DYNAMICS OF DISCRETE EVENT SYSTEMS

Ravindra Rao

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712-1188

TR-91-21 June 1991

Dynamics of Discrete Event Systems[†]

Ravindra Rao
Department of Computer Sciences
University of Texas At Austin
Austin, TX 78712
e-mail: rnr@cs.utexas.edu

Abstract: We provide a mathematical framework for the analysis of discrete dynamical systems. These are systems with a discrete state spaces. We find that Hilbert spaces are the appropriate spaces in which to embed the dynamics. In this formulation, it is possible to define and write equations of motion for the system. Although Hilbert spaces are generally associated with quantum systems in physics, they are employed here in a purely classical manner; both dynamical evolution and results of measurements are just as in classical mechanics. Two examples analysed are Euclid's algorithm and the Turing machine.

1. Introduction

This work provides a framework for describing the dynamics of discrete dynamical systems. Such systems, sometimes called discrete event systems, are so called because some aspect of the evolution of the system is discrete. It may be the case that states accessible to the system form a discrete set or it may happen that transitions of the system take place only at certain instants of time. In systems such as the digital synchronous computer, both the time and the states of the system change by discrete amounts. Such systems are the subject of this work.

It is important to understand that 'dynamical system' here stands for any system that can be described by the ideas and techniques of physical systems familiar to physicists. Systems in physics have a special property—they obey the "laws of nature." No such requirement is imposed on the systems considered here. The examples we describe in this paper do not obey the laws of nature. However they do have states and equations of motion. Therefore they qualify as dynamical systems. In other words, our interest in discrete event systems is quite independent of any connection with natural systems or with physical law.

The relationship between physics and computations has received a great deal of attention. We have deliberately avoided a discussion of this relationship as they involve ideas that are not necessary for the present discussion. We refer the reader to the article by Bennett [1] for a history of this subject.

[†] Supported by the IBM corporation through grant number 61653 and by the State of Texas through TATP Project 003658-237.

Let us begin by first classifying all known dynamical systems according as the set of all possible states of the system or the times at which the state may change is continuous or discrete, see figure 1.

Systems that evolve continuously in time, namely those that may be described by differential equations have easily the longest history and have been very well studied. These are the "classical systems" of figure 1. Quantum theory and quantum dynamical systems should leave no one in doubt that nature provides an abundance of systems with discrete state spaces. It is certainly not the case that discrete event systems are solely "man-made." As far as we know (observationally), the quantity we call time appears as a continuous parameter in all natural systems. These systems therefore occupy part of the box called "Quantum Theory." The digital computer is a fine example of a man-made system with a discrete set of states and in which transitions occur at discrete instants of time. These are placed in the lower right portion of figure 1. There are other man-made systems, such as asynchronous computers or sytems in the field of operations research where either time is a continuous parameter or the state space is continuous. These systems, including the digital computer, fall in the shaded portion of the figure and are the ones that hold our interest.

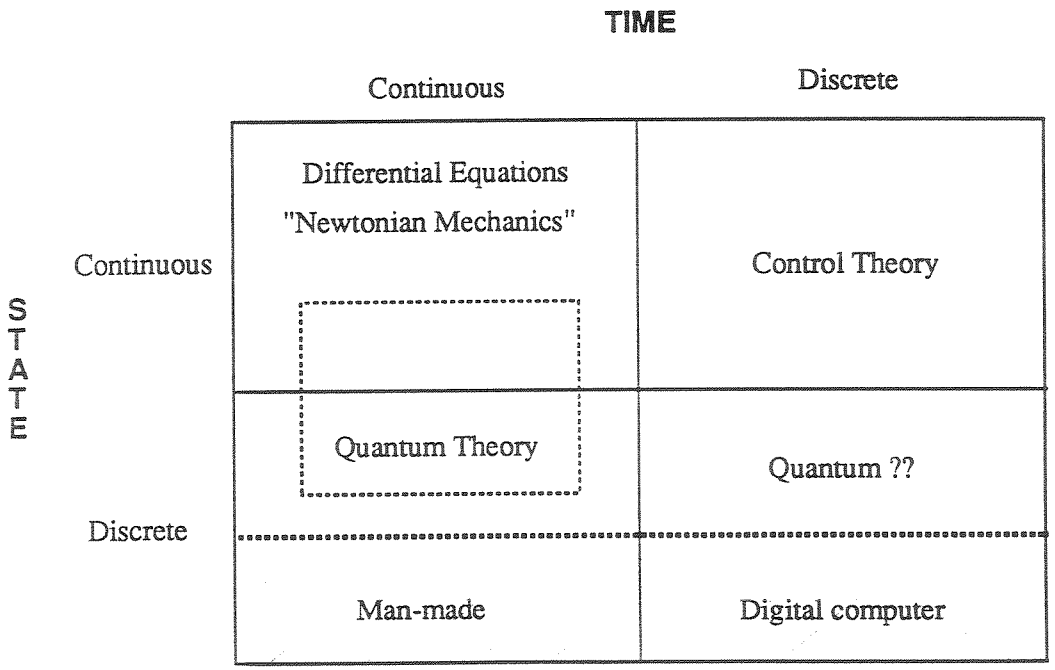

Figure 1

Our results indicate strongly that techniques developed for treating quantum systems can also be used for dealing with the dynamics of man-made systems. We are encouraged by the observation that our classification places man-made systems in the same class as quantum systems. As justification for this point of view we present two examples of discrete

systems that may be interpreted as a dynamical system in the sense just described. Both examples are of interest to the computer scientist. The first example analysed in section 2 is Euclid's algorithm for finding the greatest common divisor of two intergers. Section 3 formulates the Turing Machine as an example of a discrete dynamical system. Both examples point to suitably chosen Hilbert spaces as being the appropriate vector spaces in which to embed the dynamics. The state space of a quantum system such as a hydrogen atom is a Hilbert space[1]. Such spaces have been in use by physicists for the better part of this century and there is an abundance of literature on this subject. We do not provide a comprehensive introduction to Hilbert spaces but will intorduce only those aspects necessary for our work. For an introduction see reference [2]. Appendices A and B provide more details of the particular Hilbert spaces used in section 2 and 3 respectively.

2. Euclid's Algorithm—An Example

A very simple discrete dynamical system is the algorithm of Euclid for finding the greatest common denominator *gcd*, of two integers x and y. Dijkstra [3], presents a discussion of this algorithm in a form that is particluarly suitable for our purpose. We begin with a review of this algorithm, see figure 2.

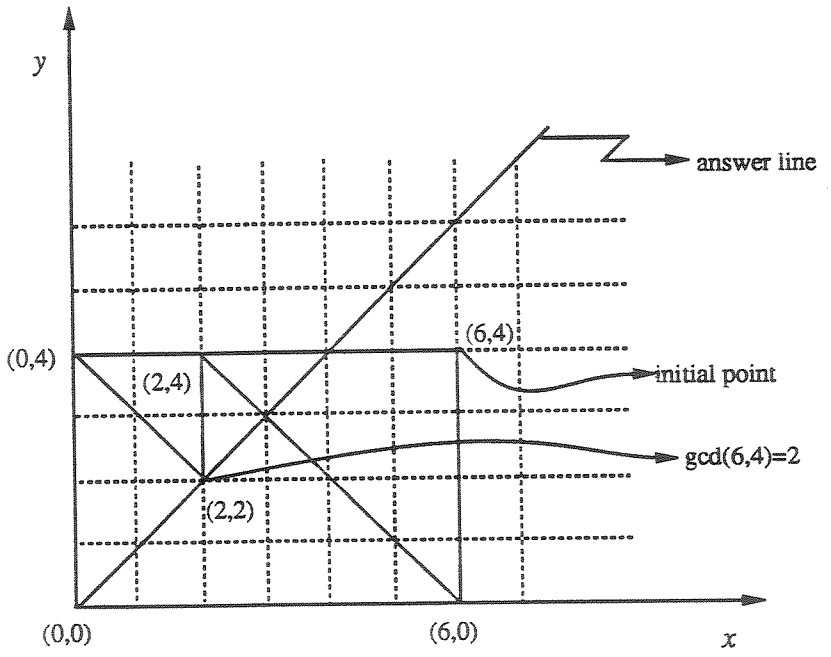

Figure 2

The line $y = x$ is the "anwer line" because $gcd(x, x) = x$. So for points not on the answer line, one proceeds by drawing the smallest right isosceles triangle with the right angle at

[1] Though of a different kind than the ones used here.

(x, y) and one acute angle on one of the axes. The new point for the next iteration is chosen by moving to the point that coincides with the other acute angle of the triangle. This is repeated until the new point falls on the answer line. We will now assume that this algorithm is correct and present the dynamical system to which it corresponds.

The two main ingredients of any dynamical system are the dynamical variables and a rule that specifies the evolution of these variables. Let us begin by describing the dynamical variables. We proceed heuristically. We will first desribe a state space and define the dynamical variables later. Let us suppose the existence of a linear vector space F with the following properties.

1. Basis vectors denoted $|i\rangle$, $i = 0, 1, 2, \ldots$
2. An inner product defined by

$$\langle i|j\rangle \ = \delta_{ij} \tag{2.1}$$

where $\langle i|$ is the complex conjugate transpose of $|i\rangle$ and δ_{ij} is the Kronecker delta defined by

$$\delta_{ij} \ = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases}$$

So the basis vectors $|i\rangle$ are an orthonormal set.

3. Two independent operators $\hat{a}$ and $\hat{a}^\dagger$ called annihilation and creation operators that operate on the basis vectors as follows;

$$\hat{a}|i\rangle \ = \begin{cases} 0, & \text{if } i = 0; \\ \sqrt{i}\ |i-1\rangle, & \text{otherwise.} \end{cases} \tag{2.2}$$

$$\hat{a}^\dagger|i\rangle \ = \sqrt{i+1}\ |i+1\rangle. \tag{2.3}$$

4. A commutation relation

$$[\hat{a}, \hat{a}^\dagger] \ = \ \hat{a}\hat{a}^\dagger - \hat{a}^\dagger\hat{a} \ = \ 1. \tag{2.4}$$

This is an operator equation where the right side is the identity operator in the space F. A vector space with these properties is called a Fock space after Vladimir Fock. Fock spaces are direct sums of Hilbert spaces. Appendix A contains a representation of F used in this discussion.

The vectors $|i\rangle$ are eignevectors of the operator $\hat{a}^\dagger\hat{a}$ since firstly

$$\hat{a}^\dagger\hat{a}|0\rangle = \ 0|0\rangle$$

and

$$\hat{a}^\dagger \hat{a}|i\rangle = i|i\rangle, i = 1, 2, \ldots.$$

The integers $0, 1, \ldots$ are in one to one correspondence with the basis vectors of F. Consider the direct product $F \otimes F$ of F with F. The basis vectors of $F \otimes F$ are defined by

$$|x, y\rangle \stackrel{def}{=} |x\rangle|y\rangle. \tag{2.5}$$

where $|x\rangle$ and $|y\rangle$ are vectors in F. The set of states written in the form $|x, y\rangle$ may each be considered as representing a point in upper right quadrant of the plane, that is those in figure 2. There are now two sets of operators, one for each of the spaces in the direct product. Let us call these $\hat{a}_1$ and $\hat{a}_1^\dagger$ and $\hat{a}_2$ and $\hat{a}_2^\dagger$. These creation and annihilation operators act on the basis vectors of the first and second F spaces in the product $F \otimes F$. The inner product in $F \otimes F$ is induced by that in F,

$$\langle x', y'|x, y\rangle = \delta_{xx'}\delta_{yy'}. \tag{2.6}$$

So that to determine the x-coordinate of a state denoted by $|x, y\rangle$, we must compute the inner product of the state $\hat{x}|x, y\rangle$ with the state $|x, y\rangle$, where $\hat{x} \stackrel{def}{=} \hat{a}_1^\dagger \hat{a}_1$.

$$\begin{aligned} \langle x, y|\hat{x}|x, y\rangle &= \hat{a}_1^\dagger \hat{a}_1^\dagger |x, y\rangle \\ &= \langle x|\langle y|\left(\hat{a}_1^\dagger \hat{a}_1|x\rangle\right)|y\rangle \\ &= x\langle x, y|x, y\rangle \\ &= x. \end{aligned} \tag{2.7}$$

Similarly for the y coordinate with $\hat{y} \stackrel{def}{=} \hat{a}_2^\dagger \hat{a}_2$

$$\begin{aligned} \langle x, y|\hat{y}|x, y\rangle &= \langle x, y|\hat{a}_2^\dagger \hat{a}_2|x, y\rangle \\ &= \langle x, y|\left(|x\rangle \hat{a}_2^\dagger \hat{a}_2|y\rangle\right) \\ &= \langle x, y|y|x, y\rangle \\ &= y. \end{aligned} \tag{2.8}$$

With these definitions, the action of the $\hat{a}$'s and $\hat{a}^\dagger$'s are identified with translations in the plane of figure 2.

with the states $|x, y\rangle$ and $|x+y, y\rangle$ as the initial and final states respectively, it can be shown that

$$\langle x+y, y|[\hat{T}, \hat{E}]|x, y\rangle \ > \ 0.$$

Our monotonicity criteria will therefore not permit the relevant coefficient $A_{(k'+l')l'k'l'}$ in this case, to be non-zero. In this way we can arrive at equation (2.9).

It would seem that the monotonicity criteria described is the equivalent of the action principle in classical mechanics. In classical mechanics, the action principle is imposed on the system "from outside." It would seem that our present derivation corresponds to the construction of Hamiltonian dynamics rather than Lagrangian dynamics; $\hat{E}$ might well be called the Hamiltonian of Euclid's system.

One might ask if it is possible to construct Euclid's system in some other way, say for instance in a Fock space that is not the direct product. It turns out that this is indeed possible. Once again we may think of the state of the system as being described by the coordinates (x, y) but we now encode these two numbers as a single number z using the well known encryption $z = 1/2((x+y)^2 + 3x + y)$. The new Fock space has basis vectors given by $|z\rangle$. The $\hat{E}$ operator is also different but its function not nearly as obvious as the $\hat{E}$ described here in detail. There are yet other ways of formulating this algorithm. We might like to think of erecting a finite dimensional Fock space at each point of the positive quadrant of the $x - y$ plane. Each such point can be in one of two states, only one of which represents the occupation of the corresponding point by a fictious particle. In this description the state of the system is defined to be the product of the state at each point. These kinds of Fock spaces are used in physics for the description of fermions. The kind introduced in the detailed presentation are used for the description of bosons. We will not present the detailed analysis of these alternatives but it is important to realize that there are many different ways of describing the same algorithm.

We have shown explicitly that Euclid's algorithm can be viewed as a dynamical system in which successive iterations of the body of the algorithm correspond to a unique evolution of the dynamical system in Fock space. As no model of computation was used in this demonstration, we assert that any system for which the state space is discrete, may be viewed as dynamical system in a similar manner.

Euclid's algorithm terminates in a finite number of iterations. The dynamical system will therefore reach its quiescent state after a finite number of operations of $\hat{E}$. However, we cannot associate a period of time with this evolution as there is no scale for time in the system. Finally, Euclid's algorithm is a deterministic algorithm for computing the gcd of two integers. So is the dynamical system described above. It is deterministic in the sense of classical mechanics where "boundary conditions" or "initial state" determine the evolution of

These two dimensional spaces are quite adequate for constructing systems with any desired number of independent states. We now present a general technique for constructing such spaces; it is used repeatedly in our description. Suppose we require a system that can be in any one of S independent states. Consider the vector space spanned by basis vectors of the form

$$|b_0, b_1, \ldots, b_{S-1}\rangle \stackrel{def}{=} \prod_{i=0}^{S-1} |b_i\rangle, \tag{3.5}$$

where it is understood that the terms in the product are written in order of increasing value of index. Hereafter, the state $|a, b\rangle$ will mean the product $|a\rangle|b\rangle$, where each member of the product is a basis vector of a distinct F. The set of annihilation and creation operators is now $\hat{b}_i$ and $\hat{b}_i^\dagger$ with anti-commutation relations

$$\begin{aligned} \{\hat{b}_i, \hat{b}_j^\dagger\} &= \delta_{ij}, \\ \{\hat{b}_i, \hat{b}_j\} &= 0, \\ \{\hat{b}_i^\dagger, \hat{b}_j^\dagger\} &= 0, \end{aligned} \tag{3.6}$$

with $i = 0, 1, \ldots, S-1$. As each of the b_i can have the value 0 or 1, the vector space in question has dimensionality 2^S. The inner product in this space is taken over from that in F.

$$\langle b_0, b_1, \ldots, b_{S-1} | b_0', b_1', \ldots, b_{S-1}' \rangle \stackrel{def}{=} \prod_{i=0}^{S-1} \delta_{b_i b_i'} \tag{3.7}$$

We can restrict ourselves to a particular S dimensional subspace by imposing the constraint that at most one of the b_i can be 1, the others must all have value 0. A simple way of imposing this constraint is by use of the number operator $\hat{n}$,

$$\hat{n} \stackrel{def}{=} \sum_{i=0}^{S-1} \hat{b}_i^\dagger \hat{b}_i. \tag{3.8}$$

From this definition of $\hat{n}$, it follows that

$$\hat{n}|b_0, b_1, \ldots, b_{S-1}\rangle = \left(\sum_{i=0}^{S-1} b_i\right)|b_0, b_1, \ldots, b_{S-1}\rangle.$$

So the basis vectors are eigenvectors of $\hat{n}$ with eigenvalues equal to the number of b_i that have the value 1. The condition we must impose is then

$$\hat{n}|b_0, b_1, \ldots, b_{S-1}\rangle = |b_0, b_1, \ldots, b_{S-1}\rangle.$$

Hereafter we will always assume that such a restriction has been imposed and in order to indicate this we will adopt the notation where the state $|\underline{i}\rangle$ stands for that basis vector in this

special subspace in which the ith member of the product is in state $|1\rangle$ the others in state $|0\rangle$. The range of the parameter that is underscored is to be understood within the context of the discussion in which it appears[3]. This subspace is henceforth the single-bit subspace. Two important operators that will play a role are the state operator $\hat{s}$ and $\hat{t}_{ij}$. These are defined as follows.

$$\hat{s} = \sum_{i=0}^{S-1} i\hat{b}_i^\dagger \hat{b}_i. \tag{3.9}$$

$$\hat{t}_{ij} = \hat{b}_i^\dagger \hat{b}_j. \tag{3.10}$$

$\hat{s}$ can be used to determine which b_i has value 1,

$$\begin{aligned} \langle \underline{i}|\hat{s}|\underline{i}\rangle &= i\langle \underline{i}|\underline{i}\rangle \\ &= i \end{aligned}$$

and

$$\hat{t}_{ij}|\underline{k}\rangle = \delta_{jk}|\underline{i}\rangle.$$

In this equation, if k is equal to j, $\hat{t}_{ij}$ changes the state from $|\underline{k}\rangle$ to $|\underline{i}\rangle$ otherwise the state is nullified. So $\hat{t}_{ij}$ is an operator that conducts transitions from one state to another[4]. t has the important property that if the initial state is in the single-bit subspace, so also is the $\hat{t}_{ij}$-transform of the initial state. Finally, a single-bit subspace of dimension D will be written as F_s^D. Let us now proceed to discuss the Turing machine proper.

Recall that our Turing machine has one external tape, one tape head and some finite set of internal states. The conventional meaning of these parts of a Turing machine is as follows. The external tape is taken to be infinite in extent and is divided into an infinite number of cells. Each cell can contain at most one symbol that is a member of some fixed set of symbols. The tape head can read at most one cell at a time. This cell is called the current cell. After reading the current cell the tape head performs some action that depends on the internal state. This action is some combination of moving along, reading from and writing to the external tape. Finally, it is assumed that initially the tape has only a finite number of marked cells, there is always only one current cell, only one internal state that is the current state and that any action of the machine modifies only a finite number of cells of the external tape.

[3] Note that the vectors $|0\rangle$ and $|1\rangle$ are quite different from $|\underline{0}\rangle$ $|\underline{1}\rangle$ respectively.

[4] Observe that $\hat{n} = \sum_{i=0}^{S-1} \hat{t}_{ii}$

The description of our Turing machine begins with the external tape. The object that will represent the external tape is a state ψ_E that is a product of the state of each of the individual cells. Let N_E be the number of independent states available to each cell. The state $C(i)$ of the ith cell is defined to be

$$C(i) \stackrel{def}{=} |h(i), \underline{s}(i)\rangle, \tag{3.11}$$

where $i = -\infty, \ldots, -1, 0, 1, \ldots, \infty$. $C(i)$ is a vector in $F \otimes F_s^{N_E}$. The value of $h(i)$ can be 0 or 1. When $h(i) = 1$ the ith cell is the current cell. The range of $s(i)$ is $0, 1, \ldots, N_E - 1$, corresponding to the N_E independent states available to a cell. What is conventionally regarded as data is here recorded as a particular state of the cell. The state of the entire tape ψ_E, is a product of the state of the individual cells.

$$\psi_E = \prod_{i=-\infty}^{\infty} C(i). \tag{3.12}$$

The annihilation and creation operators corresponding to the $h(i)$ are $\hat{h}(i)$ and $\hat{h}(i)^\dagger$. Each $\hat{s}(i)$ is constructed out the annihilation and creation operators $\hat{s}_u(i)$ and $\hat{s}_u(i)^\dagger$, $u = 0, 1, \ldots, N_E$, which act on the uth parameter of the state of the ith cell.

$$\hat{s}(i) = \sum_{u=0}^{N_E-1} u\, \hat{s}_u(i)^\dagger \hat{s}_u(i), \tag{3.13}$$

The anti-commutation relations amongst them are

$$\begin{aligned} \{\hat{s}_u(i), \hat{s}_v(j)^\dagger\} &= \delta_{ij}\delta_{uv}, \\ \{\hat{h}(i), \hat{h}(j)^\dagger\} &= \delta ij. \end{aligned} \tag{3.14}$$

All other anti-commutation relations, for example those between $\hat{h}$'s and $\hat{s}$'s vanish. From the definitions (3.4), (3.5)and (3.13), it follows that

$$\hat{s}(i)|h(i), \underline{s}(i)\rangle = s(i)|h(i), \underline{s}(i)\rangle$$

and

$$\hat{h}(i)^\dagger \hat{h}(i)|h(i), \underline{s}(i)\rangle = h(i)|h(i), \underline{s}(i)\rangle$$

The operator that changes the state of the ith cell is

$$\hat{t}_{uv}(i) = \hat{s}_u(i)^\dagger \hat{s}_v(i), \tag{3.15}$$

that which changes the cell that is marked as current is

the system uniquely. Moreover, as in classical mechanics, one is free to make measurements on the system at any time without affecting the system. The surprising feature of Euclid's system is that despite it being a "classical system," the language used for its description is that of quantum theory.

3. Turing Machines

We will show that it is possible to specify and describe the working of a Turing machine as a dynamical system in Hilbert space. Our discussion is relevant to a machine that has a single external tape, one finite control that we will henceforth be referred to as the internal tape and a device conventionally called the tape head, that is capable of communicating between the two media and reading from and writing to the external tape. The vector spaces we will need are different from the ones in section 2 let us begin by introducing these.

The vector space we will need is again a Fock space F so [2] it has all the properties of a vector space described in section 2. The crucial difference being the properties of the annihilation and creation operators $\hat{a}$ and $\hat{a}^\dagger$. These obey the relations

$$\begin{aligned} \{\hat{a}, \hat{a}^\dagger\} &\stackrel{def}{=} \hat{a}\hat{a}^\dagger + \hat{a}^\dagger\hat{a} = 1, \\ \{\hat{a}, \hat{a}\} &= 0, \\ \{\hat{a}^\dagger, \hat{a}^\dagger\} &= 0. \end{aligned} \tag{3.1}$$

Notice the plus sign in the definition where previously in section 2, there was a minus sign. These are called anti-commutation relations.These relations imply that the vector space of these operators is spanned by two basis vectors that we will denote $|0\rangle$ and $|1\rangle$. The annihilation and creation operators act on these vectors according to

$$\hat{a}|0\rangle = 0, \quad \hat{a}|1\rangle = |0\rangle, \quad \hat{a}^\dagger|0\rangle = |1\rangle, \quad \hat{a}^\dagger|1\rangle = 0. \tag{3.2}$$

The bases $|0\rangle$ and $|1\rangle$ are orthonormal in the following inner product

$$\langle 0|0\rangle = 1, \quad \langle 0|1\rangle = 0, \quad \langle 1|0\rangle = 0, \quad \langle 1|1\rangle = 1. \tag{3.3}$$

The two basis vectors are eigenvectors of the number operator $\hat{a}^\dagger\hat{a}$, that is

$$\hat{a}^\dagger\hat{a}|0\rangle = 0\,|0\rangle, \quad \hat{a}^\dagger\hat{a}|1\rangle = 1\,|1\rangle, \tag{3.4}$$

A particular representation of this vector space is presented in appendix B.

[2] We use the same name for this vector space as in section two. No confusion should arise as we will not refer to the F of section 2 here.

$$\hat{h}(i,j) \;=\; \hat{h}(i)^{\dagger}\hat{h}(j). \tag{3.16}$$

This completes our discussion of the external tape.

In the usual description of the Turing machine, the internal state and that of the current cell define a transformation of the internal state and the external tape. Following this, let us suppose that our Turing machine is always to be found in one of N_I internal states. This can be fully described by vectors in $F_s^{N_I}$.

$$\psi_I \;=\; |\underline{\sigma}\rangle \tag{3.16}$$

where $\sigma = 0, 1, \ldots, N_I - 1$ and where we have adopted the convention quantities referring to the internals of the Turing machine are taken from the Greek alphabet. There is now a further increase in the number of annihilation and creation operators, namely $\hat{\sigma}_\mu$ and $\hat{\sigma}_\mu^\dagger$ which combine to form the internal state operator $\hat{\sigma}$,

$$\hat{\sigma} \;=\; \sum_{\mu=0}^{N_I-1} \mu\, \hat{\sigma}_\mu^\dagger \hat{\sigma}_\mu. \tag{3.17}$$

The operator that changes the internal state is

$$\hat{\tau}_{\mu\nu} \;=\; \hat{\sigma}_\mu^\dagger \hat{\sigma}_\nu. \tag{3.18}$$

In order to discuss the tape head, it is necessary to first define the state Ψ of the entire Turing machine.

$$\Psi \;\stackrel{def}{=}\; \psi_I\, \psi_E. \tag{3.19}$$

The tape head is a device that first finds and determines the state of the current cell. Second, it measures the internal state and then performs some transformation on Ψ.

All these definitions can now be brought together by discussing how to specify a computation. For this purpose we introduce a set of Turing operators $\hat{T}_{\mu u}(i)$ that can act on the state of the Turing machine to transform it to another state. The meaning of the indices is as follows. The range of the Greek index, in this case μ, is from 0 to $N_I - 1$—the numbering of the cells of the internal tape. The range of the Latin index, here u, is $0, 1, \ldots, N_E - 1$—the set numbering the states of a cell of the external tape. The general rule is that indices from the Latin alphabet refer to the state of a cell of the external tape, Greek indices have the same meaning with reference to the internal tape. The parameter i that appears as an argument is the number assigned to a cell. We treat this parameter as an argument because firstly, the number assigned to cell is arbitrary to within translations of the entire tape. Secondly

treating it as an index will imply that there are an infinite number of $\hat{T}$'s but as we will see, there are in fact only a finite number. These operators are constructed out of the entire set of annihilation and creation operators introduced. For the present we will assume that there is no restriction on their form except that the action of any of them on a legal state must produce a legal state. A legal state is one in which a finite number of cells are in a state other than the state $|\underline{0}\rangle$, exactly one cell is marked as current and the machine is in one internal state. Clearly, the $\hat{T}_{\mu u}$ will have this property if they are composed only out of the transition operators, $\hat{t}_{uv}(i)$, $\hat{\tau}_{\mu\nu}$ and $\hat{h}_{ij}$.

A computation progresses by the application of a sequence of the $\hat{T}$ operators. To understand this, let us suppose that the current cell with number $i = i_0$ of the external tape is in state $u = u_0$ and that the current internal state is one with $\mu = \mu_0$. In this case the rule we would like to adopt is that the state Ψ is to be transformed by the application of the operator $T_{\mu_0 u_0}(i_0)$. In other words, Ψ is replaced by $T_{\mu_0 u_0}(i_0)\Psi$. It is in fact possible to construct an operator that at any given stage of the computation, selects the correct $\hat{T}$ operator to apply. The operator that does this is

$$\hat{H} = \sum_{\mu=0}^{N_I-1} \sum_{u=0}^{N_E-1} \sum_{i=-\infty}^{\infty} \hat{T}_{\mu u}(i)\hat{\tau}_{\mu\mu}\hat{t}_{uu}(i)\hat{h}(i,i) \tag{3.20}$$

To see how this operator works, let Ψ be a legal state of the computation and consider the action of $\hat{E}$ on Ψ. What we have to show is that although $\hat{E}$ is an infinite sum of operators, at any stage in the computation only one of these terms actually acts on the current state. Recall first that a legal state is one in which exactly one cell, call it i_0, of the external tape is marked as current, that is, only one of the $h(i)$ is 1. From equation (3.16), $\hat{h}(i,i) = \hat{h}(i)^\dagger \hat{h}(i)$. Each term of the infinite sum will be acted on by one such operator. However only that term will survive which has $i = i_0$. So we have

$$\hat{h}(i,i)\Psi = \begin{cases} \Psi, & \text{if } i = i_0; \\ 0, & \text{otherwise.} \end{cases}$$

Therefore

$$\hat{E}\,\Psi = \sum_{\mu=0}^{N_I-1} \sum_{u=0}^{N_E-1} \hat{T}_{\mu u}(i_0)\hat{\tau}_{\mu\mu}\hat{t}_{uu}(i_0)\,\Psi.$$

Similarly, the summations over u and μ collapse to just one term and we have finally,

$$\hat{E}\,\Psi = \hat{T}_{\mu_0 u_0}(i_0)\Psi,$$

as claimed. We can think of the "collapse" as taking place in any order we wish. A fixed Ψ will always be acted on by the same $\hat{T}$ regardless of this order. The reason is the vanishing of

the anti-commutation relations between the primitive annihilation and creation operators of different parts of the Turing machine. We now see that the state of the computation $\Psi(k+1)$ after $k+1$ steps is related to $\Psi(k)$ according to

$$\Psi(k+1) \;=\; \hat{E}\Psi(k). \tag{3.21}$$

As we remarked earlier, the $\hat{T}_{\mu u}(i)$ are not completely arbitrary and further more, they can be generated from a finite set. We address the latter point first. The transformations that define a conventional Turing machine cannot depend on the particular number given to a cell. After all, given some tape with a legal marking, one is free to choose any cell as cell number 0. Equivalently, one can uniformly translate all cells by any desired amount without changing the input or the computation. This means that the Turing operators must have the same *form* for each cell. That is, they are covariant with respect to translations along the external tape. So one first defines the finite set $\hat{T}_{\mu u}(i_0)$ with reference to some definite but arbitrary cell i_0. There are $N_E N_I$ operators in this set. In addition to operating on the i_0th cell, each will in general transfrom some set of neighboring cells. The set of neighbors is part of the prescription of the program and is defined relative to some fixed cell. So for each μ and u, $\hat{T}_{\mu u}(i_0)$ will involve operators of neigboring cells relative to this fiducial cell. We imagine that a copy of the space $F \otimes F_s^{N_E}$ is erected over each cell together with the entire set of $\hat{T}_{\mu u}(i_0)$ but with i_0 replaced by i—the number assigned to the particular cell. This is not quite sufficient because the space of the Turing operators is not just the space of the cells but includes the space of internal states $F_s^{N_I}$. So we must also assume that there is a single copy of this space and this copy is available simultaneously to each cell. This of course does not prove to be a problem as only one cell is current at any time.

The general structure of the Turing operators is constrained by the requirement that legal states be transformed into legal states. The two requirements are

$$[\hat{n}_I, \hat{E}] \;=\; 0 \tag{3.22}$$

where $\hat{n}_I = \sum_{\mu=0}^{N_I-1} \hat{\tau}_{\mu\mu}$ and

$$[\hat{n}_E, \hat{E}] \;=\; 0 \tag{3.23}$$

where $\hat{n}_E = \sum_{i=0}^{N_E-1} \hat{h}(i,i)$. That is, $\hat{E}$ must commute[5] with the number operators $\hat{n}_I$ and $\hat{n}_E$. These are the equations of motion for the two operators (dynamical variables) $\hat{n}_I$ and $\hat{n}_E$. These operators measure respectively, the number of internal states that and the number

[5] Where $[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}$. See section 2. The use of square brackets [] rather than curly braces {} is explained in appendix B.

of cells that are current. If the number of internal states and current cells are both fixed at the beginning of the computation, then equations (3.22)and (3.23)guarantee they will remain unchanged. For Turing machines both of these numbers are choosen to be equal to 1. Substituting for $\hat{E}$ from equation (3.20)in equation (3.22),

$$\sum_{\mu=0}^{N_I-1}\sum_{u=0}^{N_E-1}\sum_{i=-\infty}^{\infty}[\hat{n}_I\,,\ \hat{T}_{\mu u}(i)\hat{\tau}_{\mu\mu}\hat{t}_{uu}(i)\hat{h}(i,i)] = 0.$$

By use of the indentity $[\hat{A},\hat{B}\hat{C}] = [\hat{A},\hat{B}]\hat{C} + \hat{C}[\hat{A},\hat{C}]$, and noting that $\hat{n}_I$ commutes with the operators $\hat{t}_{ii}$, $\hat{\tau}_{\mu\mu}$ and $\hat{h}(i,i)$ this equation reduces to

$$\sum_{\mu=0}^{N_I-1}\sum_{u=0}^{N_E-1}\sum_{i=-\infty}^{\infty}[\hat{n}_I\,,\ \hat{T}_{\mu u}(i)]\hat{\tau}_{\mu\mu}\hat{t}_{uu}(i)\hat{h}(i,i) = 0.$$

Since the $\hat{T}_{\mu u}(i)$ are all independent, we have

$$[\hat{n}_I, \hat{T}_{\mu u}(i)] = 0,$$

for all μ, u and i. Similarly for $\hat{n}_E$ we have

$$[\hat{n}_E, \hat{T}_{\mu u}(i)] = 0.$$

The necessary and sufficient condition for these equations to be satisfied is that $\hat{T}_{\mu u}(i)$ should be composed out of the transition operators $\hat{t}_{ij}$, $\hat{\tau}_{\mu\nu}$ and $\hat{h}(i,j)$.

Specification of a particular programs will in general assign further properties to the Turing operators. We will not study particular programs here as for the present we are content with the characterization of Turing machines as dynamical systems. Program termination is also something about which we have not been explicit. The usual Turing machine has a special symbol in the set of recognized ones that is defined to signal a "halt." The Turing machine stops when the current cell is in this state. Dynamical systems do not "halt," they are continually evolving or are in stasis at a fixed point. In keeping with this spirit we note that $\hat{E}$ will in many cases have eigenvalues. That is a state Ψ which have the property

$$\hat{E}\,\Psi \;=\; \Psi$$

Clearly if a particular Turing machine reaches such a state, there can be no further change in its state. These states are fixed points in the space of states. If we wish to identify these eigenvalues with the end of the program, then we must constuct $\hat{E}$ so that it has exactly one eigenvalue for each distinct input. In addition, it should be possible to reach one of these eigenstates for every legal initial state. We must also define $\hat{E}$ so that there are no cycles in the space of states. Where by cycle we mean a sequence of states that are transformed into

each other by the action of $\hat{E}$. In the usual analysis of the Turing machine these and other questions are part of the halting problem. Such considerations specifically regarding Turing machines will be taken up in future work.

4. Equations of Motion

A complete specification of a dynamical system requires a definition of the dynamical variables, the space of states and equations of motion. In this section, we will define and discuss dynamical variables and equations of motion. Euclid's algorithm is once again used as an example but the conclusions are generally valid.

Now a dynamical variable is an object that when measured, gives information about the state of the system. The value of this measurement changes with the state of the system. In Euclid's system, the states are the vectors $|x, y\rangle$, the evolution operator is the operator $\hat{E}$. Let $\psi(k)$ be the state of Euclid's system after k applications of $\hat{E}$. If we apply $\hat{E}$ again, we have

$$\psi(k+1) \;=\; \hat{E}\psi(k). \tag{4.1}$$

Equations of motion usually involve derivatives of objects that arise in the system under consideration. As this is not possible here, we must accept equation as the nearest thing to an equation of motion. We will call this an equation of evolution because equation (4.1)resembles an integral form of the equation of motion rather than an equation of motion.

The state of the system is determined by measuring the values of x and y of the current state. As we indicated, this is accomplished by computing the expectation values[6] $\langle x, y|\hat{x}|x, y\rangle$ and $\langle x, y|\hat{y}|x, y\rangle$ respectively. Measuring the system is the same thing as computing the expectation value of some operator. Dynamical variables are now operators; expectation values of operators have physical meaning. How many dynamical variables do we have? As Euclid's algorithm is formulated on a "plane," there are two degrees of freedom one for each of the two (geometrically) orthogonal directions. The corresponding dynamical variables are the operators $\hat{x}$ and $\hat{y}$.

In general, we will always have an equation of evolution as in (4.1). The dynamical variables are those operators all of whose expectation values are required *simultaneously* to completely determine the state of the system. This means that the dynamical variables are those operators that commute (or anti-commute) with each other. In the case of Euclid's system there are two namely $\hat{x}$ and $\hat{y}$. The Turing machine has many more; in fact as the external tape is allowed to be infinitely long, there are an infinite number of dynamical variables. But as we always restrict ourselves to the single-bit subspaces the relevant ones are the

[6] Terminology that is borrowed from physics.

various "position" operators introduced in section 3. These are the operators that measure the state of each cell, $\sum_{u=0}^{N_E-1} u\hat{s}_u(i)^\dagger \hat{s}_u(i)^\dagger$, the position of the current cell, $\sum_{i=-\infty}^{\infty} i\hat{h}(i)^\dagger \hat{h}(i)$ and the internal state, $\sum_{\mu=0}^{N_I-1} \mu\hat{\sigma}_\mu^\dagger \hat{\sigma}_\mu$.

5. Conclusions

Our aim was to demonstrate that we could treat discrete systems as dynamical systems just as are systems described by continuous variables. Section 1 demonstrated our intent with a simple example—Euclid's algorithm. A surprising aspect of that example was the existence of a monotonicity criteria that almost completely specified the evolution operator. We argued there that this was similar to the action principle in classical mechanics. The action principle of classical mechanics is universal in the sense that the same principle is invoked for every dynamical system. We have not been able to find such a universal principle for discrete systems. We consider this to be an important open problem.

Now equation (4.1)is an equation that was not derived but posutulated. The evolution operators were defined with this equation in mind. It is entirely possible that there are other ways of defining such equations; even some that might resemble the diffential equations of classical mechanics. But in the form given there is a very important difference between systems described by such equations and those of classical and quantum mechanics. Linear classical systems and quantum mechanical systems all obey the principle of superposition. The objects that may be superposed are solutions to the dynamical equations. No such superposition principle is applicable here. Indeed the process of "collapse" of the evolution operator to a single term works precisely because our systems are always to be found in a state that is not a superposition. It would be interesting to find a man-made system that requires or perhaps evolves through a superposition of states.

We hope that our description of Turing machines will remove any doubt that systems with a discrete space of states can indeed be treated as dynamical systems. Moreover, we have explicitly presented the mathematics that is suitable. Attention should also be paid to the importance of the single-bit subspaces and the corresponding transition operators. These together provide the basis for constructing universal mimics. That is, systems that can mimic any other discrete system. We should stress that what we have presented here is not merely an alternative description of problems that are well understood by use of other techniques. The hope is that these methods are powerful enough to describe any discrete dynamical system—from those in operations research to those in the computing sciences including other kinds of automata and perhaps physics. Indeed by taking seriously the view that algorithms are dynamical systems [4, 5], it may be possible to develop quantum algorithms for problems that fall in the complexity class NP.

5. Acknowledgements

I would like to thank J. C. Browne for much encouragement. I have benefited greatly from his generosity. I would also like to thank Sanjay Deshpande for many discussions, reading some early versions of this manuscript and for his continuing interest.

Appendix A

This appendix displays one explicit representation of the Fock spaces used in section 1. In this representation, the basis vectors are denoted $|i\rangle$ with $i = 0, 1, 2, \ldots$, where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix}, |2\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{pmatrix}, \ldots \tag{A.1}$$

By use of the equations

$$\hat{a}|i\rangle = \begin{cases} 0, & \text{if } i = 0; \\ \sqrt{i}\ |i-1\rangle, & \text{otherwise.} \end{cases}$$

and

$$\hat{a}^*|i\rangle = \sqrt{i+1}\ |i+1\rangle.$$

The annihilation and creation operators $\hat{a}$ and $\hat{a}^*$ are realised as infinite singular matrices.

$$\hat{a} = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{2} & 0 & \cdots \\ 0 & 0 & 0 & \sqrt{3} & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{A.2}$$

$$\hat{a} = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & \cdots \\ 0 & \sqrt{2} & 0 & 0 & \cdots \\ 0 & 0 & \sqrt{3} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{A.3}$$

Here the (ij)th element of $\hat{a}$ and $\hat{a}^*$ are evaluated by computing the matrix element $\langle i|\hat{a}|j\rangle$ and $\langle i|\hat{a}^*|j\rangle$ respectively. One can easily check that indeed

$$[\hat{a}, \hat{a}^*] = \hat{a}\hat{a}^* - \hat{a}^*\hat{a} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

and that the number operator $\hat{n}$ is

$$\hat{n} \;=\; \hat{a}^*\hat{a} \;=\; \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 2 & 0 & \cdots \\ 0 & 0 & 0 & 3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Appendix B

Here we will discuss a useful representation of the two-dimensional Hilbert spaces F used in section 3. The orthonormal basis vectors are denoted $|0\rangle$ and $|1\rangle$ where

$$|0\rangle \;=\; \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (B.1)$$

and

$$|1\rangle \;=\; \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad (B.2)$$

Annihilation and creation operators $\hat{a}$ and $\hat{a}^*$ are

$$\hat{a} \;=\; \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \qquad (B.3)$$

$$\hat{a} \;=\; \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad (B.4)$$

It is then straightforward to check orthonormality of the basis vectors and anti-commutation relations.

$$\begin{aligned} \hat{a}|0\rangle &= 0, \\ \hat{a}|1\rangle &= |0\rangle, \\ \hat{a}^*|0\rangle &= |1\rangle, \\ \hat{a}^*|1\rangle &= 0. \end{aligned}$$

$$\begin{aligned} \{\hat{a}, \hat{a}^*\} &= \hat{a}\hat{a}^* + \hat{a}^*\hat{a} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \{\hat{a}, \hat{a}\} &= 0 \\ \{\hat{a}^*, \hat{a}^*\} &= 0 \end{aligned}$$

The spaces that were actually employed in section 3 were products of F. Using the notation and ideas of section 3, a system with S states is requires the single-bit subspace F_s^S. A state ψ in F_s^S is written as

$$\psi = |b_0, b1, \ldots, b_{S-1}\rangle \stackrel{def}{=} \prod_{i=0}^{S-1} |b_i\rangle$$

It is now possible to show that each step of Euclid's algorithm can be realised as the action of an operator $\hat{E}$ acting on a state $|x, y\rangle$ This operator has the form

$$\hat{E} = \sum_{k,l,k',l'=1}^{\infty} |k,l\rangle\{\delta_{kk'}\delta_{ll'}\delta_{kl} + \Theta(k',l')\delta_{k\ k'-l'}\delta_{ll'} + \Theta(l',k')\delta_{kk'}\delta_{l\ l'-k'}\}\langle k',l'| \qquad (2.9)$$

where $\Theta(k,l)$ is the step function defined by

$$\Theta(k,l) = \begin{cases} 1, & \text{if } k > l; \\ 0, & \text{if } k \leq l. \end{cases}$$

It is easy to check that the state $|x, y\rangle$ is transformed by $\hat{E}$ according to

$$\hat{E}|x,y\rangle = \begin{cases} |x-y,y\rangle, & \text{if } x > y; \\ |x,y-x\rangle, & \text{if } y > x; \\ |x,y\rangle, & \text{if } x = y. \end{cases}$$

$\hat{E}$ so defined performs the required transformations and can therefore claim to represent Euclid's algorithm. Observe that the only eigenvalues of $\hat{E}$ are states of the form $|x, x\rangle$ which of course correspond to points on the "answer line" in figure 2. Once we reach the state $|x, x\rangle$, additional applications of $\hat{E}$ do not change the state of the system.

The expression for $\hat{E}$ given in equation (2.9)follows directly by "inpsection" of figure 2. There is however a systematic way of deducing equation (2.9). Using the properties of *gcd* and imposing a monotonicity criteria on the evolution of the system, it is possible to actually derive equation (2.9)as the (almost) unique form that $\hat{E}$. This is also demonstrated in [3]. The way to do this is to first write $\hat{E}$ as a general projection operator,

$$\hat{E} = \sum_{k,l,k',l'=1}^{\infty} |k,l\rangle\ A_{klk'l'}\ \langle k',l'|,$$

where $A_{klk'l'}$ are a set of coefficients to be determined. We assume that each of these coefficients can be 0 or 1. Next, one uses the properties of *gcd* such as for instance $gcd(x, y) = gcd(y, x) = gcd(x + y, y)$ etc. to restrict the number of non-zero coefficients $A_{klk'l'}$. Although this will certainly reduce the number of non-zero coefficients, there are still an infinite number of undetermined ones. To reach the form presented in equation (2.9)we must assume that a given state is to be transformed to another state only if this transformation results in a strict decrease in the expectation value of the operator

$$\hat{T} = \hat{a}_1^\dagger \hat{a}_1 + \hat{a}_2^\dagger \hat{a}_2$$

betwen the initial and final states. One imposes this condition by examining the expectation value of the operator $[\hat{T}, \hat{E}] \stackrel{def}{=} \hat{T}\hat{E} - \hat{E}\hat{T}$ between an initial and an acceptable final state. By acceptable we mean a final state that does not violate the properties of *gcd*. For instance

An important set of operators in this space are the transition operators of section 3 where they are defined in terms of the primitive annihilation and creation operators $\hat{b}_i$ $\hat{b}_i^\dagger$, $i = 0, 1, \ldots, S-1$.

$$\hat{t}_{ij} \;=\; \hat{b}_i^\dagger \hat{b}_j .$$

It can be shown that these operators have the property

$$[\hat{t}_{ij}, \hat{t}_{kl}] \;=\; \delta_{jk}\hat{t}_{il} - \delta_{li}\hat{t}_{kj} \qquad (B.5)$$

where $i, j, k, l = 0, 1, \ldots, S-1$. They also satisfy the Jacobi identity

$$[\hat{t}_{ij}, [\hat{t}_{kl}, \hat{t}_{mn}]] + [\hat{t}_{mn}, [\hat{t}_{ij}, \hat{t}_{kl}]] + [\hat{t}_{kl}, [\hat{t}_{mn}, \hat{t}_{ij}]] \;=\; 0, \qquad (B.6)$$

Any set of objects that have these two properties form a Lie algebra. The transition operators $\hat{t}_{ij}$ therefore form Lie algebra. The basis vectors of F_s^S together with the defining equations for the $\hat{t}_{ij}$ is an explicit representation of this algebra. We will defer a detailed treatment of this aspect of our work.

Now although the transition operators are composed out of annihilation and creation operators which obey anti-commutation relations, we used the commutator bracket [,] rather than anticommutator bracket {}. Let $\hat{A}$ and $\hat{B}$ be two operators. The usual convention is to use commutation relations when either of $\hat{A}$ or $\hat{B}$ are bosonic and to use anti-commutation relations only when both are fermionic. Transition operators such as $\hat{t}_{ij}$ are bilinear in the fermionic operators $\hat{b}_i$ and $\hat{b}_i^\dagger$ and are regarded as bosonic. The general rules are that the product of two bosonic or two fermionic operators is bosonic. The product of a bosonic and fermionic operator is fermionic.

Bibiliography

1. C. H. Bennett, Notes On The History of Reversible Computation, IBM J. Res. Dev., Vol. 32, 16–23, (1988).

2. J. M. Jauch, Foundations of Quantum Mechanics, Addison-Wesley Publishing Company Inc., (1968).

3. E. W. Dijkstra, *A Discipline Of Programming,* Prentice Hall, Inc., (1976).

4. R. P. Feynman, *Quantum Mechanical Computers,* Foundations of Physics, Vol. 16, No.6 (1986).

5. R. Rao, *Quantum Mechanics And The Problem of The Traveling Salesman,* In preparation, University of Texas at Austin.