Self-similar Functions and Population Protocols: A Characterization and a Comparison

Swapnil Bhatia and Radim Bartoš

Department of Computer Science, Univ. of New Hampshire, Durham, NH
{sbhatia,rbartos}@cs.unh.edu

Abstract. Chandy et al. proposed the methodology of "self-similar algorithms" for distributed computation in dynamic environments. We further characterize the class of functions computable by such algorithms by showing that self-similarity induces an additive relationship among the level-sets of such functions. Angluin et al. introduced the population protocol model for computation in mobile sensor networks and characterized the class of predicates computable in a standard population. We define and characterize the class of self-similar predicates and show when they are computable by a population protocol.

1 Introduction

Mobile wireless sensor networks hold tremendous promise as a technology for sampling a variety of phenomena at unprecendented granularities of time and space. Such networks embody a modern-day "macroscope": an instrument that can potentially revolutionize science by enabling the measurement, understanding—and eventually—control, of a whole new class of physical, biological, and social processes. The source of potential of such networks lies in the following four capabilities endowed to each participating node: the ability to sense environmental data, the ability to compute on such data, the ability to communicate with peers in the network, and the ability to move in its environment. A network of autonomous underwater vehicles (AUVs) deployed to patrol a harbor, to map the locations of underwater mines, to monitor the diffusion of a pollutant in a river, or to build a bathymetric map are some realistic examples of missions that mobile sensor networks are charged with today.

While there has been tremendous interest in building such networks in recent years, most of this work has focused on a proper subset of the four capabilities of mobile sensor nodes described above. Work on mobile ad hoc networks has focused on mobility and communication [1,2,3] and sensor network research has mostly focused on sensing and communication [4,5]. More recently, there has been a growing interest in in-network computation and communication in static sensor networks [6,7]. We believe that all this previous work paves the way for a more comprehensive model that includes all four of the above abilities, particularly computation. Such a model would allow us to frame new questions from the point of view of the *computational mission* of the network and provide us insight into the design tradeoffs of such networks for various classes of missions. This paper represents an intermediate step toward this goal.

In this paper, we focus on two recent papers that deal with distributed computation in dynamic environments—the first by Chandy et al. [8] and the second by Angluin et al.

V. Garg, R. Wattenhofer, and K. Kothapalli (Eds.): ICDCN 2009, LNCS 5408, pp. 263–274, 2009.

[9]—and attempt to characterize the relationship between their work. Both papers are motivated by the need to understand computation in distributed systems that exist in highly dynamic environments similar to those in which mobile sensor networks are deployed. Using approaches that complement each other, these papers attempt to abstract the four capabilities of mobile sensor nodes described above to answer new questions regarding computation in such networks. Chandy et al. propose a methodology for designing algorithms for dynamic distributed systems and identify a class of functions amenable to their method. They outline a method for systematically designing such algorithms which they call "self-similar algorithms." (We call the functions computed by such algorithms self-similar functions.) The approach taken by Angluin et al. complements that of Chandy et al. in that instead of starting with a class of functions, Angluin et al. define a computational model called the population protocol model, which abstracts the four capabilities of mobile sensor nodes described above. Their model comprises a population of anonymous identical nodes, each with a small constant amount of memory, that communicate and compute opportunistically during encounters with each other. In a series of papers [9,10,11], Angluin et al. have characterized the class of predicates computable in a standard population model. The goal of this paper is to further characterize the class of functions defined by Chandy et al. and to understand its relationship to the computational model defined by Angluin et al.

Our contributions are as follows. Restricted to a finite input space (but any number of sensing nodes), we study the structure of self-similar functions and show how their definition imposes an additive relationship on the level-sets of such functions, a property that is similar to the one known to hold for predicates computable by population protocols in a standard population. Using these results and known results about population protocols, we show that although population protocols and self-similar functions are identically motivated, these two concepts do not coincide. For a given convention of representing predicates, we define and characterize the class of self-similar predicates and those computable by a population protocol. While self-similarity more generally captures the properties required of a function to be distributedly computable in a dynamic environment, the constraints that its definition imposes appear to be stronger than those imposed by population protocols. On the other hand, the notion of self-similarity appears to be more general than the notion of opportunistic computation in a population protocol. Our work constrasts these two conceptions of computation in dynamic environments in a mobile sensor network and highlights their particular strengths. We hope that this increased understanding of existing models will usher in better models of mobile sensor networks that incorporate the computational mission of such networks. We also hope that this paper will generate interest in a study of mobile sensor networks that unifies computation, communication, mobility, and sensing.

2 Self-similar Algorithms

Implicit in the paper by Chandy et al. [8] are the following questions: How can we derive distributed algorithms that compute correctly in dynamic environments? Which functions are amenable to distributed computation in dynamic environments? To answer the first question, Chandy et al. begin by enumerating properties that a computation must

possess, if it is to execute correctly in a dynamic distributed environment. They restrict their investigation to stable and idempotent functions which can be computed by what they call "self-similar algorithms." By stability, it is meant that once a computation achieves its "final" state, it remains in that state forever, thus providing a stable answer. It follows that a computation in such an environment must be conservative in the sense that it must always transition to only those states that would not result in an incorrect computation; all transitions must conserve the correct final answer. That is, if s_i is the collective state of the computational agents in the system in the ith step and f is the function to be computed, then $f(s_i) = f(s_0)$ for all i. Finally, a self-similar algorithm is one in which any "group behaves like the entire system" [8]. More precisely, suppose f is a function that is to be computed by a collection of agents. Then, a self-similar algorithm A for f is one which can be executed by any (nonempty) subset of identical agents participating in a sequence of arbitrary groupings such that the result of their "local" computation is compatible with and usually contributes to the "global" computation that is to be executed. Chandy et al. show that the above properties—stability, idempotence, conservation, and computability by self-similar algorithms—hold exactly for a class of functions they call *superidempotent*.

Definition 1. *A function f from multisets to multisets is* **superidempotent** *if $f(X \cup Y) = f(f(X) \cup Y)$ [8].*

In this paper, we shall refer to such functions as self-similar functions to emphasize their computability by a self-similar algorithm.

2.1 General Observations

It is easy to see that the class of self-similar functions excludes some familiar functions.

Proposition 1. *Any one-to-one function (except for the identity) is not self-similar, because it is not idempotent.*

On the other hand, self-similar functions include some familiar functions.

Proposition 2. *An idempotent homomorphism is self-similar.*

Proof. If f is an idempotent homomorphism, then the r.h.s. in the definition of superidempotence $f(f(X) \cup Y) = f(f(X)) \cup f(Y)$ (by homomorphism), $= f(X) \cup f(Y)$ (by idempotence) $= f(X \cup Y)$ (by homomorphism), which is the l.h.s. of the definition of superidempotence. □

Corollary 1. *Let T be a linear transformation such that $T^2 = T$. Then, T is self-similar.*

Proof. By its definition, T is idempotent and linearity implies the homomorphism property. □

Thus, all projections (i.e., linear transformations T such that $T^2 = T$) are self-similarly computable.

2.2 Finite-Valued Self-similar Functions

While Chandy et al. define self-similar algorithms over infinite input spaces, in order to compare the class of such functions with a realizable model of computation, we study self-similar functions over a finite input space (alphabet) in this paper. Let Q be the finite nonempty set of possible input values for any agent and let $|Q| = q$ be a positive integer. Consider the q-dimensional space $\mathbb{N}^q$ of nonnegative integers.

Lemma 1. *Let $Q^\star$ be the (infinite) set of all finite multisets containing elements from Q. There exists an isomorphism ϕ between the monoids $(Q^\star, \cup)$ and $(\mathbb{N}^q, +)$.* [1]

Thus, the definition of superidempotence can be translated from $(Q^\star, \cup)$ to $(\mathbb{N}^q, +)$ as follows: a function $f : \mathbb{N}^q \to \mathbb{N}^q$ is superidempotent if and only if $f(x + y) = f(f(x) + y)$ for all $x, y \in \mathbb{N}^q$. In this subsection, $f : \mathbb{N}^q \to \mathbb{N}^q$ is a self-similarly computable function. From the definition of superidempotence, we know

Fact 1. *For any $u, v \in \mathbb{N}^q$, $f(u+v) = f(f(u)+v) = f(u+f(v)) = f(f(u)+f(v))$.*

Definition 2. *For any $v \in \mathbb{N}^q$, denote by $\sum v$ the integer $\sum_{i=1}^{q} v_i$, and by H_k^q the hyperplane $\{v \in \mathbb{N}^q | \sum v = k\}$.*

We assume that a computational step is agent conserving in that the number of a agents in a group participating in a computational step does not change during the step. From this we have

Fact 2. *If $f : \mathbb{N}^q \to \mathbb{N}^q$ and $v \in \mathbb{N}^q$, $\sum v = \sum f(v)$.*

That is, any $v \in \mathbb{N}^q$ lives in the $q-1$-dimensional hyperplane $\{u : \sum u = \sum v\}$ of $\mathbb{N}^q$ and any self-similar f maps v to an $f(v)$ in the same plane. It is useful to know the number of points in each H_k^q. For each $k \in \mathbb{N}$, the number of points in H_k^q is the number of integral solutions of the equation $\sum v = k$ Therefore, $|H_k^q| = \binom{k+q-1}{q-1}$.

Definition 3. *The set of all points (multisets) x such that $f(x) = y$, for some fixed y, is called a* **fiber**. *A fiber is* **trivial** *if it contains exactly one point. Any subset of a fiber of f is called a* **contour of** *f. A contour of f containing u that also contains $f(u)$ is called a* **complete contour.** *The* **value** *of a contour is $f(u)$ for any u in the contour.*

Self-similar computations progress along trajectories that must be contained in fibers; if not then f cannot be conservative. Fibers play a central role in self-similar functions. Indeed, self-similarity induces an additive relationship between contours, as we show below.

Theorem 1 (Direct sum of contours). *If U and V are contours, then $U \oplus V = \{u + v | u \in U, v \in V\}$ is also a contour.*

Proof. Since the claim is trivially true if either U or V are empty, we assume that they are both nonempty. For any $w_1, w_2 \in U \oplus V$ let $w_1 = u_1 + v_1$ and $w_2 = u_2 + v_2$ for some $u_1, u_2 \in U$ and $v_1, v_2 \in V$. Now $f(w_1) = f(u_1 + v_1) = f(f(u_1) + f(v_1)) = f(f(u_2) + f(v_2)) = f(u_2 + v_2) = f(w_2)$, where the second and fourth equalities follow from the definition of superidempotence, and the third from the definition of a contour. □

[1] We omit this and several other easy proofs below due to lack of space; see [12].

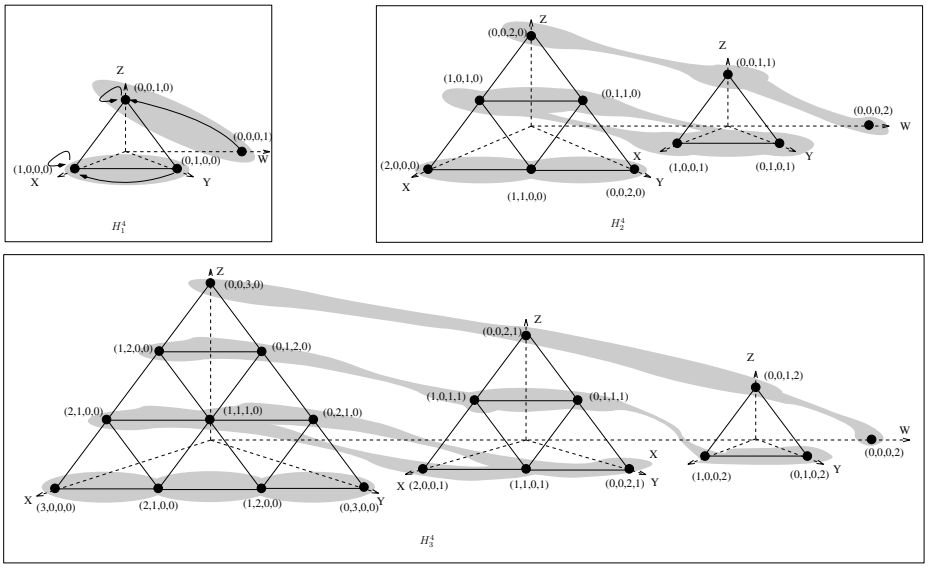

Fig. 1. Relationship between contours of a self-similar function $f : \mathbb{N}^4 \to \mathbb{N}^4$. (The axes in order are X, Y, Z, and W.) Points (black disks) included in the same shaded region form a contour. Notice that H_2^4 contains four copies of H_1^4, and H_3^4 contains four copies of H_2^4. Contours are invariant under translation from H_k^4 to H_{k+1}^4.

Corollary 2 (Translation of a contour). *If U is a contour, then for any $v \in \mathbb{N}^q$, the translation $U + v = \{u + v | u \in U\}$ of U is also a contour.*

Proof. For any $v \in \mathbb{N}^q$, $\{v\}$ is (trivially) a contour. Then by Theorem 1, $U \oplus \{v\}$ is a contour. □

Figure 1 illustrates this relationship between contours of a self-similar function $f : \mathbb{N}^4 \to \mathbb{N}^4$. Over H_1^4, it is defined as follows: $f(0,1,0,0) = f(1,0,0,0) = (1,0,0,0)$ and $f(0,0,0,1) = f(0,0,1,0) = (0,0,1,0)$. Thus, there are two fibers (and contours) in H_1^4: $\{(0,1,0,0),(1,0,0,0)\}$ and $\{(0,0,0,1),\ (0,0,1,0)\}$. (There are only $\binom{1+4-1}{4-1} = 4$ points in H_1^4 and they are thus partitioned into two fibers.) As per the above results, any translation of these two contours must also be a contour. Thus, $\{(0,0,0,1)+(1,0,0,0),(0,0,1,0)+(1,0,0,0)\} = \{(1,0,0,1),(1,0,1,0)\}$ for example, must be a contour in H_2^4. (There are three more possible translations, one along each of the axes, all of which must also be contours in H_2^4.) Since points in each contour must have the same value, the union of intersecting contours must form a single contour. For example, the intersecting contours $\{(1,0,1,0),(1,0,0,1)\}$, $\{(0,1,1,0),(0,1,0,1)\}$, $\{(1,0,1,0),(0,1,1,0)\}$, and $\{(1,0,0,1),(0,1,0,1)\}$ of H_2^4 together form a single contour, as shown by the overlapping shaded regions of the figure. Similarly, the contours in H_2^4, when translated along any of the four axes must form contours in H_3^4, as shown in the figure.

Viewing contours in translation justifies naming such functions as self-similar: contours in H_k^q are the result of translating contours in H_{k-1}^q in q ways and are thus copies

of them; those in H^q_{k-1} are copies of those in H^q_{k-2}; and so on. However, while contours are invariant under translation, the value of a contour in H^q_k, in contrast to the standard notion of self-similarity, need not bear any relationship to the value of a contour in H^q_{k-1}. For example in Figure 1, the value of any point in H^4_2 under f is not determined by its contour membership: the contour only requires that the value of all its points be the same.

The results proved above are fundamental in understanding the structure of self-similar functions. They complement the description given by Chandy et al. that self-similar algorithms are those in which "any group behaves like the entire system" [8]. Our results show that for finite input spaces, such algorithms compute functions in which self-similarity manifests itself in the form of an additive relationship between contours: larger contours are formed by translating smaller contours. This clarifies the notion of self-similarity proposed by Chandy et al. and makes our understanding of it more precise.

We now state two useful results that immediately follow from the above results.

Definition 4. *For any $u = (u_1, \ldots, u_q) \in \mathbb{N}^q$ and $v = (v_1, \ldots, v_q) \in \mathbb{N}^q$, we define the* **partial order** $\leq$ *as follows: $u \leq v \iff \forall i \in \{1, \ldots, q\} : u_i \leq v_i$.*

Lemma 2. *Let $\{v^1, \ldots, v^r\}$ be a contour in H^q_k (with k such that $1 \leq r \leq |H^q_k|$) and let $u \in \mathbb{N}^q$ such that $u \leq v^i$ for $i = 1, \ldots, r$. Then the set $\{u + \sum_{i=1}^r m_i(v^i - u) : m_i \in \mathbb{N}, \sum_{i=1}^r m_i = m\}$ is a contour in $H^q_{j+m(k-j)}$, where $j = \sum u$.*

Proof. We prove this by induction on r.

Basis. If $r = 1$, then we must show that if $\{v^1\}$ is a contour in H^q_k and $u \leq v^1$, then the set $\{u + m_1(v^1 - u) : m_1 \in \mathbb{N}, m_1 = m\}$ is a contour in $H_{j+m(k-j)}$, where $j = \sum u$. Since $v^1 \in H^q_k$, and $u \in H^q_j$, $u + m(v^1 - u) \in H^q_{j+m(k-j)}$. For any $m_1 = m \in \mathbb{N}$ the set in question contains a single vector and is therefore trivially a contour.

Induction hypothesis. Suppose the statement is true for $r = n$.

Inductive step. For $r = n$, we are given that the set $\{u + \sum_{i=1}^n m_i(v^i - u) : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\}$ is a contour in $H^q_{j+m(k-j)}$. Let $v^{n+1} \in H^q_k$. Let $U_n = \{u : u \leq v^i, i = 1, \ldots, n\}$ and $U_{n+1} = \{u : u \leq v^i, i = 1, \ldots, n+1\}$. Then it must be that $U_{n+1} \subseteq U_n$ because if $u \in U_{n+1}$, then it must necessarily be no larger than $v^1, \ldots, v^n$. Moreover, $(0, \ldots, 0) \in U_{n+1}$ and hence U_{n+1} is nonempty. Thus, the induction hypothesis holds for all $u \in U_{n+1}$, since it holds for U_n. Let $u' \in U_{n+1}$ with $\sum u' = j'$. Therefore, $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\}$ is a contour in $H^q_{j'+m(k-j')}$ as per the induction hypothesis.

Now, the set $\{m_{n+1}(v^{n+1} - u')\}$ is a contour in $H^q_{m_{n+1}(k-j')}$ for any fixed $m_{n+1} \in \mathbb{N}$ because it contains a single point. Therefore, by Theorem 1, $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\} \oplus \{m_{n+1}(v^{n+1} - u')\}$ is a contour in $H^q_{j'+m'(k-j')}$, where $m' = m + m_{n+1}$. But $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\} \oplus \{m_{n+1}(v^{n+1} - u')\} = \{u' + \sum_{i=1}^{n+1} m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^{n+1} m_i = m'\}$. Thus, we have shown that this set is a contour in $H_{j'+m'(k-j')}$, where $j' = \sum u'$. □

The union of the contours described in the above result is called a *linear* set. Such sets are closely related to the type of predicates computable in the standard population protocol model.

We now state a useful special case of the above result. We omit the proof, which follows directly from the previous result, due to space restrictions.

Corollary 3. *Let $u, v^1, v^2, \ldots, v^q \in \mathbb{N}^q$ be such that $v^i = u + e_i$ where $\{e_1, \ldots, e_q\}$ is the standard basis for $\mathbb{N}^q$. If $\{v^1, \ldots, v^q\}$ is a contour, then f is constant in each H_k^q for all points $w \geq u$.*

Lemma 3. *Any function $f : \mathbb{N}^q \to \mathbb{N}^q$ that is constant over each H_k^q and maps each H_k^q to itself is self-similar.*

Theorem 2. *There exists a function $f : \mathbb{N}^2 \to \mathbb{N}^2$ that is not computable but is self-similar.*

Proof. Let $w = w_2, w_3, \ldots$ be an infinite sequence of nonnegative integers such that $0 \leq w_k < |H_k^2|$. Let $f_w : \mathbb{N}^2 \to \mathbb{N}^2$ be a function constant over each H_k^2 such that for any $(i, k-i) \in H_k^2$, $f_w(i, k-i) = (w_k, k - w_k)$: w_k defines the value of the function in H_k^2. If $w \neq w'$ are two sequences as defined above such that $w_k \neq w'_k$, then $f_w(i, k-i) = (w_i, k - w_i) \neq (w'_i, k - w'_i) = f_{w'}(i, k-i)$. Thus, every sequence w defines a distinct function f_w that is constant over each H_k^2. By Lemma 3, each such f_w is self-similar. However, the set $\{f_w | w = w_2, w_3, \ldots; 0 \leq w_k < |H_k^2|\}$ is uncountable, whereas the set of Turing machines is countable. □

3 Population Protocols and Self-similar Functions

In a series of recent papers, Angluin et al. have defined the *population protocol* model of distributed computation and characterized its computational power [9]. A *population* is a collection of n anonymous computational agents with an undirected population graph on n vertices. Each agent is modeled as a deterministic finite automaton, with a finite set of transition rules from pairs of states to pairs of states. In the randomized variant (see [11] for details), an input symbol from an input alphabet is provided to each agent, and a fixed input function maps it to the initial state of the agent. A computation evolves in discrete steps, and at each step, an edge (i, j) of the population graph is chosen uniformly at random by the "environment": this models a pairwise random *encounter* between agents during which they communicate and compute. During such an encounter, agents i and j transition from their current states q_i and q_j to new states according to the population protocol (i.e., $(q_i, q_j) \to (q'_i, q'_j)$). The collective state of all n agents can be completely described by an n-dimensional vector over the states of the protocol, where the ith component is the current state of the ith agent. Thus, an execution is an *infinite* sequence of n-dimensional vectors. At any step, the current output of the computation can be obtained by mapping the current state of any agent to the output alphabet using a given fixed output function. A function f is stably computed by a population protocol iff for any input assignment x, the computation eventually converges to an orbit of n-vectors, all of which map to the unique $f(x)$ under the output function. We recall some definitions below [9].

Definition 5. *A* **population protocol** *$\mathcal{A}$ is a 6-tuple $\mathcal{A} = (X, Y, Q, I, O, \delta)$ where: X is the* **input alphabet***, Y is the* **output alphabet***, Q is a set of* **states***, $I : X \to Q$ is the*

input function, $O : Q \to Y$ *is the* **output function**, *and* $\delta : Q \times Q \to Q \times Q$ *is the* **transition function**.

Definition 6. *A* **population** $\mathcal{P}$ *is a set* A *of* n *agents with a directed graph over the elements of* A *as vertices and edges* $E \subseteq A \times A$. *The* **standard population** $\mathcal{P}_n$ *is the set of* n *agents* $A_n = \{a_1, \ldots, a_n\}$ *with the complete directed graph (without loops) on* A_n.

Definition 7. *A* **semilinear set** *is a subset of* $\mathbb{N}^q$ *that is a finite union of linear sets of the form* $\{u + k_1v_1 + k_2v_2 + \cdots + k_mv_m\}$ *where* u *is a* q*-dimensional base vector,* $v_1, \ldots, v_m$ *are* q*-dimensional basis vectors, and* $k_1, \ldots, k_m \in \mathbb{N}$. *A* **semilinear predicate** *is one that is true precisely on a semilinear set.*

The computational power of population protocols was characterized by Angluin et al. [10,11].

Theorem 3 (Theorem 6 in [11]). *A predicate is computable by a population protocol in a standard population if and only if the set of points on which it is true is semilinear.*

3.1 Self-similar Functions Computed by Population Protocols

Theorem 4. *If the population protocol* $\mathcal{A} = (X, X, Q, I, I^{-1}, \delta)$ *stably computes a function* $f : X^\star \to X^\star$ *from multisets to multisets over* X *in the standard population* $\mathcal{P}_n$, *then* f *is self-similar.*

Proof sketch. A population protocol $\mathcal{A}$ that correctly executes in a standard population $\mathcal{P}_n$ must also correctly execute in any population $P \subseteq \mathcal{P}_n$ because it cannot distinguish between the two populations. Partition $\mathcal{P}_n$ into P and P'. Let t be larger than the number of steps required for $\mathcal{A}$ to converge when executed in P and P'. Let $f(P)$ and $f(P')$ denote the output respectively. Now execute $\mathcal{A}$ in $\mathcal{P}_n$ such that for the first t steps no inter-partition encounter is allowed, and after t steps all encounters are allowed. The intermediate output will be $f(P) \cup f(P')$ and the final output will be $f(f(P) \cup f(P')) = f(\mathcal{P}_n)$. □

3.2 Predicates: Semilinear and Self-similar

Definition 8. *A* **predicate** *is a function* $P : \mathbb{N}^q \to \{T, F\}$. *For any predicate* P, *its* **consensus predicate** *form is a function* $f : \mathbb{N}^q \to \mathbb{N}^q$ *such that for any* $v \in \mathbb{N}^q$, $f(v) = (\sum v, 0, 0, \ldots, 0)$ *iff* $P(v) = T$ *and* $f(v) = (0, \sum v, 0, \ldots, 0)$ *iff* $P(v) = F$. *We call the consensus predicate form* **self-similar** *if* f *is self-similar.*

The consensus predicate defined above follows the "all-agents output convention" as defined by Angluin [9] which requires all agents to agree on the truth-value of the predicate. In the sequel, our results involve only those predicates that are expressible in this convention because this is one of the conventions used by Angluin et al. and we are interested in comparing self-similar predicates to population protocol computable predicates. We postpone the discussion of more robust conventions to future work.

Proposition 3. *Not all semilinear consensus predicates are self-similar consensus predicates.*

Proof. Consider the following consensus predicate: $f(i,j) = (i+j, 0)$ if $j \leq i$ and $f(i,j) = (0, i+j)$ otherwise. It is easy to show that this predicate is semilinear and idempotent but not self-similar. □

If a predicate P is always true or always false, then its consensus form function will be constant over each H_k^q, and by Lemma 3, will be self-similar. We say that a predicate is eventually constant if there is a $k \in \mathbb{N}$ such that the predicate is constant over H_k^q. (Corollary 3 implies that the predicate is then constant for all $k' \in \mathbb{N}$ such that $k' \geq k$.)

Theorem 5. *A predicate $P : \mathbb{N}^q \to \{T, F\}$ that is not eventually constant has a self-similar consensus form $f : \mathbb{N}^q \to \mathbb{N}^q$ if f is idempotent and either the set of points on which P is true or that on which P is false has a standard basis.*

Proof. Suppose the set of true points of P has a standard basis T_1^q. Thus, P is true only on points in $\mathsf{span}(T_1^q)$ and hence P is false only on points in $\mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q))$.

To show that P has a self-similar consensus form f, we must show that $\forall v \in \mathbb{N}^q : \forall u \leq v : f(v) = f(f(u) + f(v-u))$. Suppose $P(v) = T$, that is $v \in \mathsf{span}(T_1^q)$. Then $\forall u \leq v : u \in \mathsf{span}(T_1^q)$ because u must have zeroes in at least those coordinates in which v has zeroes. Now since $P(v) = T$, $f(v) = (\sum v, 0, \ldots, 0)$ by definition of the consensus form. On the other hand, $f(f(u) + f(v-u)) = f((\sum u, 0, \ldots, 0) + (\sum (v-u), 0, \ldots, 0)) = f(\sum v, 0, \ldots, 0)$. Since $f(v) = (\sum v, 0, \ldots, 0)$, and since f is idempotent, $f(\sum v, 0, \ldots, 0) = (\sum v, 0, \ldots, 0)$. Thus, we have shown that $\forall v \in \mathsf{span}(T_1^q) : \forall u \leq v : f(v) = f(f(u) + f(v-u))$.

Now suppose $P(v) = F$. Thus $v \notin \mathsf{span}(T_1^q)$, that is $v \in \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q)) = \mathsf{span}(F_1^q) \cup \mathsf{span}(F_1^q \oplus T_1^q)$. If $v \in \mathsf{span}(F_1^q)$, then the same argument as above applies because $\forall u \leq v : P(u) = F$.

If $v \in \mathsf{span}(F_1^q \oplus T_1^q)$, then $v = v_F + v_T$ for some $v_F \in \mathsf{span}(F_1^q)$ and some $v_T \in \mathsf{span}(T_1^q)$. Thus $P(v_F) = F$ and $P(v_T) = T$ and therefore $f(v_F) = (0, \sum v_F, 0, \ldots, 0)$ and $f(v_T) = (\sum v_T, 0, \ldots, 0)$. Therefore $f(f(v_F) + f(v_T)) = f(\sum v_T, \sum v_F, 0 \ldots, 0)$. Since P is not eventually constant, $(i, 0, \ldots, 0) \in \mathsf{span}(T_1^q)$ and $(0, j, 0, \ldots, 0) \in \mathsf{span}(F_1^q)$ for all $i, j \in \mathbb{N}$. Hence $(\sum v_T, 0, \ldots, 0) \in \mathsf{span}(T_1^q)$ and $(0, \sum v_F, \ldots, 0) \in \mathsf{span}(F_1^q)$ and therefore $(\sum v_T, \sum v_F, 0 \ldots, 0) \in \mathsf{span}(T_1^q \oplus F_1^q)$. Therefore, $P(\sum v_T, \sum v_F, 0, \ldots, 0) = F$ and hence $f(\sum v_T, \sum v_F, 0 \ldots, 0) = (0, \sum v, 0, \ldots, 0)$. Thus, we have shown that $\forall v \in \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q))$ f is self-similar. □

Theorem 6. *If $P : \mathbb{N}^q \to \{T, F\}$ is a predicate with a self-similar consensus form, then at least one of the following holds: Either the set of points on which P is true or that on which P is false has a standard basis; or P is eventually constant.*

Proof. Consider the q points in H_1^q. If P is true on all q points or false on all q points, then P is eventually constant. So, assume otherwise and let the true fiber $T_1^q \subset H_1^q$ and the false fiber $F_1^q \subset H_1^q$ partition H_1^q (with $e_1 \in T_1^q$ and $e_2 \in F_1^q$ as per the definition of the consensus form convention).

Now consider H_2^q and observe that $H_2^q = (T_1^q \oplus T_1^q) \cup (F_1^q \oplus F_1^q) \cup (T_1^q \oplus F_1^q)$. By Theorem 1, $T_1^q \oplus T_1^q$, $F_1^q \oplus F_1^q$ and $T_1^q \oplus F_1^q$ are all contours and thus P is constant over each of these sets in H_2^q.

For some $v \in T_1^q \oplus F_1^q$, suppose $P(v) = F$. Then, all points in $T_1^q \oplus F_1^q$ must map to $(0, 2, 0, \ldots, 0)$ since P must be false on all these points. Furthermore, $f(0, 2, 0, \ldots, 0) = (0, 2, 0, \ldots, 0)$ since f is self-similar and hence idempotent. But $(0, 2, 0, \ldots, 0) \in F_1^q \oplus F_1^q$ and hence P must be false on all the points in $F_1^q \oplus F_1^q$. Thus, we can write the false fiber $F_2^q \supseteq (F_1^q \oplus F_1^q) \cup (F_1^q \oplus T_1^q) = F_1^q \oplus (F_1^q \cup T_1^q) = F_1^q \oplus H_1^q$. (It is easy to check that $\oplus$ distributes over $\cup$.) The only points remaining in H_2^q are those in the contour $T_1^q \oplus T_1^q$. If P is false on any of these points, then P is constant on H_2^q, and thus P is eventually constant. So assume that P is true on each point in the contour $T_1^q \oplus T_1^q$. Therefore, the set of true points in H_2^q, i.e., the true fiber in H_2^q is $T_2^q = T_1^q \oplus T_1^q$ and the false fiber $F_2^q = F_1^q \oplus H_1^q$. Thus, H_2^q is partitioned into two nonemtpy fibers.

Now $\mathbb{N}^q = \mathsf{span}(H_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q) \cup \mathsf{span}(F_1^q \oplus T_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q)) = \mathsf{span}(T_1^q) \cup \mathsf{span}((F_1^q \cup F_1^q) \oplus (F_1^q \cup T_1^q)) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q \oplus H_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_2^q)$. Using Corollary 3 and considering F_2^q as the contour we obtain that the $\mathsf{span}(F_2^q) \cap H_k^q$ is a contour in every H_k^q, $k \geq 2$. If the value of this contour in any H_k^q is true, then P is constant over all of that H_k^q and thus is eventually constant. If the value of this contour is false for all H_k^q, and for some k, the value of T_k^q is also false, then P is constant over all of H_k^q and thus is eventually constant. If the value of this contour is false for all H_k^q, and the value of T_k^q is true for all H_k^q, then the set of true points has a standard basis T_1^q.

We assumed that for some $v \in T_1^q \oplus F_1^q$, $P(v) = F$. If we assume that $P(v) = T$, then we can show that the set of false points has a standard basis F_1^q. □

From this, and Angluin et al.'s Theorem 3 immediately follows

Theorem 7. *If predicate $P : \mathbb{N}^q \to \{T, F\}$ is not eventually constant and has a self-similar consensus form, then P is computable by a population protocol.*

Proof. Since P has a self-similar consensus form and is not eventually constant, the set of points on which either P or its negation is true has a standard basis and is therefore a linear set. Population protocols are closed under complement. □

For predicates that are eventually constant, self-similarity imposes no additional constraints within each H_k^q. Thus, for any $k \in \mathbb{N}$, the predicate may be true on all points in H_k^q or false on all points in H_k^q. Therefore, the computability of such predicates by a population protocol is given directly by Theorem 3.

3.3 Self-similar Functions Not Computable by Population Protocols

It is known that all predicates that are computable in the standard population are in the class NL [9], the set of functions computable by a nondeterministic Turing machine with access to memory logarithmic in the size of the input.

Theorem 8. *There exists a self-similar function that is in NL but whose predicate form is not computable by any population protocol.*

Proof. Let $f : \mathbb{N}^2 \to \mathbb{N}^2$ be the constant function such that for any $(i, k - i) \in H_k$, $f(i, k - i) = (k - \lfloor \lg k \rfloor, \lfloor \lg k \rfloor)$. By Lemma 3, f is self-similar. Since f requires

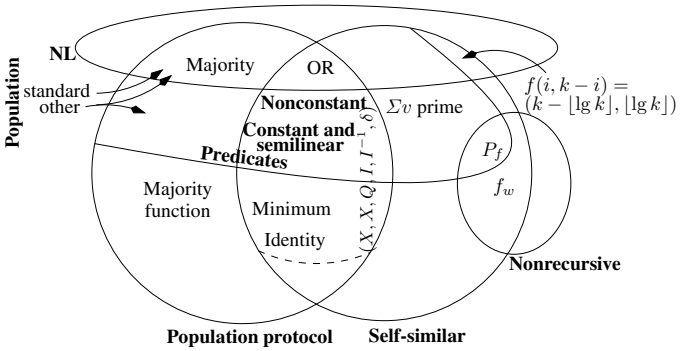

Fig. 2. Relationship between self-similar functions and functions computable by population protocols (Bold names differentiate classes from examples. Not all relationships are known).

an addition, the counting of the number of bits of the result, and a subtraction, it is in L, the set of functions computable with a deterministic Turing machine with access to memory logarithmic in the size of the input. It is known that L $\subseteq$ NL [13] and thus f is in NL. Define the predicate $P_f(v)$ over all points $v \in \mathbb{N}^2$ such that it is true if and only if $v \in H_k^2$ is the image of all $u \in H_k^2$ under f. From Theorem 3 we can deduce that the predicate P_f will be computable by a population protocol if and only if the set of its true points—which are also the fixed points of f—is semilinear. However, the set of fixed points $\{(k - \lfloor \lg k \rfloor, \lfloor \lg k \rfloor) | k = 1, 2, \ldots\}$ of f is not semilinear. □

4 Conclusions and Future Work

Starting with the class of self-similar algorithms defined by Chandy et al., we studied functions from multisets to multisets computed by such algorithms over a (finite) input alphabet. We showed how the definition of self-similarity of algorithms—a group of any size behaves identically—results in a self-similar additive relationship among the contours of the functions computed by such algorithms. We defined self-similar predicates under the consensus convention used by Angluin et al., and showed that all such predicates that are not eventually constant have a simple structure: the set of points on which they are true or the set of points on which they are false has a standard basis. Using known results about population protocols, we thus showed that nonconstant self-similar predicates are computable by population protocols. We also showed that the notion of self-similarity is more general than, though quite similar to, the notion of opportunistic computability inherent in the population protocol model by showing the existence of a self-similar function not computable by population protocols. Our results, alongwith other examples, are summarized in Figure 2.

Both models discussed in this paper are motivated by distributed computation in dynamic mobile sensor network-like environments. However, neither model attempts to capture in sufficient detail the spatio-temporal nature of the data and its impact on communication and computation. Thus, one cannot frame questions that involve the spatial distribution of data or constraints on communication in the context of these

models. If the state space of the population protocol model is endowed with a topology reflecting the space in which the network exists, then such questions may perhaps be framed. The population protocol model is intended to model a large number of frugal sensors. This may not be appropriate for AUV networks where the number of AUVs is small and each AUV is equipped with sufficient resources. While other models may allow us to ask these questions, we believe that a unified approach to studying such networks may be necessary.

Acknowledgments

We thank Prof. Michel Charpentier for reading an early version of this manuscript, the anonymous referees for their comments, and the Office of Naval Research for supporting this work.

References

1. Spyropoulos, T., Psounis, K., Raghavendra, C.: Efficient routing in intermittently connected mobile networks: The single-copy case. IEEE/ACM Trans. on Networking 16(1) (2008)
2. Grossglauser, M., Tse, D.: Mobility increases the capacity of adhoc wireless networks. IEEE/ACM Transactions on Networking 10(4), 477–486 (2002)
3. Chatzigiannakis, I.: Design and Analysis of Distributed Algorithms for Basic Communication in Ad-hoc Mobile Networks. Computer science and engineering, Dept. of Computer Engineering and Informatics, University of Patras (2003)
4. Gnawali, O., Greenstein, B., Jang, K.Y., Joki, A., Paek, J., Vieira, M., Estrin, D., Govindan, R., Kohler, E.: The TENET Architecture for Tiered Sensor Networks. In: ACM Conference on Embedded Networked Sensor Systems (Sensys), Boulder, Colorado (2006)
5. Palchaudhari, S., Wagner, R., Baraniuk, R.G., Johnson, D.B.: COMPASS: An adaptive sensor network architecture for multi-scale communication. IEEE Wireless Communications (submitted, 2008)
6. Giridhar, A., Kumar, P.R.: Computing and communicating functions over sensor networks. IEEE Journal on Selected Areas in Communications 23(4), 755–764 (2005)
7. Giridhar, A., Kumar, P.R.: Towards a theory of in-network computation in wireless sensor networks. IEEE Communications Magazine 44(4), 98–107 (2006)
8. Chandy, K.M., Charpentier, M.: Self-similar algorithms for dynamic distributed systems. In: 27th International Conference on Distributed Computing Systems (ICDCS 2007) (2007)
9. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing 18(4), 235–253 (2006)
10. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing 20(4), 279–304 (2007)
11. Aspnes, J., Ruppert, E.: An introduction to population protocols. Bulletin of the European Association for Theoretical Computer Science 93, 98–117 (2007)
12. Bhatia, S., Bartoš, R.: Self-similar functions and population protocols: a characterization and a comparison. Technical Report UNH-TR-08-01, University of New Hampshire (2008)
13. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)