A Tetris Controller - An Example of a Discrete Event Dynamic System

R. Germundsson
Department of Electrical Engineering
Linköping University
S-581 83 Linköping
SWEDEN

Email: `roger@isy.liu.se`

November 18, 1991

Abstract

In this paper we build a controller for a Tetris game. A model inspired by control theory is used and a control law derived. This control law has certain degrees of freedom, i.e. some parameters to tune. Stochastic approximation is used to derive the "optimal" controller within this framework. An implementation of this control law with an existing Tetris game has been done and all code is available inorder for more control schemes to be proposed. This process is also proposed as a test case for discrete event methods.

1 Introduction

1.1 About Tetris

The computer game Tetris consists of a well, where the player is supposed to place the incoming pieces properly at the bottom. These pieces should be placed in such a way (if possible) that no holes are present. Once a row is completed it gets deleted and the player will get some extra points. The purpose of the game is to keep playing for as long as possible or equivalently to obtain as high score as possible. See section 2 for a more detailed and graphic description.

The game itself is originally a russian (!) computer game and seems to be immenseley popular. It has no doubt found its way into many departments (and homes of course).

1.2 Motivation

I engaged in this project at a time when I had started to read up on Discrete Event Dynamic Systems (DEDS) and thought a lot about what kinds of systems and properties you where able to model in those frameworks. At the same time Tetris arrived at our department and it was simply irresistable to first of all model and secondly control such a game. In retrospect I feel this problem has many of the real world properties present when dealing with DEDS problems, i.e. large, unstructured and without apparent algebraic structure.

1.3 Discrete Event Dynamic Systems

One might wonder what makes a game of Tetris a DEDS. However DEDS are characterized by:

- Event driven in/out relation.
- Discrete state space.
- May or may not have performance criterion associated with it.

It is easily realized that Tetris satisfies all of these criteria. Unfortunately there is no uniform way of describing DEDS, but several interesting subclasses exist:

- **Finite Automata.** Most models that are analyzed (as opposed to merely simulated) are finite automata, see e.g. Ramadge and Wonham [10, 9, 7, 8, 11]
- **Petri Nets.** One important subclass is bounded Petri Nets which is no more than finite automata. See e.g. Reisig [12] or Peterson [6] for a more complete reference list.

There are also several other description languages for DEDS, e.g. Communicating Sequential Processes (CSP) [5] and Signal [1], but generally the prospect of doing analysis w.r.t. these are fairly limited.

A Tetris game is indeed a finite automaton, but with a very large state space (in the order of 2^{200}). Hence we could in principle describe Tetris using any of the given formalisms above and in some sense one could say that the game program implements a finite automata. However, due to the very large state space more detailed analysis is ruled out. Instead we turn to simulation and tuning parameters. This is akin to performing peturbation analysis (PA) of queuing networks, see Ho [4]. Unfortunately the PA methods require quite detailed knowledge of how parameters affect the statetrajectory (since you basically reconstruct the new trajectory) this detailed knowledge is in some sense available, but the reconstruction seems to be just as complicated as performing a fresh simulation.

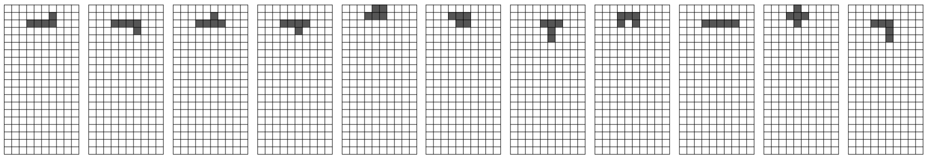


Figure 1: The various pieces available for Tetris.

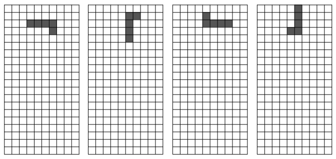


Figure 2: The four rotations of a figure.

1.4 Challenge

Actually games are not that far fetched when it comes to control theory, see e.g. Bryson and Ho [2] for the game theoretic approach to optimal control. Also animation is a good way to perceive the dynamics in processes. This controller is implemented as a modification of a public domain Tetris game and the code can be obtained from the author. Although modifying an existing code does not produce the shortest code it is probably optimal in the sense of the amount of work that has to be put in to get a working controller. I believe it is fairly straight forward to modify the controller. Also since the approach proposed in this paper certainly is suboptimal in a more global sense it should be possible to outperform this controller. *This is a challenge to the control community, i.e. build a better controller in the sense of producing consistently higher score.*

2 Game Description

In this section the game Tetris will be described and illustrated. In the next section the corresponding control theoretic formulation is given.

First of all you have a game board, in our case a 10 by 20 grid. At the top there will arrive various pieces in this case 11 different sorts, see figure 1.[1] You are then supposed to place these figures on the gameboard using your controls, i.e. rotate (see figure 2), translate (see figure 3) and drop (see figure 4), in such a way that you can handle as many figures as possible. The figures will fall to the bottom of the game board and completely filled rows will be deleted (see figure 5). What makes the game difficult (interesting?) is that it is also time

[1] This is really a nonstandard Tetris. In the standard Tetris each figure consist of four pieces. That game is a lot easier to play ...

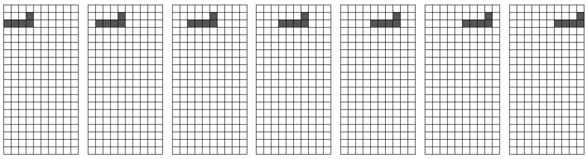


Figure 3: Translating a piece across the board.

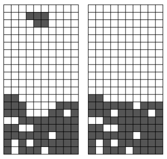


Figure 4: Drop a figure to the bottom

driven, in the sense that it simply moves the figures downward periodically and hence you have to make decisions rapidly as to where you will place your figure otherwise your options shrink fast. There is also a score function associated with the game that is set up to first of all give you credit for being able to place the figure. It is linear in the level, i.e. the pace at which pieces move downward, and linear in the height, i.e. your score is higher for each piece if you place the figure higher on the board however the pieces will fall as far to the bottom as is possible.You will also get a bonus if you drop the figure, i.e. let it fall to the bottom. This bonus is proportional (equal to) the height from which you drop the figure.

A typical sequence of game moves is shown in figure 6 where the figure to be placed is shown at the top of the grid (as is the case when you play the game).

3 Problem Formulation

In this section we provide a conceptual (control theoretic) model of the Tetris game described in the last section. The basic formulation is as an optimal control problem. This optimal control problem is solved (numerically) using stochastic approximation in the next section.

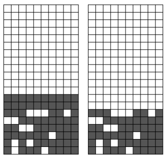


Figure 5: A row gets deleted.

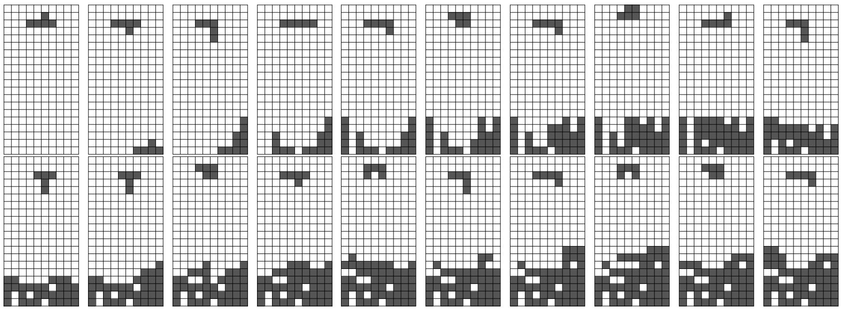


Figure 6: A typical game sequence.

In our model for the Tetris game we will overlook the time dependent nature of the process, i.e. we will build a *logical* model or an *event driven* model. This is no great restriction if we assume thet the control law is "fast enough", a common abstraction in logical DEDS. Using this simplification we can write the Tetris game process as:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), v(k)) & (1) \\ s(k+1) &= s(k) + g(x(k), u(k), v(k)) & (2) \\ x(k) &\in X & (3) \\ u(k) &\in R \times T \times D = U & (4) \\ v(k) &\in F & (5) \end{aligned}$$

with the following interpretations:

x: The current state of the tetris game, i.e. the current grid pattern. This may be conveniently thought of as an element of $Z_2^{10\times 20}$. ($X = Z_2^{10\times 20}$)

s: The current score of the game process. This is a natural integer.

u: The input signal or control, which consist of three parts $u = [r, t, d] \in R \times T \times D = U$, i.e.

- r: The current rotation. May be thought of as an element of Z_4.
- t: The current translation. May be thought of as an element of Z_{10}.
- d: If we decide to drop the incoming piece or not. This may be thought of an element of Z_2.

($U = Z_4 \times Z_{10} \times Z_2$)

v: The incoming piece. This may be thought as an element of Z_{11}. Furthermore the stochastic process $\{v(k)\}$ is white and uniformly distributed. ($F = Z_{11}$)

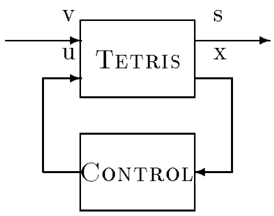


Figure 7: Block diagram representation of the Tetris controller.

where $Z_k = \{0, 1, \ldots, k-1\}$. Strictly speaking f, the transition function, is not defined on the full set $X \times U \times F$, e.g. some translations may not be possible. This is however a minor point and the actual implementation of course take notice of this by defining f over $\mathcal{XUV} = \{[x, u, v] \in X \times U \times F : Safe(x, u, v)\}$.

One *game* is then:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), v(k)) \; k = 0, \ldots, N & (6) \\ x(0) &= 0_{20\times 10} & (7) \\ x(N) &\in X_F \subset X = Z_2^{20\times 10} & (8) \end{aligned}$$

where $\min_k(x(k) \in X_F) = N$, i.e. the first instance we reach the *game over* states X_F which are defined as the states where no legal move can be made. Also define the *score*:

$$S = s(N)$$

We whish to derive a control law that maximizes S in various senses, e.g. the *average*, *peak* etc. Using the familiar block diagram notation we want something of the form depicted in figure 7. Suppose we want to maximize the average performance of our control law, then this may be stated as an optimal control problem as follows:

$$\begin{aligned} \max_u E[S] \quad &= \max_u \sum_{k=0}^{N} E[g(x(k), u(k), v(k))] & (9a) \\ \textbf{subject to} : x(k+1) \quad &= f(x(k), u(k), v(k)) & (9b) \\ x(0) \quad &= 0 & (9c) \\ \psi(x(N)) \quad &= 0 & (9d) \end{aligned}$$

Unfortunately (or interestingly) the lack of analytical expression for f and g precludes the use of traditional tools. However the Tetris game model is really a finite state model and hence dynamic programming could (in principle) be used. This process is an example of a really untractable finite state problem since assuming the control is of the form:

$$u(k) = h(x(k), v(k))$$

we would need a table of size:

$$|X \times F| = |X||F| = 2^{200} \times 11 \approx 1.76 \times 10^{61}$$

As an alternative we propose a controller of the form:

$$\begin{aligned} u(k) = h(x(k), v(k)) &= \arg \min_{u(k) \in U} W(f(x(k), u(k), v(k)) \\ &= \arg \min_{u(k) \in U} W(x(k+1)) \end{aligned} \tag{10}$$

where W is a suitable measure. In particular we propose a parametrized W of the form:

$$W(x;\theta) = \sum_{i=1}^{n} \theta_i h_i(x) \quad \theta \in R^n \tag{11}$$

For each choice of $\theta \in R^n$ we have a fixed dynamic system driven by essentially white noise. This makes $S = s(N)$ (the score) a stochastic variable with a distribution that is parametrized by θ, i.e. $S \in F(\theta)$ for some distribution F. This provides a clue to how we can optimize system performance given the structural choice of equation (11). The only degree of freedom left in the controller is the choice of the parameter θ and hence we want to calculate:

$$\theta = \arg \max_{\theta} E[S]$$

This maximation is performed in the next section using stochastic approximation, i.e. θ is calculated through:

$$\theta_{k+1} = \theta_k + \gamma_k \frac{dE[S]}{d\theta} \tag{12}$$

At this stage it is also fairly obvious that there are several other things we may choose to optimize, e.g. instead of maximizing average performance we could maximize peak performance through $\max_\theta E[S^n]$ (for some positive integer n).

4 Simulations

Using the problem formulation of last chapter we need to make a choice as to what $h_i(x)$ to include in equation (11). This choice is of course pretty much ad

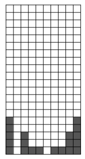
$x =$  $\Longrightarrow c(x) = [5, 0, 3, 1, 1, 0, 1, 1, 3, 5]$

Figure 8: The contour c associated with a state x.

hoc, so this leaves much room for improvement, but the following choice was found to yield good[2] results.

We hope to present our choice of the $h_i(x)$ without introducing coordinates (of course we need the coordinate version in the program, though). Let the contour be the upper bound in each column of the game grid (i.e. the state) see figure 8, Denote this contour by $c(x)$. It is my experience that some forms of this contour are better than others and my favourite contour is of the form $[2, 1, 0, 0, 0, 0, 0, 0, 1, 2]$. This means that one can allow the pieces to go higher towards the edges of the game grid, denote this contour by ic. Based on the contour c and the ideal contour ic all the proposed $h_i(x)$ may be defined. We have:

$$
\begin{aligned}
h_1(x) &= HeightMean(x) = E[c(x) - ic] && (13)\\
h_2(x) &= HeightVariance(x) = V[c(x) - ic] && (14)\\
h_3(x) &= Holes(x) && (15)\\
h_4(x) &= PeakHeight(x) = \max_i \ c_i(x) && (16)\\
h_5(x) &= CompleteRows(x) && (17)
\end{aligned}
$$

These have the following interpretations:

h_1 The average column height. This we want to keep small.

h_2 The column height variance. Generally this should be kept small in order to get a high score. One could argue that one should use standard deviation instead of variance and in fact that is what is used used in the current version of the controller.

h_3 The number of holes below the contour line. If we have many holes then we have stacked the pieces inefficiently, hence we want to keep this small.

h_4 The peak column height. To some extent this is included in h_2.

h_5 All complete rows will get deleted so this is very desirable. Also this is to some extent included in h_1.

[2] It outperformed all manual players in our group:-)

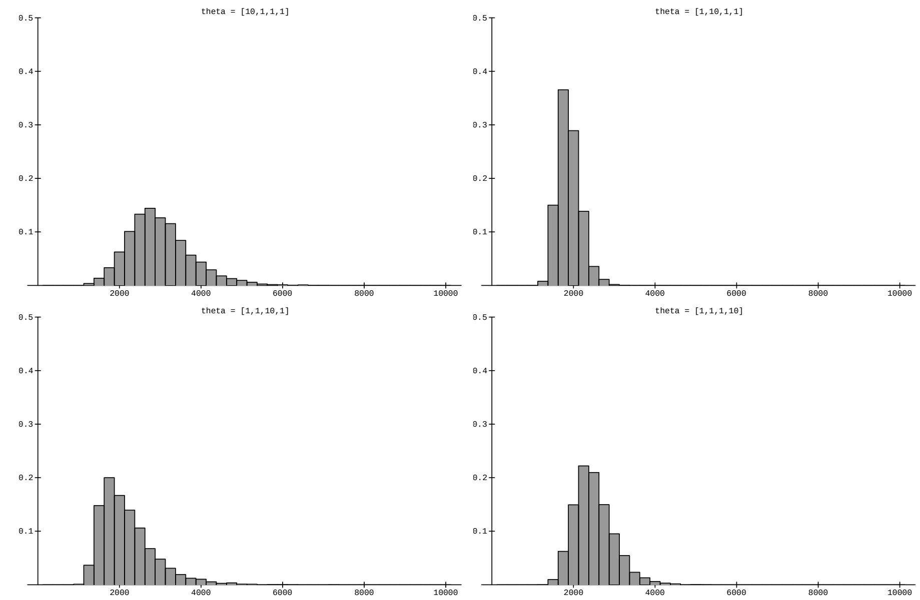


Figure 9: A Tetris score distribution with $\theta = [10, 1, 1, 1]$ (upper left), $[1, 10, 1, 1]$ (upper right), $[1, 1, 10, 1]$ (lower left) and $[1, 1, 1, 10]$ (lower right), where $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$ is the weight vector of equation (11). This is based on 6000 samples (i.e. games).

Since the control law minimizes the linear combination in equation (11), we can fix one of the weights, say $\theta_5 = -1$. Our job is now to make a good or optimal choice of $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$.

Before the optimization we could get some feeling for how we can shape the score distribution by changing the parameters θ. First we could take a look at the full distribution, see figure 9 for how the parameters affect the shape of the distribution. Also we could take a peak at how the mean ($E[S] = e(\theta)$) depends on the parameters θ, see figure 10 for this.

For the optimization a steepest ascent method is used with a variation of the Armijo-Goldstein step selection method, see [3, pg. 126]. The steepest ascent method was choosen because of the high cost of evaluating the mean score function $m(\theta) = E[S]$. One further complication was the fact that evaluating $m(\theta)$ as

$$m(\theta) \approx \frac{1}{N} \sum_{1 \leq i \leq N} s_i$$

where s_i is the score of game i, the value of $m(\theta)$ becomes very noisy and hence the value of $\nabla m(\theta)$ obtained by a difference approximation is even more noisy. To alleviate this problem we use fixed number of prechosen realizations of the

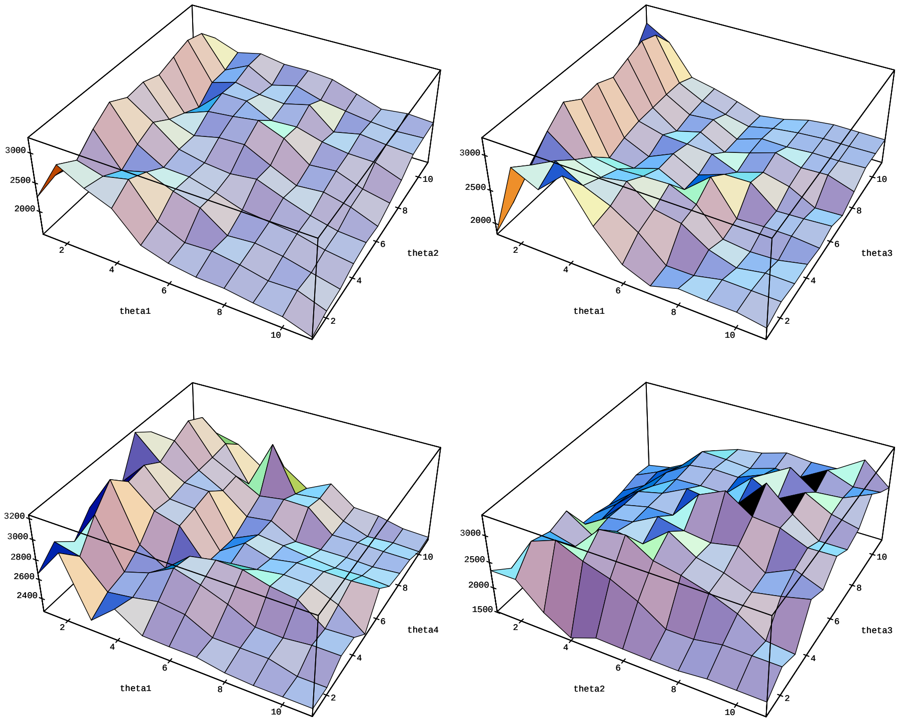


Figure 10: The mean Tetris score as we vary the θ parameters of equation (11). In each plot two parameter θ_i and θ_j are varied as the other are kept fixed ($= 1$). In particular we vary θ_1, θ_2 (upper left), θ_1, θ_3 (upper right), θ_1, θ_4 (lower left) and θ_2, θ_3 (lower right).

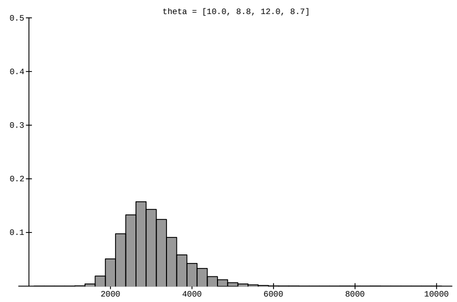


Figure 11: Resulting score distribution after some optimization of control law has been performed. In this case we have $\theta = [10.0, 8.8, 12.0, 8.7]$.

game process (i.e. setting seed for the random number generator) and optimize on these.

The practical optimization was performed by running the same C-code that defines the game, but without any of the windowing commands. This was then run under Mathematica and from Mathematicas point of view the Tetris simulator is simply a random number generator whose distribution is paramatrised by θ.

As can be expected from the plots we may not necesserily end up in a global maximum by running the optimization, but the basic results can be ascertained from figure 11. This has the following complementary data:

$$\frac{1}{6000}\sum_{1\leq i\leq 6000} s_i = 2904.6, \qquad \max_{1\leq i\leq 6000} s_i = 8425 \tag{18}$$

and $\theta = [10.0, 8.8, 12.0, 8.7]$.

This should be contrasted with the best hand tuned controller which ended up having an average of ≈ 2000 and a max of ≈ 5000. Typical manual scores are in the range $1000 - 3000$ and peak scores of ≈ 6000.[3]

5 Discussion

We have taken the computer game Tetris, viewed is a dynamical system and then outfitted it with a parametrized control law. By handtuning or optimizing this parametrization we obtained impressive performance gains.

Even though this is just a computer game we feel that many of the real world problems are present in this case:

[3] As has been previously stated this is a nonstandard version of Tetris. In the standard version with figures having four pieces the scores are substantially higher (almost an order of magnitude).

Modelbuilding. The control law needs an internal model of the game process. Also it is of great use conceptually to have this model.

Complexity. If the game is attacked in a naive manner we end up with something that is truely untractable. Even with the simplified parametric approach we end up with a large simulation burden to optimize our choice of control law (not unlike the situation with queuing networks).

Furthermore this application shows that control concepts can fruitfully be applied in a wide variety of contexts.

Lastly, but definitely not least, it is *great fun*.

A About the Program

As has been previously stated this controller has been implemented as an extension to a public domain version of Tetris. All the code associated with this application can be accessed over the internet through anonymous ftp from `joakim.isy.liu.se` or `130.236.24.1`.

Below follows a sample ftp session:

```
kajsa[joakim/roger]41 ftp joakim.isy.liu.se
Connected to joakim.
220 joakim FTP server (SunOS 4.1) ready.
Name (joakim.isy.liu.se:roger): anonymous       <--Use anonymous as login name
331 Guest login ok, send ident as password.
Password:                                       <--Use user name as password
230 Guest login ok, access restrictions apply.
ftp> cd pub/reports
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (130.236.24.104,1895) (0 bytes).
LiTH-ISY-I-1259.ps.Z
LiTH-ISY-I-1260.ps.Z
README
226 ASCII Transfer complete.
52 bytes received in 0.66 seconds (0.076 Kbytes/s)
ftp> get README
200 PORT command successful.
150 ASCII data connection for README (130.236.24.104,1896) (258 bytes).
226 ASCII Transfer complete.
local: README remote: README
269 bytes received in 0.034 seconds (7.7 Kbytes/s)
ftp> binary
200 Type set to I.
ftp> get LiTH-ISY-I-1259.ps.Z
200 PORT command successful.
150 Binary data connection for LiTH-ISY-I-1259.ps.Z (130.236.24.104,1897)
(74689 bytes).
226 Binary Transfer complete.
```

```
local: LiTH-ISY-I-1259.ps.Z remote: LiTH-ISY-I-1259.ps.Z
74689 bytes received in 0.46 seconds (1.6e+02 Kbytes/s)
ftp> quit
221 Goodbye.
```

The code is placed in `Tetris.tar.Z` and further instructions cab be found in the file `README.Tetris`.

References

[1] A Benveniste and P Le Guernic. Hybrid dynamical systems theory and th signal language. *IEEE Transactions on Automatic Control*, 35(5):535–546, May 1990.

[2] A, E Bryson and Y, C Ho. *Applied Optimal Control*. Hemisphere Publishing, 1975.

[3] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[4] Y C Ho. Peturbation analysis explained. *IEEE Transactions of Automatic Control*, 33(8):761–763, August 1988.

[5] C A R Hoare. *Communicating sequential processes*. Prentice Hall, 1985.

[6] J L Peterson. *Petri Net Theory and Modeling of Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[7] P Ramadge. Supervisory control of discrete event systems: A survey and some new results. In P Varaiya and A B Kurzhanski, editors, *Discrete Event Systems: Models and Applications*, pages 69–80. Springer Verlag, Lecture Notes in Control and Information Sciences, 1988.

[8] P J Ramadge. Some tractable supervisory control problems for discrete event systems modeled by buchi automata. *IEEE Transactions of Automatic Control*, 34(1):10–19, January 1989.

[9] P J Ramadge and W M Wonham. Modular feedback logic for discrete event systems. *SIAM J. on Control and Optimization*, 25(5):1202–1218, September 1987.

[10] P J Ramadge and W M Wonham. Supervisory control of a class of discrete event systems. *SIAM J. on Control and Optimization*, 25(1):206–230, January 1987.

[11] P J Ramadge and W M Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.

[12] W Reisig. *Petri Nets - An Introduction*. Springer-Verlag, 1982.