Computational Requirements for Nano-Machines: There is Limited Space at the Bottom

Florian-Lennert Adrian Lau
lau@itm.uni-luebeck.de
Institute of Telematics
Ratzeburger Allee 160
Lübeck, Germany 23562

Florian Büther
buether@itm.uni-luebeck.de
Institute of Telematics
Ratzeburger Allee 160
Lübeck, Germany 23562

Bennet Gerlach
bgerlach@itm.uni-luebeck.de
Institute of Telematics
Ratzeburger Allee 160
Lübeck, Germany 23562

ABSTRACT

Akyildiz et al. envisioned the use of nanonetworks as a new paradigm for computation on a very small scale. Since then, many scientists investigated dependent aspects like nanoscale communication. However, most research omitted specifying the complexity required for their respective scenarios. To close this gap, we analyzed numerous medical scenarios and extracted the formal problems to be solved. We then compared the resulting formal problems using computational complexity theory and displayed them sorted into the classes AC^0, NC^1 and L. Lastly, we describe the benefits of our results for simulation purposes and to better assess the feasibility of nanonetwork scenarios.

KEYWORDS

nanonetworks; computational complexity; nano-machines; space-complexity

ACM Reference format:
Florian-Lennert Adrian Lau, Florian Büther, and Bennet Gerlach. 2017. Computational Requirements for Nano-Machines: There is Limited Space at the Bottom . In *Proceedings of NANOCOM '17, Washington D.C., DC, USA, September 27-29, 2017,* 6 pages.
DOI: 10.1145/3109453.3109458

1 INTRODUCTION

In 2008 Akyldiz et al. proposed nanonetworks as a new paradigm for computation at the nanoscale [1]. A nanonetwork consists of numerous nano-machines of an overall size of about 1 to a few thousand nanometers. The individual abilities of such a machine are to some degree indefinite but often mentioned capabilities are: communication, sensing/acting and computational power. However, an important open question is, if and how the aforementioned capabilities can be implemented, since the minuscule size of individual machines restricts the scope of functions to be performed. Yet, many scientists from different fields have proposed to utilize nano-machines to help combat or solve open problems. Medical scenarios are among the most prominent areas of application, but other disciplines— like physics, chemistry or computer science—consider the use of nano-machines to assist in certain tasks. A prominent scenario is the detection of certain markers that indicate a disease [5, 23]. Unfortunately—for many scenarios—it remains unclear how complex a single nano-machine has to be in terms of computational power. Most scenarios establish the operations to be performed by nano-machines, for instance by giving the formulas to be computed, but don't specify the computational complexity of a single nano-machine. Sometimes they even overestimate it, as we argue below. Furthermore, the consequences such capabilities have on the realization and implementation of a nano-machine are often not discussed. This leads to the question of how capable or complex nano-machines at least have to be to fulfill the presupposed computational requirements.

A smaller or less complex nano-machine might render an actual realization easier. Firstly, basic building blocks have already been realized, as opposed to more complex constructions [9, 19]. Secondly, combining huge amounts of building blocks—the way today's CPUs are designed—to form more complex constructions, thus enabling considerable computational power might not be feasible at the nanoscale. Current CPUs, while trying to minimize their size, with transistors only a few nanometers big, still form large constructs in the size of centimeters.

Thus, to evaluate the feasibility of a given scenario, the question arises what the minimal computational requirements of a single nano-machine are in order for it to be useful to that scenario. In other words, what is the lower bound on the computational capability of a single nano-machine to be able to perform all operations necessary to the scenario? A nano-machine with less capability might be simpler to realize, but would be unfit for the task.

Much research already investigated the physical environment that nano-machines will have to deal with. Starting from communication technologies like electro-magnetic waves or molecular channels [1] and their properties, to possible active movement, for example by chemotaxis [11], or models for computation like Quantum-Dot Automata [19]. The general capabilities of single nano-machines are often described omitting precise computational ability. The presupposed capabilities of single nano-machines differ greatly and range from nanoparticles completely without computational capability to features comparable to today's microprocessors [11], for electronic as well as biological implementations. We thus try to provide a general analysis that applies to both types of machines.

To the best of our knowledge, no attempt to better specify the computational complexity of single nano-machines has been made so far and we aim to close this gap. Thus, we give a framework

NANOCOM '17, Washington D.C., DC, USA
© 2017 ACM. 978-1-4503-4931-4/17/09…$15.00
DOI: 10.1145/3109453.3109458

to specify the computational capabilities of single nano-machines more precisely. Thereby, we may assist future research by suggesting minimum requirements for certain scenarios for nano-machines and thus providing key-information on their realizability. Additionally, the framework we provide enables a better understanding of the capabilities of individual nano-machines, particularly useful when simulating them.

We achieve this by first analyzing numerous, mostly medical scenarios and identifying common tasks for nano-machines. We then extract the resulting—more formal—problems corresponding to those tasks. Using computational complexity theory, we compare and sort them into three groups of increasing capability. Finally, we explain how these groups actualize nano-machine application guidelines.

2 PROPOSED TASKS FOR NANO-MACHINES

We determine a reasonable computational model for individual nano-machines from common application scenarios. We employ nano-medicine and communication resesearch as representatives. We expect similar requirements from other settings.

Nanotechnology is constantly applied to oncology—the treatment of cancer [22]. While nowadays mostly nanoparticles are used to fight cancer, more elaborate approaches like detection and destruction of cancer cells by engineered bacteria or nano-machines emerge in the scientific community.

This approach may be generalized to the detection and treatment of diseases in the human body with—or with the help of—nano-machines. The already conducted research in this area includes possible methods of treatment for diabetes [15], the detection and localised treatment of inflammation diseases [5, 23, 24], research that focuses on the collaborate work of nano-machines [1], as well as more advanced experiments concerning detection and treatment aided by nanotechnology [9].

More complex scenarios propose a molecular automaton capable of detecting several mandatory markers for a special disease that, once all markers are present, starts a DNA-based manufactoring process simulating drug release [6].

In [14], the idea of disease detection is further generalized. Other more futuristic applications of nano-machines—like intracellular surgery—have also been envisioned. Such techniques appear to require high precision and local organization of chosen actuators and sensors with perfect timing [13].

Another popular approach is the support of the human immune system. In this scenario, nano-machines are designed to assist in tasks for which the human self-healing capabilities won't suffice. A general enhancement or the replacement of a receding body function are also plausible [1, 13, 15].

One of the most common scenarios in the nano context is health monitoring [2]. Since continuous surveillance by a physician is infeasible and regular testing is time consuming, a method that enables better feedback about the personal health status is of public interest [13].

Most of the above mentioned approaches require certain amounts of computational power or logical operations to produce a useful decision [9]. Due to the importance of communication in nanonetworks, routing and therefore addressing are also of interest. Many papers focus on advanced communication approaches in nanonetworks. For example, a nanonetwork point-to-point routing mechanism has been proposed in [25]. A more complex forwarding mechanism requires additional arithmetics, as well as a sufficient amount of data [8, 28].

Problem	Signature	Description
ADD	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Integer addition
SUB	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Integer subtraction
MULT	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Integer multiplication
DIV	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Integer division
SIGN	$\mathbb{Z} \to \{-1, 0, 1\}$	Signum function
INC	$\mathbb{Z} \to \mathbb{Z}$	Integer increment
AND	$\{0, 1\} \to \{0, 1\}$	Logical AND
OR	$\{0, 1\} \to \{0, 1\}$	Logical OR
ODD, EVEN	$\mathbb{Z} \to \{0, 1\}$	Tests an integer for even-/oddness
DIV_2	$\mathbb{Z} \times \mathbb{B} \to \mathbb{Z}$	Division by power of 2
MOD_2	$\mathbb{Z} \times \mathbb{B} \to \mathbb{Z}$	Modulo with power of 2
INV_2	$\mathbb{Z} \to \mathbb{Z}$	Binary inverse
IT-MULT	$\mathbb{Z} \times \cdots \times \mathbb{Z} \to \mathbb{Z}$	Iterated multiplication
MIN, MAX	$\mathbb{Z} \times \cdots \times \mathbb{Z} \to \mathbb{Z}$	Min/max of inputs
MAJOR_2	$\mathbb{Z} \to \{0, 1\}$	Binary majority
EXP	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Exponentiation
THRES_2	$\mathbb{Z} \times \mathbb{N} \to \{0, 1\}$	Checks for at least n positive bits
IT-ADD	$\mathbb{Z} \times \cdots \times \mathbb{Z} \to \mathbb{Z}$	Iterated addition
REG_a	$\Sigma^* \to \{0, 1\}$	Pattern matching for regex a
PARITY_2	$\mathbb{Z} \to \{0, 1\}$	Binary parity check
MOD	$\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$	Modulo
AVG	$\mathbb{Z} \times \cdots \times \mathbb{Z} \to \mathbb{Z}$	Average of n inputs
DFS, BFS	$(V, E) \times V \to \{0, 1\}$	Depth or breadth first graph search
REACH	$(V, E) \times V \to \{0, 1\}$	Graph Reachability
LOG_2	$\mathbb{Z} \times \mathbb{B} \to \mathbb{Z}$	Binary logarithm of integers
MEDIAN	$\mathbb{Z} \times \cdots \times \mathbb{Z} \to \mathbb{Z}$	Integer median

Table 1: Formally defined problems of interest to nanonetworks. We assume that all natural numbers $\mathbb{N}$ and all integers $\mathbb{Z}$ are represented as binary, in other words as $\{0, 1\}^k, k \in \mathbb{N}^+$. Further, we denote binary numbers to the power of two as $\mathbb{B}$.

3 FROM SCENARIOS TO PROBLEMS

From the given scenarios, we extract a set of *problems* that a nano-machine may need to solve—each problem corresponds to an operation a nano-machine can perform.

Table 1 provides an overview, and this section elaborates all problems and highlights computational peculiarities.

3.1 Arithmetic and Logical Operators

Most scenarios, and in fact most later problems, ubiquitously require basic arithmetic on integers and booleans. This includes basic operations, for example addition ADD, subtraction SUB and multiplication MULT/IT-ADD of two values, as well as logical operators like AND and OR. These are the basis for value aggregation, time-to-life routing schemes and complex conditional computations.

Additionally, values need to be compared, for example for positivity SIGN or even-ness EVEN/ODD, in regard to a threshold THRES, or to find a smallest or biggest value MIN/MAX, or the most prominent value MAJOR. These either guide conditional behavior or support robust and fault-tolerant sensing strategies.

More complex scenarios require computation of division DIV and modulo MOD, as part of average computation AVG, as well as exponentiation EXP/IT-MULT and sometimes logarithms.

With the binary representation, several arithmetic operations are much easier when applied to powers of two, for example division. We listed these explicitly in Table 1 as DIV_2, MOD_2, INV_2 and LOG_2.

3.2 Pattern Matching and Parity

Symbol Pattern Matching REG_a checks if a given string adheres to a pattern, typically given as a regular expression a. In the easiest case, a constant pattern, a nano-machine can check, for example, an specific activator message, whereas more complex patterns may serve for antibody detection or message parsing.

A special case of validity check is the parity check PARITY, which tests if the numbers of 1s in the binary representation of a message is even or odd, which is typically used to verify message integrity.

3.3 Communication

As all scenarios focus on nanonetworks, communication is obviously always a necessary requirement. However, proposed forwarding and routing protocols pose very different requirements.

The protocol as given in [25] performs directed routing based on 2-dimensional positioning information, and only requires basic integer addition and comparison.

The complex forwarding mechanism given in [28] requires division, square roots, logarithms and involves a set of environmental information that need to be gathered and stored; it in turn yields a very efficient hop selection mechanism.

In general, nano-machines may need to be able to distinguish and handle multiple kinds of infrastructural messages for network establishment and maintenance. Implementations may require pattern matching to detect address membership, conditional processing depending on message type and storage to handle routing information.

3.4 Complex Operations

In addition to the above problems, very complex functionality has been proposed, for example artifical neural networks [23].

As networks can be represented as directed or undirected graphs, graph algorithms are naturally of interest. These include search, either by depth first DFS or breadth first search BFS, for example to test for the existence of deviating sensor readings, and reachability REACH to monitor the overall nanonetwork health.

3.5 Storage

Next to the required processing power, the amount of memory is of interest as well. All but the simplest nano-machines need to store their current state or their current location, if a drug threshold has been reached or how often to measure a marker concentration.

Some algorithms need a varying amount of space. An important example is nano-machine addressing: If each node of the network graph should be addressed, each node needs to store its own label, which in turn requires logarithmic space "Label" relative to the graph size. As labels are required for graph traversal, the search algorithms BFS and DFS, as well as REACH also require at least that much space. Furthermore, routing algorithms that maintain routing tables will need to store several node labels.

To compute an average of a set of values AVG, a single nano-machine needs to provide storage to process all input values, or if performed in a distributed fashion, several nano-machines need to store at least two values each. This probably holds true for all statistical algorithms, for example MAX, MIN, THRES and MAJOR.

3.6 Time

Next to memory and computation requirements, nano-machines require additional facilities, a major one being a measure of time. Many algorithms need timing, in other words: Gathering sensor readings at regular intervals, or noting relative occurrence of phenomena require the ability to track progress of time.

For very timing-sensitive or long-term applications, even absolute timestamps may be required, which require network-wide time coordination. This is solved by protocols like the Network Time Protocol (NTP), however, their complexity renders their applicability at the nanoscale questionable.

As an implementation of time is mostly a physical issue of building a nanoscale clock, we do not further cover time in this work.

4 PRELIMINARIES FOR COMPUTATIONAL COMPLEXITY

Table 1 provides an unsorted list of problems likely to occur in nanonetworks. In order to derive requirements for nano-machine design, we now aim to classify their respective computational complexity. For this, we employ *complexity classes*, a fundamental concept of complexity theory. We will give a short introduction to the theory, and refer to [20] for a detailed discussion.

Put simply, a complexity class contains problems that can be solved by a specified machine model. For example, the complexity class L describes problems that a Turing-machine can solve, while using only a read/write tape of logarithmic length with respect to the length of the input. Consequently, a nano-machine that is at least as powerful as a certain machine model can safely be expected to solve all problems contained in the respective complexity class.

Next to L, we investigate two classes of importance to nano-machines. Much research assumes boolean circuit-like systems for nano-machines [18]. These circuits, consisting of INV, OR and AND gates, form exactly the basis for the complexity classes AC^0 and NC^1, among others [20]. Boolean circuits described by AC^0 are further constrained to be polynomial in size and constant in depth with respect to the number of input gates. The gates of NC^1

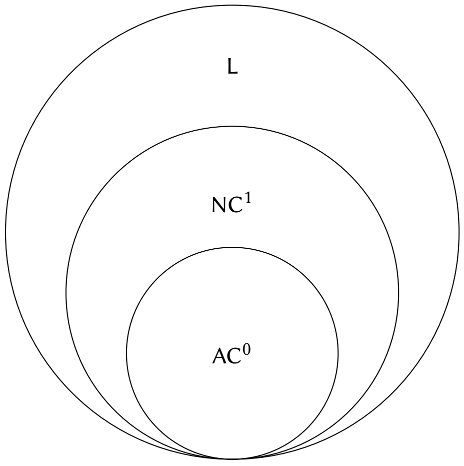

Figure 1: The relationship between the complexity classes L, AC^0 and NC^1.

are allowed to have at most two inputs per gate, but may be of logarithmic depth.

Complexity classes can contain one another. For example, the class AC^0 is contained in class NC^{1*}, which is in turn contained in L, as shown in Figure 1. Consequently, problems often lie in multiple classes: For example, since the problem LOG$_2$ is solvable in AC^0, it is also solvable in NC^1, as well as in L. Thus, if a nano-machine is at least as powerful as a Turing-machine with a logarithmic read/write tape, it can be expected to solve LOG$_2$.

To sort a problem or a nano-machine into the complexity hierarchy, complexity theory employs *reductions*. A reduction proves that a problem A is at most as difficult as a problem B. A reduces to B if the following holds: An instance i_A of problem A can be transformed with given effort to an instance i_B of problem B. Instance i_B is then solved, yielding solution s_B. Lastly, the solution s_B can be transformed with further effort to a solution s_A of problem A. For example, the problem SUB can be shown to be in the same class as ADD, namely class AC^0, by reducing it to ADD. To do so, we alter the input to ADD by changing the sign of its second input number via a negation gate. The altered circuit still has constant depth and only uses gates available to AC^0. The total effort of the altered circuit can still be performed in AC^0, making it a valid reduction. It is important to note that the efforts of both transformations need to be restricted to the capabilities of the class of problem A, as otherwise the problem A could be more easily solved by the transformation itself.

5 FROM PROBLEMS TO COMPLEXITY CLASSES

With the help of reductions, the nano-machine problems can be positioned in relation to one another in the complexity hierarchy. Table 2 notes, which problems can be computed by the machine models of the introduced complexity classes AC^0, NC^1 and L. As the classes possess an inclusion relationship as shown in Figure 1,

*The inclusion is intuitive, since AC^0-gates with many inputs can be constructed in NC^1 as a tree of two-input gates with logarithmic depth.

	Problems from scenarios	Additional problems of interest
AC^0-machines:	ADD [10, 20]	ODD/EVEN
	SUB	DIV$_2$
	SIGN	MOD$_2$
	INC	INV
	AND/OR	LOG$_2$
NC^1-machine:	MULT [10, 20]	MIN/MAX
	DIV [10]	PARITY [20]
	EXP [16]	REG [26]
	MAJOR [20]	MOD
	THRES [20]	IT-MULT [16]
	IT-ADD [10, 20]	
	AVG	
L-machine:	Label	DFS [12]
	Log mem	BFS [12]
		REACH [21, 27]
		MEDIAN
Misc.:	*Addressing*	
	Routing	
	Broadcasting	
	Forwarding	

Table 2: List of problems by nano-machine-type. A lower listed bot is capable of performing all of the above tasks. Italic items are of unclear space-complexity. We categorized the displayed problems to the best of our knowledge.

problems of enveloped classes can also be computed by the enveloping classes. Furthermore, this classification is not necessary final, as problems might move to a lower class if a suitable reduction can be found.

Starting from a set of problems for which classification results have already been published, we present short sketches for a possible reduction required for the categorization of the remaining problems:

- INV: The negation-gate is an atomic component in AC^0-circuits and obviously computable.
- AND/OR: Just as the negation-gate, AND/OR-gates are atomic components of all introduced circuit-classes and thereby computable.
- SIGN: Sign may be computed by AC^0 through inversion of a predefined sign-bit.
- SUB: The problem of subtracting two integers is computable in AC^0 by first inverting the SIGN of one of the integers followed by an addition.
- INC: INC is the addition of 1 to an integer and a special case of ADD.
- ODD/EVEN: Both problems are solvable by checking the least significant digits of a binary input.

- DIV$_2$: The division of a binary number by $k \in \mathbb{B}$ is realized by cutting the last k bits from the input.
- MOD$_2$: The binary modulo by a module $k \in \mathbb{B}$ is computed by yielding the k least significant bits of the input.
- LOG$_2$: The logarithm of a binary number is the index of the most significant 1.
- AVG: The average of n binary inputs may be computed by adding all inputs via IT-ADD followed by a DIV by n.
- MOD: n mod m may be computed by calculating SUB(n,MULT(DIV(n, m), m)).
- MEDIAN: The median may be computed by testing each number of the input to all others and counting the bigger/smaller ones.

5.1 A Problem Compendium

The presented computational models/classes are sometimes capable of calculating algorithms for additional problems that haven't been deduced from the medical nanonetworking-scenarios, but might be of future interest[†]. Table 2, column 2 displays additional algorithms that are used in wireless sensor networks.

Moreover, the following problems may also be computed by an L-machine, but are of unclear interest to nanonetworks:

- REACH/PATH/CONNECTIVITY [21, 27].
- TREE-ISOMORPHISM [17].
- BIPATIT-test[4].
- PLANARITY-test [3].
- Parametrized TREE-WIDTH.

5.2 Problems of Unknown Complexity

The "Label" and "Log mem" entries aren't problems from a computational complexity point of view. They rather reflect the capability of a Turing machine with a logarithmic space-constrained read/write-tape to hold certain amounts of information.

The italic entries in Table 2 are of unknown complexity. The system properties for nanonetworks are too vague to stipulate a context from which explicit attributes may be concluded. However, they are of special interest for digital communication processes that might occur in nanonetworks. Most of the algorithms required in communication processes or protocols may be computed by machines capable of computing problems from the previously introduced complexity classes. The physical implementation however, remains unclear. Moreover, some problems may require a more powerful machine-type or global knowledge about the network.

- Addressing: Addressing in general requires each individual nano-machine to be at least able to hold a unique identificator.
- Routing: Routing often requires a-priori information about the underlying network structure. Since mobile components might be desirable, a routing-process might be complex and is topic of current research.
- Broadcasting & Forwarding: Both operations may be part of a routing procedure and primarily describe behavior not provided by circuits or Turing-machines.

[†]See Table 1 for a small definition for the problems.

6 APPLICATIONS TO NANO-MACHINE DESIGN

The classification of the problems relevant to nanoscale applications yields important insights into the feasibility of nano-machine application scenarios, as well as guidelines for assumptions about nano-machine capabilities. Before we discuss these further, we need to highlight that our conclusions only account for the computational power of the chosen machine instance. Other open problems like communication, input/output-handling or storage are not covered. Furthermore, restrictions may be lifted by future novel innovations in the area of nanoscale computation.

For our research, we chose the smallest circuit-classes capable of at least some meaningful computation. We distinguish the classes AC^0 and NC^1, as they are proven to be not equal and have significant differences in size [7]. Moreover, we chose the L-class for the small amount of incorporated memory. The logarithmic amount is enough to uniquely identify a nano-machine in a network and might be of special importance.

These classifications allow more fine-grained specification of the exact capabilities of single nano-machines in a nanonetwork simulation. A scientist that is not familiar with this area of research might use Table 2 to better decide upon how "capable" a chosen nano-machine may be and work with a more concrete concept. For example, the problem MULT is much more difficult than all of the problems in AC^0.

To analyze an existing protocol or algorithm, we can apply Table 2 in reverse. First, we need to deassemble the algorithm into its basic operations, for example MULT, ADD, etc.. From these, we pick the operations from the strongest complexity class and count them: This will determine the final complexity class of the algorithm. All remaining operations can be added for comparatively small cost.

If a nano-machine can only perform a small set of operations, it will not neccessarily be small or simple to build. However, a careful selection of operations can help to cope with the size and resource constraints of nanoscale machine construction.

6.1 Circuit-Based Implementations

This section covers both the AC^0 and the NC^1 classes. We chose to closer investigate circuit classes, since a considerable amount of research concerning nanoscale computation is based on circuit-implementations on the basis of quantum-dot cellular automata or carbon nanotubes. Circuits possess no storage capacity (apart from input/output gates), which reflects the often-imposed resource constraints in nanonetworks [1].

Our main conclusions are as follows: If a desired nano-machine implementation is able to simulate the atomic components of AC^0-circuits and constraints for space and depth are met, it is probable that the other problems for AC^0-machines in Table 2 can be solved by a similar machine under the same conditions. For example, if the designed nano-machine is required to calculate ADD, a similar nano-machine should be able to solve SUB—among others—as well.

The same accounts for the NC^1-class, once the respective constraints are met (see Section 4). Thus, a nano-machine that is able to, for example, calculate MULT, may at least compute all problems of AC^0 and the rest of NC^1.

In medical terms, a possible scenario might be as follows: To detect a disease, a nanonetwork is sometimes required to monitor and accumulate a marker concentration over time. We know that Avg is in NC^1, as well as Thres: It is thus possible without increasing the nano-machine space or circuit-complexity to also check if the detected concentration has passed a set threshold level.

6.2 Space-Constrained Turing Machines

Space constrained Turing machines are fundamentally different from circuits. They possess a logarithmic amount of memory and may thereby compute algorithms for more difficult problems. Table 2 row 3 shows some problems of importance to nanonetworks that are in L, which is the corresponding class. Once nano-machine implementations are at least as powerful as logarithmic space constrained Turing machines, they are able to calculate exactly the problems of class L (or maybe more). All of the problems listed above in Table 2 may also be computed by a machine that may compute algorithms for problems in L.

7 CONCLUSION

While many publications feature nano-machinery, few specify the incorporated computational model for the chosen nano-machine. The provided key information often consists of the rather vague "resource constraint"-term. This paper attempts to close the gap by analyzing the tasks to be performed by a nano-machine in numerous medical scenarios. The determined problems are sorted into the three very constrained complexity classes AC^0, NC^1 and L, thereby taking the immense resource constraints into account.

The chosen complexity class yields the minimum capabilities required for the nano-machine, as well as the other readily available capabilities. If the machine resembles a specific type of circuit or L-Turing machine or is able to emulate them, the whole catalog of problems contained in the respective class and contained classes (see Table 1) is likely computable by the same machine, or a similar machine may be constructed.

However, the chosen complexity classes may not represent actual nano-machine implementations suitably. Additional components and operations for communication, input/output-handling, storage or clocking might be necessary, while not being adequately reflected by circuits or Turing machines.

Further, to fully determine the complexity of a nano-machine, one has to also consider the possible combination of operations into algorithms. An algorithm might require an operation to be performed linearly or more often in succession, perhaps exceeding the capability of the class of the operation.

Additionally, more complexity classes can be considered, further populating Figure 1, thereby specifying the complexity of operations even more precisely.

REFERENCES

[1] Ian F. Akyildiz, Fernando Brunetti, and Cristina Blázquez. 2008. Nanonetworks: A new communication paradigm. *Computer Networks* 52, 12 (2008), 2260 – 2279.
[2] Ian F. Akyildiz, Josep Miquel Jornet, and Massimiliano Pierobon. 2011. Nanonetworks: A New Frontier in Communications. *Commun. ACM* 54, 11 (Nov. 2011), 84–89.
[3] Eric Allender, David A. Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. 2009. Planar and Grid Graph Reachability Problems. *Theory of Computing Systems* 45, 4 (2009), 675–723.
[4] C. Alvarez and R. Greenlaw. 2000. A compendium of problems complete for symmetric logarithmic space. *computational complexity* 9, 2 (2000), 123–145.
[5] Paolo Amato, Massimo Masserini, Giancarlo Mauri, and Gianfranco Cerofolini. 2010. *Early-Stage Diagnosis of Endogenous Diseases by Swarms of Nanobots: An Applicative Scenario.* Springer Berlin Heidelberg, Berlin, Heidelberg, 408–415.
[6] J Christopher Anderson, Elizabeth J Clarke, Adam P Arkin, and Christopher A Voigt. 2006. Environmentally controlled invasion of cancer cells by engineered bacteria. *Journal of molecular biology* 355, 4 (2006), 619–627.
[7] Sanjeev Arora and Boaz Barak. 2009. *Computational complexity: a modern approach.* Cambridge University Press.
[8] B. Atakan, O. B. Akan, and S. Balasubramaniam. 2012. Body area nanonetworks with molecular communications in nanomedicine. *IEEE Communications Magazine* 50, 1 (January 2012), 28–34.
[9] Yaakov Benenson, Binyamin Gil, Uri Ben-Dor, Rivka Adar, and Ehud Shapiro. 2004. An autonomous molecular computer for logical control of gene expression. *Nature* 429, 6990 (27 May 2004), 423–429.
[10] Chiu, Andrew, Davida, George, and Litow, Bruce. 2001. Division in logspace-uniform NC1. *RAIRO-Theor. Inf. Appl.* 35, 3 (2001), 259–275.
[11] Luis C Cobo and Ian F Akyildiz. 2010. Bacteria-based communication in nanonetworks. *Nano Communication Networks* 1, 4 (2010), 244–256.
[12] Stephen A Cook and Pierre McKenzie. 1987. Problems complete for deterministic logarithmic space. *Journal of Algorithms* 8, 3 (1987), 385 – 394.
[13] Robert A Freitas. 2005. Current status of nanomedicine and medical nanorobotics. *Journal of Computational and Theoretical Nanoscience* 2, 1 (2005), 1–25.
[14] Siavash Ghavami, Farshad Lahouti, and Ali Masoudi-Nejad. 2012. Modeling and analysis of abnormality detection in biomolecular nano-networks. *Nano Communication Networks* 3, 4 (2012), 229–241.
[15] Zhen Gu, Alex A Aimetti, Qun Wang, Tram T Dang, Yunlong Zhang, Omid Veiseh, Hao Cheng, Robert S Langer, and Daniel G Anderson. 2013. Injectable nano-network for glucose-mediated insulin delivery. *ACS nano* 7, 5 (2013), 4194–4201.
[16] N. Immerman and S. Landau. 1995. The Complexity of Iterated Multiplication. *Information and Computation* 116, 1 (1995), 103 – 116.
[17] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. 2003. Completeness results for graph isomorphism. *J. Comput. System Sci.* 66, 3 (2003), 549 – 566.
[18] Alec A.K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. 2016. Genetic circuit design automation. *Science* 352, 6281 (2016), aac7341.
[19] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, and G. L. Snider. 1997. Realization of a Functional Cell for Quantum-Dot Cellular Automaton. *Science* 277 (8 1997), 928–930.
[20] Prof. Dr. Evangelos Kranakis (auth.) Prof. Dr. Peter Clote. 2002. *Boolean Functions and Computation Models* (1 ed.). Springer-Verlag Berlin Heidelberg.
[21] Walter J. Savitch. 1970. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. System Sci.* 4, 2 (1970), 177 – 192.
[22] Sunil Singhal, Shuming Nie, and May D Wang. 2010. Nanotechnology applications in surgical oncology. *Annual review of medicine* 61 (2010), 359–373.
[23] Mark Staples, Karen Daniel, Michael J. Cima, and Robert Langer. 2006. Application of Micro- and Nano-Electromechanical Devices to Drug Delivery. *Pharmaceutical Research* 23, 5 (2006), 847–863.
[24] Marc Stelzner, Florian-Lennert Lau, Katja Freundt, Florian Büther, Mai Linh Nguyen, Cordula Stamme, and Sebastian Ebers. 2016. Precise Detection and Treatment of Human Diseases Based on Nano Networking. In *11th International Conference on Body Area Networks (BODYNETS 2016)*. EAI, Turin, Italy.
[25] Ageliki Tsioliaridou, Christos Liaskos, Sotiris Ioannidis, and Andreas Pitsillides. 2015. CORONA: A Coordinate and Routing System for Nanonetworks. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication (NANOCOM' 15)*. ACM, New York, NY, USA, Article 18, 6 pages.
[26] Heribert Vollmer. 1999. *The NC Hierarchy.* Springer Berlin Heidelberg, Berlin, Heidelberg, 107–171.
[27] Avi Wigderson. 1992. The complexity of graph connectivity. In *International Symposium on Mathematical Foundations of Computer Science.* Springer, 112–132.
[28] Hang Yu, Bryan Ng, and Winston K. G. Seah. 2015. Forwarding Schemes for EM-based Wireless Nanosensor Networks in the Terahertz Band. In *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication (NANOCOM' 15)*. ACM, New York, NY, USA, Article 17, 6 pages.