Efficient Traffic Simulation Using the GCA Model

Christian Schäck, Wolfgang Heenes, Rolf Hoffmann

Outline

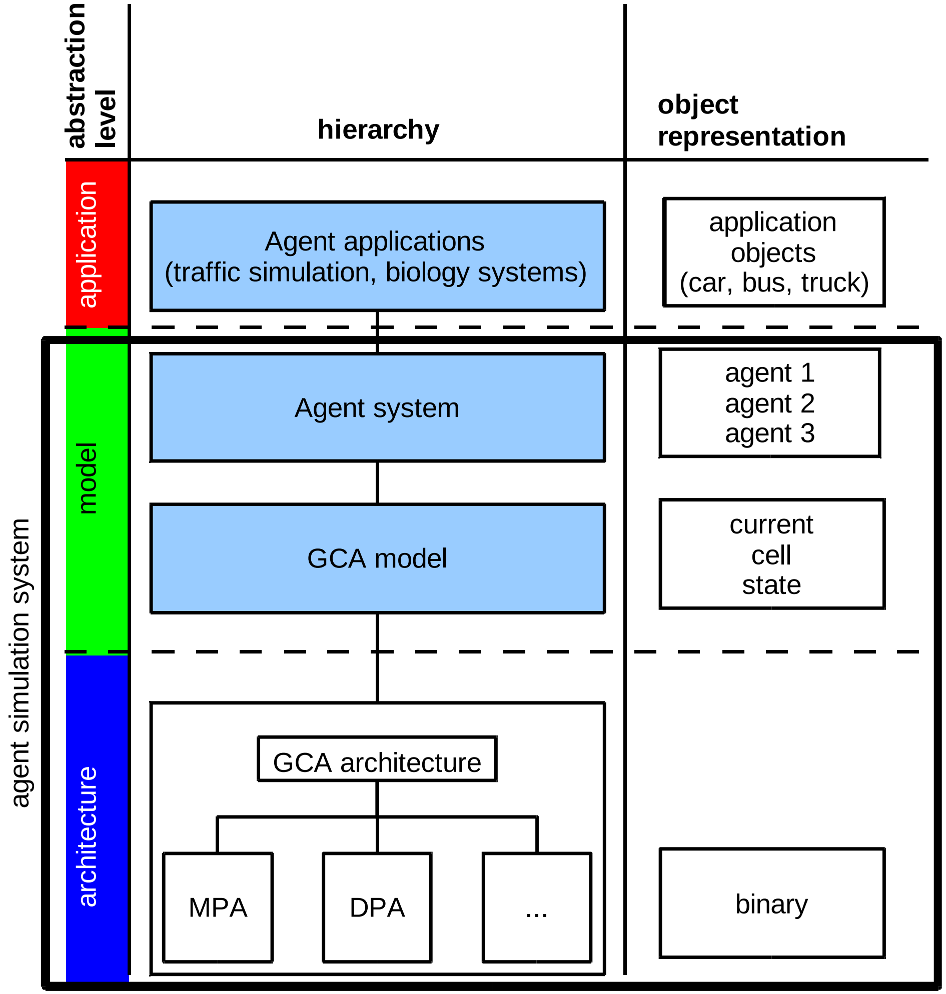
- Introduction
- Application, model and architecture hierarchy
  - Applications
  - Agent System
  - Global Cellular Automata (GCA) Model
  - Multiprocessor Architecture
- Random numbers in the GCA Model
- Traffic Simulation
  - Nagel-Schreckenberg Algorithm
  - CA Model with Searching
  - GCA Model with Linked Agents
- Results
- Summary & Outlook

Introduction

- dedicated system for Agent Simulation
- agents
  - moving entities on a n-dimensional field
  - homogeneous/heterogeneous behavior
- traffic simulation
  - Nagel-Schreckenberg algorithm
    - cars are represented as agents
- accelerate simulation using GCA model instead of CA model

System Overview

Application, Model and Architecture Hierarchy



Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

application

- general applications
  - graph algorithms
  - hypercube algorithms
  - numerical algorithms
  - graphics
  - …
- agent based applications
  - multi-agent simulation
  - traffic simulation (Nagel-Schreckenberg)
  - biology systems (honey bees)
  - …

Agent System

TECHNISCHE UNIVERSITÄT DARMSTADT

- generalization layer
- adds additional functions for agent simulations
- allows adjustment of layers underneath
- common interface regarding agent simulation
- accelerate simulation by defining hardware functions
- “library” for common used functions
  - defines function interfaces

- Agent System layer currently under investigation

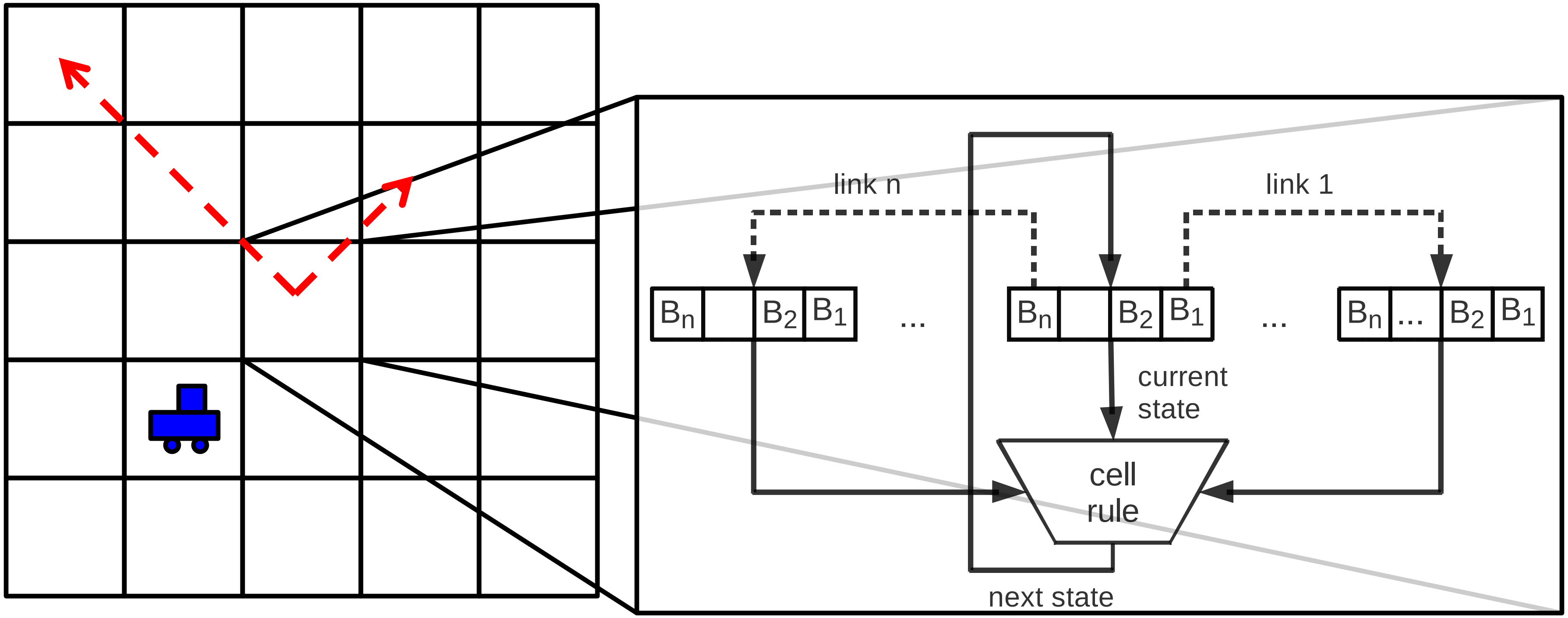
model

Global Cellular Automata Model

TECHNISCHE UNIVERSITÄT DARMSTADT

- massively parallel computational model
- extension of the classical cellular automata
- dynamic, global neighborhood (**read only**)

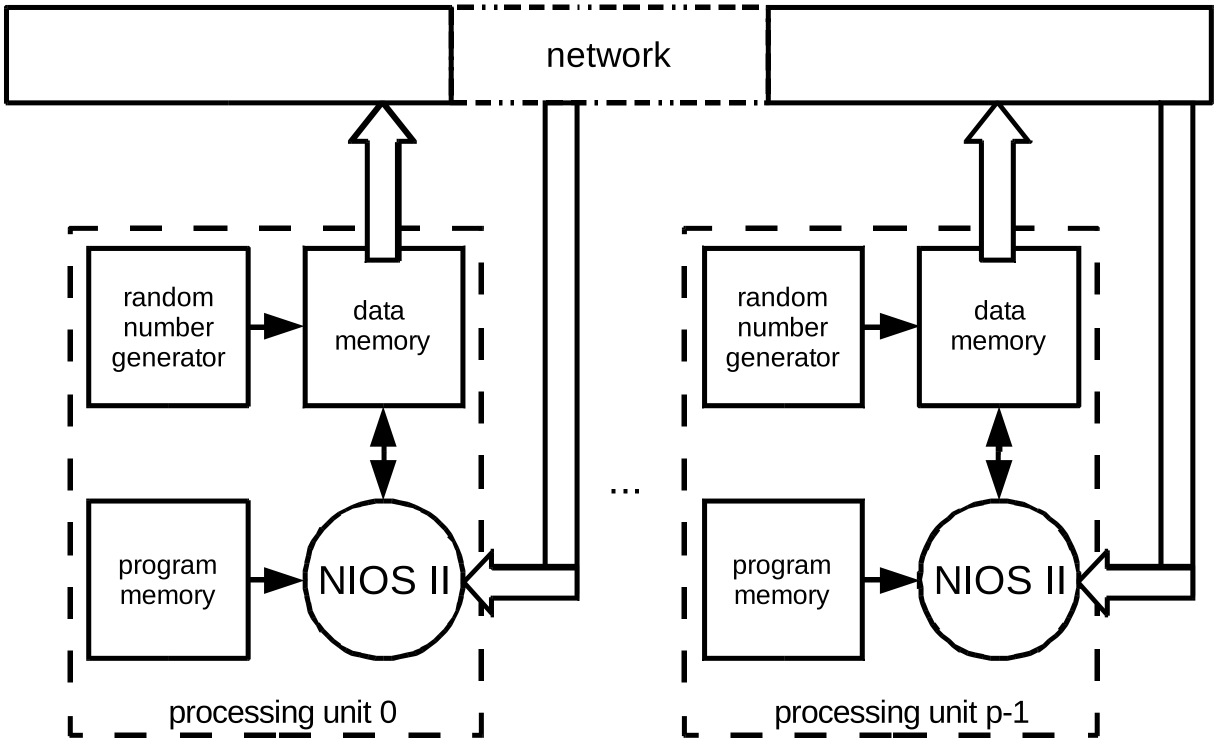
model



Multiprocessor Architecture - General Structure

- P NIOSII/f softcore processors + Custom Instructions
- internal memory (program, data)
- Interconnection network

architecture



Random Numbers

TECHNISCHE UNIVERSITÄT DARMSTADT

- necessary for individual agent behavior
- can not be generated within the processor
  - synchronization issues (delete + copy)

architecture

- random numbers generated during write
  - allows all PU's to access same random number
  - different random number for each cell and generation, but same random number while execution a generation

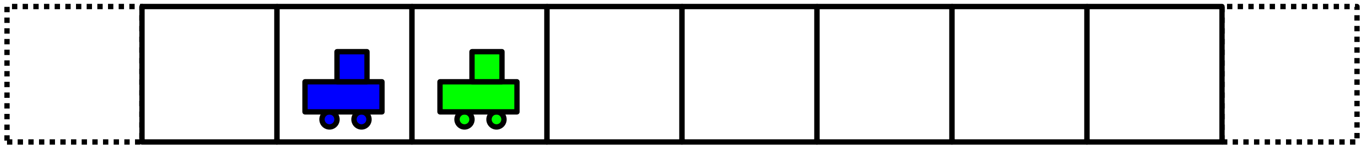
Traffic Simulation

Nagel-Schreckenberg Algorithm

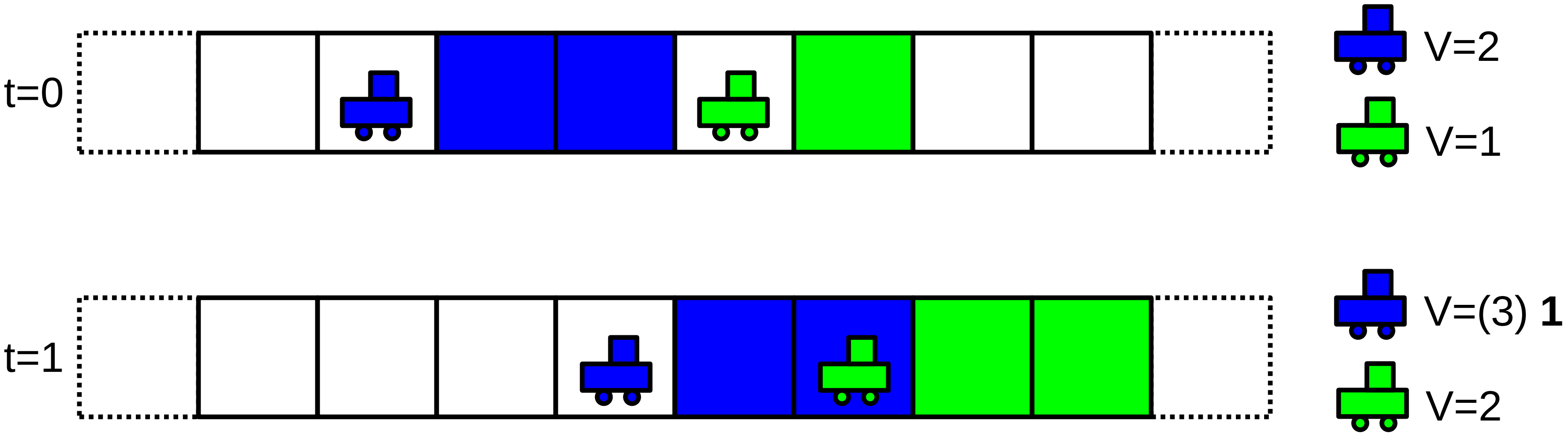
- theoretical traffic simulation model
- street consists of cells

- cell update rules:

1. if V_max not reached then V=V+1 (accelerate)
2. if gap < V then V=gap (collision free)
3. V=V-1 with probability p (dally)
4. move all vehicles



CA Model with Searching
agent cell checks



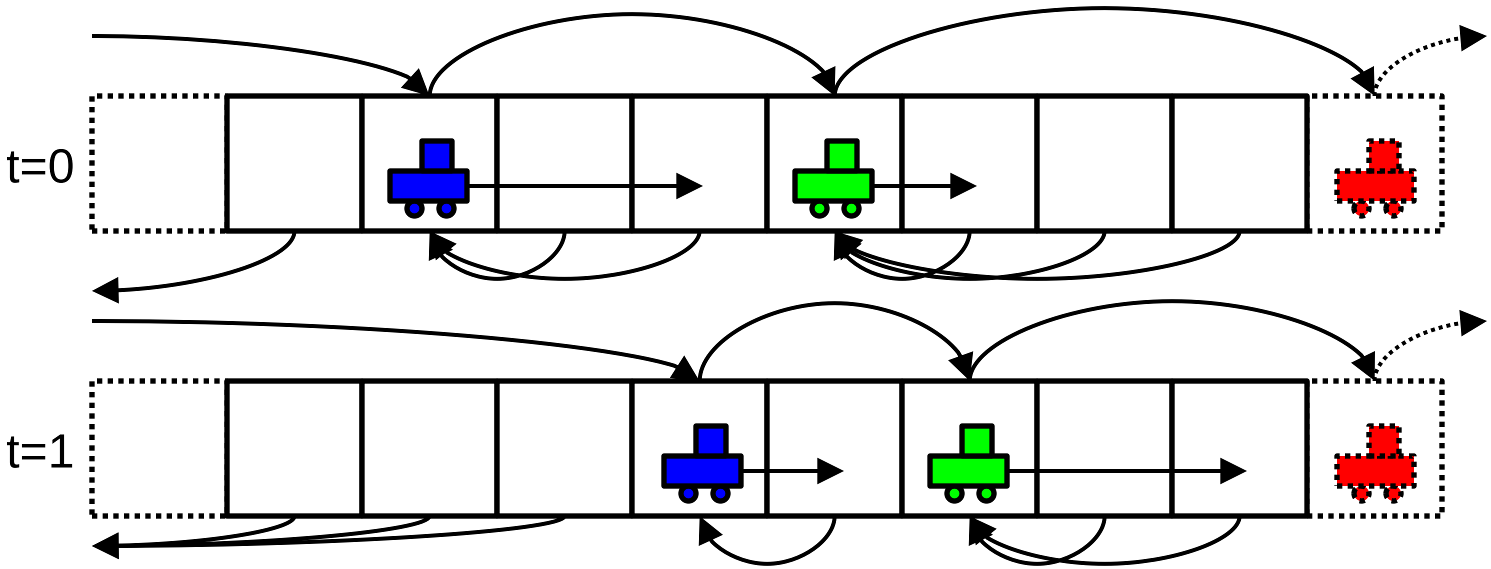
 stops checking if another agent is found or V is reached

CA Model with Searching
empty cell checks

X stops checking if an agent is found or V_max is reached

X checks two front cells 1. gap size 2. speed reduction

GCA Model with Linked Agents



Calculating speed for generation g+1 for cell i:

$$z'(i) := \begin{cases} -, & E \rightarrow E \\ max[min[z(L(i))+1, L(L(i))+z(L(L(i)))-i-1]-R, 0], & E \rightarrow A \\ -, & A \rightarrow E \\ max[min[1, L(i)+z(L(i))-i-1]-R, 0], & A \rightarrow A \end{cases}$$

not considering V_max

Results

2048 cells, p=0.5, V_max=5

P	gain (10%) 204 agents	gain (50%) 1024 agents
1	2.00	1.11
2	2.00	1.13
4	2.01	1.18
8	1.97	1.29
16	2.00	1.25

$$gain = \frac{execution\ time\ CA-Algorithm}{execution\ time\ GCA-Algorithm}$$

Scalability (50% agents):

P	execution time	speed-up
1	153.7 ms	-
2	84.2 ms	1.83
4	49.2 ms	3.12
8	28.8 ms	5.33
16	17.8 ms	8.66

Additional Results
2048 cells, p=0.5

- GCA-Algorithm performs very well for high speeds and low densities

- Low density of 1% agents (20)

V_max	gain
5	2.3
10	4.1
20	7.9
40	14.9
80	29.3

$$gain = \frac{execution\ time\ CA-Algorithm}{execution\ time\ GCA-Algorithm}$$

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- design and FPGA implementation of a multiprocessor architecture with NIOSII processors for the GCA model
- Agent System layer extension for agent simulation
- GCA-Algorithm performs faster compared to the CA-Algorithm
  - ~2x for 10% Agents
- GCA-Algorithm performs very well for
  1. low densities
  2. high maximum speeds

Outlook

- Agent System layer
  - further definition
  - evaluating HW/SW-functions for agents
- new architectures avoiding empty cells
  - based on hash functions
  - using dedicated agent memories

Thank you very much for your attention!