TECHNISCHE
UNIVERSITÄT
DARMSTADT

Cellular Automata Agents, Modeling and Applications

Rolf Hoffmann

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

PART I

- Introduction

1. Cellular Automata Models
   - 2.1 Classical Cellular Automata
   - 2.2 Cellular Automata with Write-Access (CA-w)
2. CA Agents
3. Multi-Agent-Systems, Examples
   - 4.1 CA-w Algorithms
   - 4.2 Creatures‘ Exploration, All-to-All Communication

PART II

Forming a Checkerboard Pattern

1. Introduction

What is a Multi-Agent System (MAS)?

TECHNISCHE UNIVERSITÄT DARMSTADT

- **MAS = ENV ∪ AGT**
- **Environment (ENV)**
  - fixed
  - variable by
    - itself
    - external inputs
    - agents
- **Agents (AGT)**
  - positions: fixed / moving around
  - direction: fixed / variable
  - types: uniform / non-uniform
  - can perceive / influence
    - the environment
    - other agents

Features of MAS

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Scalability**
  The problem can be solved with a **variable number of agents**, and often faster with more agents.
- **Tuneability**
  when **increasing the agent's intelligence**, the problem can be solved **faster** or **more effective** (better quality of solutions).
- **Versatility**
  similar problems can be solved by the same agents, e.g. different **shape** or **size** of the environment, **obstacles**
- **Fault-tolerance**
  when some **cells or agents are defect** the problem can still be solved in **gracefully degraded** way
- **Updating-tolerance**
  the updating-scheme (**synchronous**, **asynchronous**) only marginally influences the global attractors

MAS Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

Agents are very **flexible**, therefore they can be used for
Modeling, Designing, Simulating, Analyzing, Solving

- ***„Real" worlds***
  - Physics, Chemistry, Biology, Social behavior
- ***Artificial worlds***
  - *Artifical Life, Artificial Chemistry*
- ***Complex problems***
  - *Combinatorial, Mathematical, Algorithmical*
- ***Distributed algorithms***
  - *Distributed Hardware & Software Systems*
  - *Sensor Systems*
- ***Games, Art***

What is a Cellular Automat Agent (CAA)?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Agents modeled within the Cellular Automata (CA) paradigm, or in a closely related model.
- CA models
  - **classical CA**
  - partioned CA
  - Lattice Gas CA
  - block rule substitutions BCA
  - parallel substitution algorithm PSA
  - global CA (GCA)
  - **CA with write-access (CA-w)**

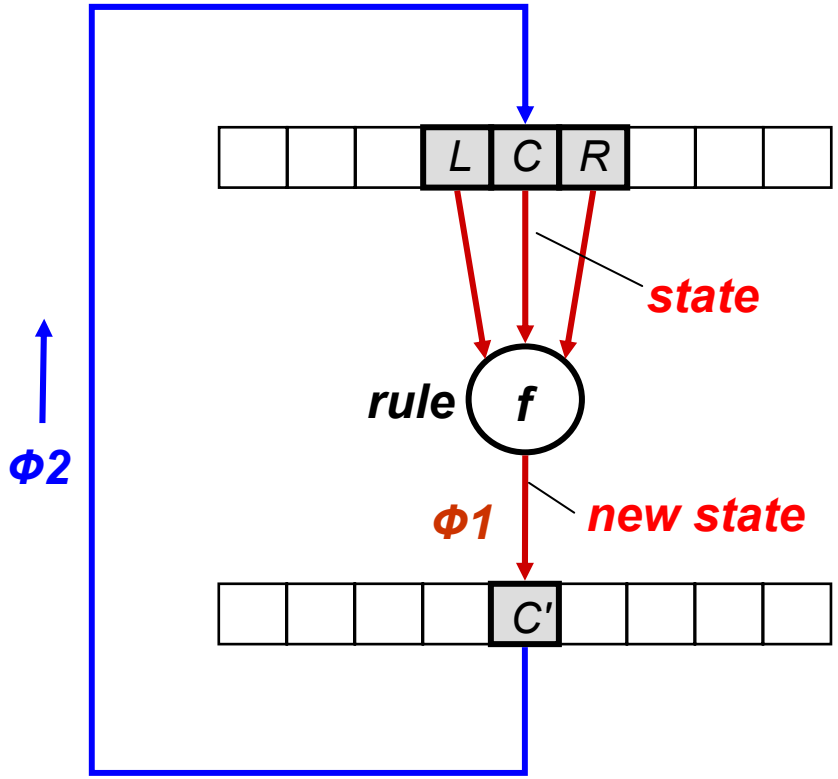
2. CA Models

2.1 Classical CA

2.2 CA with write-access (CA-w)

2.1 Classical CA, 1D

TECHNISCHE UNIVERSITÄT DARMSTADT



C(t) *configuration (global state) at time-step t*

***C*^{new}** *new configuration, stored temporarily*

CA: Synchronous Updating

TECHNISCHE UNIVERSITÄT DARMSTADT

COMPUTE PHASE Φ1, in arbitrary order

```
for i := 0 to N - 1 do

    C^new[i] := f(L[i], C[i], R[i])
```

UPDATE PHASE Φ2, in abitrary order

```
for i := 0 to N - 1 do

    C[i] := C^new[i]
```

Can easily be implemented in Hardware and in Software
Requires a global synchronization (Clock, …)

Asynchronous Updating (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

- **does not require a global clock**
- allows **more realistic** modeling of physical phenomena
- the cells are **sequentially** or partially sequential updated
- the global behavior is **non-deterministic**
- **cells are selected at random** for updating
- the cell‘s state is **not buffered**, the old configuration is not preservered

Asynchronous Updating (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

COMPUTE & UPDATE

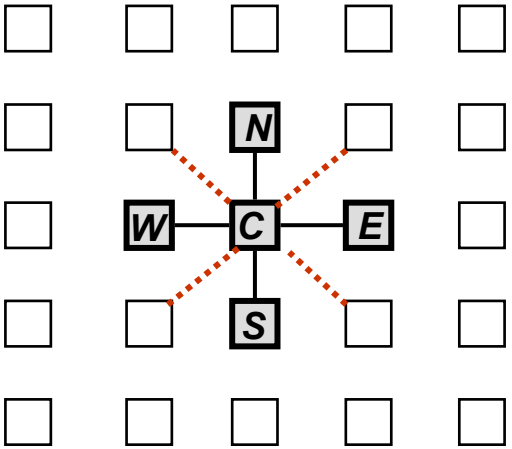
repeat N times

i : = RandomSelectFrom [0, N-1]

C[i] := f(L[i], C[i], R[i])

2D CA

TECHNISCHE UNIVERSITÄT DARMSTADT



- **Regular grid** of **active cells** with **local connections.**
- application of a **local rule** (transition function) to each cell of the array

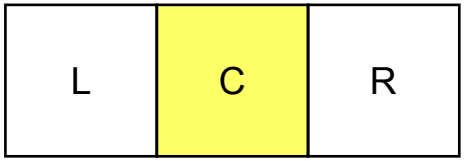
  C ← rule(C, N, E, S, W)

- allows the modeling of **complex global behaviours**.
- The model is **inherent massively parallel.**
- The cells may have a **complex state** (record) or may be considered as **objects**.
- **universal computation**

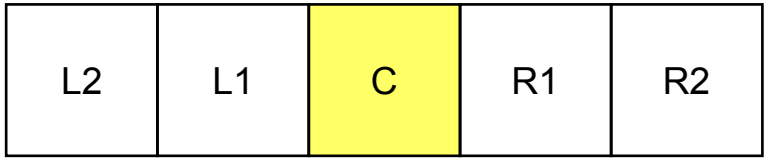
Neighborhood

TECHNISCHE UNIVERSITÄT DARMSTADT

1D

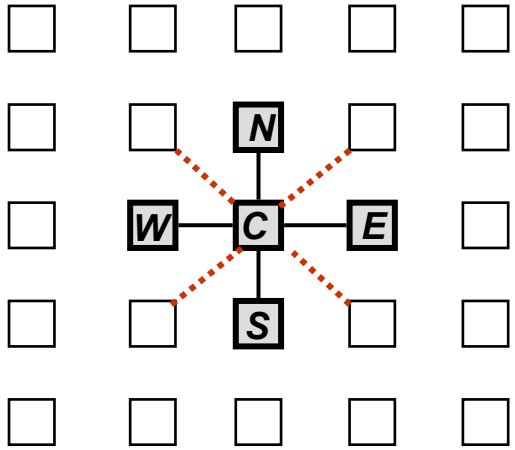


radius 1



radius 2

2D



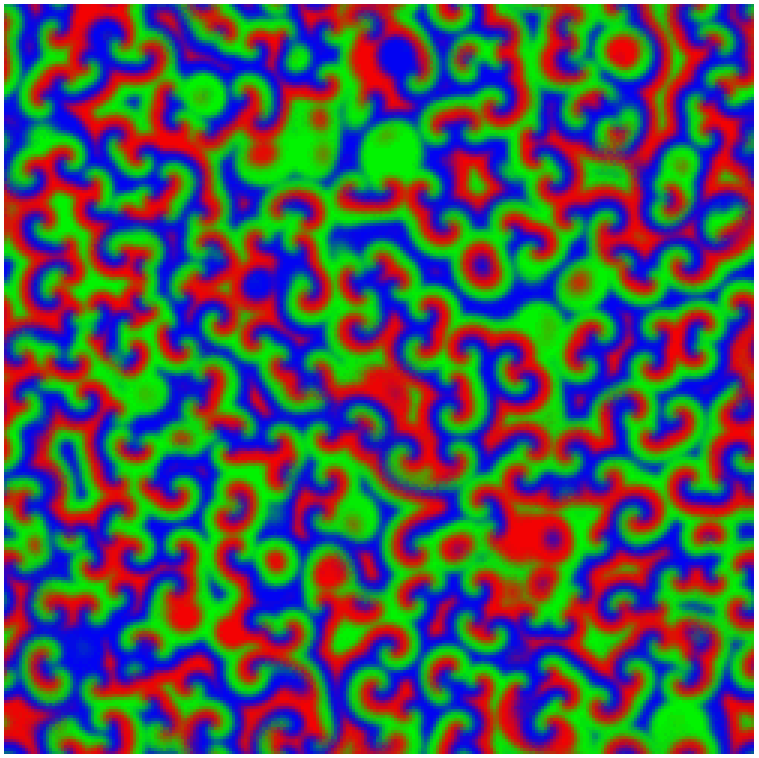
von Neumann Neighborhood
(4 nearest neighbors)

Moore Neighborhood
(8 neighbors)

Belousov-Zhabotinsky Reaction

TECHNISCHE UNIVERSITÄT DARMSTADT

Boris Pavlovich Belousov in 1930



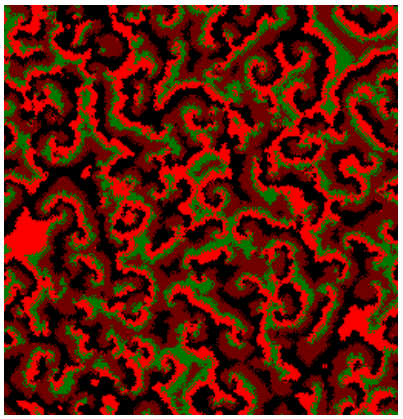
Boris P. Belousov (1959). "Периодически действующая реакция и ее механизм" [Periodically acting reaction and its mechanism]. Сборник рефератов по радиационной медицине. ***147: 145.***
Anatol M. Zhabotinsky (1964). "Периодический процесс окисления малоновой кислоты растворе" [Periodical process of oxidation of malonic acid solution]. Биофизика. ***9: 306–311***

Belousov-Zhabotinsky Reaction

TECHNISCHE UNIVERSITÄT DARMSTADT

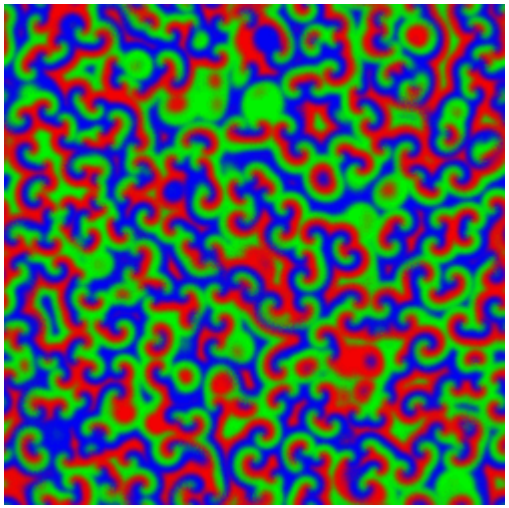
```
(1)  function fBZR(Cell, North, East, South, West);
(2)  const n=127;  g=11;
(3)  b:=(North=n) + (East=n) + (South=n) + (West=n);
(4)  a:=(0<North<n) + (0<East<n) + (0<South<n) + (0<West<n);
(5)  if (Cell=0)      then fBZR:= a/2 + b/2;
(6)  if (Cell=n)      then fBZR:= 0;
(7)  if (0<Cell<n)    then begin
(8)                          S:= (Cell+North+East+South+West);
(9)                          case a of
(10)                         0:     S:=S+g;
(11)                         1,2:   S:= S/2 +g;
(12)                         3,4:   S:= S/4 + g;
(13)                         end case

(14)                         if (S<n) then  fBZR:=S;
(15)                         else           fBZR:=n; // (S=n)
(16)               end
```



oscillating chemical process

*The number of **firing neighbours** with (state=n) is counted in b (3). The number of **excited neighbours** with (state=0..n-1) are counted in (a). If the cell state is zero, the next state depends on the number of neighbours which are firing or excited (5). If the cell is excited, cell states of the neighbours are added to S (8). If S>=n the next state will be n, otherwise it depends on a, g and S.*

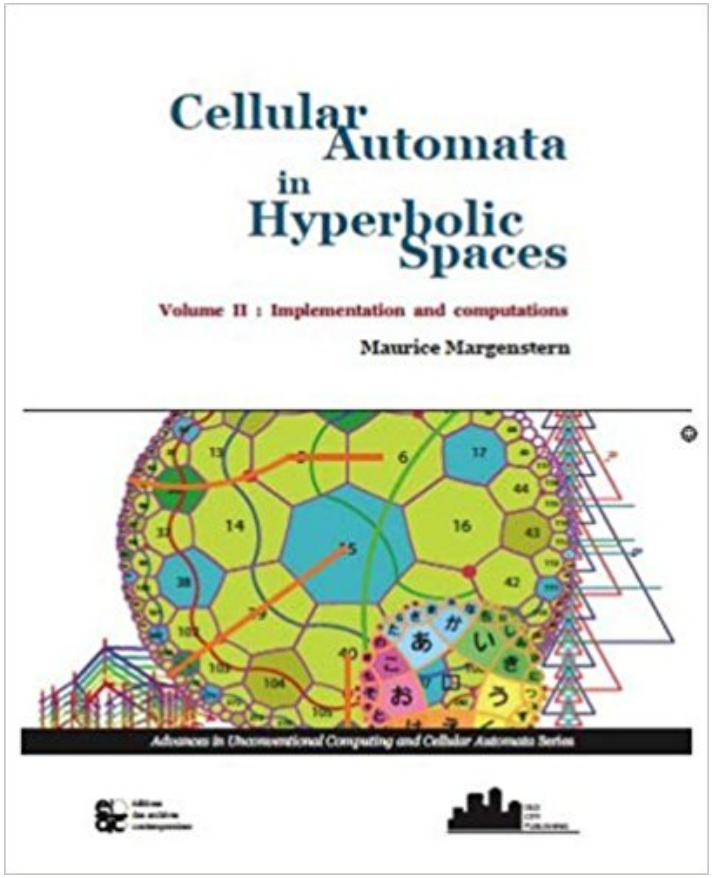


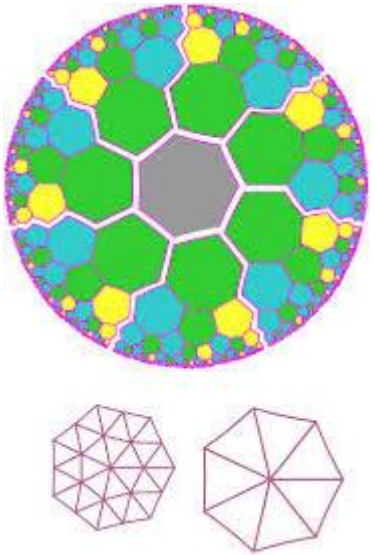
CA in Hyperbolic Space

TECHNISCHE UNIVERSITÄT DARMSTADT

Nikolai Lobachevsky

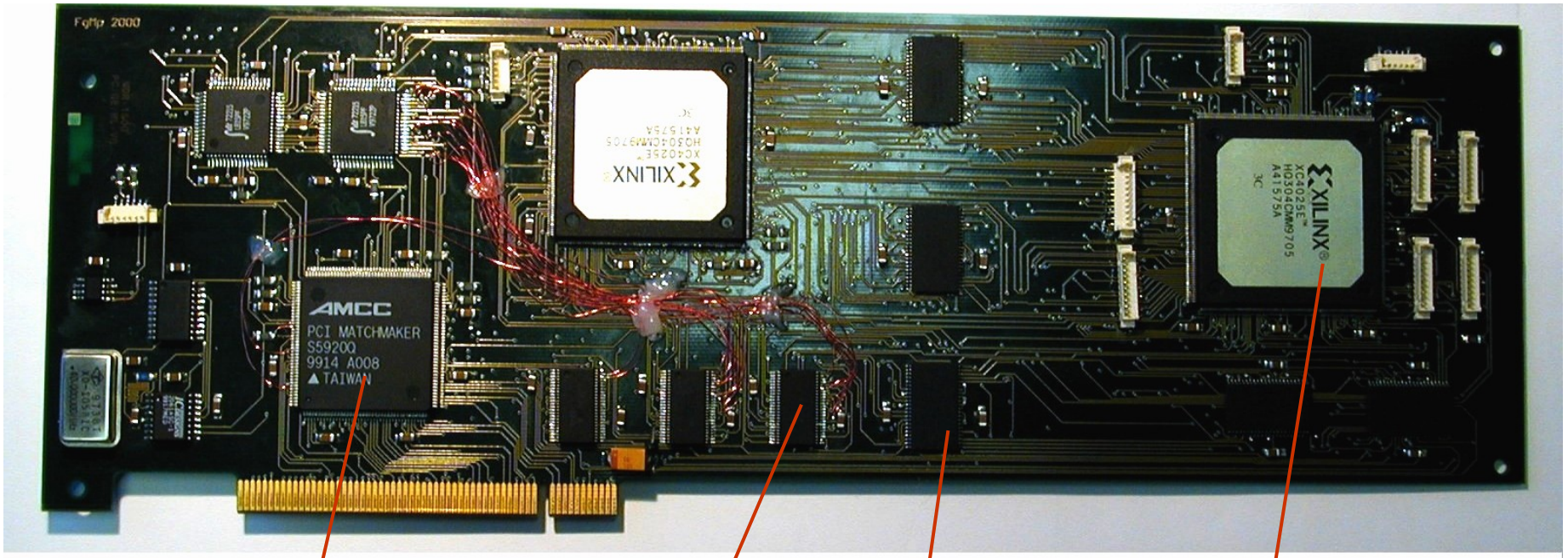
1792 –1856





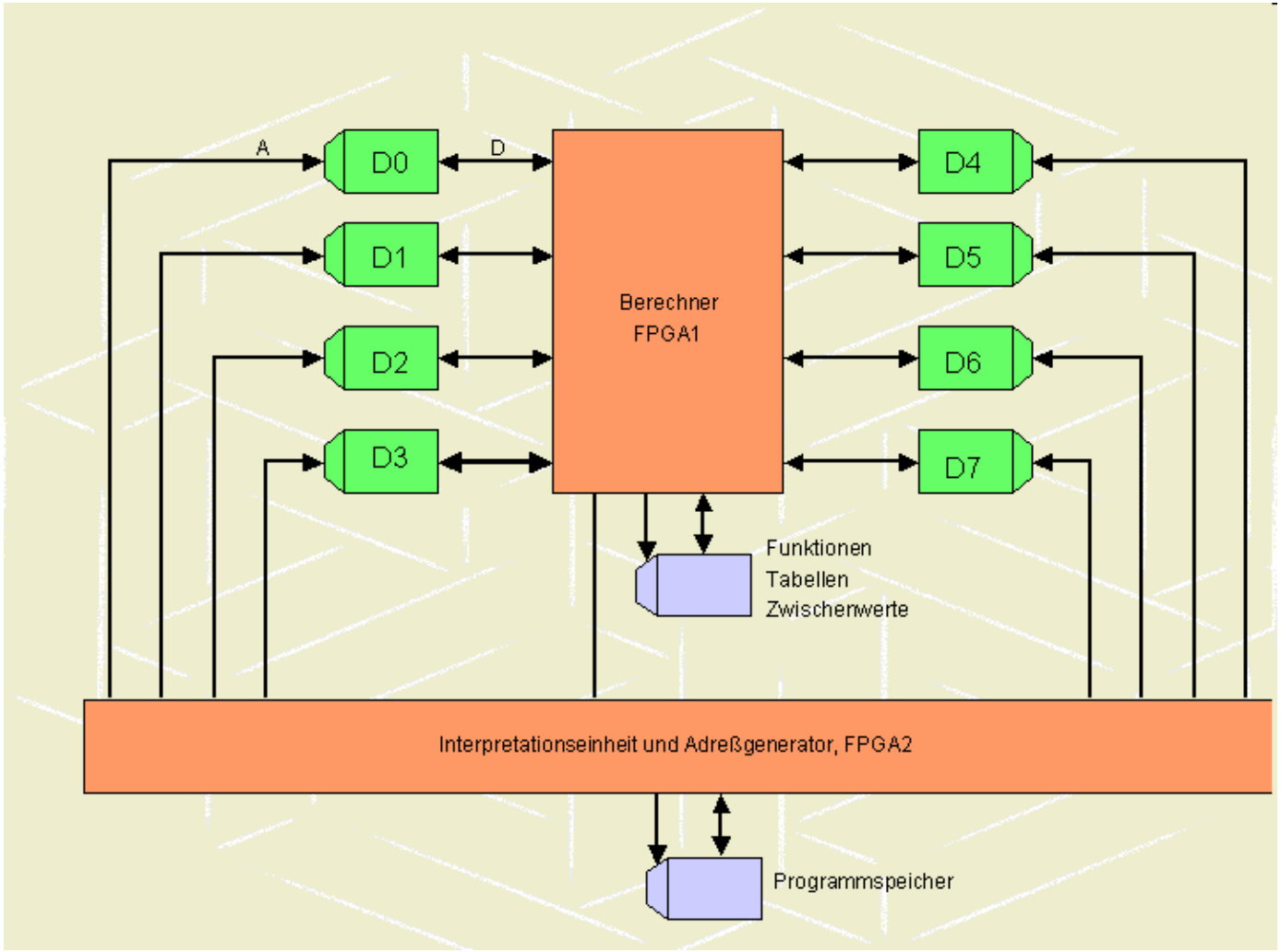
Hardware Support CEPRA-S, 2001

TECHNISCHE UNIVERSITÄT DARMSTADT



Hardware Architecture CEPRA-S, 2001

TECHNISCHE UNIVERSITÄT DARMSTADT

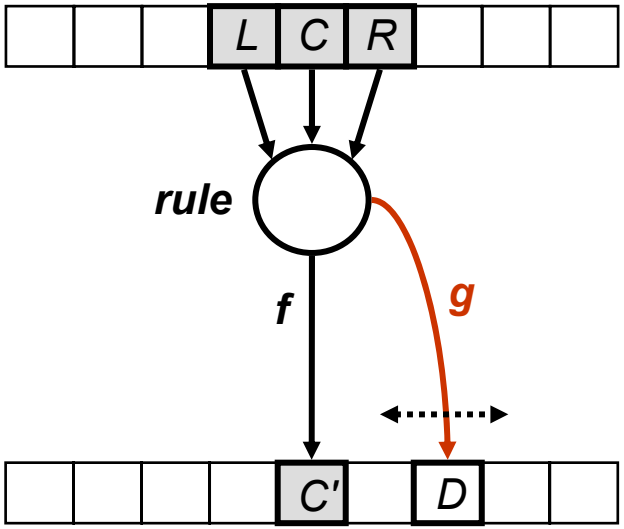


2.2 Cellular Automata with Write-Access (CA-w)

2.2 Cellular Automata with Write-Access (CA-w)

TECHNISCHE UNIVERSITÄT DARMSTADT

Idea



not only the own state but also states in the neighborhood can be modified

CA-w
Why Write-Access to Neighbor?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Writing to a neighbor can be **simulated by CA**
- But certain systems with sparse activities (agents, particles, ..) can be described **more naturally** and **more compactly**
- Write-access is an essential feature allowing to **activate** or **deactivate** cells in a decentralized way
  - **activity can be dynamically controlled**
  - **inactive cells**
    - **need not to be computed**
    - **don‘t consume power**
    - **computational resources of inactive cells may be used otherwise**
- Drawback: **Write-conflicts** may occur!

CA-w
Write-Conflicts

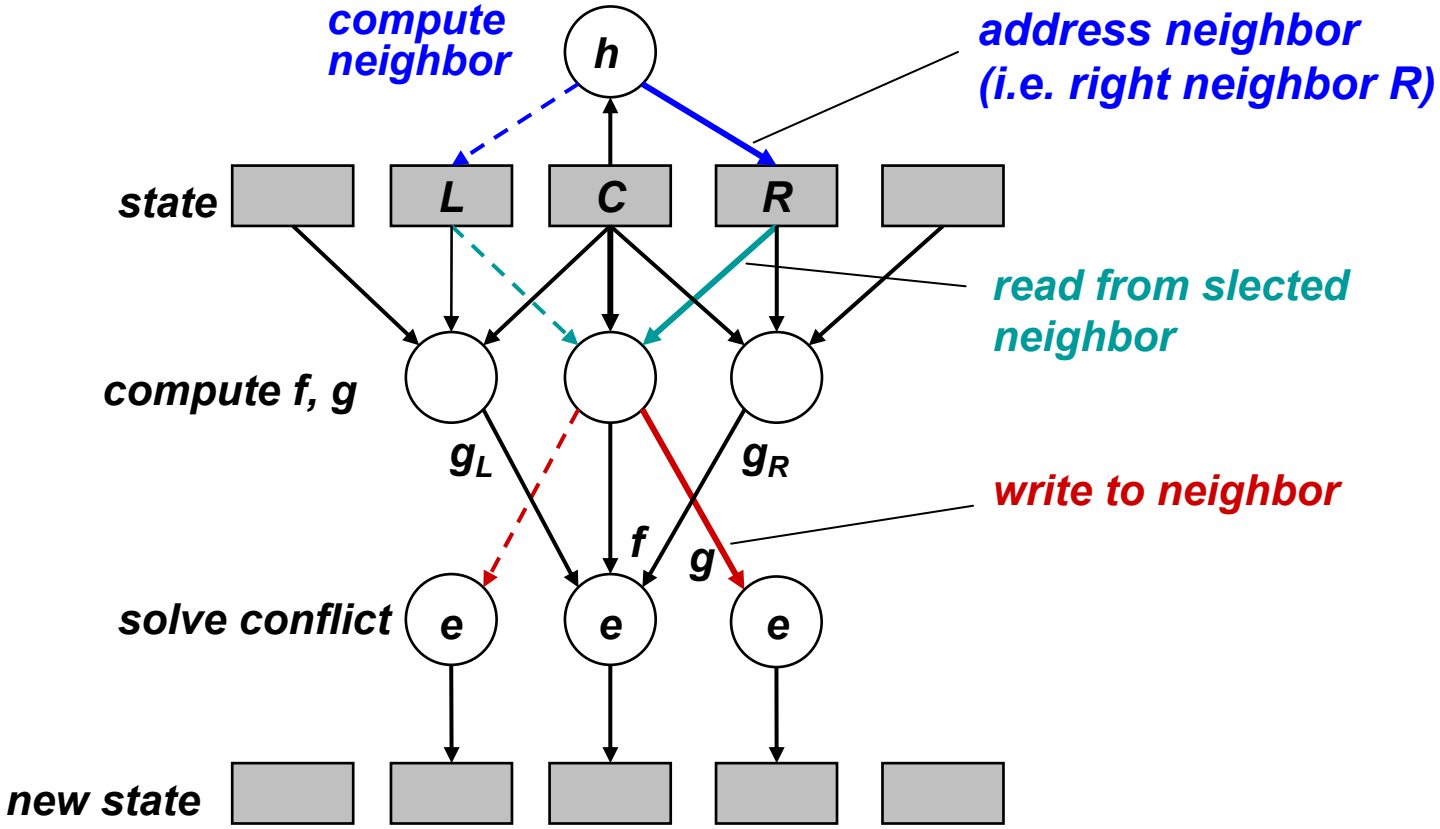
TECHNISCHE
UNIVERSITÄT
DARMSTADT

- **Exclusive write**
  - might be guaranteed by the algorithm
  - try to find **conflict-free algorithms**

- **Concurrent write**
  - extra logic for detection and resolution
  - resolution strategies
    - OR, MAX, ARBITRARY, PRIORITY, REFUSE, …

CA-w With Neighborhood Radius = ± 1

TECHNISCHE UNIVERSITÄT DARMSTADT



CA-w
Related Work

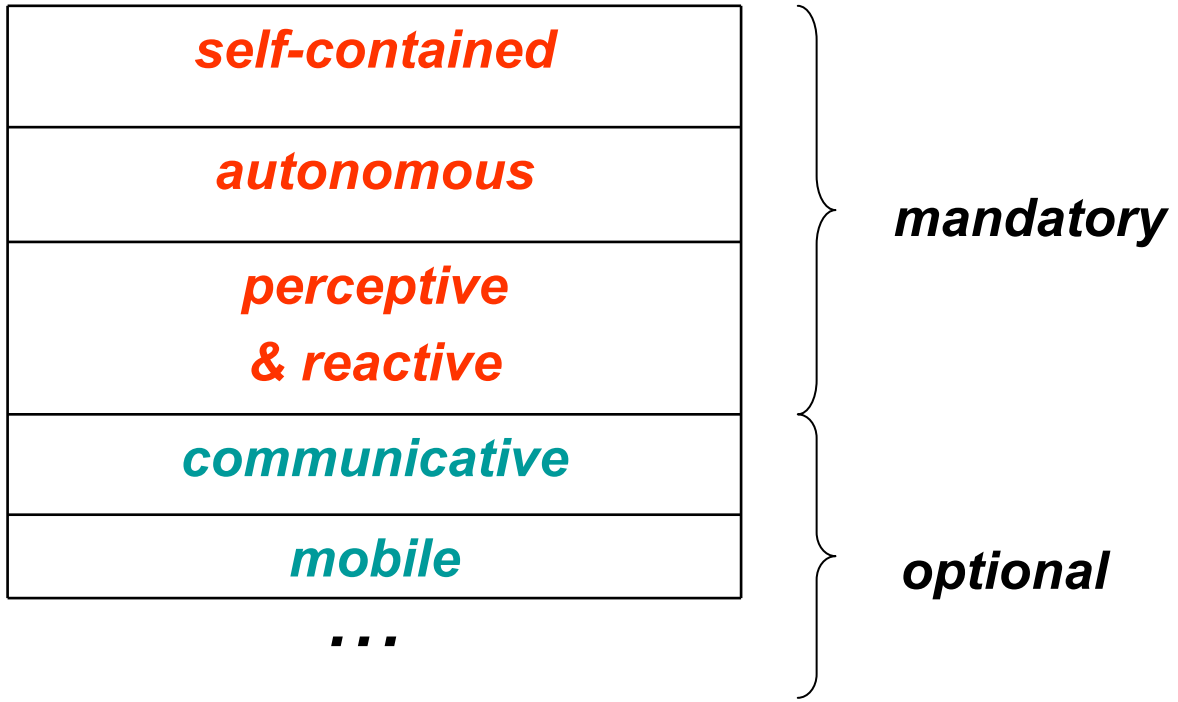
TECHNISCHE UNIVERSITÄT DARMSTADT

- **PSA**: **P**arallel **S**ubstitution **A**lgorithms
  - rules: f(BaseCells, ContextCells) → TargetCells
  - very general model based on sets of cells (name, data)
  - 1994 Achasova, S., Bandman, O., Markova, V., Piskunov, S.
- **CRCW-PRAM**: **C**oncurrent-**R**ead-**C**oncurrent-**W**rite **P**arallel **R**andom **A**ccess **M**achine
  - *p* processors have access to a common global memory and update synchronously the memory cells
  - **PRAM**
    - **physical view of *p* processors**
    - **global access to the shared memory**
  - **CA-w**
    - **logical view of *n* cells**
    - **memory distributed and local to the cells**
    - **local access to the cells**

3. CA Agents

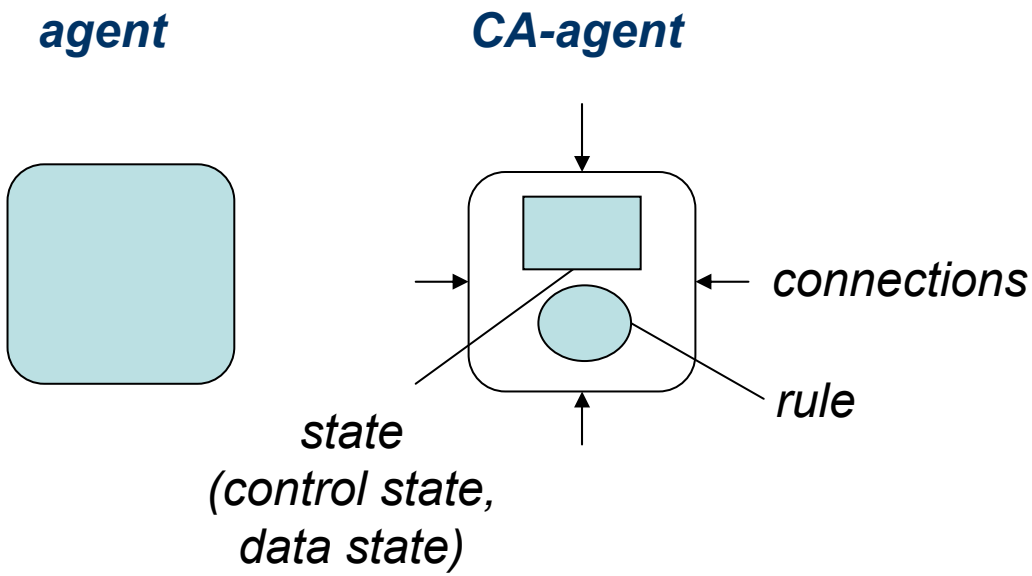
What are the Properties of an Agent?

TECHNISCHE UNIVERSITÄT DARMSTADT



Self-contained

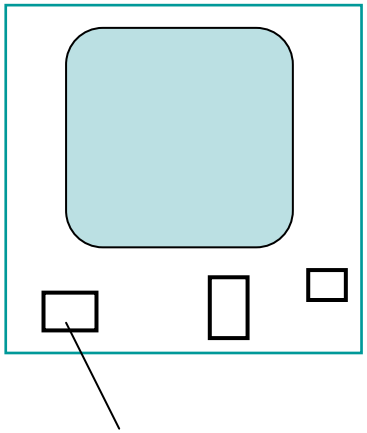
- is a discrete **individual**, **complete in itself**, containing all necessary working parts, identifiable



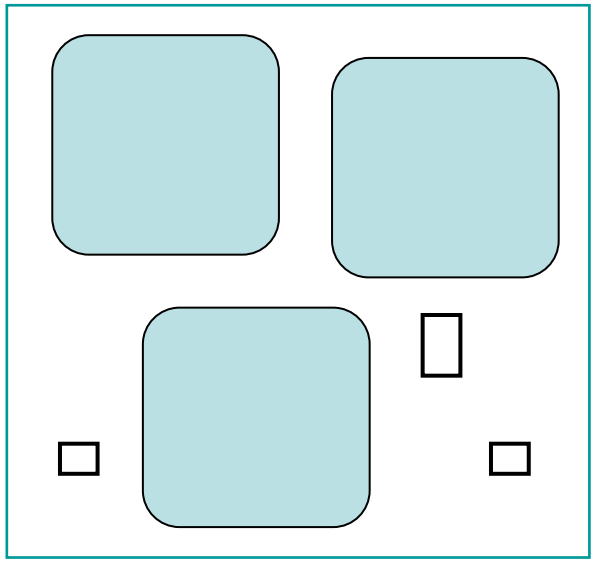
Agents, Embedded in the CA Cell Space

TECHNISCHE UNIVERSITÄT DARMSTADT

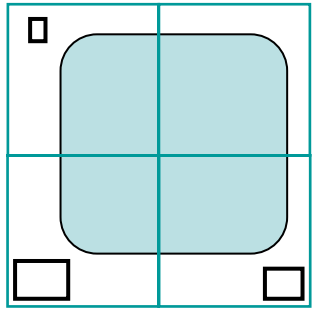
1 agent per CA cell



3 agents per CA cell



One agent modeled by 4 cells



agent is part of a CA cell

agent is distributed over several cells

Autonomous
αὐτονομία

TECHNISCHE UNIVERSITÄT DARMSTADT

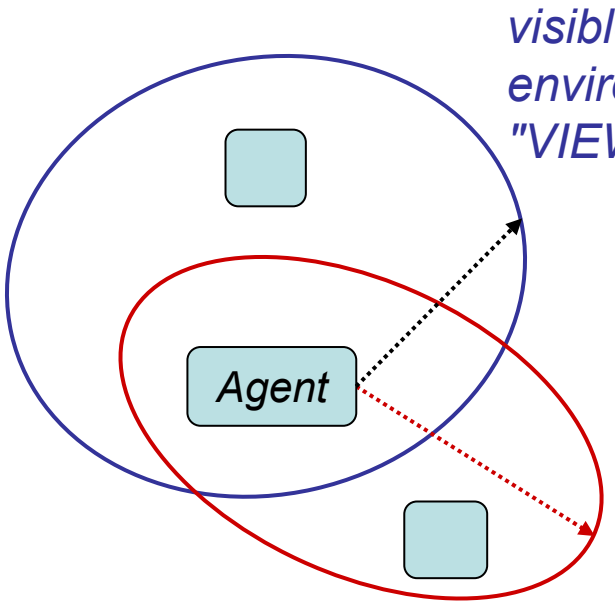
- **αὐτο = self, νομία=law**
  "one who gives oneself their own law"

- (*philosophy*) capacity of a rational individual to make an informed, un-coerced decision

- (*mechanics*) **The capacity of a system to make a decision about its actions without the involvement of another system or operator**.
  - (*robotics*) "autonomy“ means independence of central control.
  - **(*CA agent*) acts according to its own rule**

Perceptive & Reactive

TECHNISCHE UNIVERSITÄT DARMSTADT



- can perceive information about the environment (VIEW)
  - **CA**: read states of neighbors
- reacts on the perceived information (RANGE)
  - **CA**: a cell can only modify its own state
  - **CA-w**: a cell is allowed to change the state of a neighbor

Communicative

TECHNISCHE UNIVERSITÄT DARMSTADT

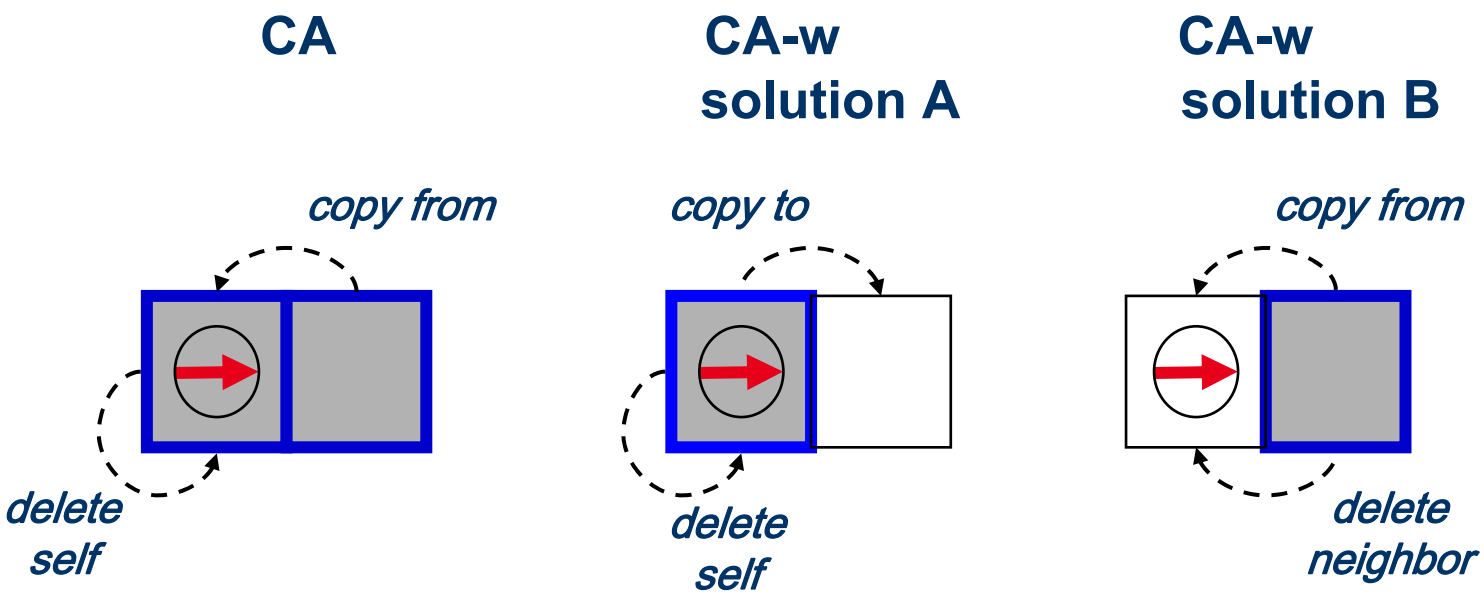
social ability, interaction with other agents, cooperation & competition

- **CA**: only by local read, e.g.
  - read the color of another agent
  - notice that there is another agent in front

- **CA-w**: an agent may also change the state of a neighbor, send messages actively
  - “let us work together”
  - “take care, I am stronger”

Mobile: Modeling the Moving of Agents

TECHNISCHE UNIVERSITÄT DARMSTADT



	CA	CA-w solution A	CA-w solution B
source active	x	x	
target active	x		x

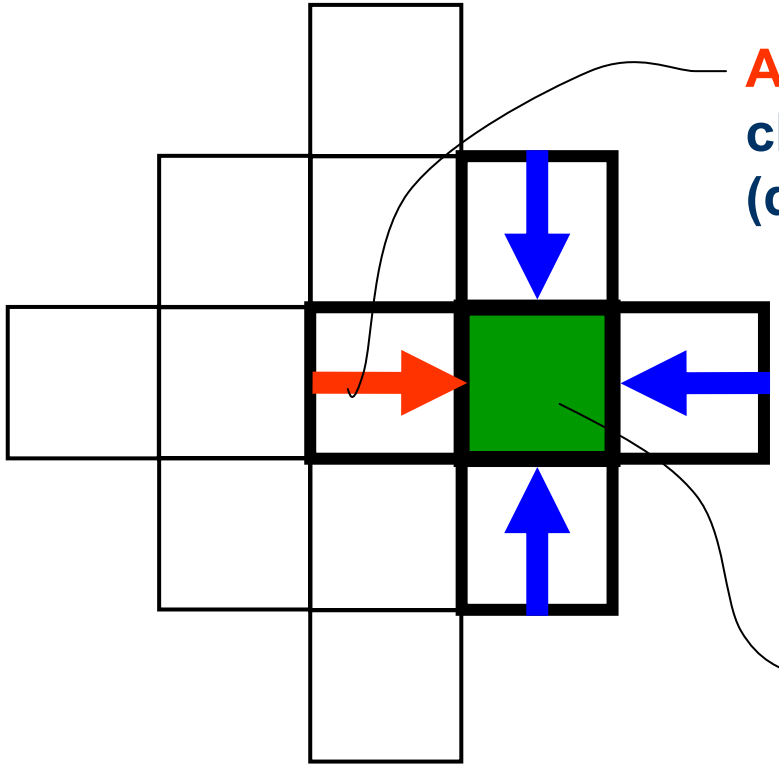
couple of two consistent rules

push rule (by source)

pull rule (by target)

How Conflicts are Detected

TECHNISCHE UNIVERSITÄT DARMSTADT



AGENT-Cell
checks 4 cells in viewing direction
(dynamic neighborhood, MH-distance 2)

EMPTY-Cell
checks N, E, S, W

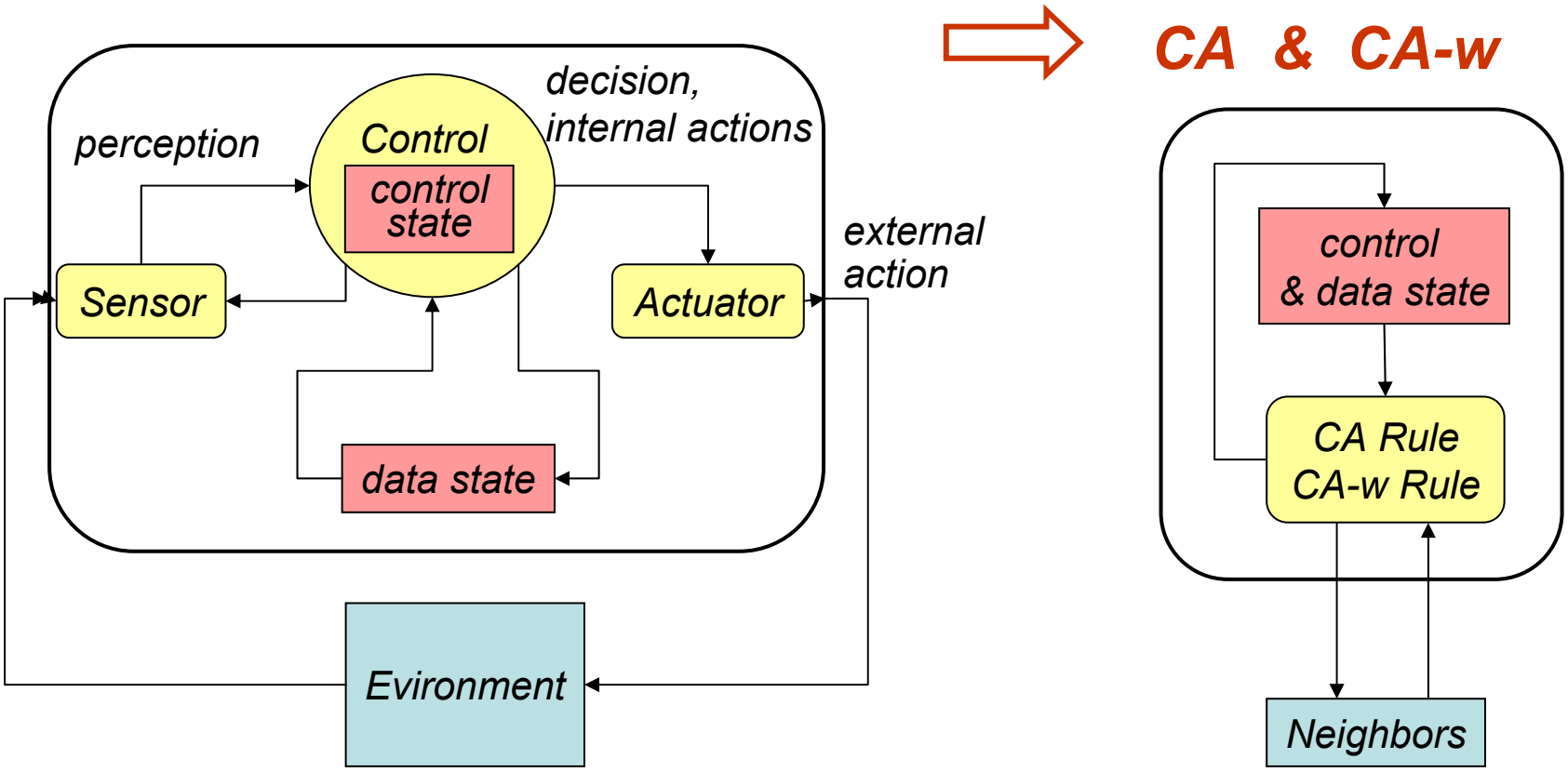
Conflict Resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1 phase model
  - requires an extended neighborhood
  - agent has to compute also the decisions of the other agents which want to move to the same target.
- 2 or 3 phases
  - compute decision (move direction)
  - arbitration (select an agent)
  - movement

Possible Structure of an Agent

TECHNISCHE UNIVERSITÄT DARMSTADT



TECHNISCHE
UNIVERSITÄT
DARMSTADT

4. Multi-Agent-Systems, Examples

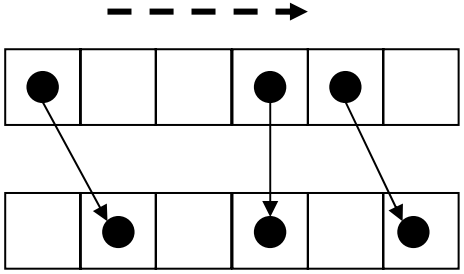
4.1 CA-w Algorithms

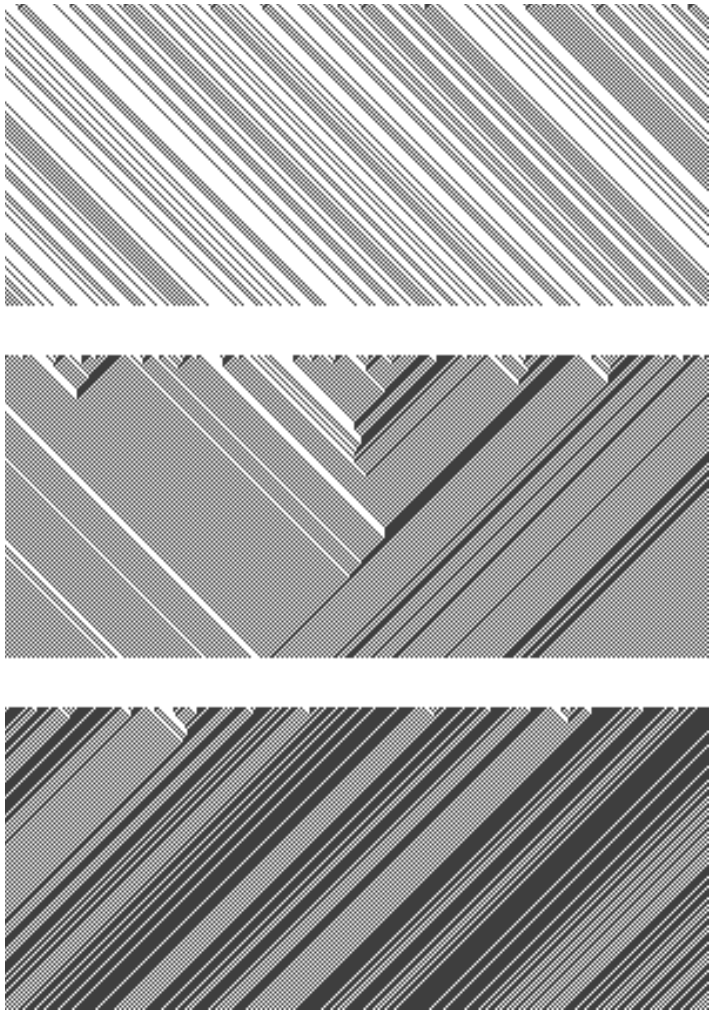
- Traffic Rule
- Leader Election

4.2 Creatures‘ Exploration, All-to-All Communication

4.1 CA-w Algorithms
Traffic Rule

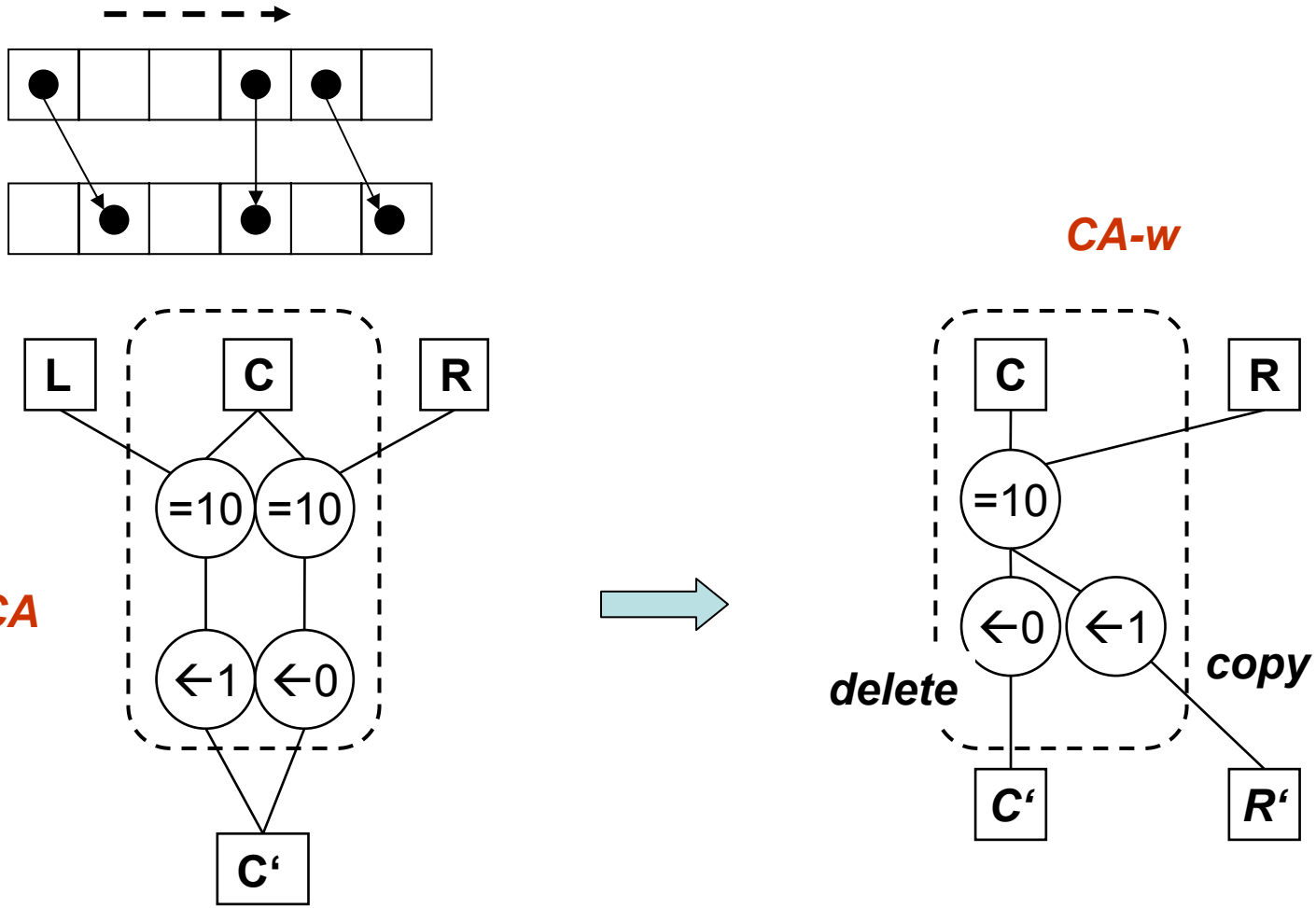
CA Rule 184: Traffic Rule





CA Rule 184: Traffic Rule

TECHNISCHE UNIVERSITÄT DARMSTADT

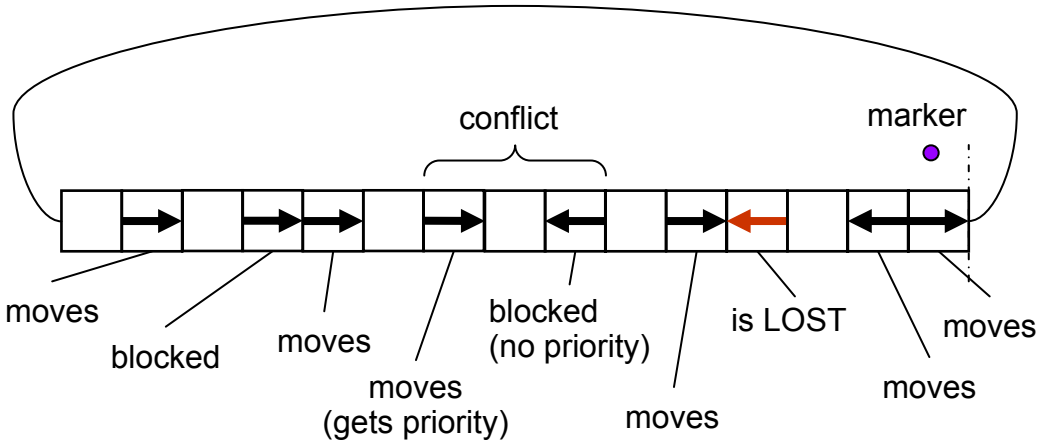


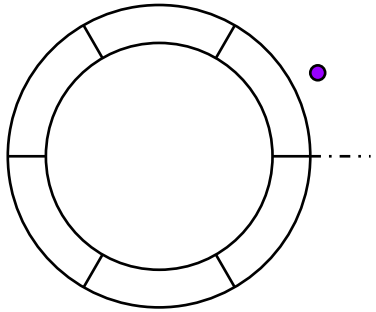
4.1 CA-w Algorithms
Leader Election

Initially p agents are in the same initial state (state = 0). The goal is to find in a decentralized way a terminal configuration in which only one agent is in the state leader (state = 3) and the others are in the state lost (state = #).

Leader Election
Initially: Random Directions

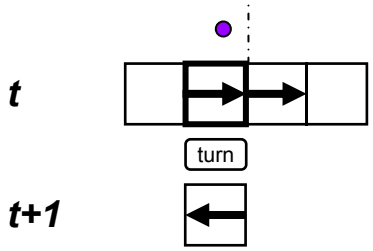
TECHNISCHE UNIVERSITÄT DARMSTADT



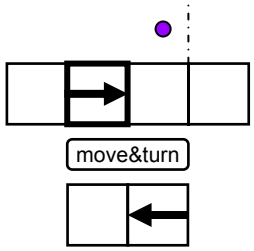


Situations and Solutions

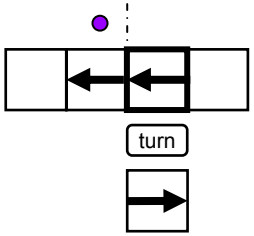
TECHNISCHE UNIVERSITÄT DARMSTADT



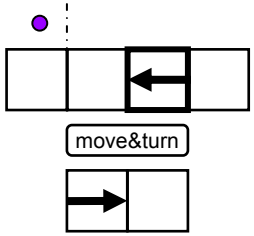
turned



moved & turned

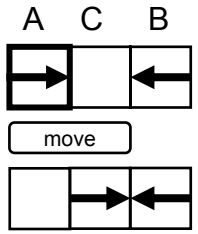


turned



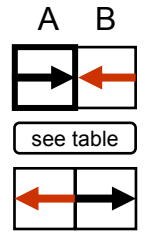
moved & turned

conflict



priority left

swap



one of the agents will be in lost state

Leader Election Simulations

TECHNISCHE UNIVERSITÄT DARMSTADT

```
      initial config:0      initial config:4      initial config:10     initial config:12
i=    0  1  2  3  4  5      0  1  2  3  4  5      0  1  2  3  4  5      0  1  2  3  4  5

t= 0  0>                    0> 0> 0>              0> 0< 0>        0<    0> 0> 0> 0> 0>
t= 1     0>                 0> 0>    0>           #> 0>    0> 0<        0> 0> 0> 0>    2<
t= 2        0>              0>    0>    0>        #>    0> #< 0>        0> 0> 0>    0> 2<
t= 3           0>                 0>    0>  2<      #> #< 0>    2<     0> 0>    0> 2< #<
t= 4              0>                 0>  0> 2<      #< #>    0> 2<     0>    0> 2< #> #<
t= 5                 2<                 0> 2< #<  #>       #> 2< #<        0> 2< #> #< #<
t= 6              2<                    2< #> #<     #>    2< #> #<        2< #> #< #> #<
t= 7           2<                       2<    #< #<        #> 2< #< #<  3>    #< #> #< #<
t= 8        2<                       2<    #<    #<           2< #> #< #<
t= 9     2<                       3>    #<    #<              2<    #< #> #<
t=10  3>                                                     3>    #<    #< #<

      initial config:13     initial config:16     initial config:17     initial config:19

t= 0  0> 0> 0> 0> 0> 0<     0< 0< 0>                 0< 0< 0> 0>           0> 0> 0> 0> 0> 0>
t= 1  0> 0> 0> 0> #< 2<        0<    0>    0<     1>    0< 0>    2<        0> 0> 0> 0> 0> 2>
t= 2  0> 0> 0> #< 0> 2<     1>          0> 0<        1> 0<    0> 2<        0> 0> 0> 0> 2< #<
t= 3  0> 0> #< 0> 2< #<        1>       #< 2<        #< 1>    2< #<        0> 0> 0> 2< #> #<
t= 4  0> #< 0> 2< #> #<           1> #<    2<     #>       1> 2< #<        0> 0> 2< #> #< #<
t= 5  #> 0> 2< #> #< #<           #< 1> 2<           #>    #< 1> #<        0> 2< #> #< #> #<
t= 6  #> 2< #> #< #> #<        #<    #< 1>              #> #< #< 3<     3> #> #< #> #< #<
t= 7  3> #> #< #> #< #<     #>    #<       3<
```

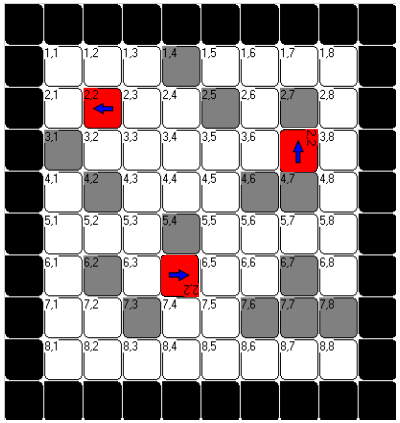
state ∈ {0, 1, 2, 3, #} ≡ {init, leftborder, rightborder, leader, lost}

4.2 CEP: Creature‘s Exploration Problem

CEP: Creature‘s Exploration Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Given is a two-dimensional grid of cells. The cells are either of type BORDER, OBSTACLE, EMPTY or CREATURE. The CREATURE is able to look one cell ahead in a certain direction and can move in this direction if there is no obstacle or border. The task of the creature is to **visit all empty cells with a minimum number of steps.**
- Find the optimal behavior of the creature
- Practical Applications
  - Mowing a lawn in shortest time
  - Vacuum cleaning a room by a robot
  - Exploring an unknown environment
  - Distribute information

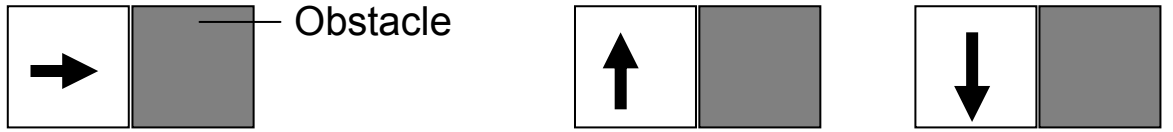


Rolf Hoffmann and Smitha Sambharaju: *Optimal Algorithm for Checking the Environment by a Moving Creature*, Report RA-1-2003, Fachgebiet Rechnerarchitektur - Fachbereich Informatik - Technische Universität Darmstadt, March 2003

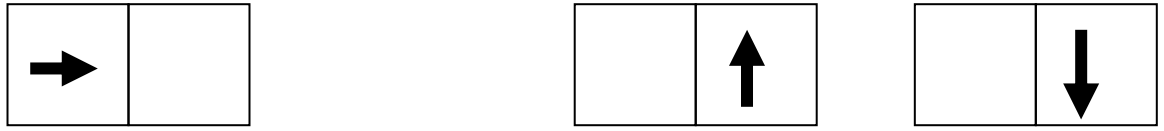
The Actions of the Creature

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Actions**
  - **R / L**: turn right / left,
  - **Rm / Lm** : (Right/Left & Move) turn right / left & move forward



(a) if not free then L or R

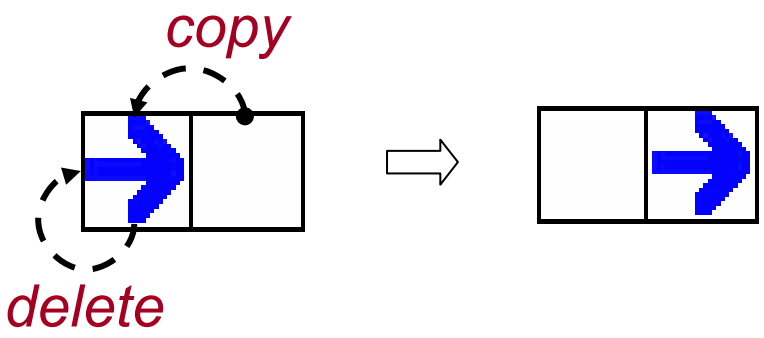


(b) if free then Lm or Rm

Moving of an Agent

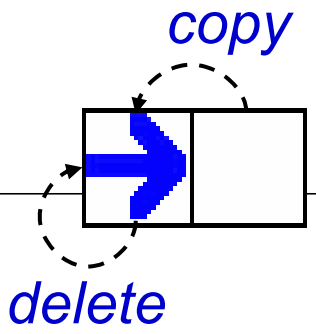
TECHNISCHE UNIVERSITÄT DARMSTADT

move forward



use a couple of consistent CA rules

CA Rule for Empty Cell



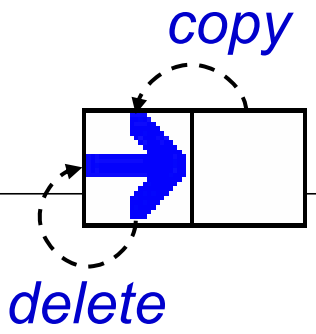
TECHNISCHE UNIVERSITÄT DARMSTADT

Rule for My.Type=EMPTY

(a1) {CASE Free}

if (Neighbor.Type = CREATURE) **and** (Neighbor.Direction points to me)
then
My.Type:= CREATURE // ***create (move by copy)***
Myl.Direction:= TurnRight/Left(Neighbor.Direction) ***// new direction***

CA Rule for Creature Cell



TECHNISCHE UNIVERSITÄT DARMSTADT

Rule for Cell.Type=CREATURE

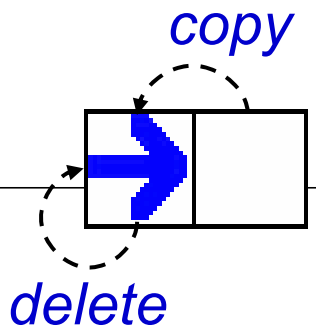
(a2) {CASE Free}

if (Neighbor.Type= EMPTY) **then** My.Type:= EMPTY *// **delete***

(b) {CASE not Free}

My.Direction:= TurnRight/Left(Cell.Direction) *// **only turn R, L***

CA Rule for Empty&Creature



TECHNISCHE UNIVERSITÄT DARMSTADT

Rule for My.Type=EMPTY

(a1) {CASE Free}

if (Neighbor.Type = CREATURE) **and** (Neighbor.Direction points to me)
then
My.Type:= CREATURE // *create (move by copy)*
Myl.Direction:= TurnRight/Left(Neighbor.Direction) // *new direction*

Coupled Action

Rule for Cell.Type=CREATURE

(a2) {CASE Free}

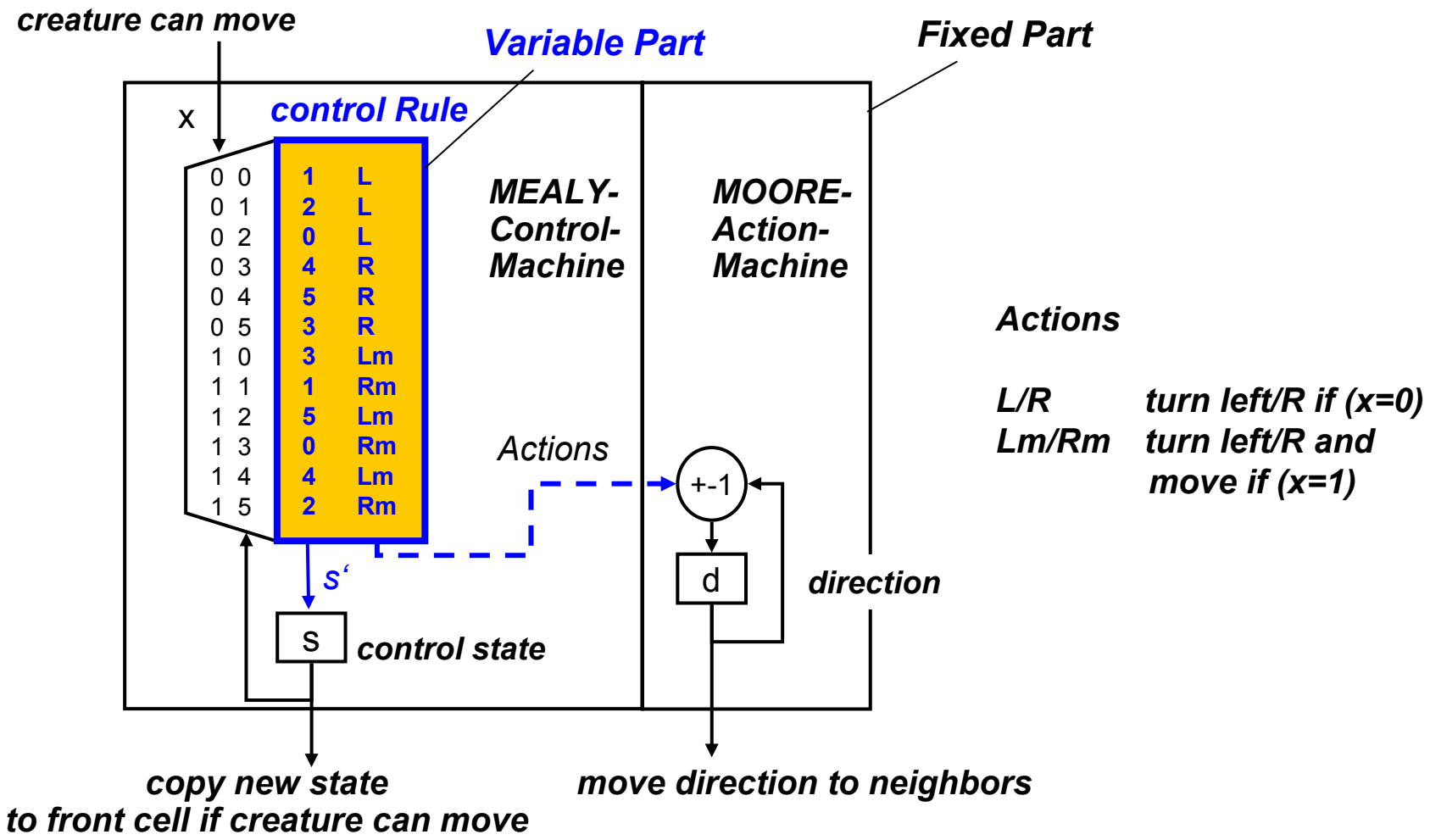
if (Neighbor.Type= EMPTY) **then** My.Type:= EMPTY // *delete*

(b) {CASE not Free}

My.Direction:= TurnRight/Left(Cell.Direction) // *only turn R, L*

Control Machine and Action Machine

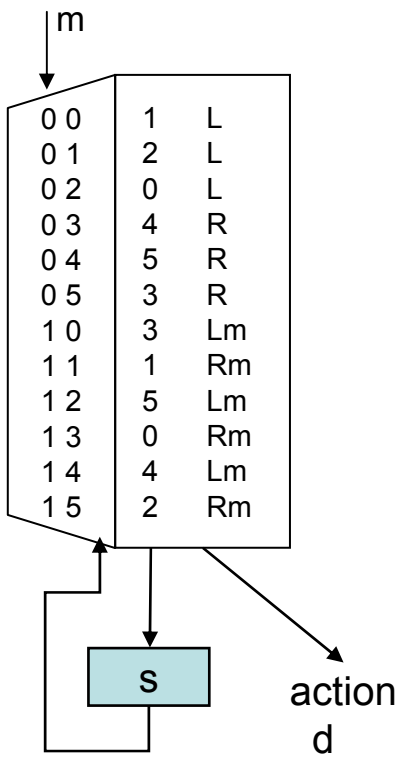
TECHNISCHE UNIVERSITÄT DARMSTADT

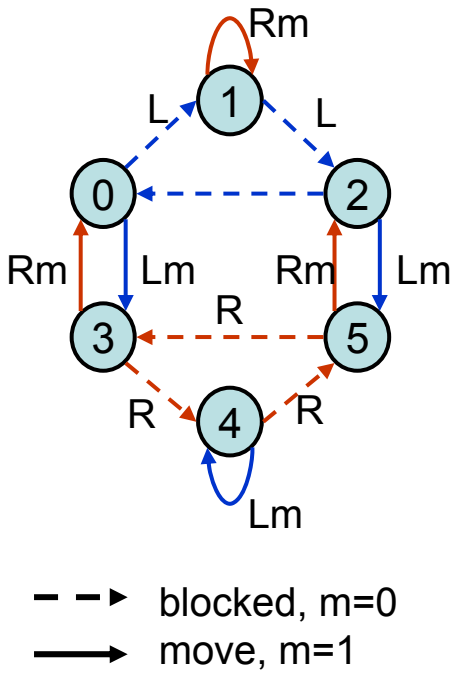


Representation of a State Algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

state/output-table ←→ state/output-graph





Number of State Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

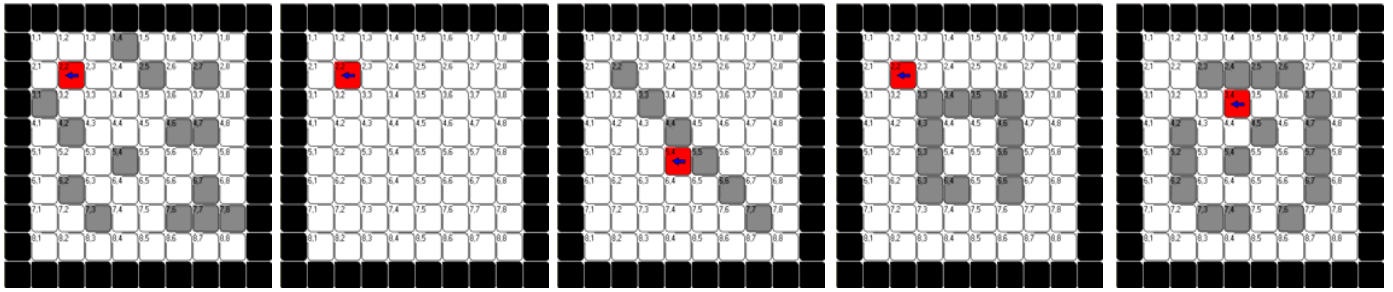
- M = Number of all control algorithms (Mealy state machines), represented by a state table
  - #s: number of states
  - #x: number of input values (blocked or free e.g.→ 2)
  - #y: number of actions (turn left or right e.g.→ 2)

- **$M = (\#s\ \#y)^{(\#s\ \#x)}$**

- **#y =2**
  - #s = 2 → M = 256
  - **#s = 4 → M = 134 217 728**

Initial Configuratons (Test Set)

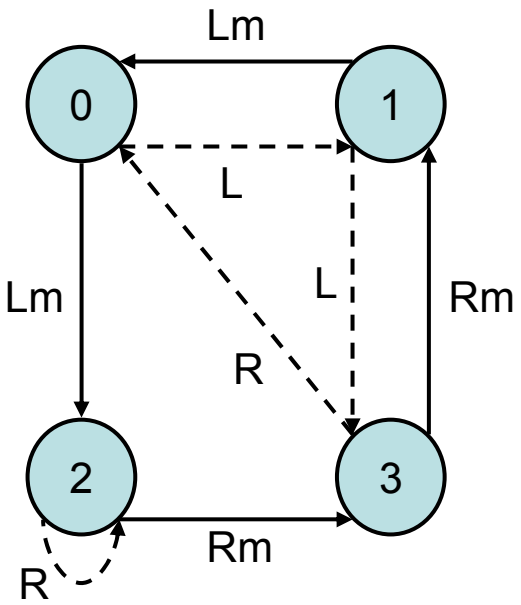
TECHNISCHE UNIVERSITÄT DARMSTADT



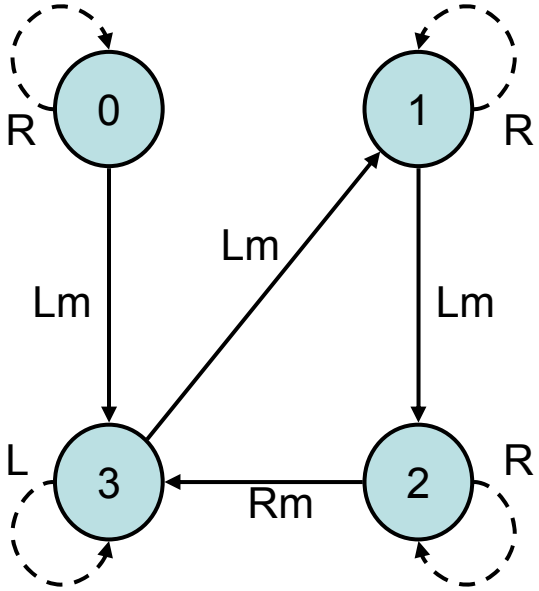
1 2 3 4 5

The Best 4-State-Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

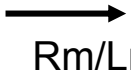


Algorithm A



Algorithm B

creature cannot move, turns Right or Left



creature can move and turns Right or Left

Visited Cells

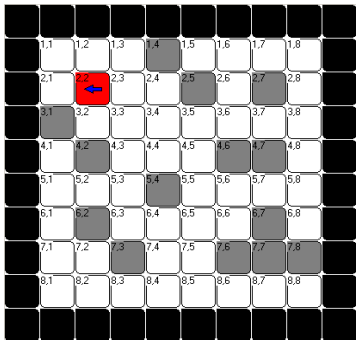
TECHNISCHE UNIVERSITÄT DARMSTADT

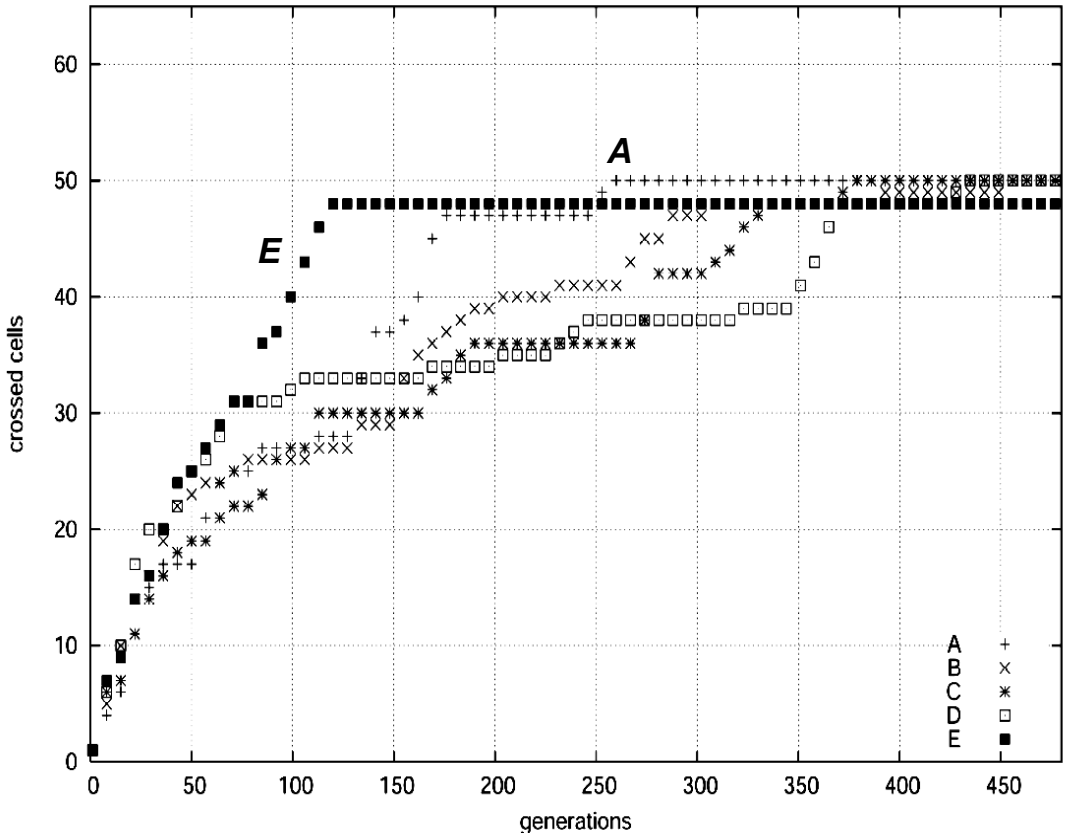
	Initial Configurations					
	#1	#2	#3	#4	#5	% avr.
maximum	50	64	58	53	48	
algorithm A	50	60	56	53	48	97 %
algorithm B	50	64	58	50	48	97 %
algorithm C	50	60	58	48	48	96 %
algorithm D	50	60	58	47	48	95 %
algorithm E	48	60	47	48	48	91 %

Number of visited cells

Speed of the Algorithms A-E for the First Initial Configuration

TECHNISCHE UNIVERSITÄT DARMSTADT



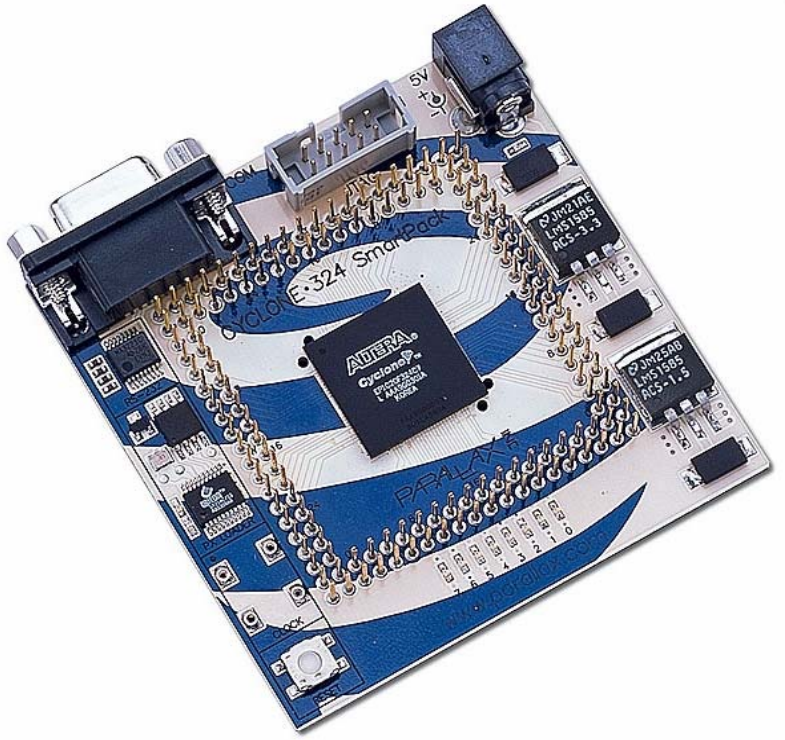


	(high)				(low)
Speed	**E** →	**A** →	**D** →	**B** →	**C**
Visited	**96%**		**100%**		

Used Hardware Platform, Different Architectures

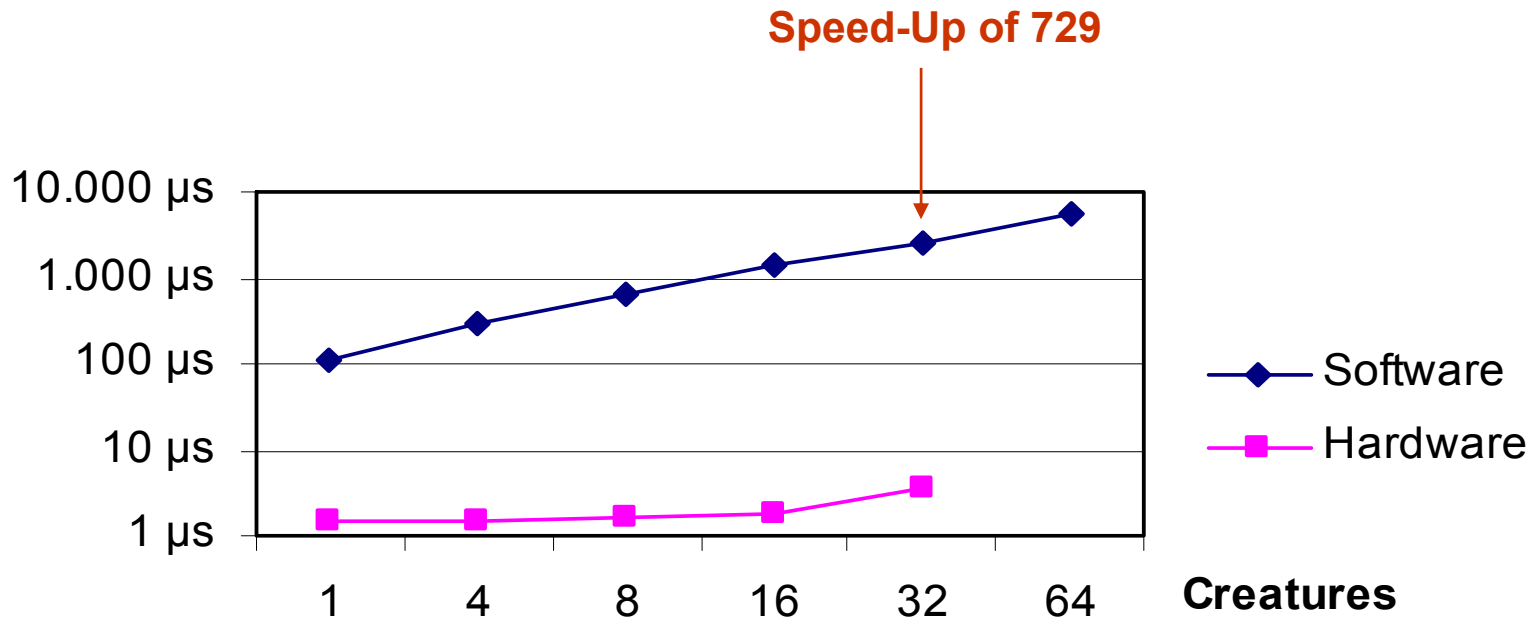
TECHNISCHE UNIVERSITÄT DARMSTADT

- FPGA Altera Cyclone EP1C20F324C7
- Altera Quartus II Tools
  - VERILOG → FPGA-Logic
- Serial Interface
  - connected to a PC
  - for control and data communication
- Computation in Hardware
  - algorithm search, simulation and evaluation in parallel
  - Hardware/Software Speed-Up: 100 to 10 000 depending on FPGA



Mathias Halbach, Rolf Hoffmann: *Parallel Hardware Architecture to Simulate Movable Creatures in the CA Model* , 9th International Conference, PaCT 2007 September 3-7, 2007, Pereslavl-Zalessky, Russia

Hardware vs. Software



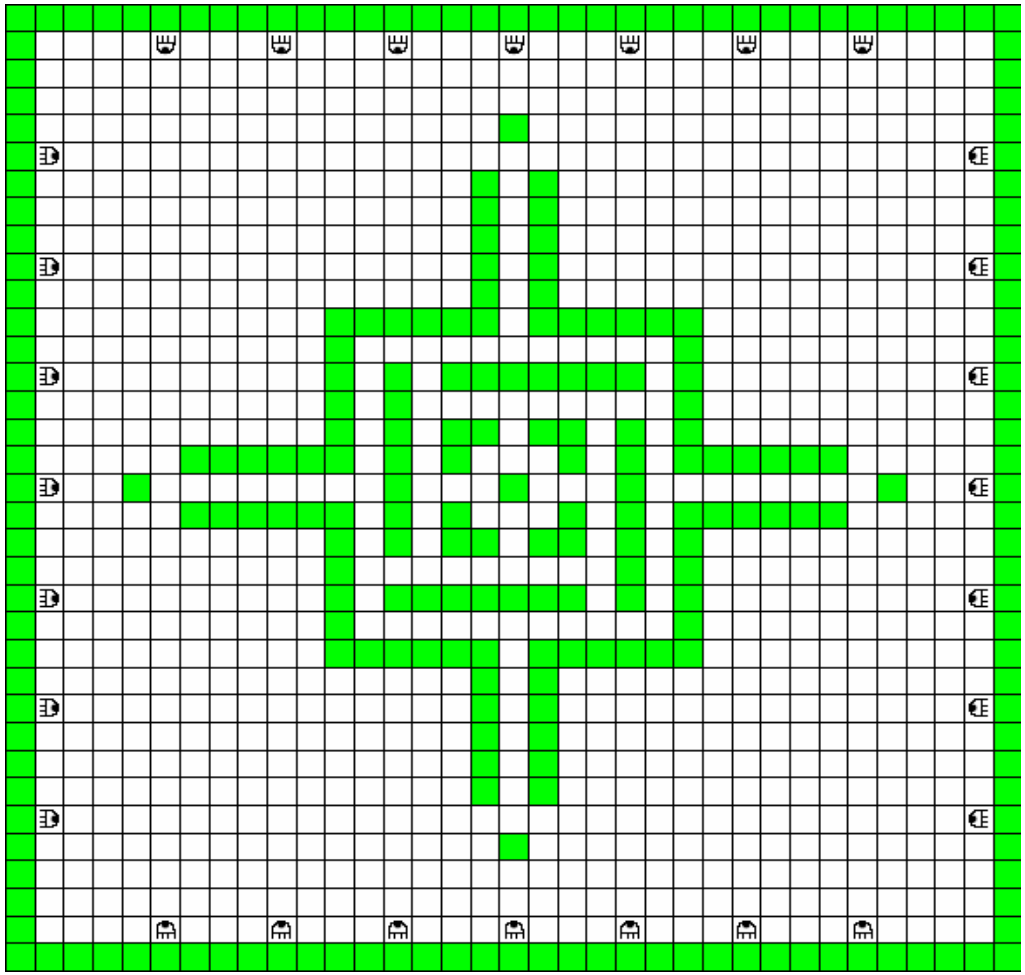
CEP Simulation Sequence

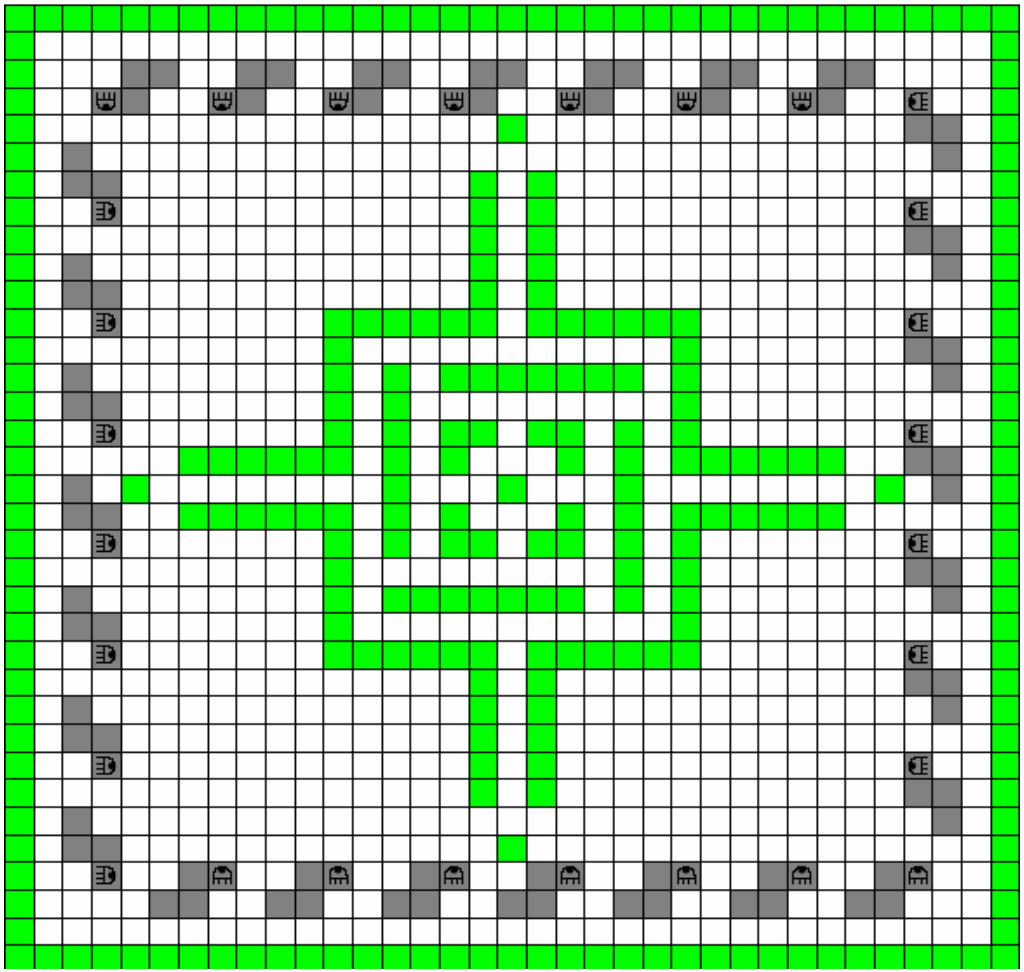
28 creatures, optimal 6-state Algorithm

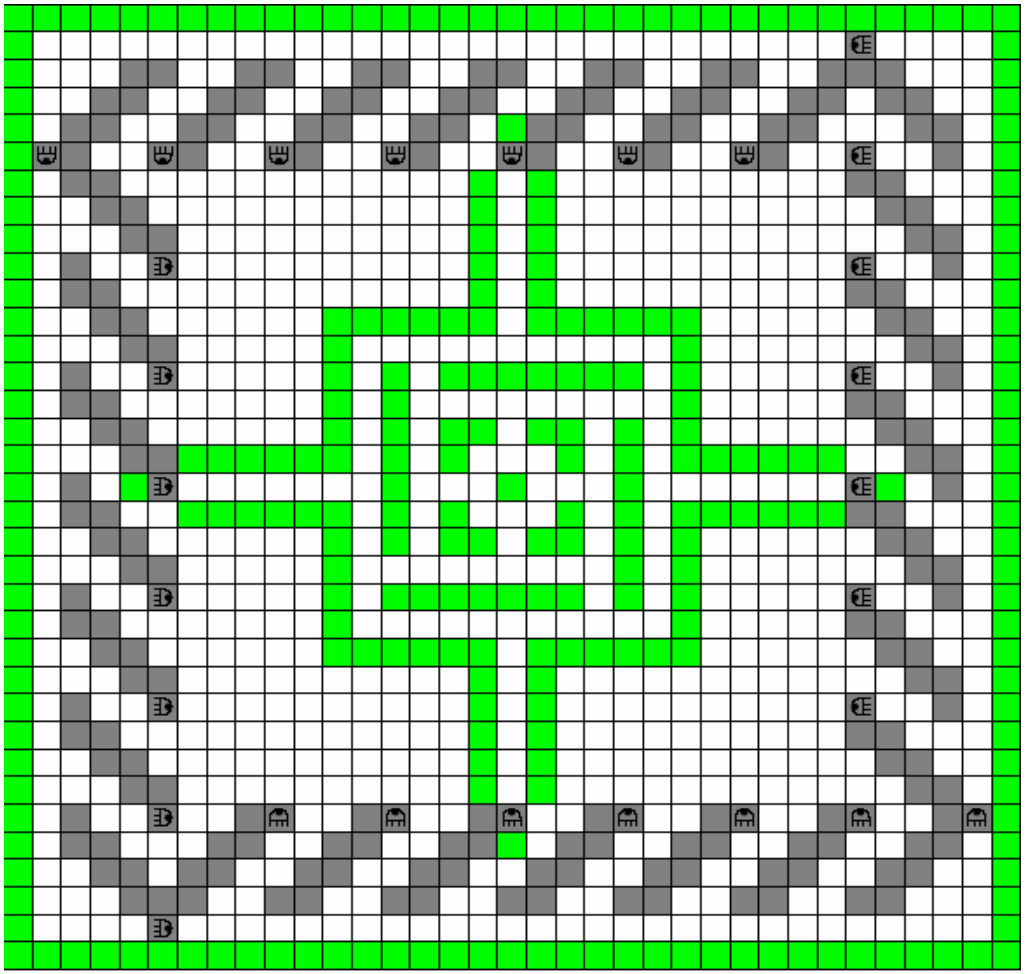
- Mathias Halbach, Rolf Hoffmann: *Solving the exploration's problem with several creatures more efficiently*, Eurocast 2007

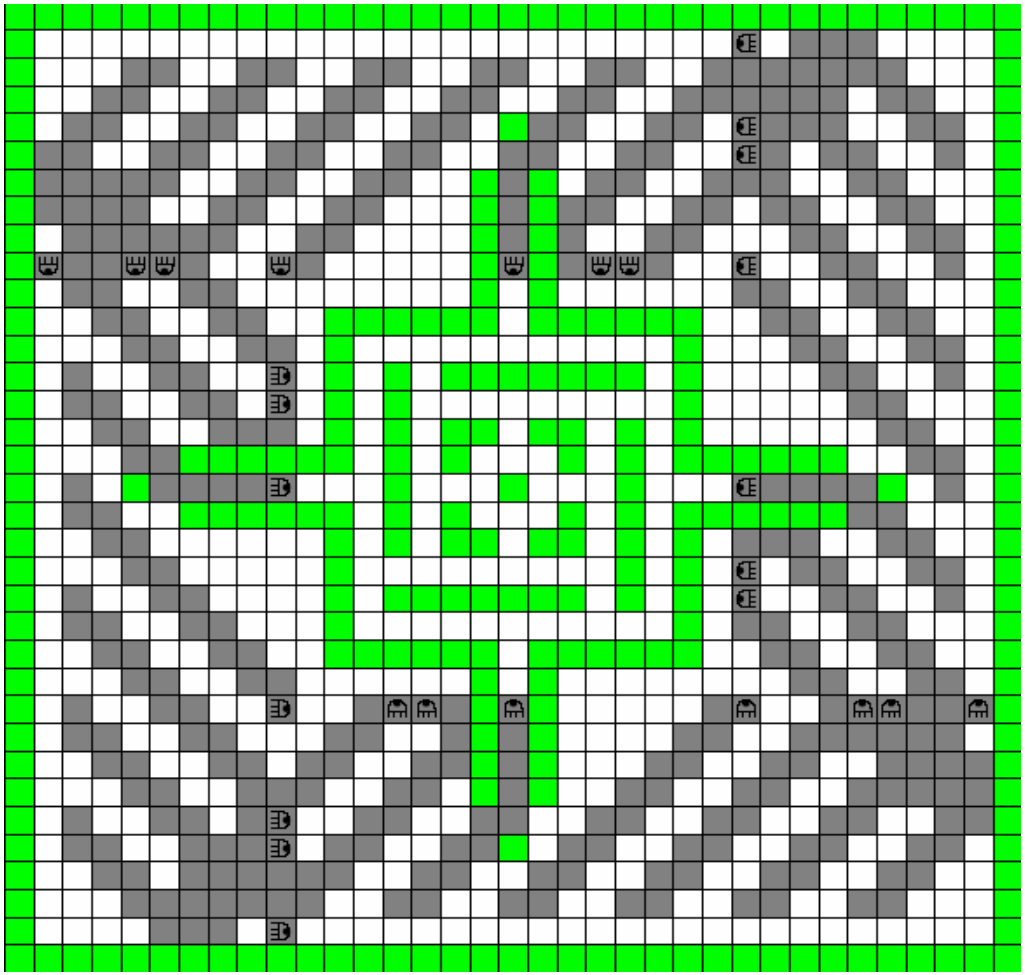
Mathias Halbach, Rolf Hoffmann: **Optimizing the Behaviour of a Moving Creature in a CA Field,** 11th Workshop on Cellular Automata (AUTOMATA) 2005, Gdansk, Poland

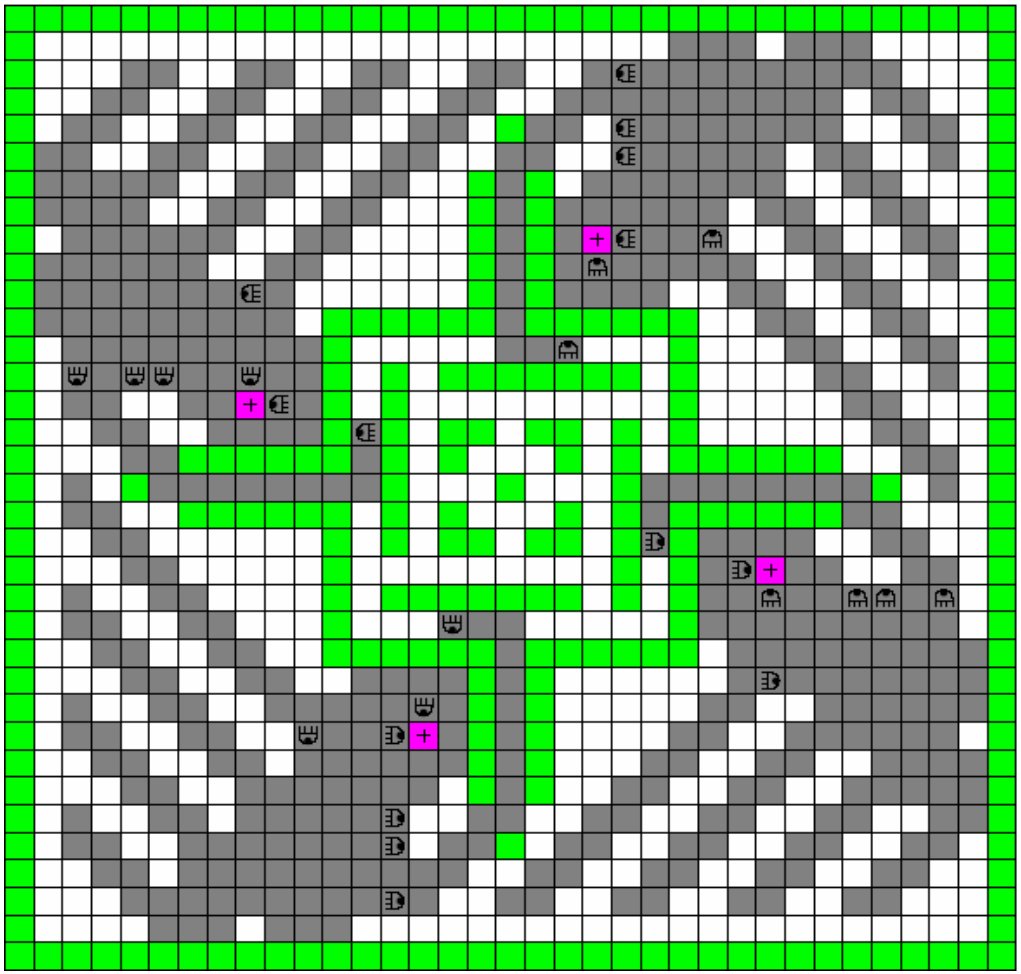
ENV3
Algorithm J-28
g_{max} = 228



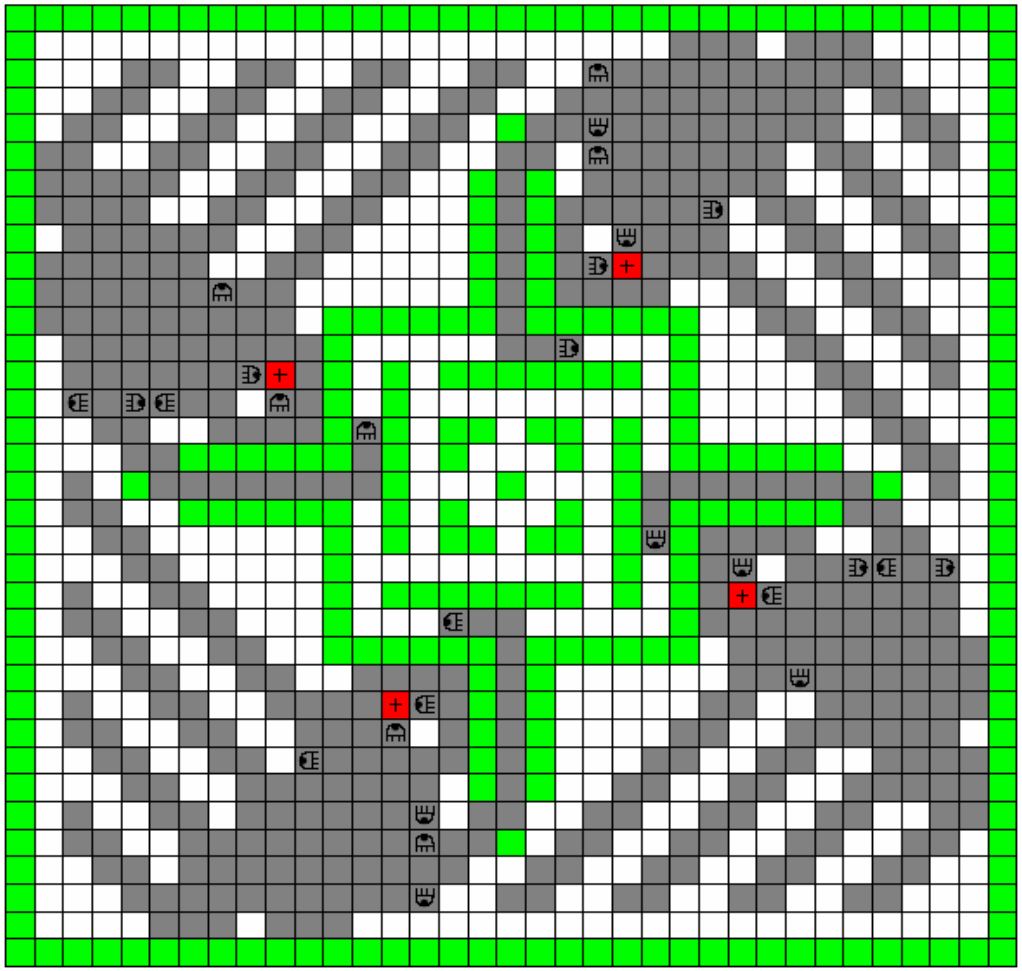




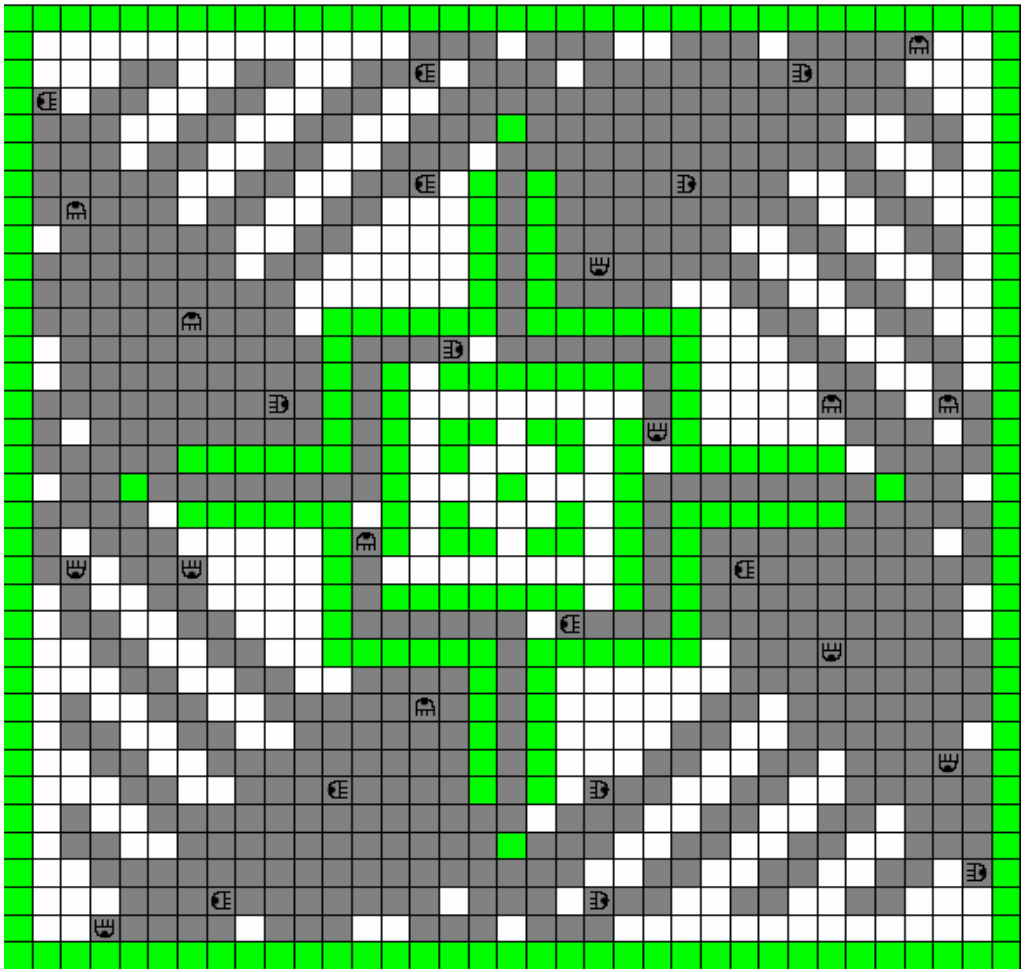


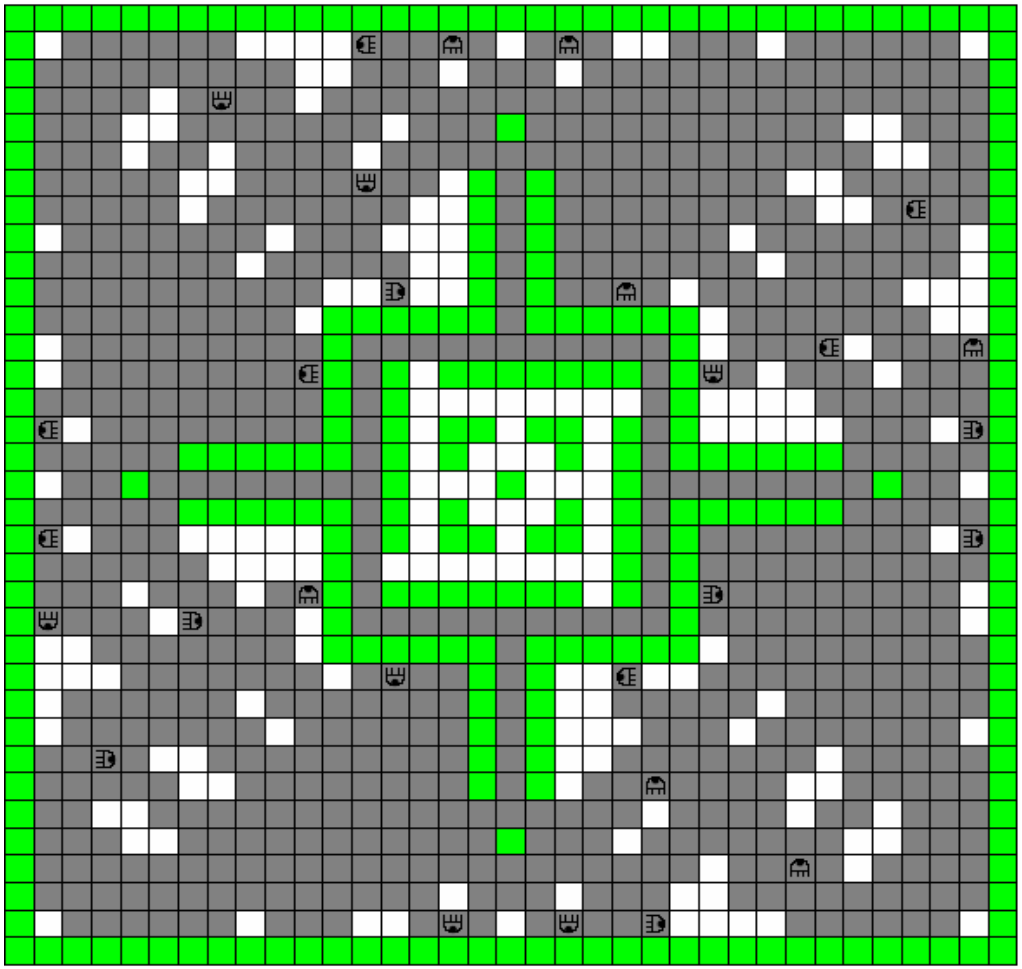


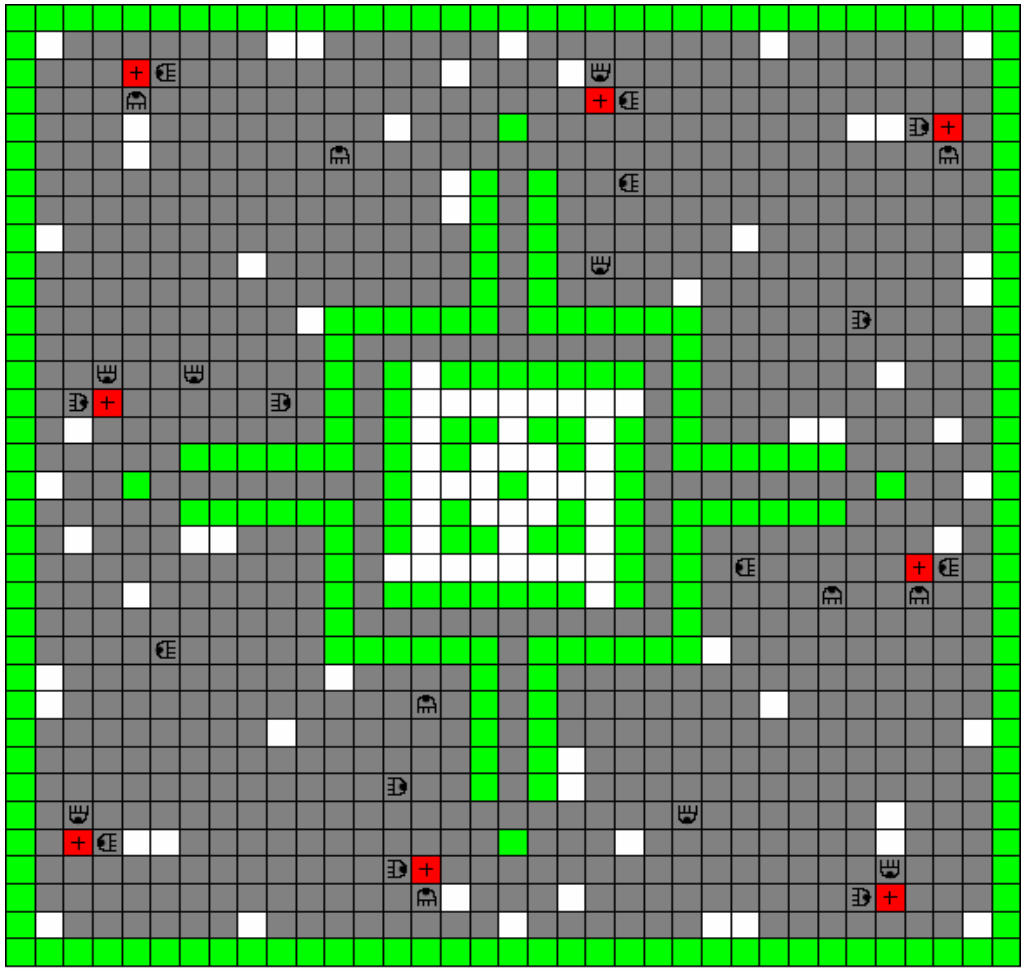
4 conflicts on not yet visited cells

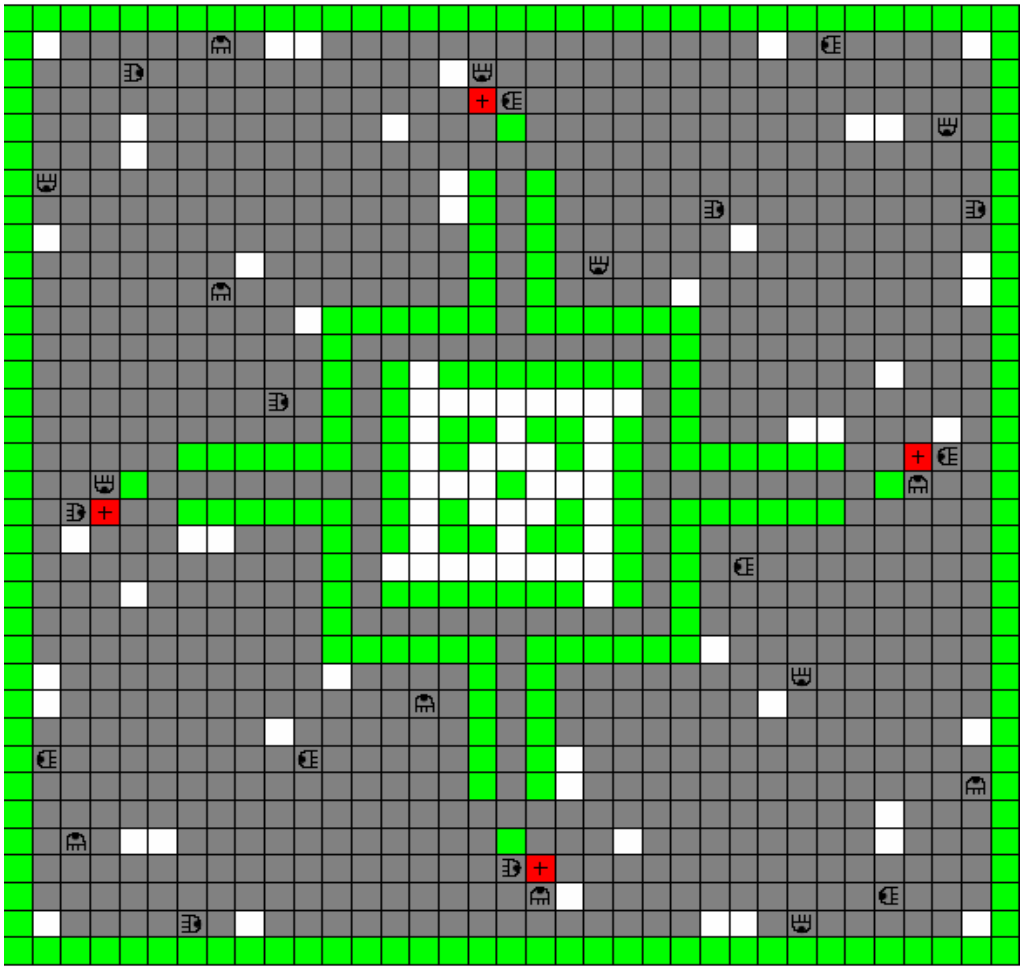


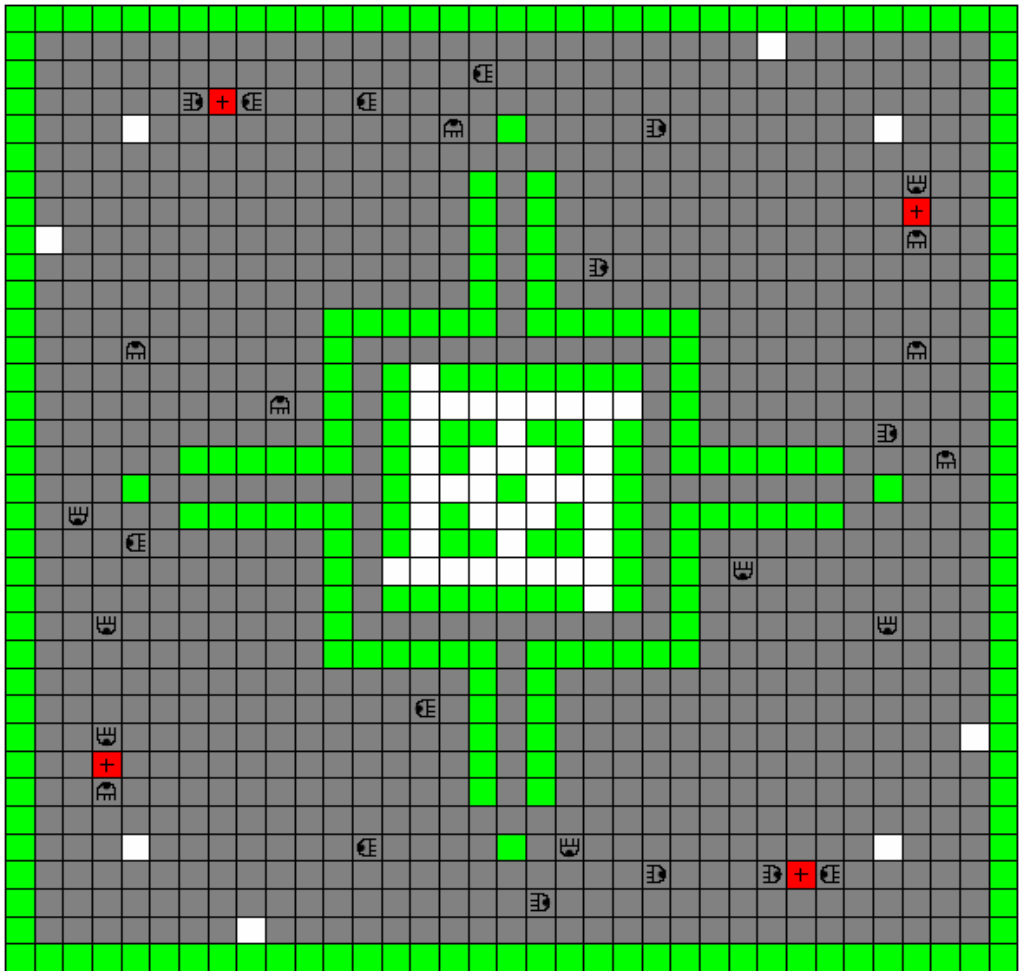
4 conflicts on already visited cells

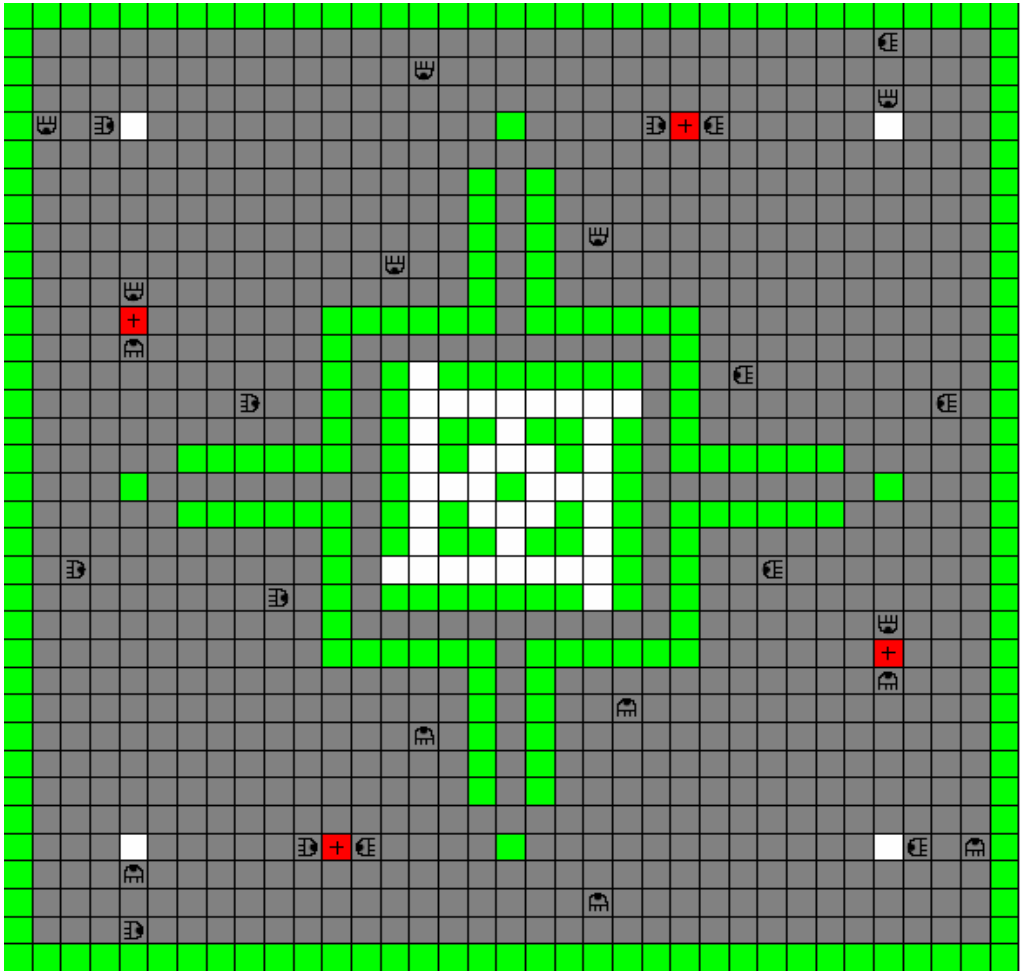


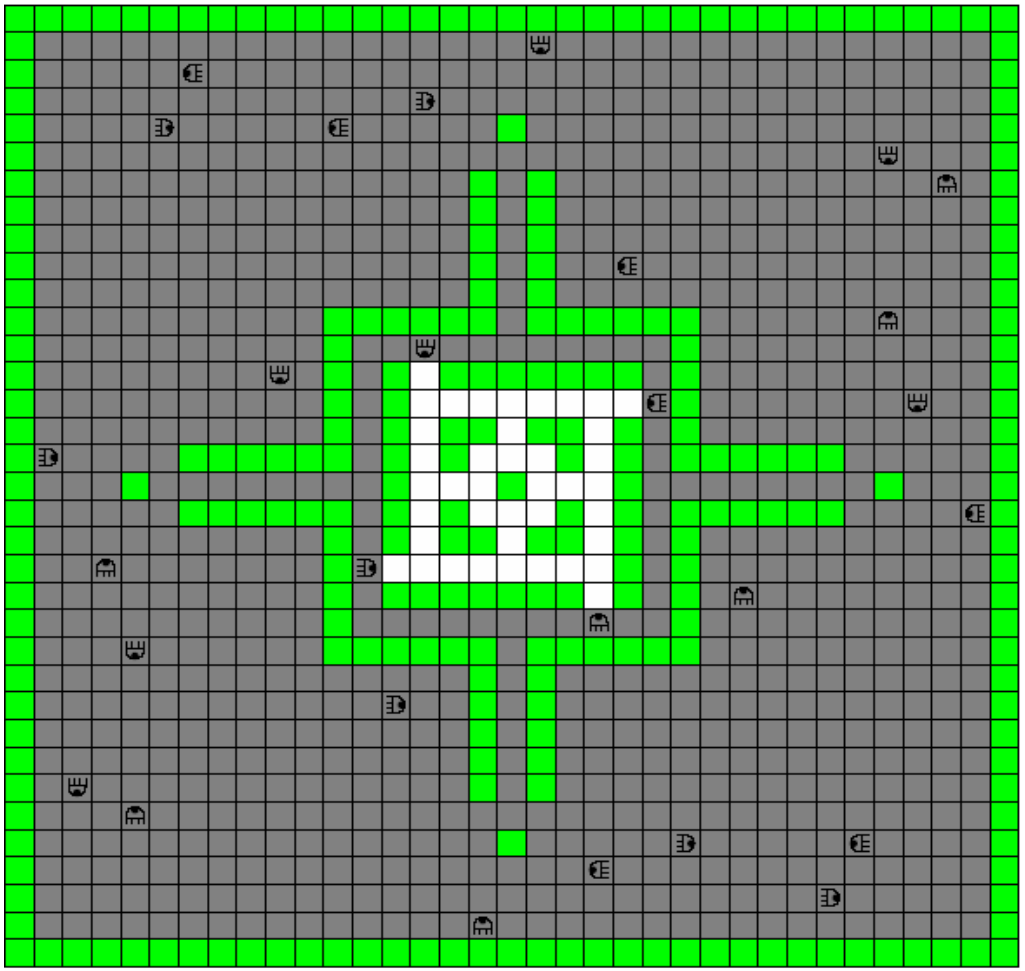


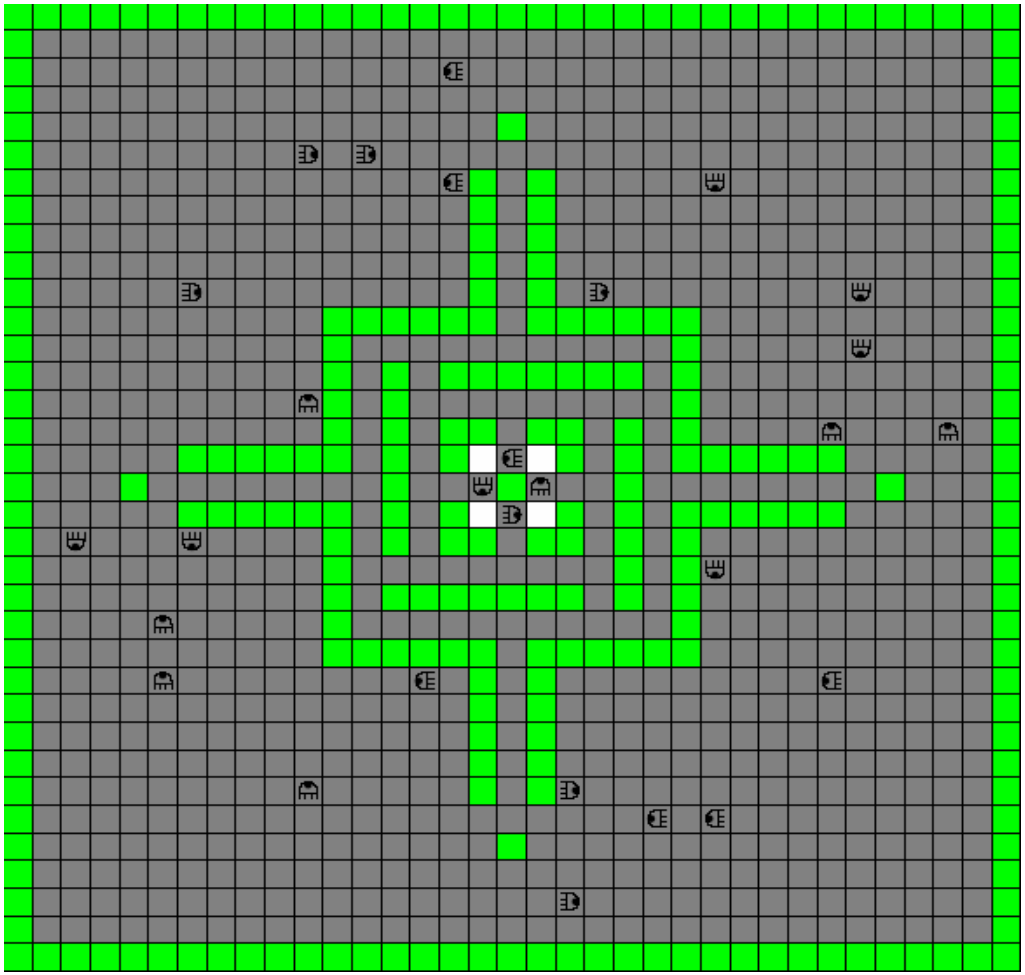






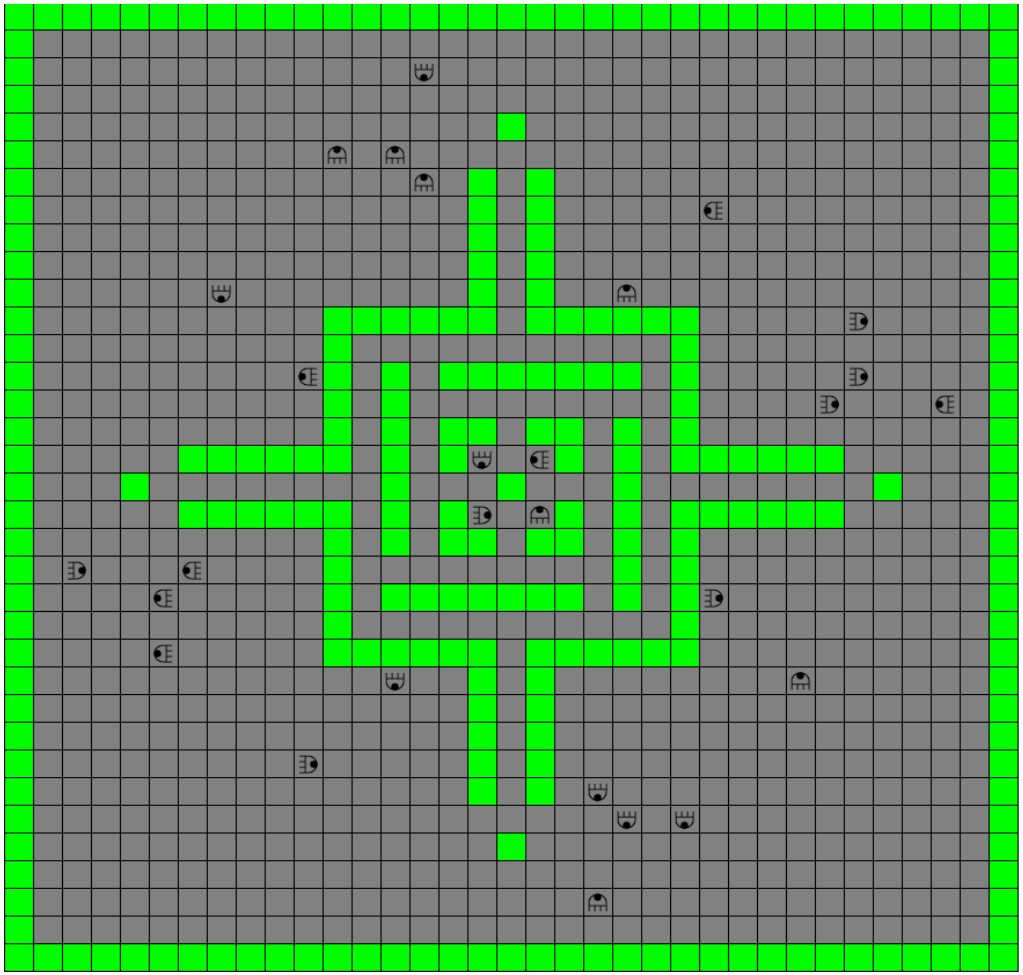






228 stop, all cells visited

TECHNISCHE UNIVERSITÄT DARMSTADT



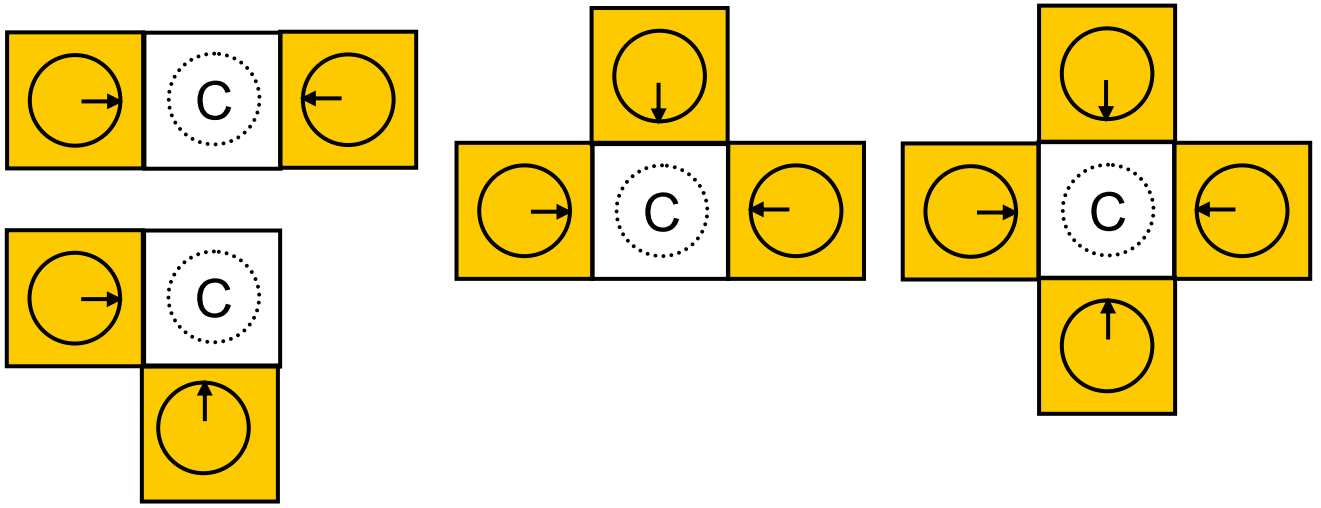
(4.2) All-to-all Communication by Agents

Patrick Ediger und Rolf Hoffmann: **Solving All-to-All Communication with CA Agents More Effectively with Flags,** 10th International Conference on Parallel Computing Technologies, LNCS, PaCT2009, Novosibirsk, Russia

All-to-all Communication by Agents

TECHNISCHE UNIVERSITÄT DARMSTADT

- Agents shall distribute their information in shortest time
- Behavior of agents defined by finite state machine
- Near optimal behaviors found by Genetic Algorithms
- Information exchange only allowed for certain communication situations

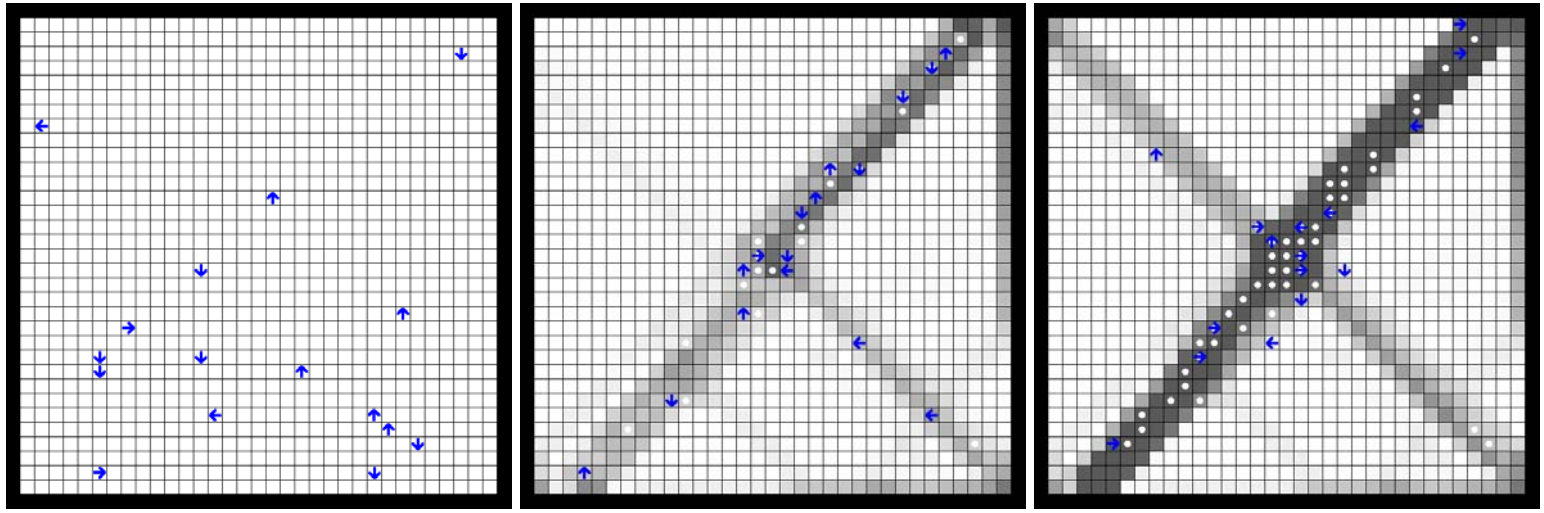


C = communicator

bitwise OR

Intelligent agents, All-to-all
Environment with border

TECHNISCHE UNIVERSITÄT DARMSTADT



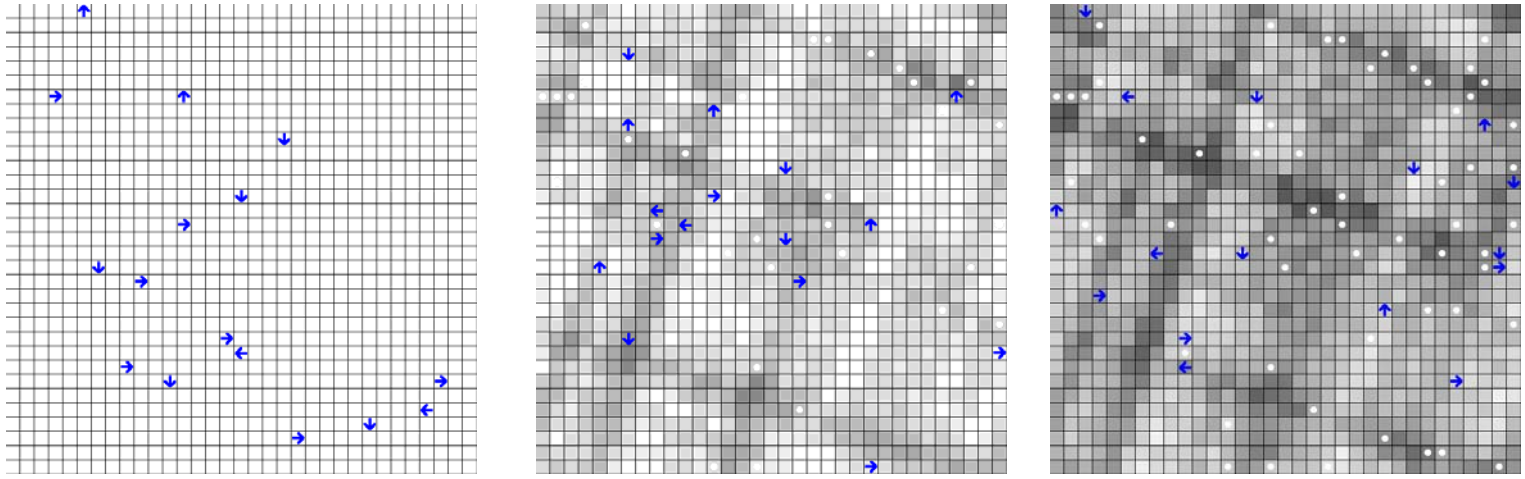
0 150 301

the darker the more often visited

Typical Pattern emerged:
trails, diagonals;
algorithm forces the agents to move on certain trails

Intelligent agents, All-to-all
Environment wrap-around

TECHNISCHE UNIVERSITÄT DARMSTADT



0 331 591

Evolving Multi-Creature Systems for All-to-all Communication
Rolf Hoffmann and Patrick Ediger
2nd International Workshop on Crowds & Cellular Automata (C&CA)
(ACRI 9 / 2008)

Typical Pattern emerged:
orthogonal mesh
turned 18,4° (3:1) to the right or to the left
"weaving pattern"

NEXT

PART II
Forming a Checkerboard Pattern