Computational Requirements for Nano-machines

Melanie Badura
Universität zu Lübeck
Lübeck
melanie.badura@student.uni-luebeck.de

ABSTRACT

This paper is a short paper for "Computational Requirements for Nano-Machines: There is limited Space at the bottom".[1]

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

ACM proceedings

ACM Reference Format:
Melanie Badura. 2021. Computational Requirements for Nano-machines. In *Proceedings of AzuNet Seminar (CR for Nano'21).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

For years, there has been talking of using nano-machines to create solutions in medicine and other subjects. Such a machine should be able to communicate and sense/act. Computational power is also a big issue. Because Nano-machines are small, one question is how to implement these capabilities. While many researchers already deal with communication technology (cite one here), computational capability is usually left out. In this paper, there is an attempt to provide a general analysis of the computational capability of nano-machines. Since the capabilities of nano-machines vary widely, from nanoparticles with no computational capabilities to microprocessors (11 cites). Nano-machines divide into three groups according to complexity theory, by analyzing the tasks that nano-machines handle.

CR for Nano'21, July 2021, Lübeck, Germany

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2 MITTELTEIL?

Most problems that a nano-machine has to solve need basic arithmetic. Pattern matching and parity solve other problems. Protocols, such as forwarding and routing, solve the communication problem. These protocols are solved in different ways so that a nano-machine can handle several types of messages. Communication is a perfect example for the use of basic arithmetics, pattern matching, storage needs. They need memory for the storage of routing information and for other values with which they must work. Nano-machines should also be able to perform more complex operations like implementing a neural network. They use graph algorithms for this.

All these problems divide into complexity classes. As the name suggests, the problems divide into classes that have approximately the same complexity. The reduction of operations chooses the complexity class. For example, subtract can be reduced to add. Here, the simplest complexity class can be used, but also with a nano-machine that has been built to solve the problems of the most difficult complexity class. However, this does not work the other way around. In table 1, the three different classes are seen. An L-machine can solve problems like a Turing machine. The class AC^0 describes boolean circuits with polynomial size and a constant depth, whereas the class NC^1 may have a logarithmic depth and two inputs per gate.

Problems can lower their class if researchers can find new reductions for them. However, problems exist that need more powerful machines or global knowledge about the network. Thus problems, like addressing, routing broadcasting/forwarding, could not be divided into the three complexity classes.

Insights into the feasibility of Nanao-machines provides the classification of the applications according to the computational power. So that when a protocol or algorithm is implemented in a nano-machine, researchers may consult Table 1. The algorithm must then be decomposed into its basic operations. Afterward one looks which operation is the most complex and decides so for a complexity class. Just because a nano-machine may perform only a few easy operations does not mean that it is simple to build, but the choice of complexity class can help.

Considering the AC^0 class, if a nano-machine can add, it may do all other operations of the complexity class. Let us

Machine	Problems		Origin
AC^0:	ADD	$ODD/EVEN$	
	SUB	DIV_2	
	$SIGN$	MOD_2	
	INC	LOG_2	
	AND/OR	INV	
NC^1:	$MULT$	MIN/MAX	
	DIV	$PARITY$	
	EXP	REG	
	$MAJOR$	MOD	
	$THRES$	AVG	
L:	$Label$	D_{FS}	
	$Logmem$	B_{FS}	
	$REACH$	$MEDIAN$	

Table 1: Complexity classes of nano-machines

take the complexity class NC^1 for a practical example. In the body, a nano-machine observes the concentration of a marker with AVG. Since $THRES$ is also in the same class, this nano-machine can also check if a defined value exceeds. This without increasing the complexity nor the memory.

Falls du auf zu wenig zeilen kommst, ein reduction beispiel auf jeden fall machen. Und vllt nochmal auf origin eingehen.

3 CONCLUSION

hier muss noch conclusion hin

REFERENCES

[1] 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 the 4th ACM International Conference on Nanoscale Computing and Communication*. 1–6.