# Computational Requirements for Nano-machines*

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

## ABSTRACT

This paper is a shortpaper 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 for problems 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.

*Produces the permission block, and copyright information
[1]


*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. Others are solved with pattern matching and parity. To solve the communication problem, there are forwarding and routing protocols. 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. However, not only for the storage of routing information they need memory but also for other values with which they must work, must be stored. 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. This is found out by reduction. 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.
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
bla
als nächstes ab 5 und du hast für mittelteil keinen titel und nicht geschaut was grammerly so sagt! Das table kommt wieder wenn du hier mehr schreibt

| 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**

# 3 CONCLUSION

## 3.1 Subsection

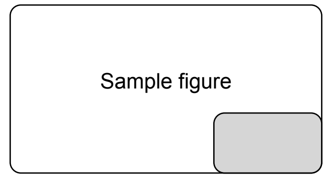


**Figure 1: Sample figure**