Mediated Population Protocols*

Ioannis Chatzigiannakis[1,2], Othon Michail[2], and Paul G. Spirakis[1,2]

[1] Research Academic Computer Technology Institute (CTI), P.O. Box 1382, N. Kazantzaki Str., 26500 Patras, Greece
[2] Computer Engineering and Informatics Department (CEID), University of Patras, 26500, Patras, Greece.
Email: {ichatz, spirakis}@cti.gr, michailo@ceid.upatras.gr

Abstract. We extend here the Population Protocol model of Angluin et al. [2] in order to model more powerful networks of very small resource-limited artefacts (agents) that are possibly mobile. The main feature of our extended model is to allow edges of the communication graph, G, to have *states* that belong to a constant size set. We also allow edges to have *readable only costs*, whose values also belong to a constant size set. Our protocol specifications are still independent of the population size and do not use agent ids, i.e. they preserve *uniformity* and *anonymity*. Our Mediated Population Protocols (MPP) can stably compute graph properties of the communication graph. We show this for the properties of maximal matchings (in undirected communication graphs), also for finding the transitive closure of directed graphs and for finding all edges of small cost. We demonstrate that our mediated protocols are stronger than the classical population protocols, by presenting a MPP for a non-semilinear predicate. To show this fact, we state and prove a general theorem about the composition of two stably computing mediated population protocols. We also show that all predicates stably computable in our model are (non-uniformly) in the class $NSPACE(|E(G)|)$.

1 Introduction

1.1 Population Protocols

In a seminal work, Angluin et al. in [2] proposed the *population protocol model* in order to represent sensor networks consisting of extremely limited agents that may move and interact in pairs. Due to their severe limitations, the agents can be represented as a population V of $|V| = n$ finite state machines. A common assumption is that a global start signal initiates computation by informing the agents to sense their environment in order to receive a piece of the input. Two agents communicate when they come sufficiently close to each other. The movement is carried out by an *adversary scheduler*. A strong global *fairness condition* is imposed on the adversary to ensure the protocol makes progress.

* This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

Formally, a population protocol $\mathcal{A}$ consists of finite *input and output alphabets* X and Y, a finite set of *states* Q, an *input function* $I : X \to Q$ mapping inputs to states, an *output function* $O : Q \to Y$ mapping states to outputs, and a *transition function* $\delta : Q \times Q \to Q \times Q$. The critical assumption that diversifies the population protocol model from traditional distributed systems is that the protocol specifications are independent of the population size (that is, need $\mathcal{O}(1)$ total memory capacity in each agent), which is known as the *uniformity* property of population protocols. Moreover, population protocols are *anonymous* since there is no room in the state of an agent to store a unique identifier.

A network communication graph $G = (V, E)$ (see [1]) describes the permissible ordered pairwise interactions. G is assumed to be a directed graph with no multi-edges or self-loops (simple) and its n nodes are numbered 1 through n. An edge $(u, v) \in E$ indicates the possibility of a communication between u and v, in which u is the *initiator* and v the *responder*. The *basic* population protocol model assumes the all-pairs family of directed communication graphs, denoted $\mathcal{G}^d_{All}$, which simply contains for each n the complete directed graph on n vertices.

A *population configuration* is a mapping $C : V \to Q$ providing a snapshot of the population states. $C \to C'$ denotes that configuration C can go to C' through a single interaction, i.e. there is an edge $e = (u, v) \in E$ such that $\delta(C(u), C(v)) = (C'(u), C'(v))$ and $C'(\omega) = C(\omega)$ for all $\omega \in V - \{u, v\}$. In this case we say that C goes to C' via encounter $e = (u, v)$, and to emphasize that, we write $C \xrightarrow{e} C'$. C' is said to be *reachable* from C, denoted $C \xrightarrow{*} C'$, if C can be converted to C' in one or more steps.

An *execution* is a finite or infinite sequence of configurations $C_0, C_1, C_2, \ldots$, where C_0 is an initial configuration and $C_i \to C_{i+1}$, for all $i \geq 0$. At any point during the execution of a population protocol, each agent's state determines its output at that time (the output of u under configuration C is $O(C(u))$, for each $u \in V$). An infinite execution is fair if for every possible transition $C \to C'$, if C occurs infinitely often in the execution, then C' occurs infinitely often. A *computation* is an infinite fair execution.

Generally, population protocols do not halt. Instead, halting is replaced by an interesting property called *stability*. Stability was defined in [2] to be a situation where computation reaches a configuration C, after which, no matter how the computation proceeds, no agent will be able to change its output value. Such a configuration C is called an *output-stable* configuration. A protocol $\mathcal{A}$ (stably) *computes* a function f that maps multisets of elements of X to Y if, for every such multiset x and every computation that starts from the initial configuration corresponding to x, the output value of every agent eventually stabilizes to $f(x)$.

For the basic population protocol model there exists an exact characterization of the computable predicates: they are precisely the *semilinear predicates* or equivalently the predicates definable by first-order logical formulas in *Presburger arithmetic* [2, 4, 5]. In the *stabilizing inputs* variant of the basic model [1], convergence to a stable common output value is only required after the inputs stop changing (here, each agent has an input field that can change over time).

The results of [4] show that again the semilinear predicates are all that can be computed by this model.

1.2 Other Previous Work

The motivation given for the population protocol model was the study of sensor networks in which passive agents were carried along by other entities. Much work has been devoted to the, now, well known fact that the set of computable predicates of the basic (complete interaction graph) population protocol model and most of its variants is exactly equal or closely related to the set of *semilinear predicates*. Also, in [2], the *probabilistic population protocol* model was proposed, in which the scheduler selects randomly and uniformly the next pair to interact. More recent work has concentrated on performance, supported by this random scheduling assumption. [6] and [9] considered a huge population hypothesis (population going to infinity), and studied the dynamics, stability and computational power of probabilistic population protocols by exploiting the tools of continuous nonlinear dynamics. In [6] it was also proven that there is a strong relation between classical finite population protocols and models given by ordinary differential equations. Distributed computing with advice (in the spirit of our model) was considered in [10]. Moreover, several extensions of the basic model have been proposed to more accurately reflect the requirements of practical systems. In [1] they studied what properties of restricted communication graphs are stably computable, gave protocols for some of them, and proposed the model extension with *stabilizing inputs*. Finally, some works incorporated agent failures and gave to the agents slightly increased memory capacity. For an excellent introduction to the subject see [5].

1.3 Our Approach

We extend the population protocol model with communication links satisfying the following properties: Each $e \in E$ is equipped with a buffer of $\mathcal{O}(1)$ total storage capacity. Before an interaction $e = (u, v)$, the interacting pair reads the contents of the corresponding buffer, that is, the state of e, to provide it to the transition function. After an interaction $e = (u, v)$, the interacting pair updates the contents of the corresponding buffer, according to the new state of e, returned by the transition function. Since the memory of the edges is constant and independent of the population size, the protocol specifications are independent of n, i.e. *both uniformity and anonymity are preserved*.

By letting the edges keeping states, the first gain is that we manage to get more computational power in comparison to the basic population protocol model. The additional computational power is limited (as expected) due to the fact that this additional set of states is again of constant cardinality (and usually we try to keep it low, i.e. one or two bits suffice). We prove that the derived model is able to compute at least one non-semilinear predicate.

The new model also gives rise to many other novel computational possibilities. By defining a natural relaxation of stability, that we call *r-stability*, we

obtain protocols that are able to locate subgraphs of the communication graph. By associating each edge with a read-only cost, under certain assumptions, we are able to devise protocols that solve optimization problems concerning the communication graph. This cost can be assumed to be stored (together with the state) in the constant size buffer of the edge.

2 The New Model

2.1 Mediated Population Protocols

A *mediated population protocol* $\mathcal{A}$ consists of finite *input and output alphabets* X and Y, a finite set of *agent states* Q, an *agent input function* $I : X \to Q$ mapping inputs to agent states, an *agent output function* $O : Q \to Y$ mapping agent states to outputs, a finite set of *edge states* S, an *edge input function* $\iota : X \to S$ mapping inputs to edge states, an *edge output function* $\omega : S \to Y$ mapping edge states to outputs, an *output instruction* r, a finite totally ordered *cost set* K, a *cost function* $c : E \to K$ assigning a cost to each edge of the communication graph and a *transition function* $\delta : Q \times Q \times K \times S \to Q \times Q \times K \times S$ (from now on we will always assume that the cost remains the same after applying δ and so we will omit specifying an output cost). If $\delta(q_i, q_j, x, s) = (q_i', q_j', s')$ (which, according to our assumption, is equivalent to $\delta(q_i, q_j, x, s) = (q_i', q_j', x, s')$), we call $(q_i, q_j, x, s) \to (q_i', q_j', s')$ a *transition*, and we define $\delta_1(q_i, q_j, x, s) = q_i'$, $\delta_2(q_i, q_j, x, s) = q_j'$ and $\delta_3(q_i, q_j, x, s) = s'$. We will call δ_1 the *initiator's acquisition*, δ_2 the *responder's acquisition*, and δ_3 the *edge acquisition* after the corresponding interaction.

In most cases we will assume that $K \subset \mathbb{Z}^+$ and that $c_{max} = \max_{w \in K} \{w\} = \mathcal{O}(1)$. Generally, if $c_{max} = \max_{w \in K} \{|w|\} = \mathcal{O}(1)$ then any agent is capable of storing at most k cumulative costs (at most the value kc_{max}), for some $k = \mathcal{O}(1)$, and we say that the cost function is *useful* (note that a cost range that depends on the population size could make the agents incapable for even a single cost storage and any kind of optimization would be impossible).

A mediated population protocol runs in a communication graph $G = (V, E)$, where V is a population of n agents, and E is an irreflexive binary relation on V, of cardinality denoted by m. In the case of an undirected graph we only require that E is also symmetric, and that for all $(u, v), (v, u) \in E$, (u, v) and (v, u) share the same buffer (in the undirected case we also assume in this work that if $\delta(q_i, q_j, x, s) = (q_i', q_j', s')$, then $\delta(q_j, q_i, x, s) = (q_j', q_i', s')$, for all $(q_i, q_j, x, s) \in Q \times Q \times K \times S$). In both cases (directed and undirected), a $(u, v) \in E$ means that interaction (u, v) is permitted in which u is the *initiator* and v the *responder*.

A *network configuration* is a mapping $C : V \cup E \to Q \cup S$ specifying the agent state of each agent in the population and the edge state of each edge in the communication graph. If we restrict our attention on the states of the agents only, we can use the mapping $C_V : V \to Q$ which is called the *population configuration* and similarly $C_E : E \to S$ is the *edge configuration*. Let C and C' be network configurations, and let u, v be distinct agents. We say that C goes to C' via encounter $e = (u, v)$, denoted $C \xrightarrow{e} C'$, if $C'(u) = \delta_1(C(u), C(v), x, C(e))$,

$C'(v) = \delta_2(C(u), C(v), x, C(e))$, $C'(e) = \delta_3(C(u), C(v), x, C(e))$, and $C'(z) = C(z)$, for all $z \in (V - \{u, v\}) \cup (E - e)$. We say that C can go to C' in one step, if $C \xrightarrow{e} C'$ for some encounter $e \in E$. We write $C \xrightarrow{*} C'$ if there is a sequence of configurations $C = C_0, C_1, \ldots, C_t = C'$, such that $C_i \rightarrow C_{i+1}$ for all i, $0 \leq i < t$, in which case we say that C' is *reachable* from C. The definitions of *execution* and *computation* are the same as in the population protocol model but concern network configurations. Note that the mediated population protocol *code* is of *constant size* and, thus, can be stored in each agent (device) of the population.

2.2 Output Interpretation

Following the definition of stability proposed in [2], we say that a network configuration C is *agent output-stable*, if $O(C'(v)) = O(C(v))$ for all C' where $C \xrightarrow{*} C'$ and for all $v \in V$ (we can also write $O(C'_V) = O(C_V)$ if we extend O to a mapping from population configurations to output assignments like in [2]). Moreover, a configuration C is said to be *edge output-stable*, if $\omega(C'(e)) = \omega(C(e))$ for all C' where $C \xrightarrow{*} C'$ and for all $e \in E$ (i.e. if $\omega(C'_E) = \omega(C_E)$) and *globally output-stable*, if it is both edge output-stable and agent output-stable.

The instruction r that we have included in the model definition is simply an instruction that tells the output viewer how to interpret the output of a protocol. For example, if a protocol is supposed to compute a predicate by reaching an agent output-stable configuration where all agents agree on the correct output value, then instruction r would be: "*Get any $u \in V$ and view its output*". In this case, an agent output-stable configuration is said to be an r-stable configuration.

A configuration C is *r-stable* if one of the following holds: If the problem concerns a subgraph to be found, then C should fix a subgraph that will not change in any C' reachable from C. If the problem concerns a function to be computed by the agents, then an r-stable configuration drops down to an agent output-stable configuration.

We will say that a protocol $\mathcal{A}$ *stably solves* a problem Π, if for every instance I of Π and every computation of $\mathcal{A}$ on I, the network reaches an r-stable configuration C that gives the correct solution for I if interpreted according to the output instruction r. If instead of a problem Π we have a function f to be computed, we will say that $\mathcal{A}$ *stably computes* f. In the special case where Π is an optimization problem, a protocol that stably solves Π will be called an *optimizing population protocol* for problem Π.

3 Some Graph Protocols

3.1 Maximal Matching

We first give a mediated population protocol $MaximalMatching$ that stably solves the following problem:

Problem 1. (*Maximal matching*) Given an undirected communication graph $G = (V, E)$, find a maximal matching, i.e., a set $E' \subseteq E$ such that no two members of

E' share a common end point in V and, moreover, there is no $e \in E - E'$ such that e shares no common end point with every member of E'.

MaximalMatching

- $X = \{0\}$, $Y = \{0, 1\}$,
- $Q = \{q_0, q_1\}$, $S = \{0, 1\}$,
- $I(0) = q_0$,
- $\iota(0) = 0$, $\omega(0) = 0$, $\omega(1) = 1$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ: $(q_0, q_0, 0) \to (q_1, q_1, 1)$

Note that we have omitted specifying costs, since there is no need for them here, and δ is of the simplified form $\delta : Q \times Q \times S \to Q \times Q \times S$. Moreover, any other possibly interacting pair not appearing in δ, e.g. $(q_0, q_1, 0)$ and $(q_1, q_1, 1)$, is assumed throughout this work to be included as an identity rule, that is, a rule that leaves all interacting components unaffected (e.g. $(q_0, q_1, 0) \to (q_0, q_1, 0)$ and $(q_1, q_1, 1) \to (q_1, q_1, 1)$). We, also, don't specify an agent output function because the protocol's correctness concerns only the edge output function.

Theorem 1. *MaximalMatching stably solves the maximal matching problem.*

Proof. Let M be the set of edges in state 1. M is theoretically updated after each interaction, i.e., at any point, any edge in state 1 belongs to M. For an edge $e = (u, v)$ to become a member of M, both its endpoints during the interaction must be in state q_0. If this holds, the edge gets in M and u, v go to state q_1 to indicate that they both have an edge incident to them that belongs to M. From now on, no edges adjacent to e can get in M, simply because their end point on which they coincide with e is in state q_1. This, together with the fact that two interactions happening in parallel cannot concern adjacent edges, proves that M is always a matching. Moreover, an edge not conflicting with M will eventually get in M (if no conflict arises in the meanwhile), since it will be in state 0 and both its end points will be in q_0. The latter proves that M is maximal. □

3.2 Transitive Closure with the Help of a Leader

Assume that $G = (V, E)$ is a graph from the $\mathcal{G}^d_{All}$ family, that is, the all-pairs family of directed communication graphs, and that a protocol has computed a subgraph of G, $G' = (V', E')$, by letting the selected edges (edges in E') be in state 1, while all the remaining edges (i.e. all $e \in E - E'$) are in state 0. Note that V' simply contains all nodes that are incident to at least one $e \in E'$. We want to solve the following problem:

Problem 2. (**Transitive Closure**) Given a communication graph $G = (V, E)$ in $\mathcal{G}^d_{All}$ with a subgraph $G' = (V', E')$ precomputed in the above manner, find the transitive closure of G', that is, find a new edge set E^* that will contain a directed edge (u, v) joining any nodes u, v for which there is a non-null path from u to v in G' (note that always $E' \subseteq E^*$).

We assume a *controlled input assignment* $W : E \to X$ that allows us to give input 1 to any edge belonging to E' and input 0 to any other edge. Moreover, we assume that initially all agents are in state q_0, except for an elected leader (by any leader election protocol) that is in state l. The assumption of a leader and the remark that this helps the protocols was first used in [2] and extensively studied in [3]. We devise a protocol, $TranClos$, with the following specification:

TranClos

- $X = Y = \{0, 1\}$,
- $Q = \{l, q_0, q_1, q_1', q_2, q_2', q_3\}$, $S = \{0, 1\}$,
- *controlled input assignment*: "$W(e') = 1$, *for all* $e' \in E'$, *and* $W(e) = 0$, *for all* $e \in E - E'$",
- $\iota(x) = x$, for all $x \in X$, $\omega(s) = s$, for all $s \in S$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ:

$$\begin{array}{ll} (l, q_0, 0) \to (q_0, l, 0) & (q_2, q_0, 1) \to (q_2', q_3, 1) \\ (l, q_0, 1) \to (q_1, q_2, 1) & (q_1, q_3, x) \to (q_1', q_0, 1), \text{ for } x \in \{0, 1\} \\ (q_1, q_2, 1) \to (q_0, l, 1) & (q_1', q_2', 1) \to (q_0, l, 1) \end{array}$$

Theorem 2. *Protocol* $TranClos$ *stably solves the transitive closure problem.*

For the proof see [8].

3.3 Edges of Minimum Cost

Let us illustrate the incorporation of edge costs in the case of optimization problems, by a simple optimizing population protocol for the following problem:

Problem 3. (*Edges of minimum cost*) Given an undirected connected communication graph $G = (V, E)$ and a useful cost function $c : E \to K$ on the set of edges, where $K \subset \mathbb{Z}^+$, design a protocol that finds the minimum cost edges of E.

MinEdges

- $X = Y = \{0, 1\}$,
- $Q = K \cup \{q_0\}$, $S = \{0, 1\}$,
- $I(x) = q_0$, for all $x \in X$,
- $\iota(x) = 0$, for all $x \in X$, $\omega(s) = s$, for all $s \in S$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ:

$$\begin{array}{rl} (q_0, q_0, c, d) \to & (c, c, 1) \\ (c_i, c_j, c, d) \to & (c, c, 1), \text{ if } c \leq \min\{c_i, c_j\} \\ \to & (\min\{c_i, c_j\}, \min\{c_i, c_j\}, 0), \text{ if } c > \min\{c_i, c_j\} \\ (c_i, q_0, c, d) \to & (c, c, 1), \text{ if } c \leq c_i \\ \to & (c_i, c_i, 0), \text{ if } c > c_i \end{array}$$

Theorem 3. *MinEdges is an optimizing population protocol for Problem 3.*

Proof. We need to show that the system reaches an r-stable configuration C, where if E_{out} is the subset of E specified by instruction r, we have $e \in E_{out}$ if and only if $c(e) = c_{opt}$, where $c_{opt} = \min_{e \in E}\{c(e)\}$.

All rules of δ together with the fairness assumption ensure that every agent will eventually get c_{opt} (a less cost encountered always replaces the current cost of an agent). At that point the system will be in an agent output-stable configuration, since there is no cost less than c_{opt} in order to replace it. From now on, after every interaction (u, v), where $e = \{u, v\}$, $E_{out} \leftarrow E_{out} \cup \{e\}$, if no interacting agent's cost is less than $c(e)$ (that is, if $c(e) = c_{opt}$) and $E_{out} \leftarrow E_{out} - \{e\}$, otherwise.

It follows that, eventually, the system will enter a configuration C where $e \in E_{out}$ will imply that $c(e) = c_{opt}$ and $e \notin E_{out}$ that $c(e) > c_{opt}$. At that time, no edge will be able to enter or leave E_{out}, and since E_{out} is the set specified by r, C will be an r-stable configuration as needed. These together imply that *MinEdges* is indeed an optimizing population protocol for Problem 3. □

4 Computability

4.1 All-pairs Directed Communication Graphs

We now investigate some aspects of the computational power of the mediated population protocol (MPP) model and show that in the special case of the all-pairs family of directed communication graphs it is in fact stronger than the basic model proposed in [2].

Definition 1. *The MPP model with the additional constraint that it runs on the all-pairs family of directed communication graphs ($\mathcal{G}^d_{All}$), will be called the* basic MPP model.

Definition 2. *We say that a predicate is* strongly stably computable *by the MPP model, if it is stably computable in the classical sense of stable computation, that is, all agents eventually agree on the correct output value.*

On the other hand, if we say that a predicate is stably computable by the MPP model (without including the word "strongly"), it is not obvious if all agents agree on the same output value or not. Finally, when we say that a predicate is stably computable by the population protocol model, it always means that all agents eventually agree on the correct output value, since in this case we always follow the classical definition of stable computation, as appears in [2].

Theorem 4. *The population protocol model is a special case of the MPP model.*

Proof. Ignoring the edge functions, the edge states, the edge costs, and the output instruction r in the mediated population protocol model, makes the two models equivalent. □

All the following corollaries are immediate consequences of Theorem 4, since it shows that the population protocol model can be simulated by the mediated population protocol model. There is nothing that the population protocol model does that the MPP model is not capable of doing.

Corollary 1. *Any predicate stably computable by the population protocol model is also strongly stably computable by the mediated population protocol model.*

Corollary 2. *Any predicate stably computable by the basic population protocol model is also strongly stably computable by the basic MPP model.*

Corollary 3. *Any predicate stably computable by the population protocol model with stabilizing inputs is also strongly stably computable by the similar extension of the mediated population protocol model with stabilizing inputs.*

It is well known that Presburger arithmetic does not allow multiplication of variables. Moreover, any semilinear predicate can be described by first-order logical formulas in Presburger arithmetic and it is known that a predicate is computable in the basic population protocol model if and only if it is semilinear. To demonstrate that the basic MPP model is stronger, it suffices to show that there is at least one non-semilinear predicate that is strongly stably computable under this model.

It is obvious that the predicate "the number of c's is the product of the number of a's and the number of b's" is not semilinear. This holds, because multiplication of variables cannot be described by first-order logical formulas in Presburger arithmetic. Let N_q denote the multiplicity of state q in the input configuration multiset. Then, $N_c = N_a \cdot N_b$ is a shorthand of the above predicate and the mediated protocol $VarProduct$ that we will now describe, stably computes it in $\mathcal{G}^d_{All}$.

VarProduct

- $X = \{a, b, c, 0\}$, $Y = \{0, 1\}$,
- $Q = \{a, \dot{a}, b, c, \bar{c}, 0\}$, $S = \{0, 1\}$,
- $I(x) = x$, for all $x \in X$, $O(a) = O(b) = O(\bar{c}) = O(0) = 1$, and $O(c) = O(\dot{a}) = 0$,
- $\iota(x) = 0$, for all $x \in X$,
- r: "*If there is at least one agent with output 0, reject, else accept.*",
- δ: $(a, b, 0) \to (\dot{a}, b, 1)$, $(c, \dot{a}, 0) \to (\bar{c}, a, 0)$, $(\dot{a}, c, 0) \to (a, \bar{c}, 0)$

Theorem 5. *Protocol $VarProduct$ stably computes (according to our relaxed definition of stable computation) predicate $N_c = N_a \cdot N_b$ in $\mathcal{G}^d_{All}$.*

Proof Sketch. Notice that the number of links leading from agents in state a to agents in state b equals $N_a \cdot N_b$. For each a the protocol tries to erase b c's. Each a is able to remember the b's that has already counted (for every such b a c has been erased) by marking the corresponding links. If the c's are less than the product then at least one $\dot{a}$ remains and if the c's are more at least one c

remains. In both cases at least one agent that outputs 0 remains. If $N_c = N_a \cdot N_b$ then every agent eventually outputs 1. For a full proof see [8].

It is easy to see that $VarProduct$'s states in every computation eventually stop changing. Any protocol with the above property will be called a protocol with *stabilizing states*. Keep in mind that original stable computation of population protocols requires that all agents agree on their output value, and that it is correct. But $VarProduct$ does not seem to strongly stably compute $N_c = N_a \cdot N_b$, since the agents do not always agree on their final output value. Now we are about to prove that with a slight addition it does.

Note that instruction r defines a semilinear predicate on multisets of states (members of Q). To see this, simply write r formally as $(N_c > 0) \vee (N_{\dot{a}} > 0)$. The fact that it is semilinear suffices to prove that there is a population protocol $\mathcal{B}'$ with stabilizing inputs from the set Q of $VarProduct$'s states, that stably computes it. This follows from a result in [4], stating that any population protocol for fixed inputs can be adapted to work with stabilizing inputs. Moreover, Corollary 3 implies that there is a mediated protocol $\mathcal{B}$ with stabilizing inputs that is equivalent to $\mathcal{B}'$ (the one that ignores edges and does the same things as $\mathcal{B}'$), that is, it strongly stably computes the predicate defined by r. Finally, $VarProduct$ has stabilizing states, so its composition with $\mathcal{B}$ (their product construction) provides stabilizing inputs to $\mathcal{B}$. If the protocol's answer is now taken from $\mathcal{B}$'s output, then it is trivial to see that their composition strongly stably computes $N_c = N_a \cdot N_b$.

We state now a composition theorem to generalize this remark. Its proof can be found in [8]. In fact, our composition theorem holds for any family of directed and connected communication graphs $\mathcal{G}$.

Theorem 6. *Any MPP $\mathcal{A}$, that stably computes a predicate p with stabilizing states in some family of directed and connected communication graphs $\mathcal{G}$, containing an instruction r that defines a semilinear predicate t on multisets of $\mathcal{A}$'s agent states, can be composed with a provably existing MPP $\mathcal{B}$, that strongly stably computes t with stabilizing inputs in $\mathcal{G}$, to give a new MPP $\mathcal{C}$ satisfying the following properties:*

- *$\mathcal{C}$ is formed by the composition of $\mathcal{A}$ and $\mathcal{B}$,*
- *its input is $\mathcal{A}$'s input,*
- *its output is $\mathcal{B}$'s output, and*
- *$\mathcal{C}$ strongly stably computes p in $\mathcal{G}$.*

Definition 3. *Let* SEM *be the class of predicates stably computable, according to the classical sense of stable computation, by the basic population protocol model (precisely the semilinear predicates), and* MP *the class of number predicates strongly stably computable by the basic mediated population protocol model.*

Theorem 7. *SEM is a proper subset of MP.*

Proof. Corollary 2 implies that $SEM \subseteq MP$. Theorem 5 shows that there is a non-semilinear predicate, $p : N_c = N_a \cdot N_b$, that does not belong to SEM but

is stably computable by the basic MPP model (according to our definition of stable computation). The mediated protocol $VarProduct$ that stably computes p, contains an output instruction r that defines a semilinear predicate t on multisets of $VarProduct$'s states. Consequently, Theorem 6 applies and we get that p is also strongly stably computable by the basic MPP model (i.e. stably computable according to the classical sense of stable computation), in other words, p belongs to MP, and the theorem follows. □

4.2 Any Family of Communication Graphs : Non-uniform upper bounds on computability

Definition 4. *Let* UMP *be the class of predicates stably computable by the MPP model in any family $\mathcal{G}$ of undirected, connected communication graphs, and* DMP *the class of predicates stably computable by the MPP model in any family $\mathcal{G}'$ of directed, connected communication graphs.*

Let m denote the number of edges of any communication graph G.

Theorem 8. *All predicates in DMP and UMP are in the class $NSPACE(m)$.*

Proof. Let $\mathcal{A}$ be a mediated protocol that stably computes such a predicate p in some family of graphs $\mathcal{G}$, and let $G \in \mathcal{G}$ be any graph of this family. Since G is always connected, we have that $m \geq n - 1$. A network configuration can be represented explicitly, by storing a state per node and a state per edge of G. This takes $\mathcal{O}(m)$ space. Note that w.l.o.g. we can talk about languages instead of predicates. So, $\mathcal{A}$ stably computes the language L corresponding to p (its *support*).

We will now present a nondeterministic Turing machine $M_{\mathcal{A}}$ that decides L in space $\mathcal{O}(m)$. $M_{\mathcal{A}}$ works as follows: To accept input x, $M_{\mathcal{A}}$ must verify two conditions: That there exists a configuration C reachable from $I(x)$ (the initial configuration corresponding to x), in which all relevant states satisfy the output instruction r, and that there is no configuration C' reachable from C, in which r is violated.

The first condition is verified by guessing and checking a sequence of network configurations, starting from $I(x)$ and reaching such a C. $M_{\mathcal{A}}$ guesses a C_{i+1} each time, verifies that $C_i \rightarrow C_{i+1}$ (begins from $C_0 = I(x)$, i.e. $i = 0$) and, if so, replaces C_i by C_{i+1}, otherwise drops this C_{i+1}. The second condition is the complement of a similar reachability problem. But $NSPACE$ is closed under complement for all space functions $\geq \log n$ (see [11]). Thus, $M_{\mathcal{A}}$ decides L in $O(m)$ space. □

Note that, as far as a DMP is concerned, even a "standard" (not mediated) population protocol whose G is a directed line can simulate a deterministic linear space Turing machine [2]. Thus, by applying Savitch's theorem [12], one can say informally that DMP is between $NSPACE(\sqrt{n})$ and $NSPACE(m)$. However, for UMP, we only know that $SEM \subset UMP \subseteq NSPACE(m)$.

5 Future Work

We are currently experimenting with Population Protocols (see [7]) and investigating the possible graph properties that can be computed by Mediated Population Protocols.

Acknowledgements. We thank Maria Serna for posing the question of Transitive Closure and its importance.

References

1. D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In Proc. Distributed Computing in Sensor Systems: 1st IEEE International Conference, pages 63-74, 2005.
2. D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *23rd Annual ACM Sympsium on Principles of Distributed Computing* (*PODC*), pages 290-299, New York, NY, USA, 2004. ACM.
3. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3): 183-199, Sept. 2008.
4. D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292-299, 2006.
5. J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98-117, October 2007. Columns: *Distributed Computing*, Editor: M. Mavronicolas.
6. O. Bournez, P. Chassaing, J. Cohen, L. Gerin, and X. Koegler. On the convergence of population protocols when population goes to infinity. To appear in *Applied Mathematics and Computation*, 2009.
7. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Experimental verification and performance study of extremely large sized population protocols. FRONTS Technical Report FRONTS-TR-2009-3, http://fronts.cti.gr/aigaion/?TR=61, Jan. 2009.
8. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Mediated Population Protocols. FRONTS Technical Report FRONTS-TR-2009-8, http://fronts.cti.gr/aigaion/?TR=65, Feb. 2009.
9. I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In *Distributed Computing*, *22nd International Symposium*, *DISC*, volume 5218 of *Lecture Notes in Computer Science*, pages 498-499, 2008.
10. P. Fraigniaud, C. Gavoille, D. Ilcinkas, and A. Pelc. Distributed computing with advice: Information sensitivity of graph coloring. *34th International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.
11. N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935-938, Oct. 1988 (see also page 153 C. H. Papadimitriou "Computational Complexity").
12. W. J. Savitch. Relationship between nondeterministic and deterministic tape classes. *J.CSS*, 4, pages 177-192, 1970 (see also page 149-150 C. H. Papadimitriou "Computational Complexity").