Research Statement

Rati Gelashvili

Introduction

Distributed algorithms govern some of the most complex systems in the world, from data center clusters to multicore computing devices to biological systems. One of the key obstacles that distributed systems face is *asynchrony*: the nodes in distributed systems compute independently of each other, often at different speeds or even with a possibility of crashing. In particular, the nodes cannot rely on a perfect shared source of time (a "clock in the sky"), as is the case in synchronous computation.

Dealing with asynchrony in distributed algorithms introduces unique technical challenges, which are often ignored by synchronous distributed models and high-level programming languages and frameworks. These abstractions rely on clever *synchronization algorithms* to mask the underlying asynchrony. The existence and efficiency of these synchronization algorithms is hence of profound importance and guides system design decisions. For instance, in multicore architectures, the set of instructions that are supported *atomically* (i.e. ones that take effect as a single indivisible step) in hardware is chosen to facilitate efficient synchronization.

Broadly speaking, my research is centered on the complexity of synchronization. It involves both understanding the power of *synchronization instructions* and determining the optimal complexity of important *synchronization tasks* in given distributed environments. I do this by designing algorithms and proving lower bounds. My work requires diverse mathematical tools ranging from probability theory and graph theory to logic and combinatorial topology.

I will first describe the space hierarchy that we developed to quantify power of synchronization instructions and new space lower bounds that we proved for important synchronization tasks. Then, I will outline results in the biologically inspired population protocols model and list some other contributions including a time complexity upper bound of leader election and a result related to neuroscience. I will conclude by discussing future directions.

Distributed computing is a mature field with long standing questions, but also emerging areas. My work so far makes considerable headway on long standing open questions such as space complexity bounds, the space hierarchy, and the time complexity of leader election, and also in emerging areas such as population protocols for biological modeling and computational neuroscience.

Preliminaries: In the standard asynchronous shared memory model, which captures the computation in multicore systems, nodes are *processors*. Communication takes place using shared *objects* and *registers*. An object is a memory location to which processors can apply a fixed set of synchronization operations and a register is an object that supports only read and write operations. If an algorithm that a processor is executing is *wait-free*, then the processor will return after performing a finite number of operations. If the algorithm is *lock-free*, then one of the processors executing it will return after performing finitely many operations. If the algorithm is *obstruction-free*, then the processor will return after performing some finite number of operations while no other processors apply any operations. In the consensus task, processors receive private inputs from $\{0, 1\}$ and processors that do not crash must agree on one of the input values.

Contributions

The Space Hierarchy

A fundamental question in multicore systems is to characterize the relative power of synchronization instructions that processors apply to shared memory locations. For more than 25 years, the best scholarly

explanation has been Herlihy's consensus hierarchy [25]. Herlihy's hierarchy classifies an object by its *consensus number*, the maximum number of processors for which it is possible to solve consensus in a wait-free way using instances of this object and registers. Consensus is universal: the ability to solve consensus among a set of processors allows any (sequentially specified) object shared by those processors to be implemented.

In [8], in collaboration with Faith Ellen, Nir Shavit and Leqi Zhu, I showed that Herlihy's hierarchy is too simplistic a model for existing computer architectures. We showed that there are operations that have consensus number 1 if applied to separate objects, i.e. they are at the lowest level in the hierarchy, yet can be combined to solve consensus among any number of processors if both operations are applied to the same object. This "operation" based modeling is more realistic since any instruction that is supported in hardware can be applied to any of a computer's memory locations. Our work has shown that, contrary to common belief, shared-memory computability does not require multicore architectures to support synchronization instructions like *compare-and-swap*, whose corresponding object sits at the top of Herlihy's hierarchy. Instead, from a pure computability standpoint, combinations of instructions like *decrement* and *multiply*, whose corresponding objects are at the bottom of Herlihy's hierarchy, will suffice.

In [8], we not only showed the limitations of Herlihy's hierarchy, but also proposed an alternative new hierarchy based on of sets of synchronization instructions (as opposed to objects) and classified by the minimum number of memory locations required by such sets to solve consensus in an obstruction-free way. Obstruction freedom is a natural progress measure used in state-of-the-art synchronization constructs, e.g. hardware transactions [27] do not guarantee more than obstruction freedom. The key to our proof is is proving tight space upper and lower bounds. The classification resulting from our hierarchy seems to fit with one's intuition about how useful some instructions are in practice, while questioning the effectiveness of others.

Most efficient concurrent data-structures rely heavily on *compare-and-swap* for swinging pointers, and in general, for conflict resolution. In [11], together with Idit Keidar, Alexander Spiegelman and Roger Wattenhofer, I provided an alternative by designing a Log data-structure that can be used in a lock-free universal construction of [25] to implement any concurrent object from its sequential specification. A Log supports two operations: a lock-free *append*(*item*), which appends the item to the log, and a wait-free *get-log*(), which returns the appended items so far, in order. Our implementation used atomic *read*, *xor*, *decrement*, and *fetch-and-increment* instructions (whose corresponding objects have consensus numbers one and two) supported on X86 architectures, and provides similar performance to a *compare-and-swap*-based solution. Our work raised a fundamental question about the minimal sets of synchronization instructions that shared-memory multicore architectures must support to allow fully expressive computability.

Space Lower Bounds

As a part of developing the space hierarchy, one must understand how many memory locations (traditionally called "registers") are required to solve consensus in an obstruction-free way. This was a long-standing open question. It's possible to solve consensus for n processors using n registers in an obstruction-free way or in a randomized wait-free way. In randomized wait-free protocols non-faulty processors are required to terminate with probability 1. A lower bound of $\Omega(\sqrt{n})$ by Ellen et al. [21] dates back to 1993.

In [10], I proved an asymptotically tight $\Omega(n)$ lower bound for *anonymous* processors, which was the first progress on the problem since [21]. Anonymous processors have no identifiers and run the same code: all processors with the same input start in the same initial state and behave identically until they read different values. On a technical side, [10] introduced new ideas that utilized the anonimity of processors for proving space lower bounds.

This was followed by Zhu's proof that any obstruction-free protocol solving consensus for n processors requires at least $n-1$ registers [33]. In our hierarchy paper [8], we generalized read and write instructions to a family of buffered read and buffered write instructions $\mathcal{B}_\ell$ for $\ell \geq 1$ and extended the techniques of [33] to prove that $\lceil \frac{n-1}{\ell} \rceil$ memory locations supporting $\mathcal{B}_\ell$ are necessary. We showed that this is tight when $n-1$

is not divisible by ℓ. Moreover, by way of an interesting combinatorial argument, we proved that the lower bound holds within a factor of 2 even in the presence of atomic multiple assignment. Multiple assignment can be implemented by simple transactions, so our results imply that such transactions cannot significantly reduce space complexity.

In my most recent work with Faith Ellen and Leqi Zhu [9], we developed a novel method for proving space lower bounds, enabling us to obtain strong results for classical synchronization tasks, for which previously no good space lower bounds were known. In particular, in the k-set agreement problem, n processors, each with an input value, are required to return at most k different input values. This is a generalization of consensus, which is the case $k = 1$. The best known upper bound on the number of registers needed to solve k-set agreement among $n > k$ processors in an obstruction-free way is $n - k + 1$ registers, while no general lower bound better than 2 was known. We proved that processors must use at least $\lfloor \frac{n-1}{k} \rfloor + 1$ registers. In particular, this gives a tight lower bound of n for consensus.

Our main technical contribution was a simulation that serves as a reduction from the impossibility of deterministic wait-free k-set agreement [26, 31], a celebrated result in distributed computing which relies on a connection to combinatorial topology. Our simulation converted any obstruction-free protocol for k-set agreement that uses too few registers to a protocol that solves wait-free k-set agreement, which is impossible. We also used this new technique to prove a space lower bound of $\lfloor \frac{n}{2} \rfloor + 1$ for ϵ-approximate agreement for sufficiently small ϵ, an important task that requires participating processors to return values within ϵ of each other. This is within a factor of 2 of the best known known upper bound, while previously no general lower bounds were known. A critical component of the simulation is the ability of simulating processors to revise the past of simulated processors. We introduced a new *augmented snapshot object*, which facilitates this.

Obstruction freedom and randomized wait freedom are known to be closely related [24, 22]. In [9], we also showed that any space lower bound on the number of registers used by obstruction-free protocols applies to randomized wait-free protocols. Hence, our result for k-set agreement implies a tight lower bound of exactly n registers for obstruction-free and randomized wait-free consensus.

Population Protocols

Population protocols [13] are a model of distributed computing consisting of n nodes which have little computational power and interact randomly. They were originally introduced to model animal populations equipped with sensors [13], they have proved a useful abstraction for settings from wireless sensor networks [30, 20], to gene regulatory networks [15], and chemical reaction networks [17]. There is an intriguing line of applied research showing that population protocols can be implemented at the level of DNA molecules [18], and that some natural protocols are equivalent to the computational tasks solved by living cells in order to function correctly [16].

In the population protocol model, majority is a central task: each node starts in state A or B and the goal is to collectively determine whether more nodes were initially in state A or B. Another key task is leader election, which requires the system to stabilize to final configurations with exactly one node is in a special *leader* state. Two complexity metrics are important: the time that a protocol requires to stabilize to an output decision, and the size of the state space that each node requires to do so. The applications of the model dictate that we would like protocols to use as few states as possible and be *fast*, i.e. stabilize in time polylogarithmic in n.

Until a few years ago, the best known protocols for majority [20, 29] used only 4 states, but required a superlinear time to stabilize when the initial discrepancy between the number of nodes in states A and B was small. For leader election, the existence of a fast protocol with constant number of states was an open question [14].

In [6], with Dan Alistarh and Milan Vojnovic, I designed the first fast protocol for majority. In [4], with Dan Alistarh, I also designed a fast leader election protocol using $O(\log^3 n)$ states. At the same time, Doty and

Soloveichik [19] showed that fast leader election requires a superconstant number of states.

In [2], in collaboration with Dan Alistarh, James Aspnes, David Eisenstat and Ronald Rivest, I pushed these results further. We gave fast algorithms for both majority and leader election using $O(\log^2 n)$ states, based on a novel *synthetic coin flipping* technique. The nodes are deterministic, but synthetic coin flipping allows them to generate almost-uniform local coins within a constant number of interactions, by exploiting the randomness in the scheduler. Proving this requires analyzing a random walk on the hypercube. This technique is generally useful and has since been used in other contexts. In [2], we proved a lower bound of $\Omega(\log \log n)$ states for fast protocols for both majority and leader election, extending the result of [19].

Finally, in [3], with Dan Alistarh and James Aspnes, I devised a new majority protocol that uses $O(\log n)$ states, and stabilizes in $O(\log^2 n)$ expected time. Central to the protocol is a new *leaderless phase clock* technique, which allows agents to synchronize in *phases* of $\Theta(n \log n)$ consecutive interactions. It exploits a new connection between population protocols and load balancing mechanisms. On the negative side, we provided a new lower bound of $\Omega(\log n)$ states for any majority protocol which stabilizes in $O(n^{1-c})$ expected time, for any constant $c > 0$, conditional on a couple of assumptions, satisfied by all known protocols.

In [3], we also employed our phase clock to build a fast leader election algorithm with a state space of size $O(\log n)$. Gąsieniec and Stachowiak [23] independently designed a leader election protocol using $O(\log \log n)$ states. This is optimal by our lower bound [2]. Combined, our results and [23] demonstrate a separation between the state complexities of majority and leader election.

Other Contributions

In [5], we designed a leader election protocol in asynchronous message-passing system with $O(\log^\star k)$ time complexity for k participants and proved that the message complexity of our protocol is optimal. This is a progress on a long-standing open problem of determining the optimal time complexity of leader election in asynchronous distributed systems. The best previously known protocol relied on a tournament-like structure and had time complexity of $\Theta(\log k)$.

In [1], with Dan Alistarh, James Aspnes, Michael Bender and Seth Gilbert, we provided the first asynchronous shared-memory algorithm for the dynamic version of the classical task allocation problem, where a set of potentially faulty nodes must cooperate to perform a set of tasks. We used competitive analysis to show that the total amount of work performed by nodes is a polylogarithmic factor away from optimal on any given sequence of inputs.

I am also interested in problems beyond the standard distributed models. The human brain is a prime example of a distributed system that allows for rich computation to take place. Many models of neural computation exist [28, 32]. In my work [7] with Zeyuan Allen-Zhu, Silvio Micali and Nir Shavit, we considered a theoretical construction that neuroscientists had suggested as playing a role in information compression in the brain. We showed how to make this construction neurobiologically plausible.

I have also amassed significant practical experience through industrial internships at Google, Facebook, Akamai, Dropbox and the D.E. Shaw Group. My most notable achievements were at Akamai and Dropbox, where I contributed to complex projects that have had large practical impact. At Akamai, I designed and implemented a high-performance concurrent data structure in the Linux Kernel, that relies on fine-grained synchronization (i.e. the read-copy-update mechanism) and efficiently serves a huge number of requests. At Dropbox, I developed automatic targeted throttling of requests for a massive scale distributed database service, powering all of Dropbox's products.

Future Research Directions

Concrete Problems: I would like to resolve a number of important and perplexing problems motivated by the space hierarchy [8]: extending our hierarchy to involve time complexity, establishing the exact power of certain conventional instructions, and determining whether there is a complexity-based version of universality, i.e. whether it is possible to implement any (sequentially specified) object using the same amount of space (to within a constant factor) as is used for solving consensus. We expect that new methods will likely be needed to answer at least some of these questions.

The simulation technique introduced in [9] is a completely novel approach. It is conceivable that it can be further extended to apply to other tasks or prove a tight lower bound for k-set agreement.

There are a number of interesting questions in population protocols. This includes getting rid of certain assumptions in our majority lower bound [3], applying our techniques to different tasks, and considering protocols that allow small probability of error or are robust to certain chemical phenomena called *leaks* [12].

General Directions: The topological framework for analyzing distributed algorithms currently has two major limitations: it is not known how to represent algorithms that use randomization or multi-writer registers. I would like to extend the state-of-the-art to overcome these limitations. This is a substantial undertaking, but we may have a way to attack the multi-writer register case. Our intuition that led to the results in [9] stemmed from the connection between shared-memory computability and combinatorial topology. Understanding and properly formalizing what the simulation means in terms of the topological framework is a great starting point.

I also intend to explore intersections with other fields, including more systems research and applications of my expertise in other fields, such as optimization and machine learning, which are becoming increasingly large-scale and thus, reliant on distributed platforms.

Finally, I am currently working on some problems related to neural network architectures.

My Publications

[1] Dan Alistarh, James Aspnes, Michael A Bender, Rati Gelashvili, and Seth Gilbert. Dynamic task allocation in asynchronous shared memory. In *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms*, (SODA), pages 416–435, 2014.

[2] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms*, (SODA), pages 2560–2579, 2017.

[3] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms*, (SODA), 2018.

[4] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming*, (ICALP), pages 479–491, 2015.

[5] Dan Alistarh, Rati Gelashvili, and Adrian Vladu. How to elect a leader faster than a tournament. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing*, (PODC), pages 365–374, 2015.

[6] Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 34th ACM Symposium on Principles of Distributed Computing*, (PODC), pages 47–56, 2015.

[7] Zeyuan Allen-Zhu, Rati Gelashvili, Silvio Micali, and Nir Shavit. Sparse sign-consistent johnson–lindenstrauss matrices: Compression with neuroscience-based constraints. *Proceedings of the National Academy of Sciences (PNAS)*, 111(47):16872–16876, 2014.

[8] Faith Ellen, Rati Gelashvili, Nir Shavit, and Leqi Zhu. A complexity-based hierarchy for multiprocessor synchronization:[extended abstract]. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing*, (PODC), pages 289–298, 2016.

[9] Faith Ellen, Rati Gelashvili, and Leqi Zhu. Revisionist simulations: A new approach to proving space lower bounds. arXiv preprint arXiv:1711.02455. Submitted to STOC '18.

[10] Rati Gelashvili. On the optimal space complexity of consensus for anonymous processes. In *Proceedings of the 29th International Symposium on Distributed Computing*, (DISC), pages 452–466, 2015. **Best Paper Award**.

[11] Rati Gelashvili, Idit Keidar, Alexander Spiegelman, and Roger Wattenhofer. Brief announcement: Towards reduced instruction sets for synchronization. In *Proceedings of the 31th International Symposium on Distributed Computing*, (DISC), pages 53:1–53:4, 2017.

Related Work

[12] Dan Alistarh, Bartłomiej Dudek, Adrian Kosowski, David Soloveichik, and Przemysław Uznański. Robust detection in leak-prone population protocols. In *Proceedings of the 23rd International Conference on DNA Computing and Molecular Programming*, (DNA), pages 155–171, 2017.

[13] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, March 2006.

[14] Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008.

[15] James M Bower and Hamid Bolouri. *Computational modeling of genetic and biochemical networks*. MIT press, 2004.

[16] Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Nature Scientific Reports*, 2:656:1–656:9, September 2012.

[17] Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, 2015.

[18] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srnivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature Nanotechnology*, 8(10):755–762, September 2013.

[19] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. In *Proceedings of the 29th International Symposium on Distributed Computing*, (DISC), pages 602–616, 2015.

[20] Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, May 2012.

[21] Faith Fich, Maurice Herlihy, and Nir Shavit. On the space complexity of randomized synchronization. In *Proceedings of the 12th ACM Symposium on Principles of Distributed Computing*, (PODC), pages 241–249, 1993.

[22] Faith Ellen Fich, Victor Luchangco, Mark Moir, and Nir Shavit. Obstruction-free algorithms can be practically wait-free. In *Proceedings of the 19th International Conference on Distributed Computing*, (DISC), pages 78–92, 2005.

[23] Leszek Gąsieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms*, (SODA), 2018.

[24] George Giakkoupis, Maryam Helmi, Lisa Higham, and Philipp Woelfel. An $\mathcal{O}(\sqrt{n})$ space bound for obstruction-free leader election. In *Proceedings of the 27th International Symposium on Distributed Computing*, (DISC), pages 46–60, 2013.

[25] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, January 1991.

[26] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of The ACM (JACM)*, 46(6):858–923, November 1999.

[27] Intel. Transactional synchronization in Haswell. `http://software.intel.com/en-us/blogs/2012/02/07/transactional-synchronization-in-haswell`, 2012. Manual.

[28] Wolfgang Maass. On the computational power of noisy spiking neurons. In *Proceedings of the Advances in Neural Information Processing Systems*, (NIPS), pages 211–217, 1996.

[29] George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*, (ICALP), pages 871–882, 2014.

[30] Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *Proceedings of the 28th IEEE Conference on Computer Communications*, (INFOCOM), pages 2527–2535, 2009.

[31] Michael Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing*, 29(5):1449–1483, 2000.

[32] Leslie G Valiant. *Circuits of the Mind*. Oxford University Press, 2000.

[33] Leqi Zhu. A tight space bound for consensus. In *Proceedings of the 48th ACM Symposium on Theory of Computing*, (STOC), pages 345–350, 2016.