Outline

• Motivation / Introduction
• Research Questions
• Background
• Results (some)
  – Evolvability
  – Complexity
  – Scalability
• Conclusions and Further Work
Motivation
Motivation
Engineering: top-down
Nature: bottom-up
Conventional Engineering: attempts to analyze (top-down) systems

Bio-Inspired Computation: attempts to synthesize (bottom-up) life-like behaviors within computers and other artificial media
Examples: bio-inspired

- Conway 1970
- Funes 1997
- Christensen, Grady, Dorigo 2009
- Hornby, Al Globus, Linden, Lohn 2006
- Doursat, Sanchez, Dordea, Fourquet, Kowaliw 2014

(Funes 1997)

(Conway 1970)

(Doursat, Sanchez, Dordea, Fourquet, Kowaliw 2014)

(Hornby, Al Globus, Linden, Lohn 2006)
Research Focus

- How to apply artificial evolution and development for the design of cellular machines that can produce complex computation and modelling?
Research Questions
Research Questions

RQ1:

• What kind of information must be present in the genome in order to produce computation in any of the computational classes?

Universality classes: CA computational behavior (Wolfram)

– What information must be present in the genome?
– What information processing capability must be available in the gene regulation?
– What cellular actions are required to be expressed as to be able to develop a target organism?
Research Questions

RQ2:

- How to quantify developmental complexity, i.e. emergent phenotypic complexity?

Development process as a whole
Phenotypic changes: trajectory, transient, attractor
Research Questions

RQ3:

- Do genome parameters give any information on the evolvability of the system? And if yes, can genome information be used to guide evolutionary search in favourable areas of the search space where the wanted emergent behavior is more likely to be found?
Research Questions

RQ4:

• How can scalability of artificial EvoDevo systems be improved towards achieving systems that can fully unleash their inherent complexity, e.g. potentially at the levels of complexity found in nature?

Gene duplication
Open ended
## Contributions

<table>
<thead>
<tr>
<th>Paper N.</th>
<th>Title</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the Correlations Between Developmental Diversity and Genomic Composition</td>
<td>A.1</td>
</tr>
<tr>
<td>2</td>
<td>Genome Parameters as Information to Forecast Emergent Developmental Behaviors</td>
<td>A.2</td>
</tr>
<tr>
<td>3</td>
<td>Measuring Phenotypic Structural Complexity of Artificial Cellular Organisms</td>
<td>B.1</td>
</tr>
<tr>
<td>4</td>
<td>Evolution of Incremental Complex Behavior on Cellular Machines</td>
<td>C.1</td>
</tr>
<tr>
<td>5</td>
<td>Investigation of Genome Parameters and Sub-Transitions to Guide Evolution of Artificial Cellular Organisms</td>
<td>C.2</td>
</tr>
<tr>
<td>6</td>
<td>Evolutionary Growth of Genome Representations on Artificial Cellular Organisms with Indirect Encodings</td>
<td>D.1</td>
</tr>
<tr>
<td>7</td>
<td>Evolutionary Growth of Genomes for the Development and Replication of Multicellular Organisms with Indirect Encodings</td>
<td>D.2</td>
</tr>
<tr>
<td>8</td>
<td>Trajectories and Attractors as Specification for the Evolution of Behavior in Cellular Automata</td>
<td>E.1</td>
</tr>
<tr>
<td>9</td>
<td>Discrete Dynamics of Cellular Machines: Specification and Interpretation</td>
<td>E.2</td>
</tr>
<tr>
<td>10</td>
<td>On the Edge of Chaos and Possible Correlations Between Behavior and Cellular Regulative Properties</td>
<td>E.3</td>
</tr>
</tbody>
</table>
Chronological Structure

- **A**: genome parameters, Lambda, forecast emergent behaviour
  - A1
  - A2

- **B**: structural complexity
  - B1

- **C**: parameters evolution, incremental evolution
  - C1
  - C2

- **D**: evolutionary growth of genomes
  - D1
  - D2

**Timeline**
- **Begin Sep. 2010**
- **End Sep. 2014**

Legend:
- Blue square: paper included
- Red square: paper not included
Background
Background

- Development
- Evolution
- Cellular Automata
- Genotype-Phenotype mapping & representations
- Genome parameters
- Complexification

Artificial EvoDevo
Artificial Development

DNA ~ 22000 – 25000 genes
Human body ~3.72x10^{13} cells
EvoDevo

(Darwin 1859)
EvoDevo systems - CA

- Cellular Automata can be considered as developmental systems

- Organisms can develop (e.g. grow) from a zygote to a multi-cellular organism (phenotype) according to specific local rules, represented by a genome (genotype)

- The genome specifications and the gene regulatory information control the cells’ growth and differentiation

- The behavior of the CA is represented by the emergent phenotype, which is subject to shape and size modification, along the developmental process
Cellular Automata

John von Neumann
Precursor of CA,
Universal constructor
Self-reproduction

Stephen Wolfram
1D CA classes
Edge of Chaos & Genome Parameters

\[ \lambda = 1 - \frac{1}{k} \]

Other parameters:
- MFP (Li)
- Sensitivity (Binder)
- Z (Wuensche)
- …
Genotype-to-Phenotype Encodings

(adapted from "Developmental Mappings and Phenotypic Complexity", Lehre P.C., 2003)
Genotype-to-Phenotype Encodings

• One-to-one direct Gtype-to-Ptype mapping
• Redundant / many-to-one, neutrality
• Indirect
  – Nature
  – Challenges
    • Restricted set (estimate)
    • If scaled-up?
    • Variable length (speciation)
    • Open ended search
Complexification

The natural and biological process of incremental genome growth and elaboration

- Genomes of different species have different lengths
- LUA (Last Univestal Ancestor): all species diverged from a common ancestor ~ 3.5 - 3.8 billion years ago
- Gene duplication: novelty & potential evolutionary innovation
- Duplicated: redundant but < selection pressure
- 38% of Homo Sapiens genome = gene duplication

- Complexification with direct encodings (Federici & Downing, Stanley & Miikkulainen), e.g. NEAT (NeuroEvolution of Augmenting Topologies)

- Complexification with development
Results
Results Summary

- A.1: not presented
- A.2: in details
- B.1: not presented
- C.1: not presented
- C.2: in details
- D.1: in details
- D.2: shortly (if time allows)
- E.1-E.2-E.3: not included in the thesis
A.2 – Complexity/Evolvability

• Measure genomic properties
• **Predict** emergent phenotypic properties of artificial organisms
• Genome parameters: $\lambda, M, \mu$
• How the composition of genome information and gene regulation influences the developmental trajectory
CA model

• minimalistic developmental system

• 3 cell types
  (type 0: quiescent, type 1 and type 2 for multicellularity)

• all possible $3^5 = 243$ regulatory input combinations are represented in a development table
Measurements of the Phenotypic Behavior

- **trajectory and attractor length**: may indicate information about structural and adaptive properties of the organism
  - does development create a stable organism (point attractor) or does the organism end with a self-reorganizing structure that changes form in a cyclic manner (cyclic attractor)?

- **growth and change rate**: may give information on the activity (internal properties) of the developmental processes
  - growth phase: the organism expand in size toward an "adult" form
  - change phase: changes in the adult organism (measurement of the adult life of the organism)
Genome Parameters

Evaluation of the genetic information

- $\lambda$ (Lambda): purely regulatory output

- $M$ (Majority): regulatory input and relative output, each entry considered independently

- $\mu$ (Sensitivity): overall parameter calculated out of genetic dependency properties
Experimental Setup

State space:
3by3 = $3^9 = 19.683$
4by4 = $3^{16} = 43.046.721$
5by5 = $3^{25} = 847.288.609.443$
Results - $\lambda$

Measurements in correlation to $\lambda$, average over 1000 tests for each $\lambda$ value

Average trajectory and attractor length

Average growth and change rate
Results - M

Measurements in correlation to M, average over 1000 tests for each M value

Average trajectory and attractor length

Average growth and change rate
Results - $\mu$

Measurements in correlation to $\mu$, average over 1000 tests for each $\mu$ value

Average trajectory and attractor length

Average growth and change rate
Comparison
Conclusion A.2

- Parameters as measurement of genomic composition
- Predict developmental behavior
- Relation between genomic composition and developmental properties
- Each genome parameter has a specific ability to measure properties of the resulting organism
- Knowledge of probable developing properties may be helpful at the design stage of an EvoDevo system, if information on the desired target phenotype is known
- Possible to use more parameters together to compose desired developmental behaviors, not achievable with a single parameter
C.2 Evolvability

Goal: genome information (parameters) to guide evolution

Nature: evolved robust genomes

Robustness VS Evolvability

• Robust: no change in functionality after mutation
• Evolvable: genetic variation, adaptive evolution

EA: sensitive to mutations
Genotype & phenotype distance

(adapted from “Developmental Mappings and Phenotypic Complexity”, Lehre P.C., 2003)
Experimental Setup

- Genetic Algorithm (details in paper)
- Initial population = most unfit (all transitions to quiescent state)
- Standard fitness VS parameter in fitness

\[ CFitness = Fitness + Fitness \times \frac{Abs(Hi\lambda - \lambda)}{Hi\lambda} \times \text{ratio} \]
Results - target 1000 dev. steps

<table>
<thead>
<tr>
<th>GA</th>
<th>Void genomes (plotted)</th>
<th>Randomized genomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference GA</td>
<td>443,60</td>
<td>372,00</td>
</tr>
<tr>
<td>Lambda fitness</td>
<td>365,62</td>
<td>351,33</td>
</tr>
</tbody>
</table>
5000, 10000, 15000
Sub-Transitions

- Growth
- Differentiation
- Death (Lambda – quiescent state)
- No-Change

Lambda: single sub-transition parameter
More states = more sub-transition classes, Lambda less meaning, possible to build custom parameter
Conclusion C.2

- Genome information to guide evolution
- Vast search space, indirect G-Ptype mapping, development
- Where the target behavior is more likely to be found
- Lambda in fitness to speedup convergence
- Sub-Transitions
- Composite parameters
- Growth – Death transition
- Filter
D.1 Scalability

Motivation (Why):
- In nature genomes of different species have different lengths
- Scaling of artificial systems (state, search and solution space)
- Genotype representation problem (estimated, heuristics)

Genome Growth (How)
- Allows speciation
- Through gene duplication (in nature)
- Complexification (incremental elaboration)
- Compare full vs restricted vs growing (genomes)
Regulation mechanisms:
• Upper bound, duplication rate, optimization time, elitism

Selection:
• $\Sigma$ (actual fitness, exploitation parameter, innovation parameter)
Scalability in search space – genome comparison
Scalability in state space
Scalability in solution space - geometry
Conclusion D.1

- Evolutionary growth of genome representations
- Compact and effective genomes
- Scalability of search space
- Scalability of state space
- Scalability of phenotypic resources
- Start in low dimensional space
- Incrementally increase genotype complexity
D.2

- Genome Growth
- Instruction-Based Development

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Meaning</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>[N(op_1) = N(op_1) \land N(op_2)]</td>
<td>AND operation</td>
<td>0</td>
</tr>
<tr>
<td>OR</td>
<td>[N(op_1) = N(op_1) \lor N(op_2)]</td>
<td>OR operation</td>
<td>1</td>
</tr>
<tr>
<td>XOR</td>
<td>[N(op_1) = N(op_1) \oplus N(op_2)]</td>
<td>XOR operation</td>
<td>2</td>
</tr>
<tr>
<td>NOT</td>
<td>[N(op_1) = \neg N(op_1)]</td>
<td>NOT operation</td>
<td>3</td>
</tr>
<tr>
<td>INV</td>
<td>[N(op_1) = n - N(op_1)]</td>
<td>Inverse state</td>
<td>4</td>
</tr>
<tr>
<td>MIN</td>
<td>[N(op_1) = \min (N(op_1), N(op_2))]</td>
<td>Minimum</td>
<td>5</td>
</tr>
<tr>
<td>MAX</td>
<td>[N(op_1) = \max (N(op_1), N(op_2))]</td>
<td>Maximum</td>
<td>6</td>
</tr>
<tr>
<td>SET</td>
<td>[N(op_1) = N(op_2)]</td>
<td>Set value</td>
<td>7</td>
</tr>
<tr>
<td>INC</td>
<td>[N(op_1) = N(op_1) + 1]</td>
<td>Increment</td>
<td>8</td>
</tr>
<tr>
<td>DEC</td>
<td>[N(op_1) = N(op_1) - 1]</td>
<td>Decrement</td>
<td>9</td>
</tr>
<tr>
<td>SWAP</td>
<td>[N(op_1) \leftrightarrow N(op_2)]</td>
<td>Swap</td>
<td>10</td>
</tr>
<tr>
<td>ROR</td>
<td>[LCR \rightarrow RLC]</td>
<td>Rotate right</td>
<td>11</td>
</tr>
<tr>
<td>ROL</td>
<td>[LCR \rightarrow LCR]</td>
<td>Rotate left</td>
<td>12</td>
</tr>
<tr>
<td>ROU</td>
<td>[UCD \rightarrow CDU]</td>
<td>Rotate up</td>
<td>13</td>
</tr>
<tr>
<td>ROD</td>
<td>[UCD \rightarrow DUC]</td>
<td>Rotate down</td>
<td>14</td>
</tr>
<tr>
<td>NOP</td>
<td>[N(op_1) = N(op_1)]</td>
<td>No operation</td>
<td>15</td>
</tr>
<tr>
<td>Fig.</td>
<td>Table-based Evolution</td>
<td>Instruction-based Growing Evolution</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
<td>-----------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Success Rate %</td>
<td>Genotype Size (# genes)</td>
<td>Generations</td>
</tr>
<tr>
<td>2a</td>
<td>58</td>
<td>32 32 32 0</td>
<td>1336 2294</td>
</tr>
<tr>
<td>2b</td>
<td>69</td>
<td>32 32 32 0</td>
<td>2254 2501</td>
</tr>
<tr>
<td>2c</td>
<td>19</td>
<td>1024 1024 1024 0</td>
<td>5002 3157</td>
</tr>
<tr>
<td>2d</td>
<td>23</td>
<td>32 32 32 0</td>
<td>2668 2942</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Instruction-based Growing Evolution</th>
<th>Table-based Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>98</td>
<td>31 14.34 5 8.4318</td>
</tr>
<tr>
<td>2b</td>
<td>98</td>
<td>31 15.28 5 7.0973</td>
</tr>
<tr>
<td>2c</td>
<td>46</td>
<td>46 19.65 6 9.2236</td>
</tr>
<tr>
<td>2d</td>
<td>100</td>
<td>13 5.25 4 1.4097</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Instruction-based Growing Evolution</th>
<th>Table-based Evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>100</td>
<td>7 2.93 2 1.1742</td>
</tr>
<tr>
<td>2b</td>
<td>100</td>
<td>6 2.84 2 1.1166</td>
</tr>
<tr>
<td>2c</td>
<td>100</td>
<td>8 3.06 2 1.2128</td>
</tr>
<tr>
<td>2d</td>
<td>100</td>
<td>5 1.38 1 0.8012</td>
</tr>
</tbody>
</table>
- Example of evolved program for the development of structure 2c – patch structure
- After development step 9 the structure remains stable (point attractor)
- The program is composed by 14 instructions (one instruction each gene)

- INSTRUCTION CODE, OPERAND 1, OPERAND 2 (if the operand is not applicable for the given instruction, the value is ignored)

- Operands: UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, CENTRE = 4.
Examples of evolved solutions for the replication of the structures 2d, 2c and 2e.
Conclusions and Further Work
Conclusion

• Genome parameters, forecast emergent developmental phenotypes (A.1, A.2)
• Abstract measures of developmental complexity (B.1, A.1, A.2)
• Genome parameters as evolvability evaluation (C.1)
• Genome parameters to guide evolution (C.2)
• Evolutionary growth of genomes (D.1, D.2)

GUIDANCE ON HOW TO BUILD EVODEVO SYSTEMS
Contributions

RQ1: "What kind of information must be present in the genome in order to produce computation in any of the computational classes?"

- Genome parameters: plausible indication of developmental properties
- Do not guarantee developmental behavior (the other way around)
- Parameters generalized for different dimensionalities, CA size, cell types
- Genome sub-transitions and sub-parameters (death – growth)
Contributions

RQ2: "How to quantify developmental complexity, i.e. emergent phenotypic complexity?"

- As measure of phenotypic & developmental properties: developing organism as a whole, phenotypic changes
- Trajectory and attractor length: abstract (application / computational task independent)
- Approx. of Kolmogorov Complexity: compression algorithms
Contributions

**RQ3**: ”Do genome parameters give any information on the evolvability of the system? And if yes, can genome information be used to guide evolutionary search in favourable areas of the search space where the wanted emergent behavior is more likely to be found?”

- Genomes with given parameter value are likely to evolve to similar behaviors, as long as offspring has similar parameter value (evolvability)
- Parameters (as Lambda or sub-transitions) to guide evolution
Contributions

**RQ4:** "How can scalability of artificial EvoDevo systems be improved towards achieving systems that can fully unleash their inherent complexity, e.g. potentially at the levels of complexity found in nature?"

- Complexification: evolutionary growth of genomes
- Indirect encoding are a necessity (if target nature levels of complexity)
- Gene duplication is a plausible mechanism
- Different genotype-to-phenotype mappings
Further Work

- Robustness of solutions (at gtype and ptype level)
- Other parameters: Sensitivity, MFP, Z, sub-transitions
- Growth in state space (true complexification)
- Self-modifying instructions
Key References


Thanks!

Questions?
Bonus Slides
Cellular Automata

Countable array of discrete cells \( i \)

Discrete-time update rule \( \Phi \)
(operating in parallel on local neighborhoods of a given radius \( r \))

Alphabet: \( \sigma^i_t \in \{0, 1, \ldots, k-1\} \equiv A \)

Update function: \( \sigma^i_{t+1} = \Phi(\sigma^{i-r}_t, \ldots, \sigma^{i+r}_t) \)

\( s_t \in A^N \)

Global update \( \Phi: A^N \rightarrow A^N \)

\( s_t = \Phi s_{t-1} \)
Lambda Parameter

\[ \lambda = \frac{K^N - n}{K^N} \]

- \( n \) = number of transitions to the quiescent state (state 0)
- \( K \) = number of cells types = 3 (in our model)
- \( N \) = neighborhood size = 5 (Von Neumann neighborhood)
Majority Parameter

• how many neighborhood configurations in the rule table follow the majority state to determine the next state

\[ M = \sum_{(V_1V_2 \ldots V_m)} [V(m + 1) = \text{maj}(V_1V_2 \ldots V_m)] \]

• \( m \) = number of cells in the neighborhood
• \( V(m+1) \) = value of the cell being considered, at the next time step
• \( \text{maj}() \) = function that retrieves the most present cell type (or the set of most present cell types) in the neighborhood
Sensitivity Parameter

- measures the number of changes in the output of the transition table based on a change in the neighborhood, one cell at a time, over all the possible neighborhoods of the rule being considered.

\[ \mu = \frac{1}{n \cdot m \cdot (K-1)} \sum_{(V1V2...Vm)} \sum_{q=1}^{m} \frac{\delta \phi}{\delta V_q} \]

- \( m \) = number of cells in the neighborhood
- \( n \) = possible neighborhood configurations (\( V1V2...Vm = 3^5 = 243 \))
- \( K \) = number of cell types
The developmental path shown as a trajectory.

1 - Initial state
32 - Intermediate state
64 - Final state

NTNU – Trondheim
Norwegian University of Science and Technology
von Neumann architecture
- 1 complex processor
- tasks executed sequentially

cellular computing
- myriad of small and unreliable parts: cells
- simple elements governed by local rules
- cells have no global view
Results for 4x4 organisms plotted as function of $\lambda d$.
1000 tests for each $\lambda d$.

Results for 5x5 organisms plotted as function of $\lambda d$.
1000 tests for each $\lambda d$. 
The graphs illustrate the development steps for trajectory and attractor lengths across different values of $\lambda_d$ for 4x4 and 5x5 configurations. The graphs show the average trajectory length and average attractor length, with peaks at specific $\lambda_d$ values.
Average growth and change rate in correlation to $\lambda d$ on a 4x4 organism.
Average over a 1000 tests for each $\lambda d$ value
B.1

Image from "A New Kind of Science", Stephen Wolfram (2002), Wolfram Media
Goal

- Can genome information be used to predict emergent structural complexity?

1. Measure phenotypic structural complexity of artificial cellular organisms (approximation of Kolmogorov complexity)

2. Relate Lambda genome parameter to the measured structural complexity. Estimate developed organisms’ phenotypic complexity
Kolmogorov Complexity

**Image from xkcd.com**
Kolmogorov Complexity

• The notion of complexity is used differently in distinct fields of computer science.

• **Definition (Kolmogorov complexity):** Fix a Turing Machine $U$. We define the Kolmogorov function, $C(x)$ as the length of the smallest program generating $x$.

  $$C(x) = \min_p \{ |p| : U(p) = x \}$$

• **Invariance Theorem:** the particular choice of the universal machine only affects $C(x)$ by a constant additive factor

  $$\forall x, \ C(x) \leq |x| + c$$
Incomputability

- Kolmogorov complexity is incomputable. Proof by contradiction or by reduction to the non-computability of the halting problem (Turing equivalent)

- Approximations by data compression: hardly compressible strings have presumably high Kolmogorov complexity. Complexity is proportional to the compression ratio

- Incompressibility Lemma: some strings are not compressible, i.e. random strings
  Formally, a string $x$ is $c$-incompressible if $C(x) \geq |x| - c$
Lempel-Ziv compression

• Compression algorithms tend to compress repeated patterns and structures, thus being able to detect structural features in phenotype states.

• Deflate: variation of LZ77, loseless data compression algorithm, computationally inexpensive, independent of the dimensionality of the state
• 1D CA: string representing the state of the system at a certain time step compressed directly

• 2D CA (3x3 example, same for 3D where all the rows are listed for all the depth levels)

\[
\begin{array}{ccc}
0 & 1 & 0 \\
1 & 1 & 2 \\
1 & 0 & 0 \\
\end{array}
\rightarrow r = "010112100"
\]

\[
t = \text{Deflate}(r)
\]

\[
q = \text{Length}(t)
\]

\[
r_{\min} = "000000000"
\]

\[
r_{\max} = "012345678"
\]

\[
q_{\min} = \text{Length(Deflate}(r_{\min}))
\]

\[
q_{\max} = \text{Length(Deflate}(r_{\max}))
\]

\[
c = (q - q_{\min}) / (q_{\max} - q_{\min})
\]
The CA state is represented as a concatenated string and directly compressed.

The CA state is rotated in all the possible orientations and the correspondent state strings are compressed. The average is computed.

The CA state is shifted in all the possible positions and the correspondent state strings are compressed. The average is computed.

Both point 2 and 3. The CA state is rotated in all the possible orientations. Each of them is shifted in all the possible positions and the correspondent state strings are compressed. The overall average is computed.
# Experimental Setup

<table>
<thead>
<tr>
<th>Dimensionality</th>
<th>Size</th>
<th>Cells</th>
<th>Neighborhood radius</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experiment 1:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D</td>
<td>9</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>1D</td>
<td>9</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>1D</td>
<td>16</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>1D</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2D</td>
<td>3x3</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>2D</td>
<td>4x4</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>3D</td>
<td>2x2x2</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td><strong>Experiment 2:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D</td>
<td>25</td>
<td>25</td>
<td>3</td>
</tr>
<tr>
<td>1D</td>
<td>27</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>1D</td>
<td>25</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>1D</td>
<td>27</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td>2D</td>
<td>5x5</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>3D</td>
<td>3x3x3</td>
<td>27</td>
<td>7</td>
</tr>
</tbody>
</table>

State space sizes:

- $3^{3} = 3^9 = \text{19.683}$
- $3^{16} = 3^{4} \times 3^{12} = 19.683 \times 2^{12} = 43.046.721$
- $3^{25} = 3^{5} \times 3^{20} = 19.683 \times 43.046.721 = 847.288.609.443$
Results (exp. 1)

Figure 1.1: 1D 9 cells, 3-neighbourhood

Figure 1.3: 1D 16 cells, 5-neighbourhood

Figure 1.6: 2D 16 cells (4 x 4), 5-neighbourhood

Figure 1.7: 3D 8 cells (2 x 2 x 2), 7-neighbourhood
Results (exp. 2)

(a) Average trajectory and attractor length.
(b) Structural complexity of trajectory and attractor.

Figure 2.1: 1D 25 cells, 3-neighbourhood

(a) Average trajectory and attractor length.
(b) Structural complexity of trajectory and attractor.

Figure 2.2: 1D 27 cells, 3-neighbourhood

(a) Average trajectory and attractor length.

Figure 2.3: 1D 25 cells, 5-neighbourhood
Results (exp. 2)

(a) Average trajectory and attractor length.

(b) Structural complexity of trajectory and attractor.

Figure 2.4: 1D 27 cells, 7-neighbourhood

(a) Average trajectory and attractor length.

(b) Structural complexity of trajectory and attractor.

Figure 2.5: 2D 25 cells (5 × 5), 5-neighbourhood

(a) Average trajectory and attractor length.

(b) Structural complexity of trajectory and attractor.

Figure 2.6: 3D 27 cells (3 × 3 × 3), 7-neighbourhood
Conclusion

- Phenotypic structural complexity is strongly related to Lambda genome parameter and its ability to detect different behavioral regimes
- Dimensionality independent (1D, 2D, 3D CA)
- Possible to characterize parameter space when dimensionality, # states and neighborhood are rather small. Not possible with transient and attractor length
C.1

Results
(dead genomes)
A.2 Genomes generation with $\lambda$ parameter

Genomes generated with predefined values of $\lambda$

Similar method to Langton’s random table method

For every entry in the development table:
• with probability $(1 - \lambda)$ the cell type at the next developmental step is quiescent (type 0)
• with probability $(\lambda)$, the cell type at the next developmental step is generated by a uniform random distribution among the other cell types (type 1 or 2)
Genomes generation with $M$ parameter

A.2

- if there are more than 3 occurrences of a cell type:
  - with probability $M$ the cell type at the next developmental step follows the most present cell type in the neighborhood
  - with probability $1-M$ the cell type at the next developmental step is generated by a uniform random distribution among the other two cell types (the minority in the neighborhood)

- If there are 2 cell types with occurrence 2
  - with probability $M/2$ one of those 2 cell types is chosen
  - with probability $1-M$ the cell type at the next developmental step has the same type as the less present cell type in the neighborhood
Genomes generation with $\mu$ parameter

A.2

$\mu$ is easily computable for a specific development table

Much harder to generate a development table with a target $\mu$ value, because of entry dependencies

A Genetic Algorithm is used