RW778: Implementation and Application of Automata, 2006
Week 4 Lecture 1

L. van Zijl

Department of Computer Science
University of Stellenbosch

2006

Robot Path Planning with Cellular Automata

References:

1. Handout: Tutorial on CA.
2. Behring, Bracho, Castro, Moreno, An algorithm for robot path planning with CA.
3. *Marchese, A path-planner for a non-holomic mobile robot with generic shape using multilayered CA.
4. *Bandini, Mauri, Multilayered cellular automata.

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

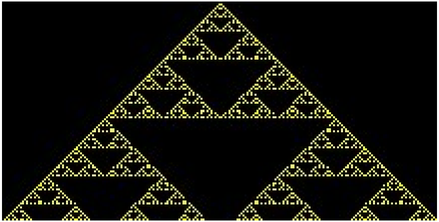
Cellular Automata

Definition: Handout (ca.html)

- Array of automata with neighbour influence (Game of Life)
- Visualization over time
- Immediate update of all cells in one time step
- Simplify: Binary (0–1 alphabet)
- Initial configuration (start states for every one)
- Ignore final states for the moment
- 1D vs 2D vs nD
- Hybrid vs uniform
- Boundary conditions (null, periodic)

Cellular Automata
Definition: Handout

Example:
Suppose $s_{t+1}(m) = s_t(m-1) \oplus s_t(m+1)$.



Robot Path Planning with Cellular Automata

The robot path planning problem:

- Real-time planning of collision-free route for robot to move from start to goal position.
- Possible approaches: randomized potential field methods, roadmaps, etc.
- Problem: compute intensive.
- CA: inherently parallel, allow real-time computation.

Robot Path Planning with Cellular Automata

The robot path planning problem:

- Real-time planning of collision-free route for robot to move from start to goal position.
- Possible approaches: randomized potential field methods, roadmaps, etc.
- Problem: compute intensive.
- CA: inherently parallel, allow real-time computation.

Robot Path Planning with Cellular Automata

The robot path planning problem:

- Real-time planning of collision-free route for robot to move from start to goal position.
- Possible approaches: randomized potential field methods, roadmaps, etc.
- Problem: compute intensive.
- CA: inherently parallel, allow real-time computation.

Robot Path Planning with Cellular Automata

The robot path planning problem:

- Real-time planning of collision-free route for robot to move from start to goal position.
- Possible approaches: randomized potential field methods, roadmaps, etc.
- Problem: compute intensive.
- CA: inherently parallel, allow real-time computation.

Robot Path Planning: Definitions
Robot Path Planning (Behring)

- Configuration space:
  - Euclidian space
  - Finite number obstacles B_i (region)
  - Path in configuration space
- CA: 2D

Robot Path Planning: Definitions

Robot Path Planning (Behring)

- Configuration space:
  - Euclidian space
  - Finite number obstacles B_i (region)
  - Path in configuration space
- CA: 2D

Robot Path Planning: Definitions

Robot Path Planning (Behring)

- Configuration space:
  - Euclidian space
  - Finite number obstacles B_i (region)
  - Path in configuration space
- CA: 2D

Robot Path Planning: Definitions
Robot Path Planning (Behring)

- Configuration space:
  - Euclidian space
  - Finite number obstacles B_i (region)
  - Path in configuration space
- CA: 2D

Robot Path Planning: Definitions
Robot Path Planning (Behring)

- Configuration space:
  - Euclidian space
  - Finite number obstacles B_i (region)
  - Path in configuration space
- CA: 2D

Robot Path Planning: Algorithm I

Robot Path Planning (Behring)

- Three phases: input, obstacle growth, distance calculations
- Robot assumptions: no dynamic, kinematic, orientation constraints: Point.
- (See Marchese: orientation path planning)
- Path: discrete sequence of configurations $(q_{\text{init}}, q_1, \ldots, q_{\text{goal}})$

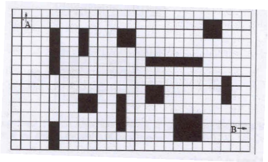


Figure: Example configuration space

Robot Path Planning: Algorithm I

Robot Path Planning (Behring)

- Three phases: input, obstacle growth, distance calculations
- Robot assumptions: no dynamic, kinematic, orientation constraints: Point.
- (See Marchese: orientation path planning)
- Path: discrete sequence of configurations $(q_{\text{init}}, q_1, \ldots, q_{\text{goal}})$

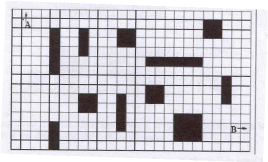


Figure: Example configuration space

Robot Path Planning: Algorithm I

Robot Path Planning (Behring)

- Three phases: input, obstacle growth, distance calculations
- Robot assumptions: no dynamic, kinematic, orientation constraints: Point.
- (See Marchese: orientation path planning)
- Path: discrete sequence of configurations $(q_{\text{init}}, q_1, \ldots, q_{\text{goal}})$

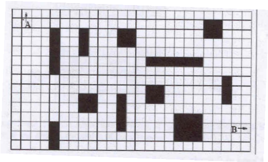


Figure: Example configuration space

Robot Path Planning: Algorithm I
Robot Path Planning (Behring)

- Three phases: input, obstacle growth, distance calculations
- Robot assumptions: no dynamic, kinematic, orientation constraints: Point.
- (See Marchese: orientation path planning)
- Path: discrete sequence of configurations $(q_{\text{init}}, q_1, \ldots, q_{\text{goal}})$

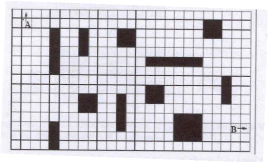


Figure: Example configuration space

Robot Path Planning: Algorithm Phases 1,2

Robot Path Planning (Behring)

- CA cell states: free, obstacle, initial, goal (robot?)
- Evolve: use Moore neighbourhood.
- $s_i(t)$ state of automaton i at time t
- $\nu_i(t)$ Moore neighbourhood of cell i at time t
- $s_i(t+1) = \begin{cases} 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t. } s_x(t) = 1 \\ s_i(t) & \text{otherwise.} \end{cases}$

Robot Path Planning: Algorithm Phases 1,2

Robot Path Planning (Behring)

- CA cell states: free, obstacle, initial, goal (robot?)
- Evolve: use Moore neighbourhood.
- $s_i(t)$ state of automaton i at time t
- $\nu_i(t)$ Moore neighbourhood of cell i at time t
- $s_i(t+1) = \begin{cases} 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t. } s_x(t) = 1 \\ s_i(t) & \text{otherwise.} \end{cases}$

Robot Path Planning: Algorithm Phases 1,2

Robot Path Planning (Behring)

- CA cell states: free, obstacle, initial, goal (robot?)
- Evolve: use Moore neighbourhood.
- $s_i(t)$ state of automaton i at time t
- $\nu_i(t)$ Moore neighbourhood of cell i at time t
- $s_i(t+1) = \begin{cases} 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t. } s_x(t) = 1 \\ s_i(t) & \text{otherwise.} \end{cases}$

Robot Path Planning: Algorithm Phases 1,2

Robot Path Planning (Behring)

- CA cell states: free, obstacle, initial, goal (robot?)
- Evolve: use Moore neighbourhood.
- $s_i(t)$ state of automaton i at time t
- $\nu_i(t)$ Moore neighbourhood of cell i at time t
- $s_i(t+1) = \begin{cases} 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t. } s_x(t) = 1 \\ s_i(t) & \text{otherwise.} \end{cases}$

Robot Path Planning: Algorithm Phases 1,2

Robot Path Planning (Behring)

- CA cell states: free, obstacle, initial, goal (robot?)
- Evolve: use Moore neighbourhood.
- $s_i(t)$ state of automaton i at time t
- $\nu_i(t)$ Moore neighbourhood of cell i at time t
- $s_i(t+1) = \begin{cases} 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t. } s_x(t) = 1 \\ s_i(t) & \text{otherwise.} \end{cases}$

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t+1) = \begin{cases} s_x(t) + 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) & \text{otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t+1) = \begin{cases} s_x(t) + 1, & \text{if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) & \text{otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t+1) = \begin{cases} s_x(t) + 1, \text{ if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) \text{ otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t{+}1) = \begin{cases} s_x(t) + 1, \text{ if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) \text{ otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t+1) = \begin{cases} s_x(t) + 1, \text{ if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) \text{ otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

- Result from phase 2 initial configuration of 2D CA.
- Evolve: calculate Manhattan distance between initial, goal positions.
- CA cell states: free, obstacle, initial, goal, Manhattan distance to goal.
- Evolve: flood from goal to initial.

$$s_i(t+1) = \begin{cases} s_x(t) + 1, \text{ if } s_i(t) = 0 \& \exists x \in \nu_i(t) \text{ s.t.} s_x(t) \geq 3 \\ s_i(t) \text{ otherwise.} \end{cases}$$

- Flood stopped when initial reached, or when all cells not zero (no path)
- Heuristics: check Behring Sect. 4.

Robot Path Planning: Algorithm Phase 3

Robot Path Planning (Behring)

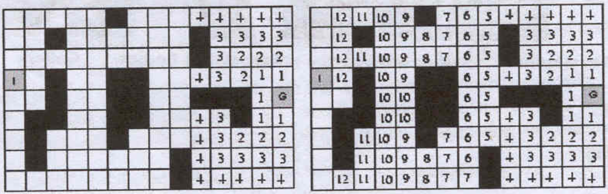


Figure 2: The Flood Dynamics. The second cellular automata dynamics computes the Manhattan distance (numbers in the cells) between each cell and the goal position, - dark point at the right size of each figure -. a) 4th iteration. b) 13th iteration.

For examination purposes:

- Study section 5 (Complexity of the algorithm) in Behring
- Read section 6 (Experiments) in Behring

Robot path planning

Homework

Homework:

1. Write a program to implement a 1D CA array. Assume each automaton has two states and one alphabet symbol. Assume the dependence function as XOR. Let the initial configuration be given as a binary vector, and the dependence for each cell also as a binary vector, stating its dependence on other cells. Assume cells are numbered $0, 1, \ldots$. In the example below, we have a binary hybrid 1D CA with 3 cells. Use the input format illustrated below. For output purposes, simply use the Wolfram blob pictures.
2. Now adapt your implementation to the 2D case.
3. Adapt your implementation of 2D CA to implement Behring's robot path planning algorithm.

Robot path planning
Homework

Homework:

1. Write a program to implement a 1D CA array. Assume each automaton has two states and one alphabet symbol. Assume the dependence function as XOR. Let the initial configuration be given as a binary vector, and the dependence for each cell also as a binary vector, stating its dependence on other cells. Assume cells are numbered $0, 1, \ldots$. In the example below, we have a binary hybrid 1D CA with 3 cells. Use the input format illustrated below. For output purposes, simply use the Wolfram blob pictures.
2. Now adapt your implementation to the 2D case.
3. Adapt your implementation of 2D CA to implement Behring's robot path planning algorithm.

Homework:

1. Write a program to implement a 1D CA array. Assume each automaton has two states and one alphabet symbol. Assume the dependence function as XOR. Let the initial configuration be given as a binary vector, and the dependence for each cell also as a binary vector, stating its dependence on other cells. Assume cells are numbered $0, 1, \ldots$. In the example below, we have a binary hybrid 1D CA with 3 cells. Use the input format illustrated below. For output purposes, simply use the Wolfram blob pictures.
2. Now adapt your implementation to the 2D case.
3. Adapt your implementation of 2D CA to implement Behring's robot path planning algorithm.

Input format:

0 DEPENDS 012
1 DEPENDS 12
2 DEPENDS 0
INITIAL 02
BOUNDARY null