Appendix A. Glossary

A.1 Notations and Definitions

a, b	Learning parameters for linear reinforcement scheme
$\underline{\alpha}$	Set of outputs of an automaton
α_i	i^{th} action of the automaton
$\alpha(n)$	Automaton action at time step n
$\beta(n)$	Environment response at time step n
c_i	Probability of (receiving) penalty for i^{th} action (P-model environment)
e_j^r	Unit vector of dimension r with j^{th} element equal to 1
s_i	Penalty strength for i^{th} action
$E[\cdot]$	Mathematical expectation
$E[\cdot \mid p_i(n)]$	Mathematical expectation given the probability $p_i(n)$
$F(\cdot)$	State transition function for an automaton (Chapter. 3); mapping for teacher responses (Chapter. 4)
f_{ij}^{β}	Probability of transition from state I to state j given the environment response β
g, h	Continuous nonnegative functions of probability vector p
$G(\cdot)$	Output function for an state-output automaton
$H(\cdot,\cdot)$	Output function for an automaton
$H(p)$	Update function used in nonlinear reinforcement scheme NL_H
k	Learning parameter for nonlinear reinforcement schemes
$L_{I\text{-}P}$	Linear inaction-penalty reinforcement scheme ($a = 0$)
$L_{R\text{-}I}$	Linear reward-inaction reinforcement scheme ($b = 0$)
$L_{R\text{-}P}$ or $L^{=}_{R\text{-}P}$	Linear reward-penalty reinforcement scheme with $a = b$
$L^{\neq}_{R\text{-}P}$	Linear reward-penalty reinforcement scheme with $a \neq b$
$M(n)$	Average penalty to the automaton at time step n
M_o	Average penalty for a pure chance automaton
μ_{ij}, η_{ij}	Coefficients of difference equation for action probabilities
NL_H	Nonlinear reinforcement scheme with update function H
$\|x\|_k$	k^{th} norm of vector x
$\underline{p}(n)$	Action probability vector
$p_i(n)$	Probability of choosing i^{th} action at time step n
$\underline{\Phi} = \{\varphi_1, \varphi_2, \ldots \varphi_s\}$	Set of internal states of an automaton

ϕ_i, ψ_i	Update functions of probability vector for nonlinear reinforcement schemes
r	Number or automaton actions
σ_{ij}^2	Covariance of $p_i(n)$ and $p_j(n)$
θ	Learning parameter for nonlinear reinforcement schemes
$V(x)$	Lyapunov function

A.2 Acronyms and Abbreviations

AHS	Automated Highway System
ASTM	American Society for Testing and Materials
ATIS	Advanced Traveler Information System
ATMS	Advanced Traffic Management System
AVC	Automatic Vehicle Control
AVCS	Advanced Vehicle Control System
AVI	Automatic Vehicle Identification
AVL	Automatic Vehicle Location
DOT	Department of Transportation
ELP	Electronic License Plate
ETT	Electronic Toll Tags
FHWA	Federal Highway Administration
FLASH	Flexible Low-cost Automated Scaled Highway (Laboratory)
GPS	Global Positioning System
ITE	Institute of Transportation Engineers
ITS	Intelligent Transportation Systems
IVHS	Intelligent Vehicle Highway System
LA	Learning Automata
NAHSC	National Automated Highway System Consortium
PATH	Program on Advanced Technology for the Highway, California
PSA	Precursor Systems Analyses
RIC	Remote Intelligent Communications
RVC	Roadside-Vehicle Communications
SAE	Society of Automotive Engineers
TRB	Transport Research Board
VRC	Vehicle-to-roadside Communications
VORAD	Vehicle On-Board Radar

Appendix B. Proof of Convergence of the "Optimal" Action with the Linear Inaction-Penalty Scheme $L_{I\text{-}P}$

The linear inaction-penalty scheme $L_{I\text{-}P}$ can be derived by a modification of the general linear reward-penalty scheme $L_{R\text{-}P}$ (See Section 6.1). This reinforcement scheme, as the name suggests, does not update the action probabilities when the environment response is affirmative:

$$\begin{aligned} &\textit{if } \alpha(n) = \alpha_i \\ &\quad \textit{for } \beta(n) = 0 \quad \begin{cases} p_i(n+1) = p_i(n) \\ p_j(n+1) = p_j(n) \qquad j \neq i \end{cases} \\ &\quad \textit{for } \beta(n) = 1 \quad \begin{cases} p_i(n+1) = (1-b)\ p_i(n) \\ p_j(n+1) = \dfrac{b}{r-1} + (1-b)\ p_j(n) \qquad j \neq i \end{cases} \end{aligned} \tag{B.1}$$

From the definition above, the expected value of the probability of an action at the next step can be written as:

$$\begin{aligned} E\left[p_i(n+1)\middle|p_i(n)\right] &= \sum_{k=1}^{r} E\left[p_i(n+1)\middle|p_i(n) \quad \alpha(n)=\alpha_k\right]\ p_k(n) \\ &= \sum_{k=1}^{r} c_k \ \ E\left[p_i(n+1)\middle|p_i(n) \quad \alpha(n)=\alpha_k \quad \beta(n)=1\right]\ p_k(n) \\ &\quad + \sum_{k=1}^{r} (1-c_k)\ \ E\left[p_i(n+1)\middle|p_i(n) \quad \alpha(n)=\alpha_k \quad \beta(n)=0\right]\ p_k(n) \\ &= c_i p_i(n)(1-b)\, p_i(n) + \sum_{k\neq i}^{r} c_k\, p_k(n)\left[\frac{b}{r-1} + (1-b)p_i(n)\right] \\ &\quad + (1-c_i)p_i(n)\, p_i(n) + \sum_{k\neq i}^{r} (1-c_k)\, p_k(n) p_i(n) \end{aligned} \tag{B.2}$$

We will again consider the "ideal case" where the probability of penalty for the optimal action is equal to zero, i.e., $c_\alpha = 0$ and $0 < c_{j\neq\alpha} < 1$. Then, for $i = \alpha$, we have:

$$E\left[p_\alpha(n+1)\middle|p_\alpha(n)\right] = 0 + \sum_{k\neq\alpha}^{r} c_k\, p_k(n)\left[\frac{b}{r-1} + (1-b)\, p_\alpha(n)\right] + p_\alpha^2(n) + \sum_{k\neq\alpha}^{r} (1-c_k)\, p_k(n) p_\alpha(n) \tag{B.3}$$

More explicitly:

$$\begin{aligned} E\left[p_\alpha(n+1)\middle|p_\alpha(n)\right] &= \frac{b}{r-1}\sum_{k\neq\alpha}^{r} c_k p_k(n) + p_\alpha(n)\sum_{k\neq\alpha}^{r} c_k p_k(n) - bp_i(n)\sum_{k\neq\alpha}^{r} c_k p_k(n) \\ &\quad + p_\alpha^2(n) + \sum_{k\neq\alpha}^{r} p_k(n)p_\alpha(n) - \sum_{k\neq\alpha}^{r} c_k p_k(n)p_\alpha(n) \\ &= \frac{b}{r-1}\sum_{k\neq\alpha}^{r} c_k p_k(n) - bp_\alpha(n)\sum_{k\neq\alpha}^{r} c_k p_k(n) + p_\alpha^2(n) + \sum_{k\neq\alpha}^{r} p_k(n)p_\alpha(n) \end{aligned} \tag{B.4}$$

Now, taking the expectation of both sides, and using the fact that $\sum_{i=1}^{r} p_i(n) = 1$, we obtain:

$$\begin{aligned} E\left[p_\alpha(n+1)\right] &= \frac{b}{r-1}\sum_{k\neq\alpha}^{r} c_k E\left[p_k(n)\right] - b\ E\left[p_\alpha(n)\sum_{k\neq\alpha}^{r} c_k p_k(n)\right] + E\left[p_\alpha^2(n)\right] \\ &\quad + E\left[\left(1-p_\alpha(n)\right)p_\alpha(n)\right] \end{aligned} \tag{B.5}$$

The simulations of the linear inaction-penalty scheme showed that the probability of the optimal action converges to a steady-state value, as seen in Figure B.1 For such a steady-state result, we can assume that:

$$As\ n \to \infty \qquad \begin{cases} E\left[p_\alpha(n)\right] = cons\tan t \equiv \overline{p} \\ \sigma^2_{p_\alpha(n)} = 0 \\ E\left[p_\alpha(n)\,p_j(n)\right] = E\left[p_\alpha(n)\right]\ E\left[p_j(n)\right] \end{cases} \tag{B.6}$$

Conditions above state that, as we approach the steady-state value for the probability of the optimal action, the expected value approaches the same value; the variance of the probability is negligibly small and the probability of all other actions are independent of the optimal action's probability, although the sum of probabilities must be equal to 1. (Since the probability of the optimal action is a constant value, this value and other probabilities can be treated as independent variables.) Using these facts, the identity $E\left[x^2\right] = E[x]^2 - \sigma_x^2$, and renaming $E\left[\sum_{k\neq\alpha}^{r} c_k p_k(n)\right] \equiv A$, we can simplify the equality in Equation B.5 as follows:

$$\begin{aligned} E\left[p_\alpha(n+1)\right] &= \frac{b}{r-1}\sum_{k\neq\alpha}^{r} c_k E\left[p_k(n)\right] - b\ E\left[p_\alpha(n)\right]\ E\left[\sum_{k\neq\alpha}^{r} c_k p_k(n)\right] \\ &\quad + E^2\left[p_\alpha(n)\right] + E\left[p_\alpha(n)\right] - E^2\left[p_\alpha(n)\right] \end{aligned} \tag{B.7}$$

Or:

$$\overline{p} = \frac{b}{r-1}\ A - b\overline{p}A + \overline{p}^2 + \overline{p} - \overline{p}^2 \ \Leftrightarrow\ 0 = \frac{b}{r-1}\ A - b\overline{p}A$$
$$\Leftrightarrow\ \overline{p} = \frac{bA}{bA(r-1)} = \frac{1}{(r-1)} \qquad \text{(B.8)}$$

where r is the number of actions. This proves that a learning automaton using the linear inaction-penalty scheme in a stationary environment where there is one (and only one) "optimal" action is not ε-optimal (except where r = 2). Furthermore, the probability of the optimal action converges to 1/(r-1). Figure B.1 shows the probabilities of five actions where the optimal action's probability converges to 1/(5-1) = 0.25.

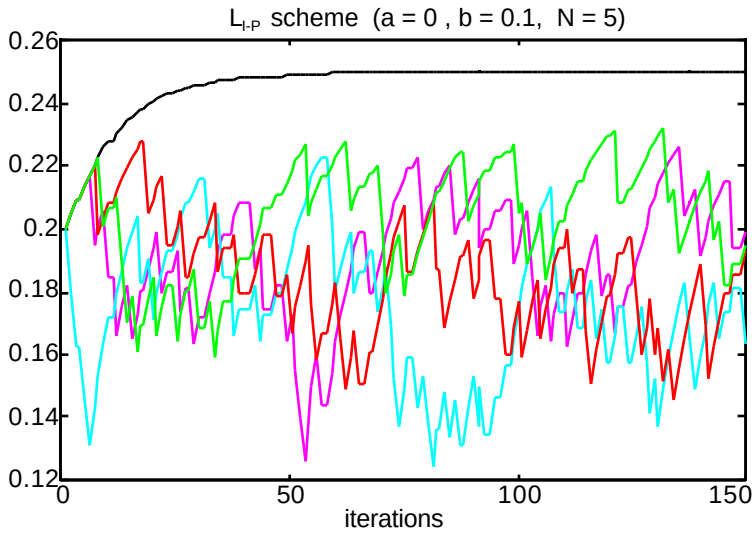

Figure B.1. Probabilities of five actions in the $L_{I\text{-}P}$ scheme; only $c_1 = 0$.

Appendix C. Simulation

C.1. The Program

The simulation for the work described in this dissertation is written in Matlab® [Matlab96]. It consists of a group of subroutines and functions coded as *m-files*, and can be run in Matlab version 4.2 or higher. Graphic user interfaces are designed on a Sun® workstation running Solaris 2.4® and Openwindows®, but can be displayed on any platform running Matlab 4.2 or higher, after making minor changes in the GUI subroutines.

Figure C.1. shows the overall structure of the simulation. The description of the files are given in Table C.1. Subroutine *m_run.m* is the main simulation program which calls several subroutines at each time step. At each time step, sensor outputs are evaluated and action decisions are made. Then, those actions are carried out for each automated vehicle on the highway. Sensor modules and flags are implemented as subroutines that take vehicle index and returns corresponding output. The learning, decision (planning), and regulation subroutines work similarly. When evaluations are complete, the highway is updated. The main program repeats the loop until the final time is reached or the program is interrupted by the user.

Learning algorithm, learning parameters, processing speed and memory vector sizes cannot be changed during the simulation run. Other parameters such as current vehicle speed, desired speed, current vehicle lane, desired lane, permitted speed variations can be changed during the simulation as well as other display parameters. For detailed description of some of the graphic user interfaces, see Figures C.3-C.7. Command line interface can be used to change parameters before the simulation run. It also displays several parameters and/or actions during the simulation run (Figure C.2).

Files described in Table C.1 and Figure C.1 are provided here as zipped archives in two different formats:

- Unix: mfiles.tar.Z
- Windows: mfiles.zip

File	Inputs	Output	Description
laneb.m	Vehicle index	Integer indicating desired lane shift	Evaluates current vehicle lane based on link layer information
mlma.m	-	-	Main program
move2.m	Vehicle index	-	Updates vehicle position
m_init.m	-	-	Initializes all parameters, arrays, etc.
m_mvec.m	-	-	Plots simulation data
m_movie.m	-	-	Shows the movie of the simulation
m_plap.m	-	-	Plots simulation data
m_plot.m	-	-	Subroutine for Plot GUI
m_redo.m	-	-	Subroutine for Scenario GUI
m_reinit.m	-	-	Reinitializes all parameters, arrays, etc.
m_run.m	-	-	Main subroutine for simulation runs
m_sandl.m	-	-	Plots simulation data
m_setgui.m	-	-	Displays the main graphic user interface
m_traj.m	-	-	Plots simulation data
pinch.m	Vehicle index	Binary value indicating pinch condition	Function for pinch module
phmin.m	Action probabilities, action, environment response	New action, new action probabilities	Function for nonlinear nonlinear scheme of Baba [Baba85].
Phfun.m	Action probabilities, action, environment response	New action, new action probabilities	Function for nonlinear reinforcement scheme of Section 6.2.
plan2.m	Vehicle index	-	Subroutine for planning layer
plot_h2.m	Vehicle positions	-	Subroutine for display updates
plrp.m	Action probabilities, action, environment response	New action, new action probabilities	Function for general linear reward-penalty learning scheme.
Reg2.m	Vehicle index	-	Subroutine for regulation layer
sb_fr.m	Vehicle index	Binary value indicating module response	Function for headway module
sb_lft.m	Vehicle index		Function for left sensor module
sb_lftn.m	Vehicle index		Function for extended left sensor module
sb_sp.m	Vehicle index		Function for speed module
sb_rgt.m	Vehicle index		Function for right sensor module
sb_rgtn.m	Vehicle index		Function for extended right sensor module
speedb.m	Vehicle index	Speed flag	Evaluates vehicle speed condition

Table C1. Description of the subroutines and functions for multiple lane, multiple automata intelligent vehicle simulation.

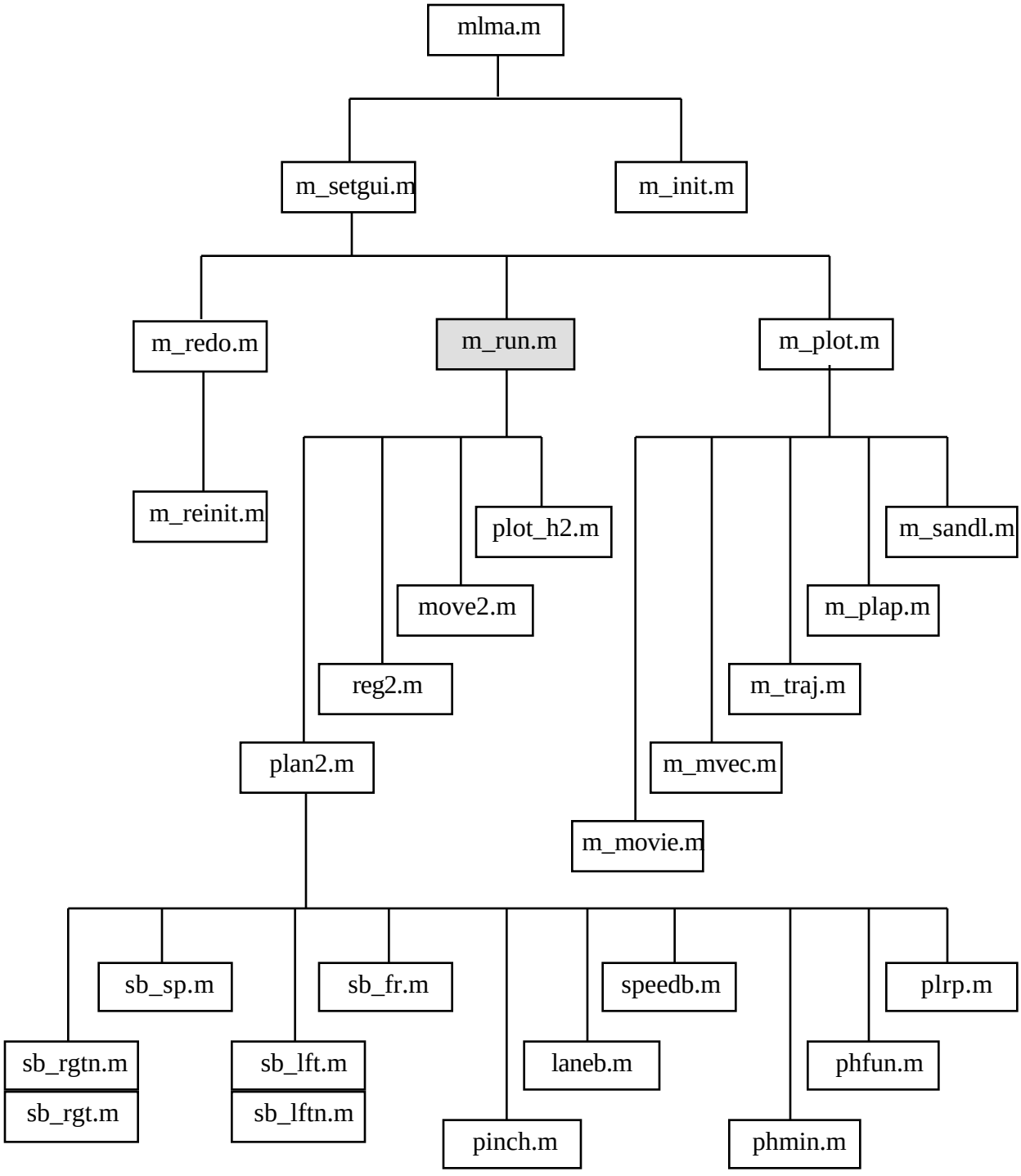

Figure C1. Structure of the simulation program.

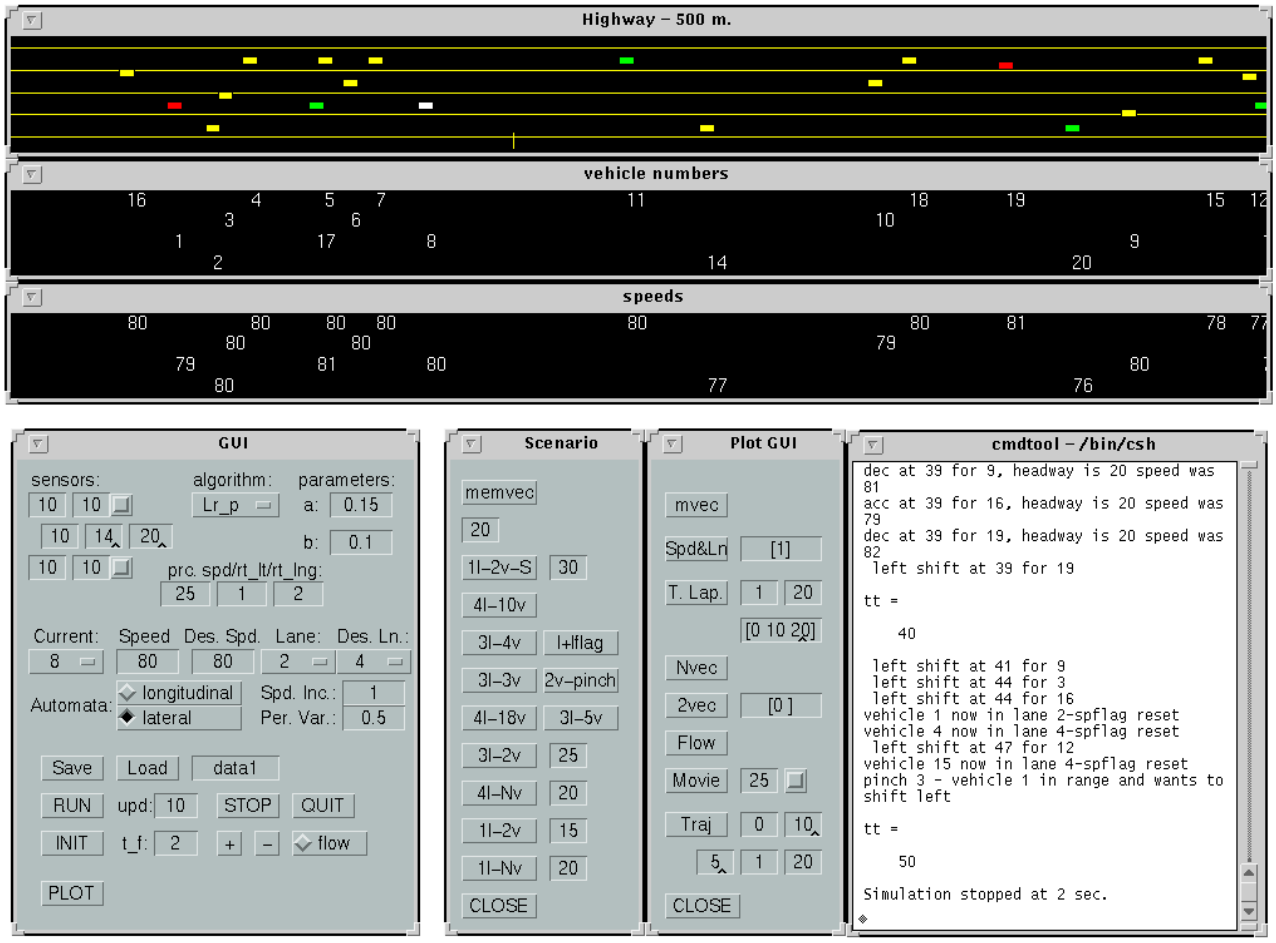

Figure C.2. The simulation.

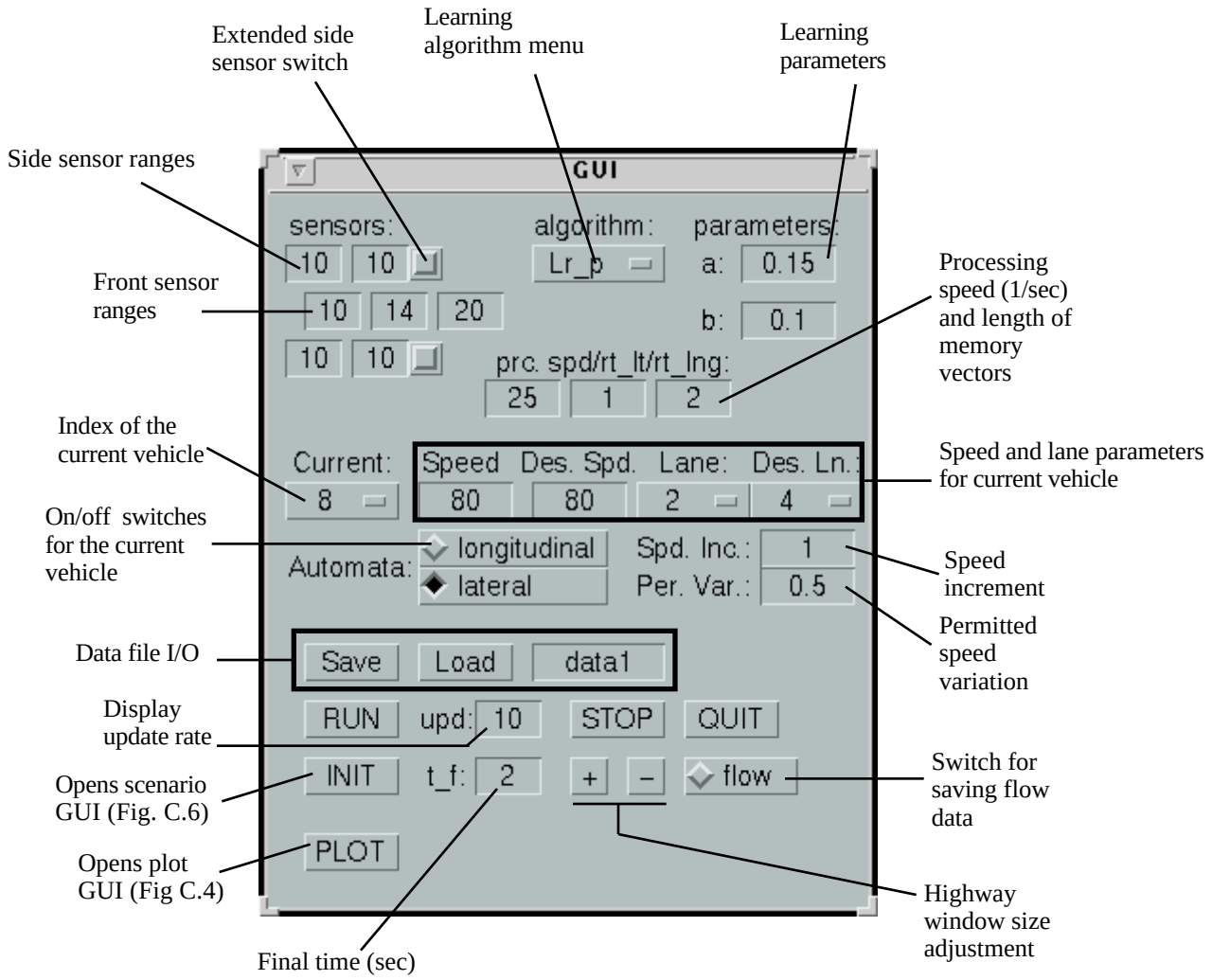

Figure C.3. Graphic User Interface.

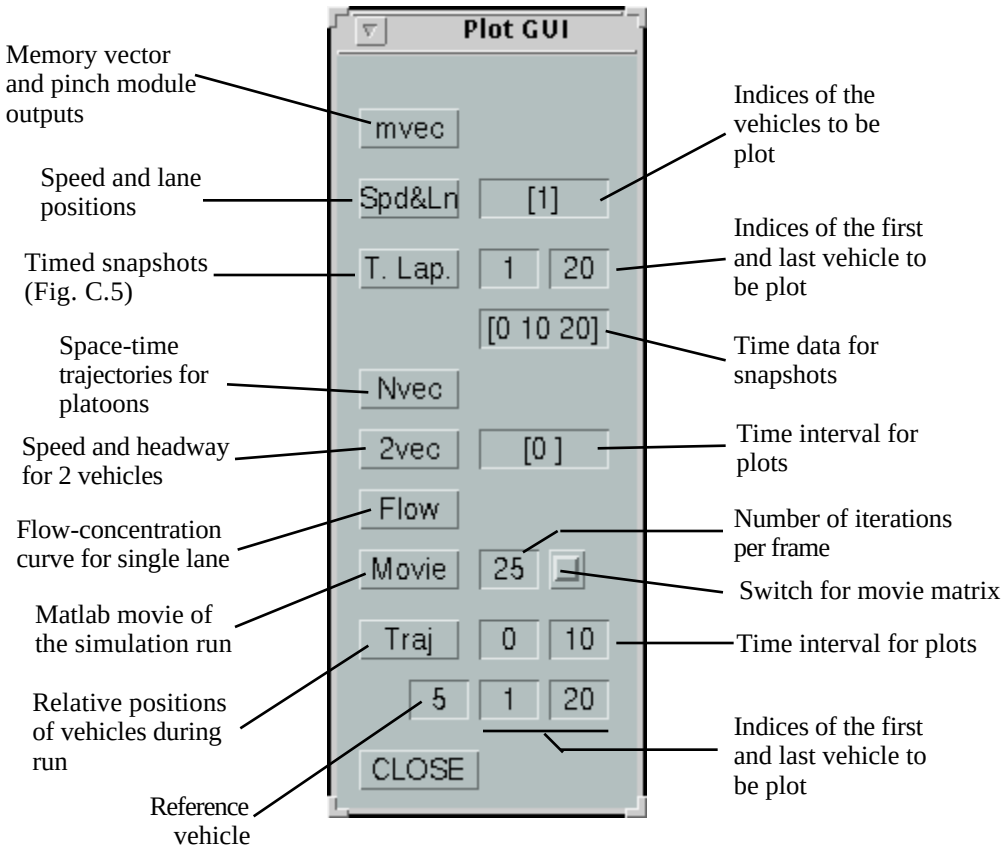

Figure C.4. GUI for data visualization.

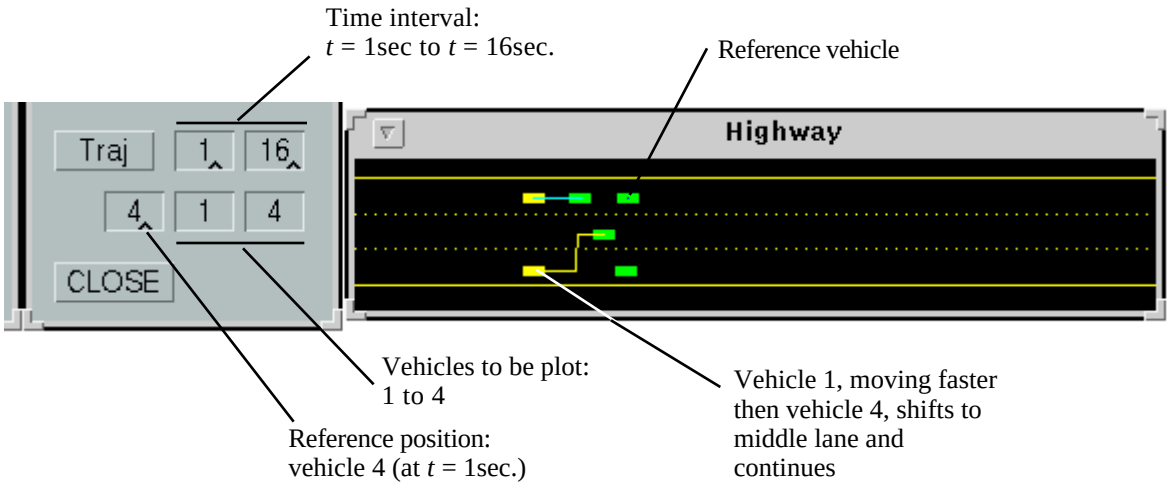

Figure C.5. "Trajectory" command for relative position plots.

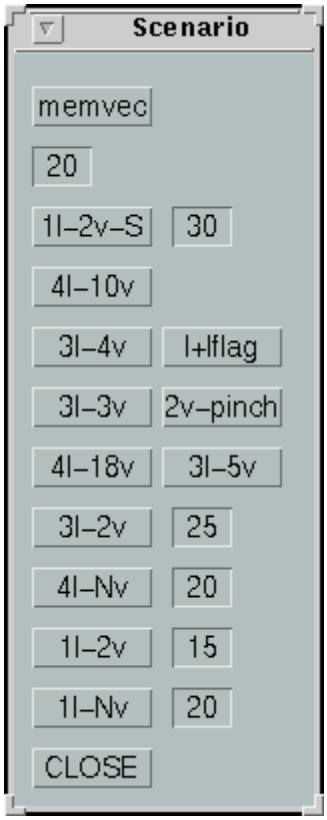

Figure C.6. Scenario window: Clickable buttons initialize several different scenarios.

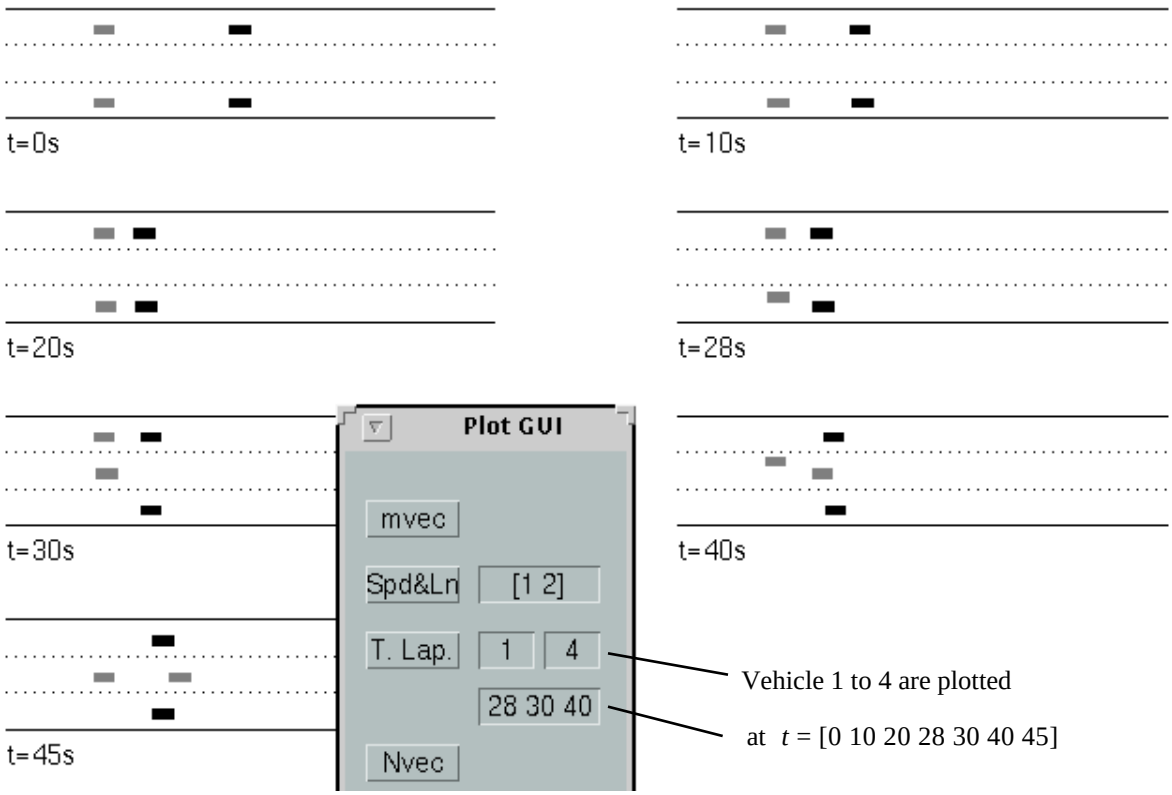

Figure C.7. Timed snapshots of a simulation run.

C.2. Additional Simulation Files

Additional simulation examples of the different highway situations are given in this section. The following table gives the description of the situations and the *mpeg* movie file of the test run.

File	Simulation Parameters and Description
appCm1.mpg 0.263Mb 40sec 5 frames/sec	20 vehicles with various initial speed definitions changing to the same lane: - Desired speed and lane for all vehicles are 85kmh and lane 2 respectively. - Algorithm: NL_H with $k = 3$, $\theta = 0.05$. - Memory vector sizes: 9 (lateral and longitudinal) - Sensor ranges: 10m side, 11m, 15m, and 20m front sensor. - All vehicles shift to lane 2, but due to speed variations, some of the vehicles shift left and right to avoid collision. At t = 40sec, average platoon speed is approximately 85kmh.
AppCm2.mpg 0.264Mb 40 sec. 5 frames/sec	Three platoons with various memory vector sizes: - 3 ten-vehicle platoons with same initial speeds of 80mph and inter-platoon distances of 15m. Leaders have a desired speed of 70mph. - Algorithm: $L_{R\text{-}P}$ with $a = 0.15$ and $b = 0.10$. - Memory vector sizes: Platoon 1 (yellow) - lateral: 25, longitudinal: 13 Platoon 2 (green) - lateral: 13, longitudinal: 13 Platoon 3 (red) - lateral: 13, longitudinal: 9 - Followers in the platoon try to match leaders speed change. In platoons 1 and 2, some of the vehicle (4, 5, and 10) shifts lane to avoid collisions. The size of the lateral memory vector affects the time to shift lanes slightly. The effect of the longitudinal memory vector is more significant. No vehicle shift lane in platoon 3 because of faster reactions to headway changes.
appCm3.mpg 0.209Mb 40 sec. 5 frames/sec	Three separate group of four vehicles in the situation of Section 7.4.3: - $V_1 = V_2$ =85mph, $V_1 = V_2$ =805mph; initial vehicle separations are 16m; no lane preferences are set for vehicles 1 and 2. - Algorithms and memory vector sizes: Group 1 (yellow) - L_{R-P} ($a = 0.15$, $b = 0.10$) lateral: 25, longitudinal: 13 Group 2 (green) - NL_H ($k = 10$, $\theta = 0.05$) lateral: 25, longitudinal: 13 Group 3 (red) - NL_H ($k = 10$, $\theta = 0.05$) lateral: 13, longitudinal: 9 - Speed flags in groups 1 and 2 are set approximately at the same time (10.04 and 8.8 seconds for group 1; 9.64 and 9.36 for group 2). The lane changes are made also made approximately at the same time. For group 1, lane changes occur before the flags are set, due to short lateral memory vector.
Sensor ranges for the last two simulations are 10m side, 10m, 15m, and 20m front. Processing speed for all simulations is 25Hz.	

Table C.2. Description of the situations and simulation files.

Appendix D. States of the Environment for Multiple Vehicle Interactions

The tables given in this section show the possible positions of vehicles in a highway for the scenarios described in Chapter 7. They indicate relative positions of multiple vehicles with respect to each other's sensor ranges. Each row in a matrix corresponds to a lane; each square illustrates a road section which falls into the side sensor range of an automated vehicle. A dark square indicates the presence of a vehicle. Not all possibilities are considered; instead, only the situations that are of interest for a specific scenario are listed. Similar situations are then combined into a single 'state' and simplified if necessary. Two situations are said to be similar if the sensor module outputs and/or possible actions are the same for both. Three scenarios of the Chapter 7 are:

- Two Vehicles on a Three-Lane Highway
  Considering the immediate neighborhood of the vehicles, there are C(9,2) = 36 possible situations shown in Figure D.1. Most of these situations are similar and, therefore, can be combined as shown in Figure D.2. Further simplifications are illustrated in Figure D.3. These final 12 situations are the states given in Section 7.4.

- Three Vehicles on a Three-Lane Highway
  Considering the immediate neighborhood of the vehicles, there are C(9,3) = 84 possible situations shown in Figure D.4. Some of these situations are similar and, therefore, are combined as shown in Figure D.5. These final 64 situations are the states given for 3-vehicle scenario in Section 7.4.

- Four vehicles on a Three-Lane Highway
  This four vehicle situation of Figure 7.13 can be treated as a two-vehicle situation with other vehicles treated as obstacles. The two vehicles in front are traveling at their desired speed and lane, unaware of the other approaching vehicles (See Section 7.4.3). Therefore, there are C(7,2) = 28 possible situations shown in Figure D.6. Since the conflict will be solved when the two trailing vehicles are both in the middle lane, the states E3, E4, and/or F3 are goal states. Again, we considered only immediate neighborhoods, and therefore, the physical locations of the vehicles are shown using a 3x4 matrix.

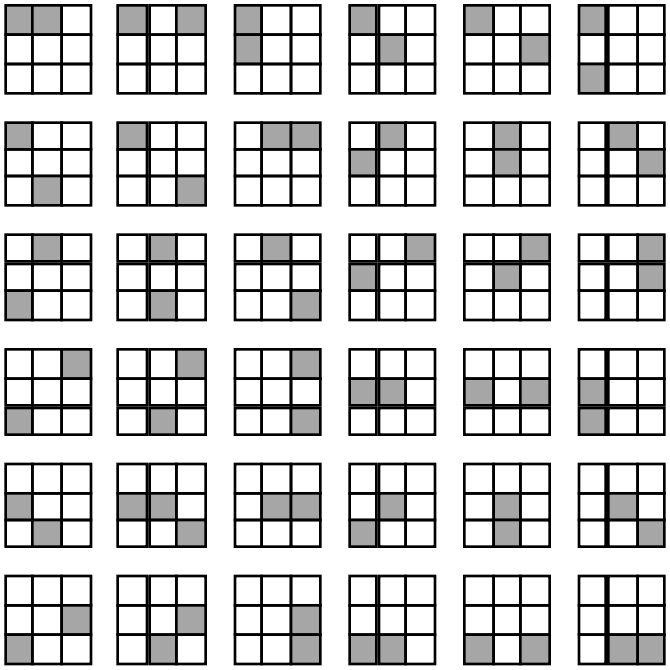

Figure D.1. All possible immediate neighborhood situations for two vehicles.

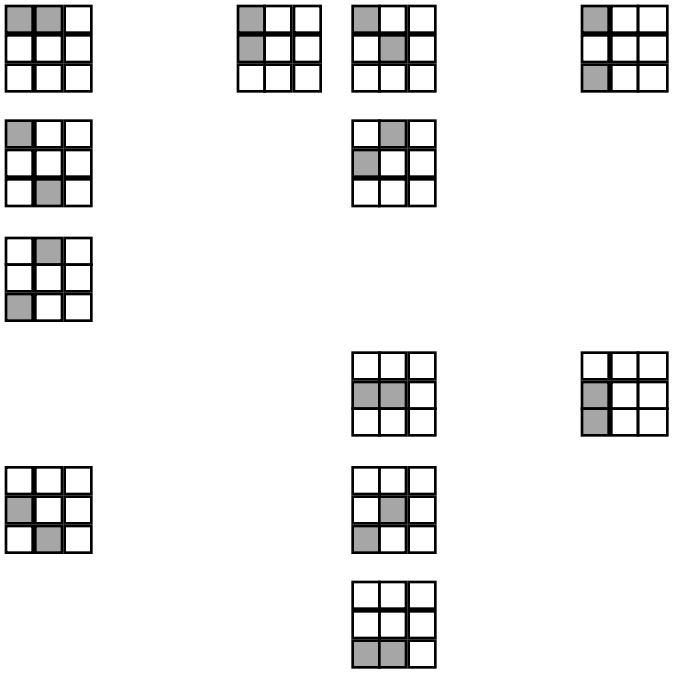

Figure D.2. Combined states for two vehicles: states not shown are identical to those given here.

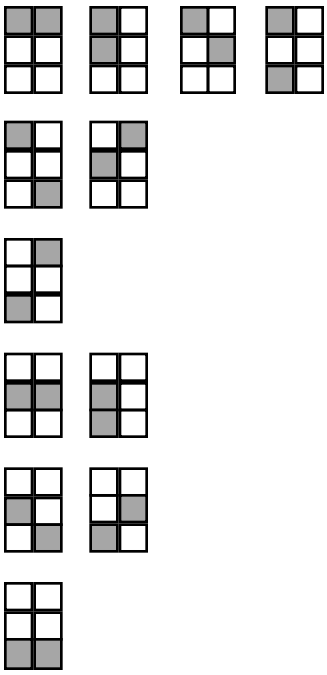

Figure D.3. Further simplified states.

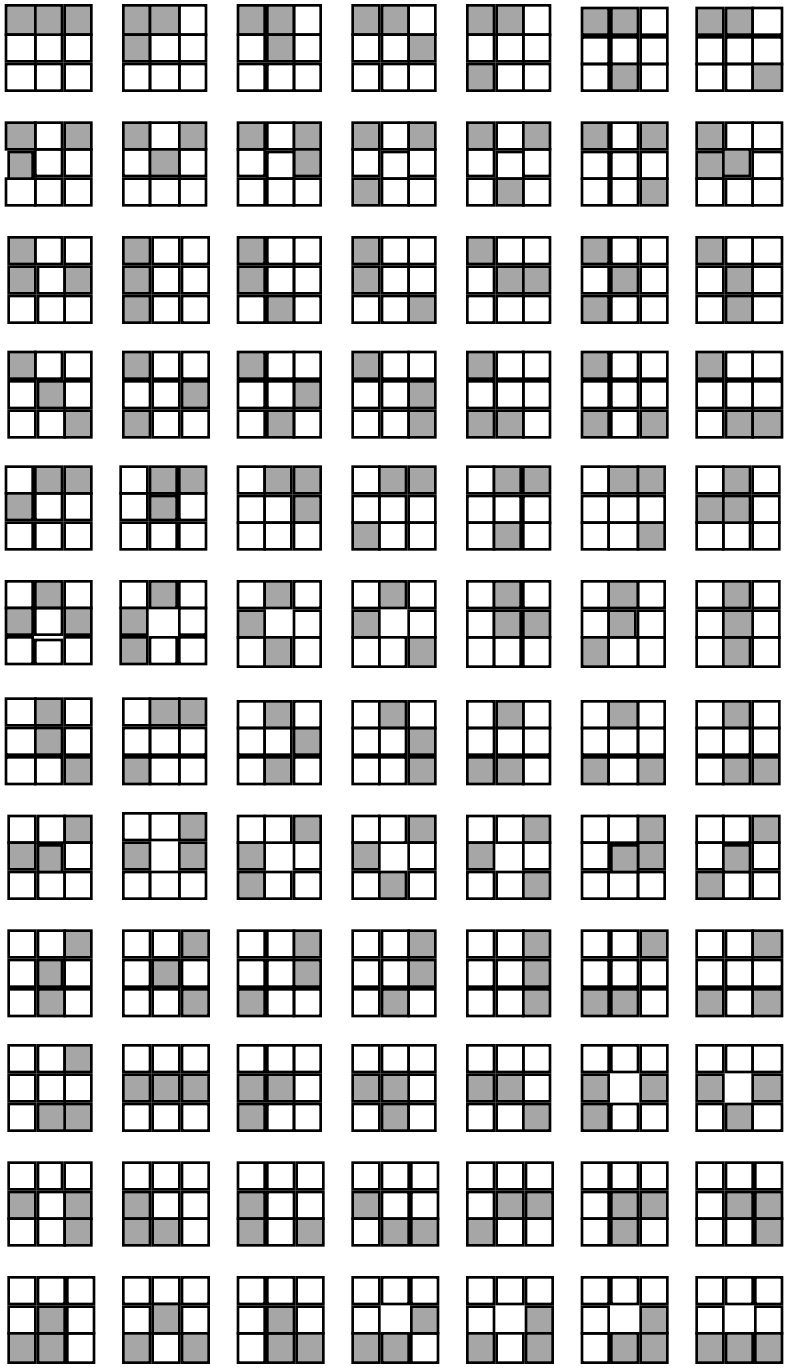

Figure D.4. All possible immediate neighborhood situations for three vehicles in a three-lane highway.

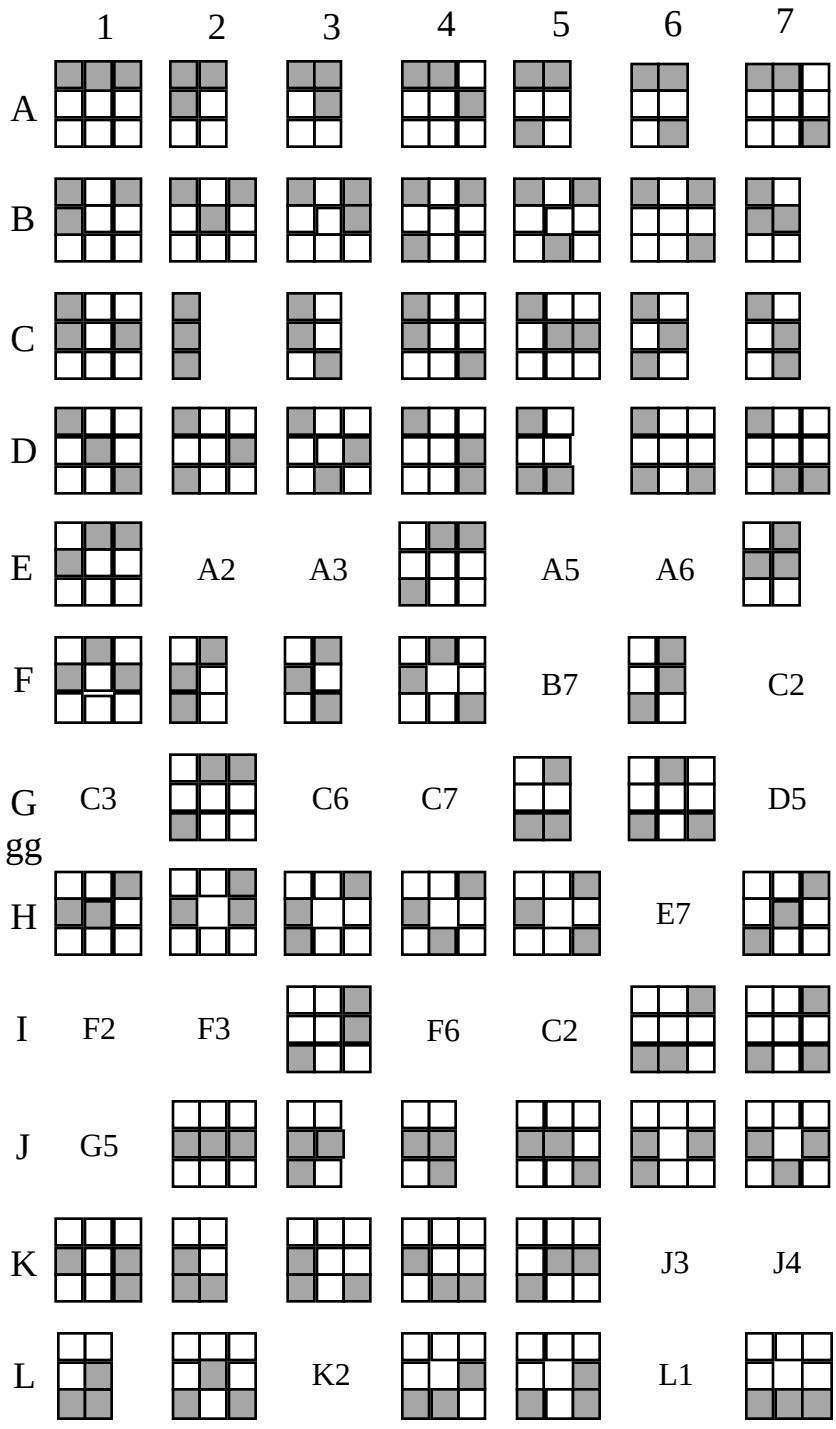

Figure D.5. Combined states for three vehicles: states not shown are identical to those indicated.

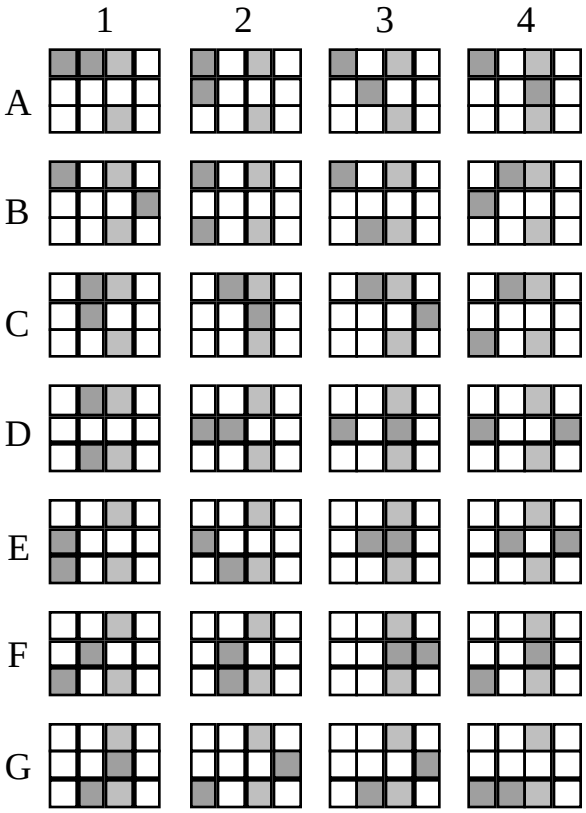

Figure D.6. Possible states for four vehicles ¾ actually a two-vehicle situation.