POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

CELLULAR AUTOMATA

K. Ming Leung

Abstract: Dynamic properties of some one dimensional cellular automata are discussed.

Directory

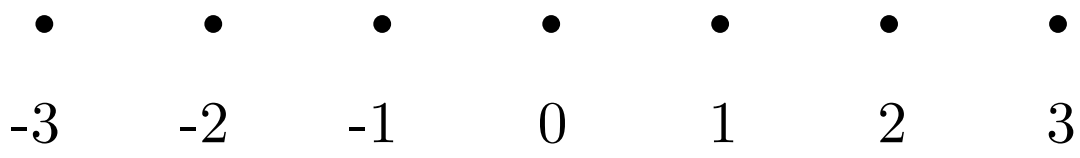
Last Revision Date: April 27, 2004

Table of Contents

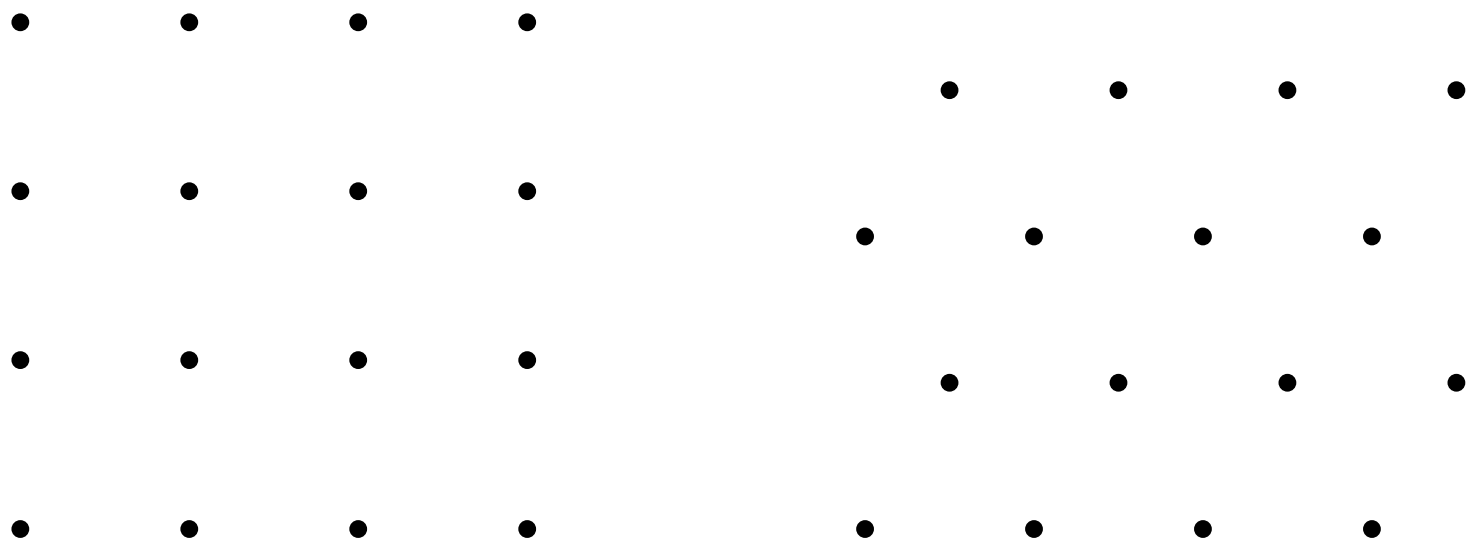
1. Introduction

Cellular automata are automata that are placed at the nodes of a periodic lattice of points that are invariant under discrete translation and rotation operations. The first model introduced by von Neumann and Ulam was aim at understanding self-reproduction in biology. Therefore the lattices are referred to as cells.

In one dimension, the points can be labeled by the infinite set of integers.



In two dimension, some common lattices are the square, triangular and hexagonal (honeycomb) lattices.

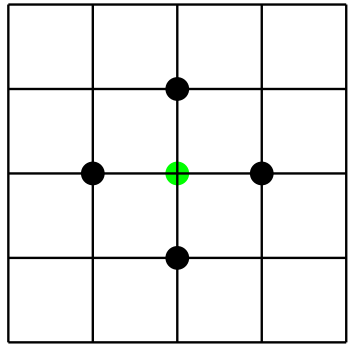


It is a common practice to assume that all the automata are identical to each other and apply the same transition rules to all the automata in the lattice. These cellular automata are often referred to as homogeneous. Cellular automata with each automaton obeying a different transition rule have also been studied. These cellular au-

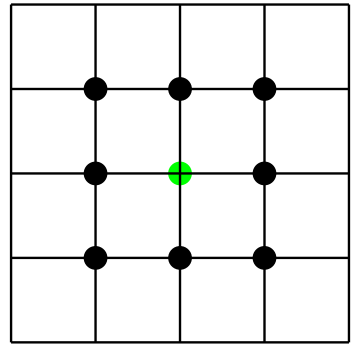
tomata are referred to as inhomogeneous. The parallel iteration mode is almost always used to update the states of the automata.

1.1. Connectivity

The connectivity of the automata must be specified. It is customary in the case of cellular automata to connect an automaton with its nearest neighbors only. In two dimension, two common choices of neighborhoods are the von Neumann and the Moore neighborhoods.



von Neumann



Moore

1.2. Handling Finite Lattices

In principle all these lattices are supposed to extend to infinity and have no edges. In reality in order to be able to simulate these cellular

automata, the lattices must be finite and therefore must have edges. Automata near the edges have a different environment compared with those in the interior. Since we are interested primarily on the bulk behavior of the cellular automata, edges effects must be suppressed.

This is normally handled by imposing periodic boundary conditions which amounts to physically connecting opposite ends, edges, or surfaces (depending on the dimensionality) of the lattices together. In a one dimensional lattice, the first lattice point is connected to the last point to form a circle of points.

In a two dimensional lattice, we can first join together the top with the bottom edges to form a tube, the tube is then bent around so that the two ends are connected together. The resulting lattice then has the form of a torus (bagel or donut).

The same procedure can be carried out in three or higher dimensions, although the resulting geometry of the lattices are more difficult to visualize.

Implementation of periodic boundary conditions is best handled by introducing extra layers of lattices at the boundaries.

For a one dimensional lattice with lattice points at $1, 2, \ldots, L$,

extra lattice points are added at 0 and $L+1$. The values at 0 and $L+1$ are always the same as those at L and 1, respectively. During each time step, we use the parallel iteration mode and compute new values for the states of the automata at positions $1, 2, \ldots, L$ based on the previous values of the states at positions $0, 1, \ldots, L, L+1$,

The idea can easily be generalized to higher dimensions. Of course if the connectivity of the automata involves more than the nearest neighbors, the dimensionalities of these extra layers have to be increased correspondingly.

2. One Dimensional Cellular Automata

We now consider in more detail a one dimensional cellular automata in which the neighborhood of an automaton consists of itself and its two adjacent neighbors. Therefore each automaton has 3 inputs. Its state is updated according to its old value and those of it two neighbors. We represent the input by 3 bits, the first of which is the state of the neighbor on its left, the second the state of the automaton in question and the state of the neighbor on the right.

Then there are $2^3 = 8$ different possible input vectors and $2^8 = 256$ different state change rules. These 256 rules, numbered from 0 to 255, encode in decimal notation the eight output bits corresponding to the eight input configurations, from 000 for the LSB to 111 for the MSB.

For example, rule 90 (01011010 in 8-bit binary notation) represents the following transition rule:

Input state	111	110	101	100	011	010	001	000
Output	0	1	0	1	1	0	1	0

2.1. Results

One simple way to picture the dynamics of these one dimensional cellular automata is to use a representation in which each row represents the configuration of the lattice at time t, and the following row the configuration at time $t + 1$. Time then increases from top to bottom. Such a picture can be displayed on the screen in straightforward character mode. To improve visualization of the picture, 0's may be displayed as blanks and 1's as asterisks.

All of the 256 one dimensional cellular automata with 3 inputs have been studied. It is found that there are only three possible classes of behaviors.

1. Certain cellular automata have such strong attractors that practically all configurations converge rapidly toward an invariant homogeneous configuration, composing of all 0's (as in rule 128 or all 1's (as in rule 250). More specifically, for example in the case of rule 128, with the exception of the initial configuration of all 1's, which being invariant, constitutes an isolated fixed point, all other initial configurations with at least one 0 converge toward the attractor containing only 0's.
2. There are short-period attractors, which have period of 1 or 2, depending on the initial configurations (rules 108 and 178). The number of different attractors is very large, in fact growing exponentially with the number of automata, N.
3. There are attractors of very long periods, too long to be easily observable (rules 90 and 126). These periods grow exponentially with N.

A period that is of the order of a power of N is considered short. A period is considered long if it grows exponentially with N.

Associated with these long period attractors are "spatial patterns" that seem to have rather regular textures. In the case of rules 90 and 126, numerous self-similar downward pointing triangles of various sizes can be seen.

3. Traffic Modeling

We will model the flow of traffic along a one dimensional (single-lane) highway using cellular automata.[3] Space is discretized into cells placed on the x-axis. Periodic boundary condition is imposed to handle the cells at the two ends. Each car occupies the space of a single cell. Not all the cells have cars in them. The density of cars is an important parameter of the model and have to be specified at the outset. The initial arrangement of the cars on the lattice and the initial speeds of the cars are also important, but are usually treated by random assignments.

In this this model, the transition rules of the cellular automata

have to be derived from a multi-step rule to obtain a one-step rule. We consider a model that describes traffic on a single-lane highway and the following assumptions will be made:

1. Cars move on the single-lane highway lane in one direction (say to the left) only.
2. Cars do not suddenly appear or disappear, except possibly at the boundaries.
3. Each car occupies a single cell in the lattice.
4. Cars cannot pass each other.

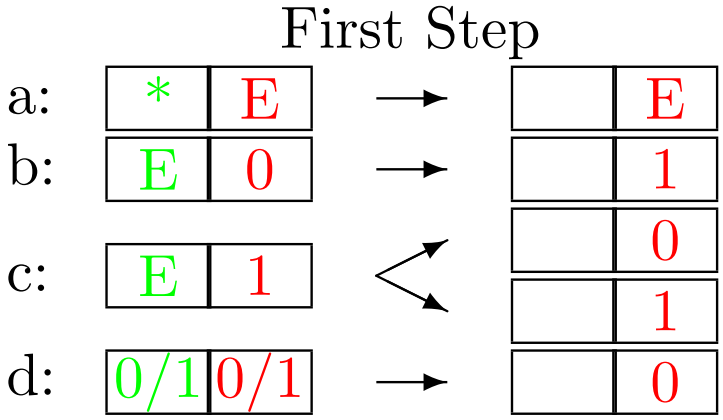
3.1. A Single-Speed Model

Our first attempt is to formulate a single-speed model in which a car is either stationary, then it has a speed of 0, or it moves with a speed of 1. Each car occupies exactly the space of one cell. Therefore each automaton has three possible state values, s. Each cell is either empty ($s = E$), occupied by a stationary car ($s = 0$), or by a moving car ($s = 1$). Then in each step, if a car has another car in front of it,

it cannot move, otherwise it moves. To model the non-deterministic behavior of human drivers, we introduce an additional probabilistic rule: a moving car may brake and stop for whatever reasons with a probability p. Let us assume that drivers are usually not that crazy and therefore p is normally chosen to be small.

We must now formulate the transition rules to update the states of the automata at each time interval. We follow a two-step procedure. In the first step, we determine for each cell occupied by a car the speed with which it moves in this current time. In the second step, each car is displaced accordingly and the state of each automaton is updated.

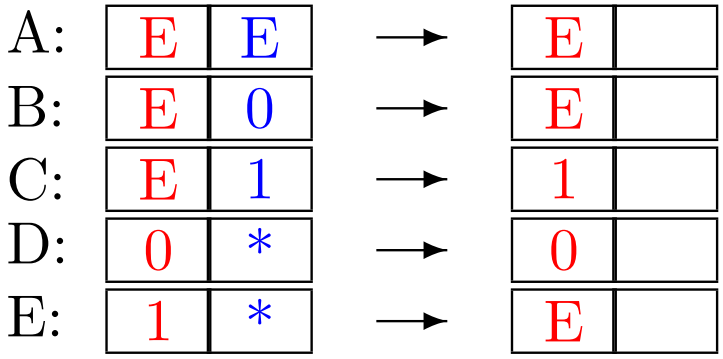
For the first step, to determine the speed with which a car at a given cell (in red) will move in the current time, we also need to consider the state of the cell in front (to the left) of it (in green). Applying the rules of the model, we obtain the following results:



In the diagram, an asterisk represents any state value, and a blank represents a value we are presently not interested in. We can describe the rules in words as follows: (a) a gap on the highway with no cars, (b) a stopped car with a space in front of it accelerates into that empty space, (c) a moving car with a space in front of it can either continue to move or stop, (d) a car with any other car right in front of it must stop regardless whether or not the other car is moving.

In the second step, the states of the automata (shown in the diagram below in red) are updated. To accomplish this we also need to know the states of the automata on its right (shown in blue). The results are:

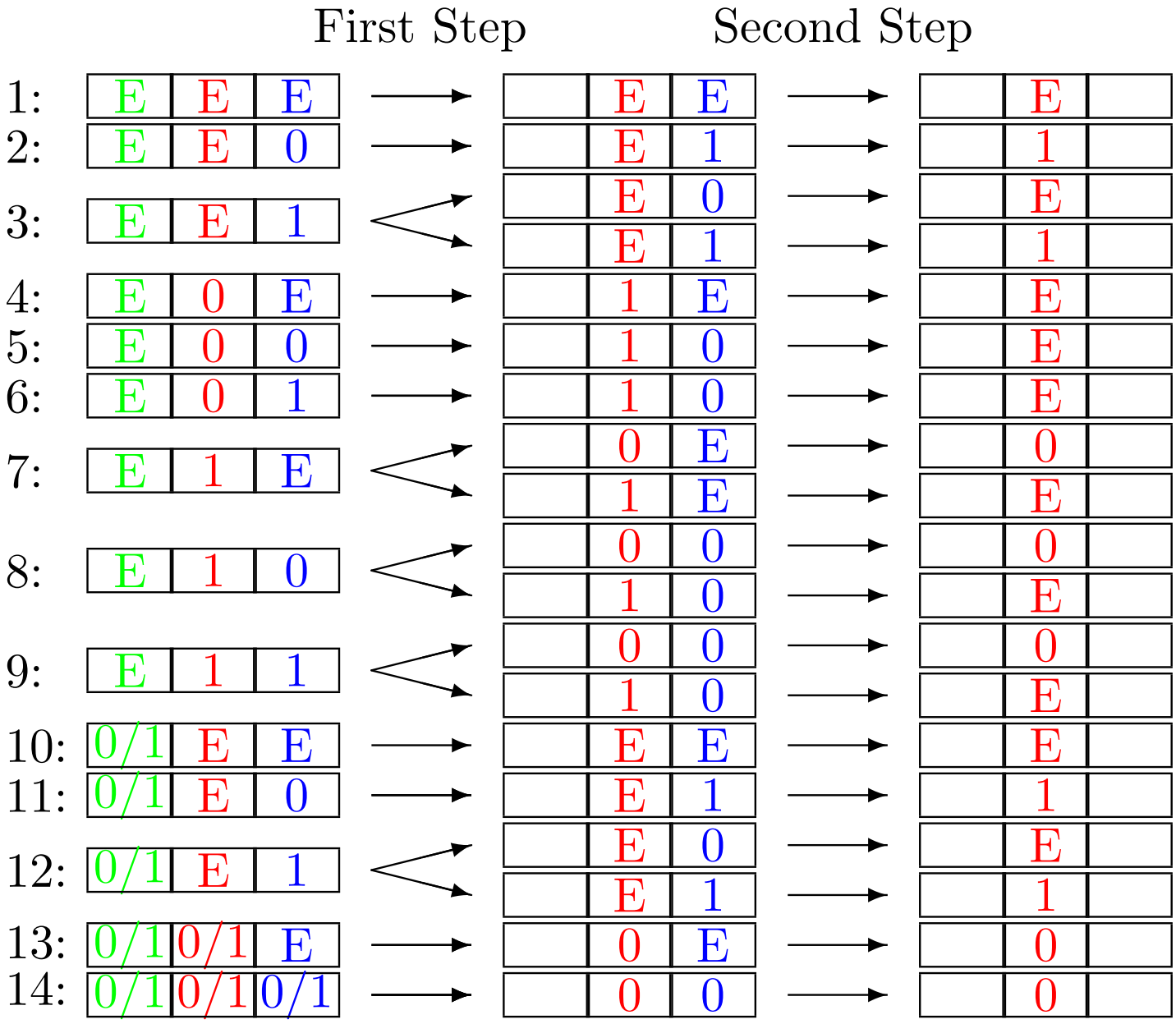
Second Step



The above action can be explained in words as follows:

(A) space remains empty because no car is behind it (B) space remains empty, car behind is stopped (C) space is filled by the moving car behind it (D) car is stopped and cannot be moved (E) car moves out of the cell

Now we try to write down the combined transition rule for an automaton. For this we need to consider the states of the neighbor on its left (in green) and on its right (in blue). We first apply the rules for the first step to both pairs of sites that appear in the three-cell configuration. Then the rules for the second step are then applied to update the state for the automaton under consideration (in red). The results are shown in the following diagram.



Unfortunately, there is a subtle mistake in this construction of the

combined transition rule. The problem arises because of the probabilistic choice. Notice that such a choice is present in the first step, but not in the second step. To see the mistake in the combined rules, we can consider a single car on an empty highway so the automaton representing it (color in red) is in state 1. We will consider updating the following configuration:

E	E	E	1	E	E

We need to update the states for the green, red and blue automata, and so the relevant combination of cells are:

E	E	1	,	E	1	E	,	1	E	E

Using the transition rules on the first group of automata, we see that the green automaton has a probability p of remaining in state E and a probability of $1-p$ of switching to state 1. For the second group of automata, the red automaton has a probability p of switching to state 0 and a probability of $1-p$ of switching to state E. For the last group of automata, we see that the blue automaton must remain in state E. Since the choices for different automata are independent

of each other, we have four possible outcomes (since two relevant combinations involve probabilistic choices). They are:

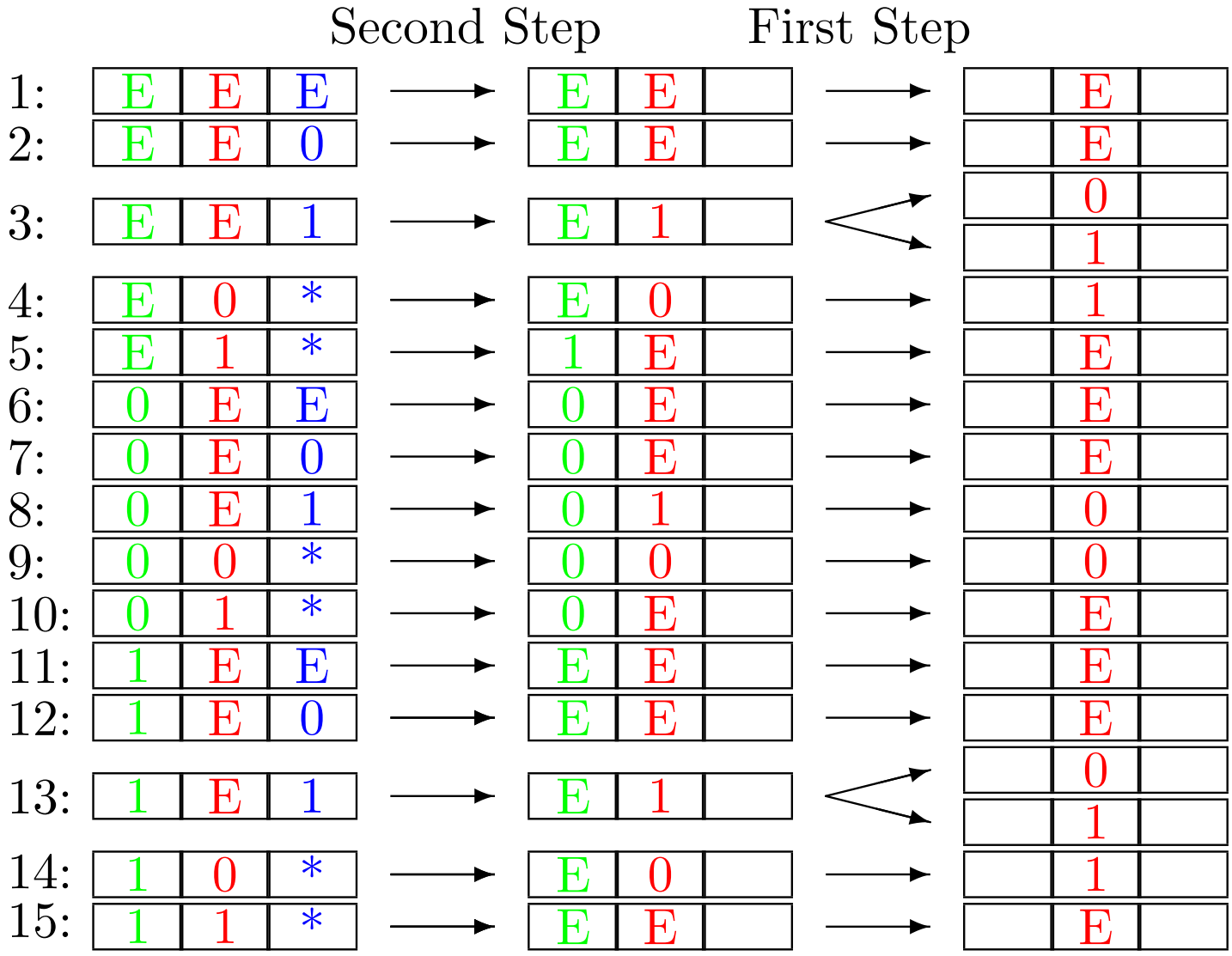
E	E	E	0	E	E
E	E	1	0	E	E
E	E	E	E	E	E
E	E	1	E	E	E

with probability p^2
with probability $(1-p)p$
with probability $p(1-p)$
with probability p^2

First of all, the probabilities are all incorrectly given by the rules. Secondly, the second configuration cannot be allowed since it has one extra car compared with the original configuration. Also the third configuration must be wrong since the original car has completely disappeared. The problem is that in the original two-step formulation, the probabilistic choice was only applied once. In the combined rule it is applied twice in two overlapping areas. This leads to the possible disappearance or duplication of cars, as well as to incorrect probabilities. This problem does not appear for deterministic rules where $p = 0$.

This problem can be avoided by performing the two steps in the

reverse order. That means that step 2 is performed before step 1. This second possibility gives the rule



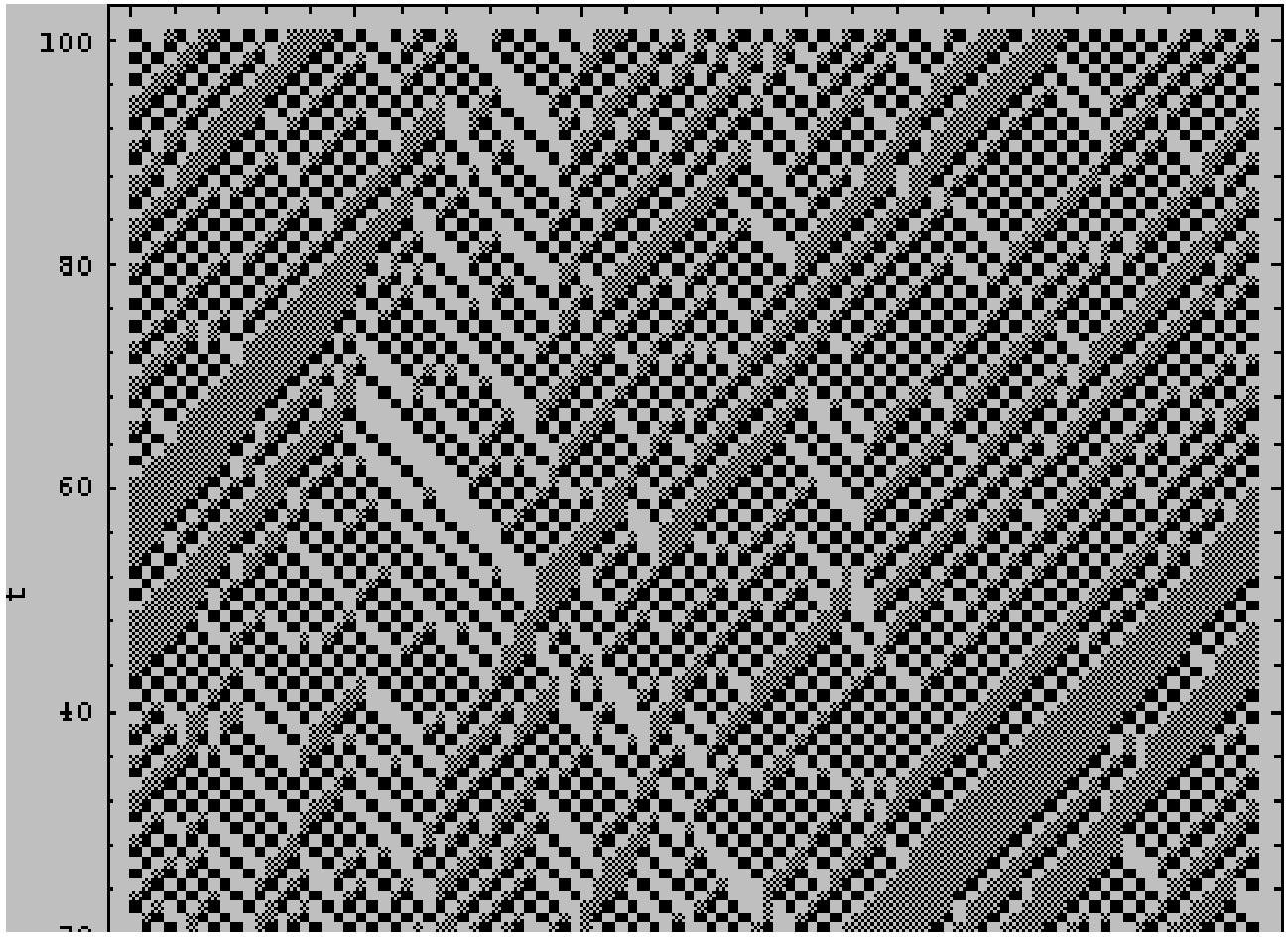
Here †marks input configurations that are not permitted, since a car is trying to pass another car. With this rule, the single car example

E	E	E	1	E	E

has the correct outcomes

E	E	0	E	E	E	with probability p
E	E	1	E	E	E	with probability $1-p$

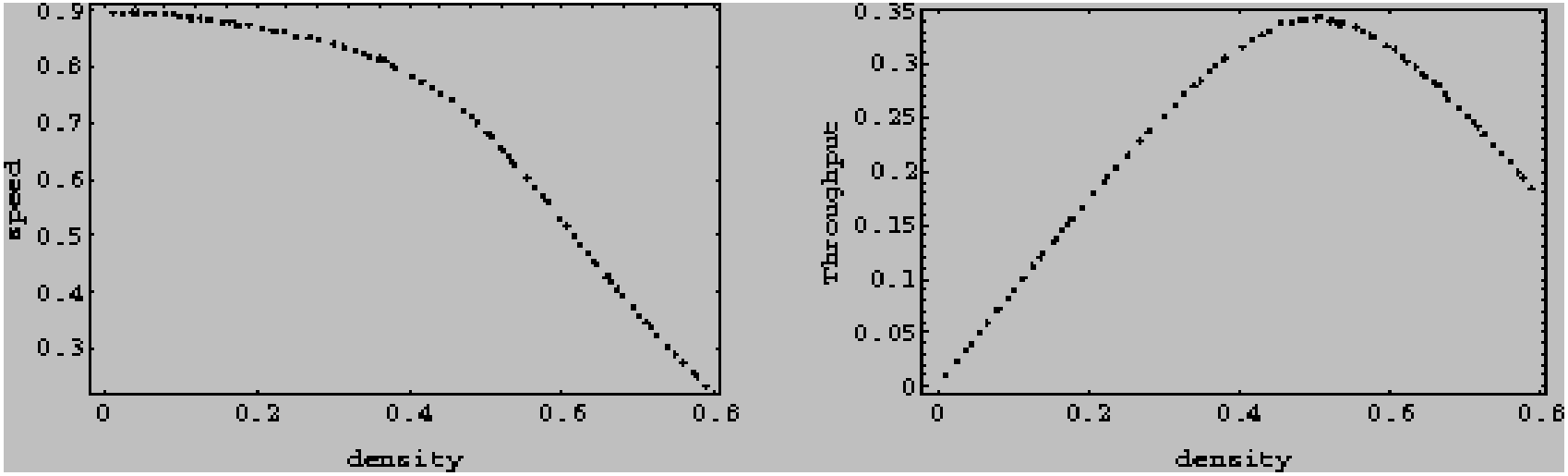
The difference between the two choices is that for the first choice, the velocity of the car is the velocity of the previous moving step, while for the second choice, it is the velocity of the next step. Because the probabilistic choice is done only in the last sub-step, it is done just once for each car, and therefore the combined rule is correct.



space-time trace for the traffic on a one-lane highway can be found in a book by Jörg R. Weimar[3]. The number of cells used is 100 and 55 cars are initially placed randomly on the highway. The unit of length is measured by the length of a cell and the unit of time is measured in units of a simulation time step. Therefore all quantities

will be in dimensionless units. Therefore in these units, the density of cars d is 0.55. Periodic boundary condition is imposed and the braking probability $p = 0.1$. Moving cars are indicated by black dots, stopped cars by dark grey dots, and empty spaces by a light grey background. We see cars moving (to the left), and some traffic jams (the dark grey areas indicating a pile-up of stopped cars), which propagate against the flow of traffic.

Simulation not only enable us to visualize events and phenomena we are interested in, very useful quantitative information can also be extracted. The most important quantities to extract from a traffic simulation are the average velocity v and the average throughput T, i.e., the number of cars passing a given point in a given time. These two values depend on the probability p, but more importantly, they depend on the density of cars on the highway. At high densities, cars are stalled most of the time, whereas for low densities, traffic can flow freely. In fact, we expect an average velocity of $1 - p$ for an almost empty highway, and a velocity of zero for a completely full highway. The throughput T, which can also be written as $T = vd$, should therefore be zero both at very low and at very high densities.



The following figure, obtained from the book written by Jörg R. Weimar http://www.tu-bs.de/institute/WiR/weimar shows the average speed and throughput as a function of the density of cars. The measurements were done on a lattice with 500 cells, and averaged for 2000 time steps. The braking probability is $p = 0.1$. We see that the throughput has a maximum value of 0.35 near the density of $d = 0.5$. This is the density at which cars can move at maximum speed while keeping the prescribed distance (of 1 cell in this case).

3.2. A Multi-Speed Model

The model can be made more realistic by extending it so that the cars can move at more than one speed.[3] The allowed values for the speed, v, are given by $0, 1, \ldots, v_{\max}$, where $v_{\max} > 1$ here. The single speed model corresponds to $v_{\max} = 1$. However the cars are still restricted to move along a single lane highway with traffic flowing to the left only. Consequently, as in the single speed model, the cars cannot pass each other.

Since $v_{\max} > 1$ the model can now incorporate the following two features. First, a stopped car with a lot of space in front of it will need a certain amount of time to build its speed up gradually before finally reaching its maximum velocity. Second, (as emphasized in the drivers hand-book) a driver must leave some space, w, between the car and the car in front. The higher the speed is the more space one has to keep.

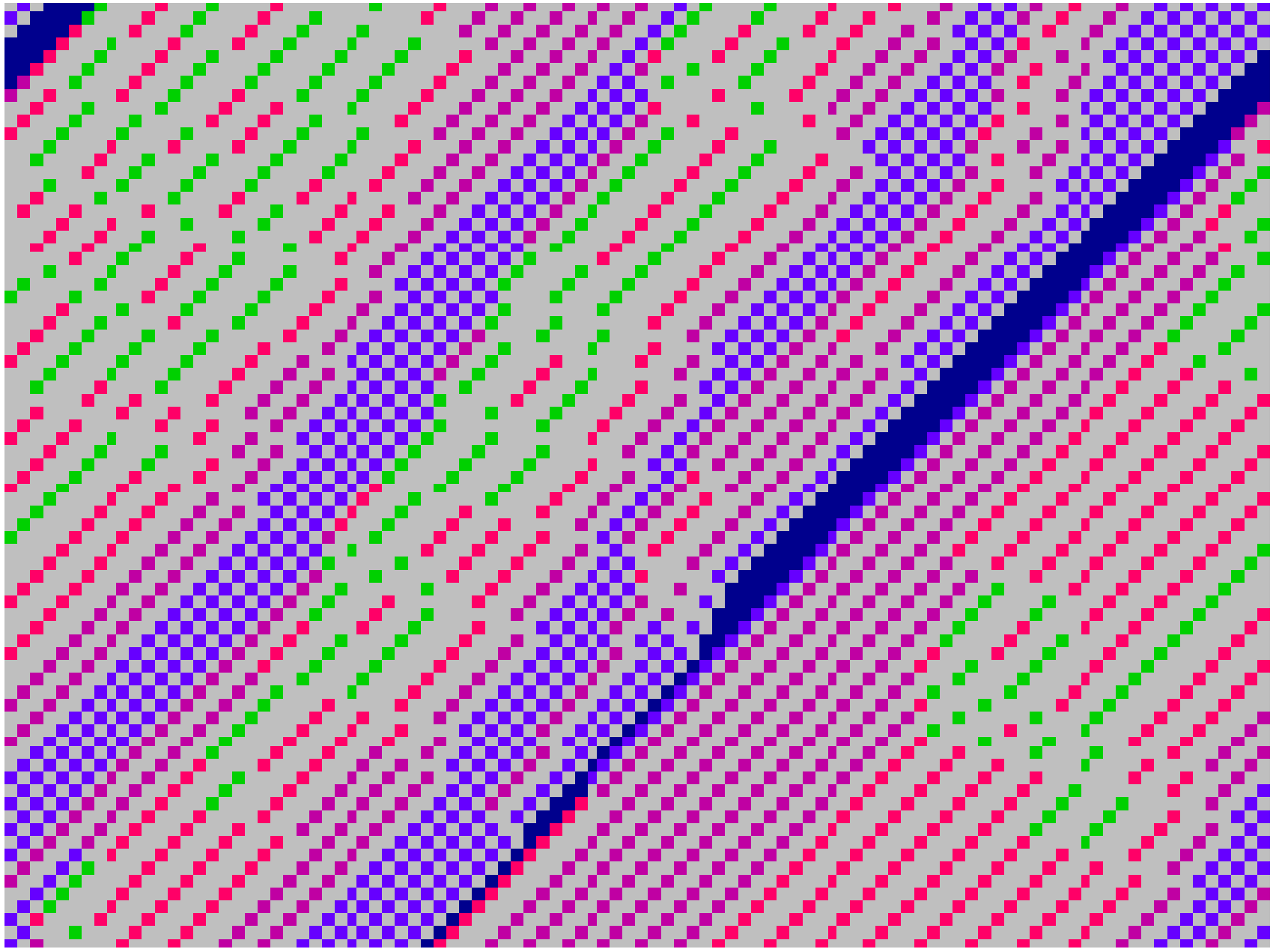
The present model uses the following multi-step rule: The rules are

1. Acceleration: If $v < v_{\max}$ then v is increased by 1.

2. Slowing down due to other cars: If there is a car located a distance w ahead and if $w < v$, then the velocity is reduced to $v = w - 1$.

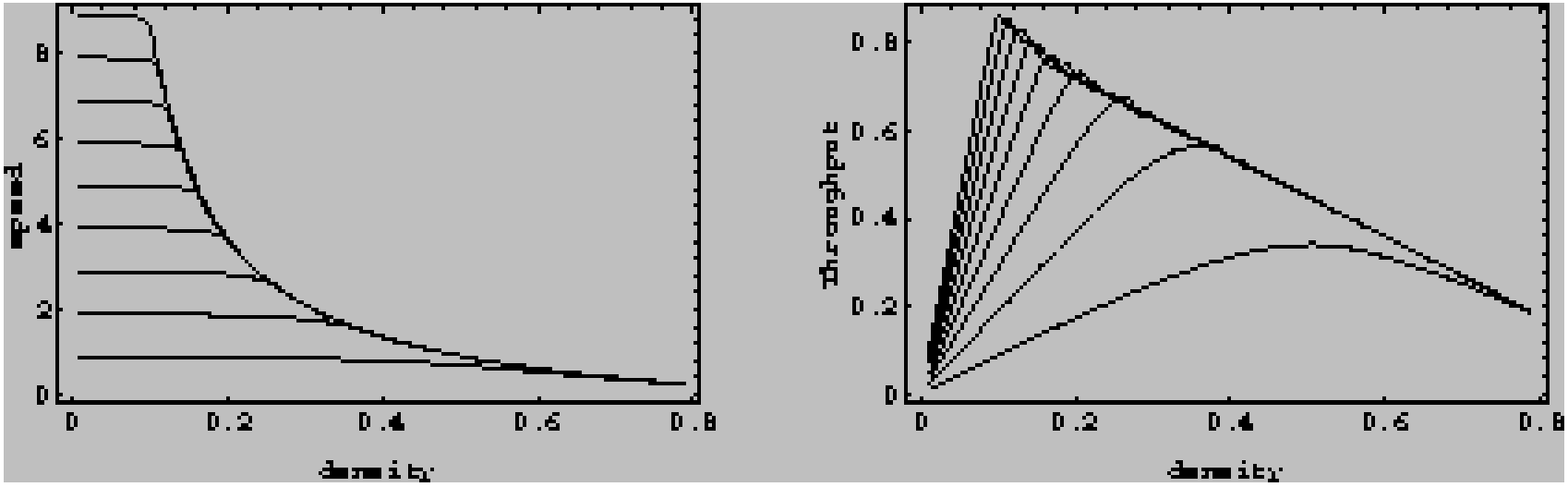
3. Randomization: The velocity of a car is decreased by 1 with a small probability p, unless it is already zero. That is v is replaced by $\max(0, v - 1)$.

4. Motion: Advance the car position by v cells.



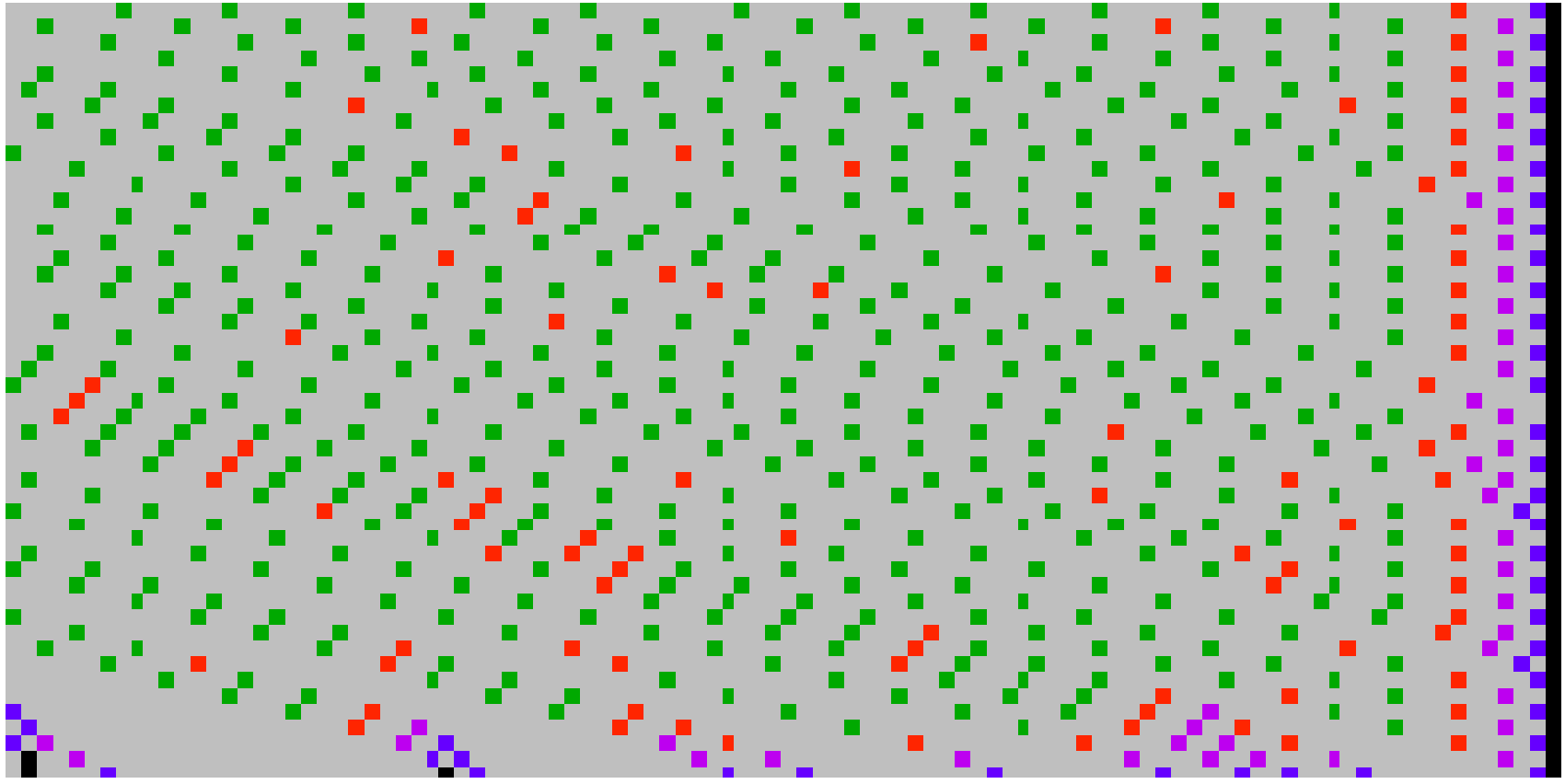
These rules can be implemented in the same way as before to

produce a cellular automaton. The resulting simulation for a lattice containing 100 cells is shown in the following figure. The figure is a space-time plot of the multi-speed traffic model with $v_{\max} = 4$, a density of cars $d = 0.3$ and $p = 0.1$. As in the single speed model, the periodic boundary condition is used. Light grey cells represent empty spaces. Green, red, purple, blue and dark blue cells represent cars moving with speed $4, 3, 2, 1$ and 0, respectively.



Simulations can be carried out for different values of $v_{\max}$. As in the single speed model, the average speed and average throughput can be computed as a function of the density. The results for different $v_{\max}$ are shown in the following figure where the lowest curve is for $v_{\max} = 1$,

and the highest for $v_{\max} = 9$. We see that the critical density at which the throughput has a maximum gets smaller for larger $v_{\max}$, and that the maximum is sharper. Above the critical density, an increased $v_{\max}$ does not significantly increase the average speed nor the throughput.



in a bottleneck situation where the road is fully filled on the right, and free out-flow is permitted on the left.

Another experiment can be done by simply changing the boundary

conditions: We fill the right boundary with cars with zero velocity, and delete all cars on the left boundary. This corresponds to the situation of a bottleneck, i.e., a section of the road that is single lane and being fed by a multi-lane highway. This leads to a jam at the beginning of the single lane section, and to a free flow at the end. Space-time plot of this situation with $v_{\max} = 4$ is shown in the figure. The traffic flow in the constricted region has a density of $d = 0.123$ and average velocity is 3.82. The throughput is therefore given by 0.47. This value is less than optimal for the case of $v_{\max} = 4$.

References

[1] G. Weisbuch, tr. by S. Ryckebysch, *Complex Systems Dynamics: An Introuction to Automata Networks*, Addison-Wesley publishing Company, 1990.

[2] S. Wolfram, Theory and Applications of Cellular Automata, World Scientific, 1986.

[3] Jörg R. Weimar, *Simulation with Cellular Automata*, Logos-Verlag, Berlin, 1998. 10, 20, 23

[4] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods: Applications to Physical Systems*, Second Edition, Ch. 11, Sec. 6. (Addison Wesley, New York, 1996).