed agent in every configuration reachable from an input configuration. □

14.3 Delayed observation

Here, we prove the converse of Theorem 8, showing that the delayed observation model is the weakest of the one-way models we defined.

Theorem 41 *Suppose ψ is stably computed by a delayed observation protocol. Then ψ is in* $\mathbf{COUNT}_1$.

Proof We show that for any input x, if $\sigma \in x$ then $\psi(x+\sigma) = \psi(x)$, which implies that ψ is completely determined by the

presence or absence of each input symbol and hence is in $\mathbf{COUNT}_1$.

Consider the finite graph whose nodes are configurations reachable from $I(x)$ that contain no messages in transit, with a directed edge from c to c' if $c \xrightarrow{*} c'$. A **final** strongly connected component of this graph is one from which no other strongly connected component of the graph is reachable. From $I(x)$ we can reach a configuration in a final strongly connected component $\mathcal{F}$ of this graph. Let $\hat{\mathcal{F}}$ denote all the configurations d, including those with undelivered messages, such that $c \xrightarrow{*} d$ for some $c \in \mathcal{F}$. For any configurations d and d' in $\hat{\mathcal{F}}$, $d \xrightarrow{*} d'$ by first delivering all messages in d. This implies that all configurations in $\hat{\mathcal{F}}$ are output stable.

The set T of states that occur in configurations in $\hat{\mathcal{F}}$ is closed, that is, if $p, q \in T$ and $(p, q) \mapsto (p, q')$, then $q' \in T$. To see this, assume not. Then, take a configuration c in $\hat{\mathcal{F}}$ that contains p and let an agent in state p send a message, putting p into messages in transit. Now mimic a computation from d to a configuration d' in $\hat{\mathcal{F}}$ containing q, leaving the message p undelivered. Then deliver p to an agent in state q, arriving at a configuration in $\hat{\mathcal{F}}$ containing q', a contradiction.

Now consider any σ in x. Consider an execution that begins in $I(x)$ and ends in an output stable configuration c in $\mathcal{F}$. There must be a sequence of states $q_0, q_1, \ldots, q_m$ where $q_0 = \iota(\sigma)$, q_m appears in c, and $(p_i, q_{i-1}) \mapsto (p_i, q_i)$ for some state p_i that appeared in some configuration during the execution. Starting from the configuration $I(x + \sigma) = I(x) + q_0$, we can reach a configuration $c + q_0$ that also has one additional copy of each message $p_1, \ldots, p_m$ left in transit. By delivering all of the m messages to the agent in state q_0 in order, we reach the configuration $c + q_m$. Because T is closed and the states of $c + q_m$ are all in T, $c + q_m$ is output stable and therefore $\psi(x + \sigma) = \psi(x)$. □

If we assume that no agent ever receives its own message, then the power of the model increases only slightly: it becomes possible for an agent in a unique state to observe that it never encounters a twin, and the stably computable predicates are $\mathbf{COUNT}_2$ in this case. The proof is a straightforward extension of the proof of Theorem 41.

15 Local fairness is weak even with unbounded states

In this section, we consider a strongly anonymous message-passing model with the following local fairness condition: if some agent sends a particular message m infinitely often, then each agent receives message m infinitely often. This model turns out to be surprisingly weak. Even if the states of processes and the lengths of messages may grow without bound, protocols in this model cannot distinguish two multisets of inputs if the same set of values appears in each. Since this model subsumes the one-way finite-state population protocol models, this result supports the choice of the stronger global fairness condition assumed in the rest of the paper.

Theorem 42 *Let Σ denote the (finite or countably infinite) set of possible input values. A predicate ψ on finite nonempty multisets of elements from Σ is stably computable in the asynchronous message-passing model with the weak fairness condition if and only if ψ is completely determined by the set of input values present in the initial configuration.*

Proof We describe a protocol in which each agent eventually determines the set of all the inputs that occur in the initial configuration, and therefore the correct value of ψ. The state of each agent is the set consisting of its initial input value together with every value that has occurred in a message it has received. Whenever an agent runs, it sends its current state. Whenever an agent receives a message, it updates its state to be the union of its previous state and the set of values in the message.

Clearly every message sent is a subset of the set of input values in the initial configuration, so there are only finitely many possible messages in each computation. Every message sent by an agent with input value σ contains the element σ, and it sends infinitely many messages, so eventually every agent receives a message containing σ. Thus, the state of every agent eventually consists of the set of values in the input configuration, and each outputs the correct value of ψ.

For the converse, assume that we have a protocol that stably computes a predicate ψ, and let x and x' be any two nonempty multisets of values from Σ such that the same set of values appears in each. Let $n = |x|$ and $n' = |x'|$. Let c_0 and c'_0 be initial configurations corresponding to inputs x and x', respectively. We construct two executions α and α' starting from c_0 and c'_0. Let $m_1, m_2, \ldots$ be an arbitrary sequence of messages where every possible message appears infinitely often. We construct the executions α and α' in phases, where phase i will ensure that message m_i gets delivered to everyone if that message has been sent enough times. Let c_i and c'_i be the configurations of α and α' at the end of phase i.

Our goal is to prove the following claim: for all $i \geq 0$ and for all $\sigma \in \Sigma$, the state in c_i of each agent with original input σ is the same as the state in c'_i of each agent with original input σ. Assume that we have constructed the first $i - 1$ phases of the two executions so that the claim is satisfied. Suppose we run all processes in lock step from c_{i-1} and c'_{i-1} without delivering any messages. There are two cases.

Case (i): Eventually, after r_i rounds, the run from c_{i-1} will have at least n copies of m_i in transit and, after r'_i rounds, the run from c'_{i-1} will have at least n' copies of m_i in transit. Then, the ith phase of α and α' is constructed by running each agent for $\max(r_i, r'_i)$ rounds without delivering any messages, and then delivering one copy of m_i to every agent. This ensures the claim will be true for c_i and c'_i.

Case (ii): Otherwise, we allow every agent to take one step without delivering any messages. (This clearly satisfies the claim for c_i and c'_i.)

It remains to show that both α and α' satisfy the weak fairness condition, and then it will follow from the claim that $\psi(x) = \psi(x')$. First, notice that every agent takes infinitely many steps in α and α'. If some agent v sends a message m infinitely many times in α or α', it will also be sent infinitely many times by an agent with the same input value in the other execution (since an agent with a particular input experiences the same sequence of events in both executions). Suppose m is never delivered after phase i to some agent w in one of the two executions. Eventually, there will be n copies of m in transit in C_j for some $j > i$ and n' copies of m in transit in $C'_{j'}$ for some $j' > i$. Consider the first occurrence of m in the sequence $m_1, m_2, \ldots$ that comes after m_j and $m_{j'}$. During the corresponding phase, m will be delivered to every agent, including w, a contradiction. Thus, α and α' satisfy the weak fairness condition. □

16 Conclusions and discussion

We have shown that the predicates stably computable by population protocols in the family of all-pairs communication graphs are semilinear, answering the main open question in [3,4]. This result also shows that the model of population protocols with stabilizing inputs, introduced in [1], is equal in computational power to the standard model of population protocols in the family of all-pairs communication graphs, resolving another open question.

Our semilinearity proof is given for an abstract model that includes not only population protocols, but generalizations permitting rules that replace one finite multiset of elements of a configuration with another finite multiset of elements. Thus, even a generalization of population protocols in which rules may add or delete agents or involve interactions between more than two agents will stably compute only semilinear predicates, answering a question in [2].

We have introduced several models of one-way communication in population protocols: queued transmission, immediate and delayed transmission and immediate and delayed observation, and exactly characterized the classes of predicates stably computable in these models in the family of all-pairs communication networks by natural subclasses of the semilinear predicates. These one-way models are more closely related to asynchronous message-passing models, and illuminate the effects of capabilities such as the ability to refuse to accept incoming messages temporarily.

Another natural variant of the population protocol model considers "reversing moves" and requires that a protocol stably compute a predicate despite the possible presence in an execution of a finite number of reversing moves, that is, steps in which a rule $(p, q) \mapsto (p', q')$ is used in reverse: $(p', q') \mapsto (p, q)$. It is not difficult to show that the basic protocols for modulo and threshold predicates are resistant to such reversing moves, and therefore that all the semilinear predicates can be computed by protocols resistant to reversing moves. Thus, requiring such resistance does not reduce the computational power of population protocols in the family of all-pairs communication graphs.

Despite promising results for population protocols in restricted communication graphs [1], much more remains to be understood about their computational power. In particular, the computational power of one-way protocols in restricted communication graphs has not been studied.

Acknowledgments This research was carried out while the third author was an undergraduate at the University of Rochester. The authors would like to thank the reviewers of the extended abstracts on which this paper is based [6,7] and also the referees of the current paper for their thoughtful comments and suggestions.

References

1. Angluin, D., Aspnes, J., Chan, M., Fischer, M.J., Jiang, H., Peralta, R.: Stably computable properties of network graphs. In: Prasanna, V.K., Iyengar, S., Spirakis, P., Welsh, M. (eds) Distributed Computing in Sensor Systems: First IEEE International Conference, DCOSS 2005, Marina del Rey, CA, USA, June/July, 2005, Proceedings, volume 3560 of Lecture Notes in Computer Science, pp. 63–74. Springer, Berlin (2005)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Urn automata. Technical Report YALEU/DCS/TR-1280, Yale University Department of Computer Science (2003)
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: PODC '04: Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, pp. 290–299. ACM, New York (2004)
4. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006)
5. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. In: Distributed Computing: 20th International Symposium, DISC 2006: Stockholm, Sweden, September 2006: Proceedings, pp. 61–75 (2006)
6. Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: Proceedings of the 25th ACM Symposium on Principles of Distributed Computing, pp. 292–299 (2006)
7. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: On the power of anonymous one-way communication. In: Principles of Distributed Systems; 9th International Conference, OPODIS 2005; Pisa, Italy; December 2005; Revised Selected Papers, volume 3974 of Lecture Notes in Computer Science, pp. 396–411 (2005)
8. Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing population protocols. In: Principles of Distributed Systems; 9th International Conference, OPODIS 2005; Pisa, Italy; December 2005; Revised Selected Papers, volume 3974 of Lecture Notes in Computer Science. pp. 103–117 (2005)
9. Angluin, D., Fischer, M.J., Jiang, H.: Stabilizing consensus in mobile networks. In: Proceedings of Distributed Computing in Sensor Systems: Second IEEE International Conference, volume 4026 of Lecture Notes in Computer Science, pp. 37–50 (2006)

10. Attiya, H., Gorbach, A., Moran, S.: Computing in totally anonymous asynchronous shared memory systems. Inform. Comput. **173**(2), 162–183 (2002)
11. Angluin, D.: Local and global properties in networks of processors. In: Proceedings of the 12th ACM Symposium on Theory of Computing, pp. 82–93 (1980)
12. Aspnes, J., Shah, G., Shah, J.: Wait-free consensus with infinite arrivals. In: Proceedings of the 34th ACM Symposium on Theory of Computing, pp. 524–533 (2002)
13. Bailey, N.T.J.: The Mathematical Theory of Infectious Diseases, Second Edition. Charles Griffin & Co., London (1975)
14. Buhrman, H.: Alessandro Panconesi, Riccardo Silvestri, and Paul Vitanyi. On the importance of having an identity or, is consensus really universal?. Distrib. Comput. **18**(3), 167–176 (2006)
15. Boldi, P., Vigna, S.: Computing anonymously with arbitrary knowledge. In: Proceedings of the 18th ACM Symposium on Principles of Distributed Computing, pp. 173–179 (1999)
16. Boldi, P., Vigna, S.: An effective characterization of computability in anonymous networks. In: Distributed Computing, 15th International Conference, pp. 33–47 (2001)
17. Diamadi, Z., Fischer, M.J.: A simple game for the study of trust in distributed systems. Wuhan Univ. J. Natural Sci. **6**(1–2), 72–82 (2001). Also appears as Yale Technical Report TR–1207, January 2001
18. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: When birds die: Making population protocols fault-tolerant. In: Proceedings of the Second IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '06), pp. 51–66 (2006)
19. Dickson, L.E.: Finiteness of the odd perfect and primitive abundant numbers with *n* distinct prime factors. Am. J. Math. **35**(4), 413–422 (1913)
20. Daley, D.J., Kendall, D.G.: Stochastic rumours. J. Inst. Math. Appl. **1**, 42–55 (1965)
21. Eğecioğlu, Ö., Singh, A.K.: Naming symmetric processes using shared variables. Distrib. Comput. **8**(1), 19–38 (1994)
22. Fischer, M.J., Jiang, H.: Self-stabilizing leader election in networks of finite-state anonymous agents. In: Tenth International Conference on Principles of Distributed Systems, volume 4305 of Lecture Notes in Computer Science, pp. 395–409 (2006)
23. Fich, F., Ruppert, E.: Hundreds of impossibility results for distributed computing. Distrib. Comput. **16**(2–3), 121–163 (2003)
24. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. A **104**, 1876–1880 (2000)
25. Gillespie, D.T.: A rigorous derivation of the chemical master equation. Physica A **188**, 404–425 (1992)
26. Guerraoui, R., Ruppert, E.: Anonymous and fault-tolerant shared-memory computing. Distrib. Comput. (2007, in press)
27. Higman, G.: Ordering by divisibility in abstract algebras. Proc. Lond. Math. Soc. **3**(2), 326–336 (1952)
28. Hopcroft, J., Pansiot, J.: On the reachability problem for 5-dimensional vector addition systems. Theoret. Comput. Sci. **8**(2), 135–159 (1978)
29. Ibarra, O.H., Dang, Z., Egecioglu, O.: Catalytic P systems, semilinear sets, and vector addition systems. Theor. Comput. Sci. **312**(2–3), 379–399 (2004)
30. Jiang, H.: Distributed Systems of Simple Interacting Agents. PhD thesis, Yale University (2007)
31. Jayanti, P., Toueg, S.: Wakeup under read/write atomicity. In: Distributed Algorithms, 4th International Workshop, volume 486 of LNCS, pp. 277–288 (1990)
32. Kutten, S., Ostrovsky, R., Patt-Shamir, B.: The Las-Vegas processor identity problem (How and when to be unique). J. Algorithms **37**(2), 468–494 (2000)
33. Lang, S.: Algebra (revised third edition). Springer, Berlin (2002)
34. Lay, S.R.: Convex Sets and their Applications. Krieger Publishing Company, (1992)
35. Lipton, R.J., Park, A.: The processor identity problem. Inform. Process. Lett. **36**(2), 91–94 (1990)
36. Parikh, R.J.: On context-free languages. J. ACM **13**(4), 570–581 (1966)
37. Panconesi, A., Papatriantafilou, M., Tsigas, P., Vitányi, P.: Randomized naming using wait-free shared variables. Distrib. Comput. **11**(3), 113–124 (1998)
38. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes-Rendus du I Congrès de Mathématiciens des Pays Slaves, pp. 92–101, Warszawa (1929)
39. Sakamoto, N.: Comparison of initial conditions for distributed algorithms on anonymous networks. In: Proc. 18th ACM Symposium on Principles of Distributed Computing, pp. 173–179 (1999)
40. Teng, S.-H.: Space efficient processor identity protocol. Inform. Process. Lett. **34**(3), 147–154 (1990)