Biological and Computational Tools for Systems Biology: Application to Fas Signaling Pathways in T cells

by

Rayka Yokoo

B.S. Computer Science
Massachusetts Institute of Technology, 2003

B.S. Biology
Massachusetts Institute of Technology, 2003

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2004

Signature of Author:____
Department of Electrical Engineering and Computer Science
January 16, 2004

Certified by:____
Luk Van Parijs
fessor of Biology
Thesis Supervisor

Accepted by:____
Arthur Smith
Professor of Electrical Engineering and Computer Science
Chairman, Committee for Graduate Theses

Biological and Computational Tools for Systems Biology: Application to Fas Signaling Pathways in T cells

by

Rayka Yokoo

Submitted to the Department of Electrical Engineering and Computer Science on January 16, 2004 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

With the development of new experimental technologies, biologists have begun to take a more global view into cell function, approaching its study in a more systematic manner than previously possible. This thesis develops three new tools to perform systems biology studies of cell death in T cells: A modeling program, JDesigner; high throughput T cell apoptosis assays; and an RNAi sequence prediction program. These tools are then applied to a biological and mathematical analysis of Fas signaling pathways in T cells.

Thesis Supervisor: Luk Van Parijs
Title: Assistant Professor of Biology

Table of Contents

Chapter 1. Introduction

Before the advent of high throughput biological research technologies, such as multi-color flow cytometry, researchers could only measure a few cellular components at a time. Such experimental limitations led to the study of detailed mechanisms of interaction between a small number of molecules and generally led to the assumption that the modules of molecules could be studied in isolation. However, scientists now have the tools to measure, simultaneously, the global expression and activity of many different variables in cells. While this new global view allows biologists to analyze the cell more realistically as a system, this approach also presents challenges because the cell is a particularly complicated and non-intuitive system with multiple feedback loops and nonlinearities, the interactions of which are poorly understood. In order to enable a productive systems biology approach to understand cell function, new research tools and platforms must be developed.

The current efforts in systems biology generally fall into two categories, detailed and statistical.

The detailed approach involves modeling and gathering large data sets for a small number of components at high resolution. Similar to the older, pre-systems biology approach, the detailed approach studies variations and interactions between individual molecules to deduce cause and effect relationships. Unlike the traditional approach, the systems biology approach can study effects that have multiple causes. For example, the cleavage of a protein may result from two or three different pathways. Using the old approach, biologists might study the five proteins in each pathway independently. Using the systems approach, biologists can study all ten to fifteen proteins simultaneously and determine how the pathways can cooperate to cleave the final protein. As illustrated by this example, the detailed approach requires some prior knowledge to select the proteins to be studied.

In contrast, the statistical approach involves modeling and gathering large data sets for a large number of components at low resolution. The statistical approach generally searches for correlation rather than cause and effect. For example, one may measure thirty proteins and discover that the combined activity of two proteins correlates well with cell death. The choice of proteins to study requires less prior knowledge but also provides less insight into the mechanism.

This project aims to identify and create research tools and platforms to enable systems biology analysis of cell death in T cells. For the detailed approach to systems biology this project identifies a modeling program, JDesigner. JDesigner allows biologists to represent signaling pathways mathematically as a series of chemical reactions in the cell. Using this representation the biologist can integrate data from multiple pathways, test hypotheses in silico, and observe the effect of non-intuitive behaviors such as feedback loops. To enable statistical approaches to study cell function, this project develops the apoptosis assay, a high throughput assay for the outcome of T cell stimulation, programmed cell death. This assay is especially appropriate for the biological problems addressed in this project dealing with the function of the Fas pathway, which is recognized as an important apoptosis pathway in T cells. In addition, this project develops a program to predict sequences for RNAi, a method to systematically repress gene expression in cells.

This thesis will introduce the Fas signaling pathways in chapter 2, then each tool, JDesigner, apoptosis assays, and RNAi in chapters 3, 4, and 5 respectively. The application of each tool to the Fas pathway will be discussed in chapters 6, 7 and 8.

Chapter 2. Fas

Introduction

The immune system functions to protect the body from foreign pathogens such as viruses and bacteria. In most vertebrates including humans, the immune system can be divided into two branches, the innate and adaptive immune system. The innate immune system includes defenses such as the skin and phagocytic cells and is not specific for a particular pathogen but has broad reactivity to classes of frequently encountered pathogens. The adaptive immune system responds to pathogens with a high degree of specificity and consists of B and T cells. A T cell recognizes pathogens through an interaction between the T cell receptor (TCR) and the major histocompatibility complex (MHC) which presents peptides, including foreign peptides when the body is infected.

Generally a MHC-self peptide complex results in tolerance while a MHC-foreign peptide complex activates T cells and initiates an immune response including T cell proliferation. Occasionally a TCR will recognize a MHC-self peptide complex as foreign. The T cell presenting this TCR will mount an immune response to a self peptide, potentially leading to autoimmunity.

T cells will upregulate Fas and Fas ligand (FasL), cell surface proteins involved in programmed cell death (apoptosis), when the TCR is repeatedly stimulated. The Fas-mediated apoptosis of T cells after frequent stimulation is known as activation induced cell death (AICD) and is believed to remove self-reactive T cells as well as extra post-infection T cells from the T cell population. The importance of Fas/FasL mediated cell death is highlighted by patients with mutations in the Fas receptor. These patients suffer from autoimmune lymphoproliferative syndrome (ALPS) which is characterized by enlarged lymph nodes and spleen, where B and T cells generally reside, as well as autoimmunity (Jackson et al. 1999). Resistance to Fas mediated apoptosis can also be an important step towards developing lymphomas (Igney and Krammer 2002).

Details of Fas Signaling Pathways

The general architecture of the Fas signaling pathways is fairly well established (Goldsby et al. 2003). As illustrated in figure 2.1, the signal begins when FasL binds Fas, a trimer (Siegel et al. 2000). Each Fas molecule can then recruit a Fas associated death domain containing protein (FADD). Each FADD has a protein-protein interaction domain termed the death effector domain (DED) which can recruit proteins with the same domain via homeotypic interactions. FLICE or caspase-8, a member of the family of *c*ysteine *asp*artate prote*ases* (caspases), contains two DEDs as does c-FLIP (FLICE-like Inhibiting Protein). The high local concentration of caspase-8 when recruited to FADD is believed to lead to cleavage and activation (Peter and Krammer 2003). Activated caspase-8 can cleave proteins in two different pathways starting with caspase-3 and Bid (Luo et al. 1998). In contrast, c-FLIP contains a mutated active domain which abolishes activity (Peter and Krammer 2003). This suggests c-FLIP may function as a competitive inhibitor of caspase-8 by binding to FADD and preventing downstream signaling.

There are two signaling pathways downstream of activated caspase-8 resulting in the activation of effector caspases, such as caspase-3. Effector caspases cleave a wide array of proteins which are required to trigger the cellular changes associated with apoptosis. In the type I pathway, caspase-3 is cleaved directly by caspase-8. In the type II pathway, caspase-8 initiates a cascade of reactions that result in cytochrome c and Smac release from the mitochondria and caspase-9 cleavage of caspase-3. In detail, caspase-8 cleaves Bid (Peter and Krammer 2003). Truncated Bid (tBid) then oligomerizes Bak or Bax to induce the mitochondria to release cytochrome c and Smac (Wei et al. 2000, Sun et al. 2002). Bcl-2 can inhibit this release (Scaffidi et al. 1998, Sun et al. 2002). After its release from the mitochondria, Smac functions to release cleaved caspase-3 and caspase-9 from inhibition by the X-linked inhibitor-of-apoptosis protein (XIAP) (Sun et al. 2002). Cytochrome c interacts with Apaf-1 to recruit and activate caspase-9. Activated caspase-9 then cleaves other caspases including caspase-3 (Slee et al. 1999). Some evidence supports the presence of a positive feedback loop with caspase-3 cleaving caspase-8 though this is not yet widely accepted (Slee et al. 1999, Engles 2000).

Questions about the Function of Fas in T cells

This thesis applies systems biology to address two questions about the function of Fas in T cells. First, what changes in cell state dictate whether a cell exhibits type I versus type II cell death characteristics. Second, at what level do Fas signaling pathways interact with TCR signaling pathways, and what are the biological consequences of this interaction.

Type I versus Type II cells

Scaffidi et al. (1998, 1999) have proposed that cells can be assigned to one of two types, type I or type II, based on the effect of Bcl-2 overexpression on Fas-mediated apoptosis. Bcl-2 overexpression in type II cells blocks caspase-8 and caspase-3 activation, as well as apoptosis. Scaffidi et al. (1998) hypothesize that the type II cells might activate only low levels of caspase-8 initially and that the mitochondrial pathway serves to amplify these weak signals. Bcl-2 overexpression would prevent amplification of a weak caspase-8 signals through the mitochondrial pathway, preventing the activation of caspase-3 and further activation of caspase-8 via feedback. However, Bcl-2 overexpression in type I cells does not inhibit caspase-8 or caspase-3 activation indicating that these cells use a different pathway to signal Fas-mediated apoptosis.

To establish the type I and type II cell paradigm, Scaffidi et al. investigated a large number of cell types, including T cell lines that exhibit either type I (H9 cells) or type II (Jurkat cells) killing by Fas (Scaffidi et al. 1998). Dr. Fei Hua, a postdoctoral researcher in our laboratory, has found that Jurkat cells were more sensitive to Fas-mediated killing than H9 cells, yet had fewer Fas receptors on the surface (Figure 2.2). She also found that Jurkat cells had higher levels of caspase-8, but lower levels of Bcl-2 and caspase-3 compared to H9 cells (Figure 2.3). The lower number of Fas receptors and the higher levels of caspase-8 initially seem to counteract each other making the Scaffidi hypothesis of low initial caspase-8 activation difficult to test. The consequences of systematically changing Fas and caspase 8 levels can be explored experimentally, but would require the creation of new reagents. Furthermore, Sun et al. (2002) propose that the ratio of XIAP to active caspase-3 and Smac may be most important in determining type I versus type II behavior. A quantitative model would help to clarify the importance of Fas receptor number versus caspase-8 concentration, and XIAP versus caspase-3 and Smac concentration, on the activation of caspase-3. The search for a modeling program and the creation of an ordinary differential equation model of the Fas signaling pathways is discussed in chapters 3 and 6 respectively.

Interaction between Fas and TCR

Before TCR activation, there is very little Fas and FasL present on the cell surface. Thus TCR signaling is required to upregulate Fas and FasL and for Fas signaling to occur (Siegel et al. 2000). However, several papers suggest that more complex interactions may also occur between the signaling pathways for TCR and Fas. Holstrom et al. (2000) show that Jurkat cells in which the TCR is activated 30min before stimulation of the Fas receptor have about one half the levels apoptosis of cells treated with Fas only. Furthermore, they show that this suppression of Fas-mediated apoptosis correlates with Erk activity, a molecule downstream of the T cell receptor pathway. Allan et al. later demonstrated that Erk specifically phosphorylates and inhibits caspase-9. Kennedy et al. (1999) show that FasL treatment of T cells, together with TCR stimulation, increases proliferation compared with TCR stimulation alone. The apoptosis assay, developed and applied in chapters 4 and 7 respectively, aims to establish conditions which maximize the interaction of the Fas and TCR pathways for further investigation.

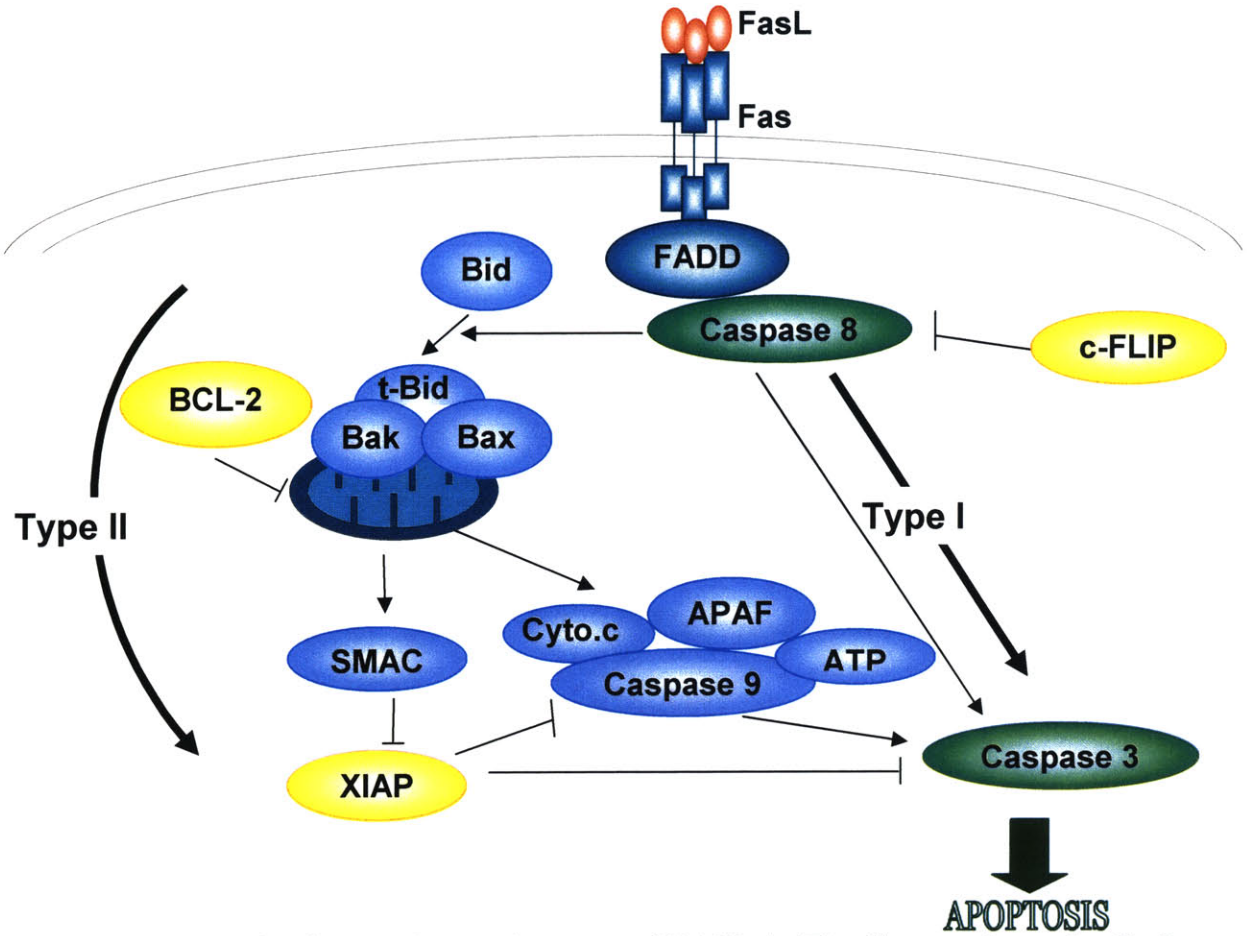


Figure 2.1 Schematic of Fas Pathways (courtesy of Fei Hua). The diagram shows that FasL binds Fas as a trimer which then recruits FADD and caspase 8. Activated caspase 8 can cleave caspase 3 by two pathways. In the type I pathway, caspase 8 cleaves caspase 3 directly. In the mitochondrial pathway, caspase 8 cleaves Bid to form t-Bid (truncated Bid). t-Bid dimerizes Bax or Bak causing mitochondrial release of Smac and cytochrome c. Smac releases the inhibition of caspase 9 by XIAP and allows the formation of a complex known as the apoptosome. The apoptosome which consists of APAF, cytochrome c, caspase 9, and ATP cleaves caspase 3. Cleavage of caspase 3 then induces apoptosis. These pathways are regulated by several molecules. Activation of caspase 8 can be inhibited by c-FLIP. The mitochondrial pathway can be inhibited by Bcl-2.

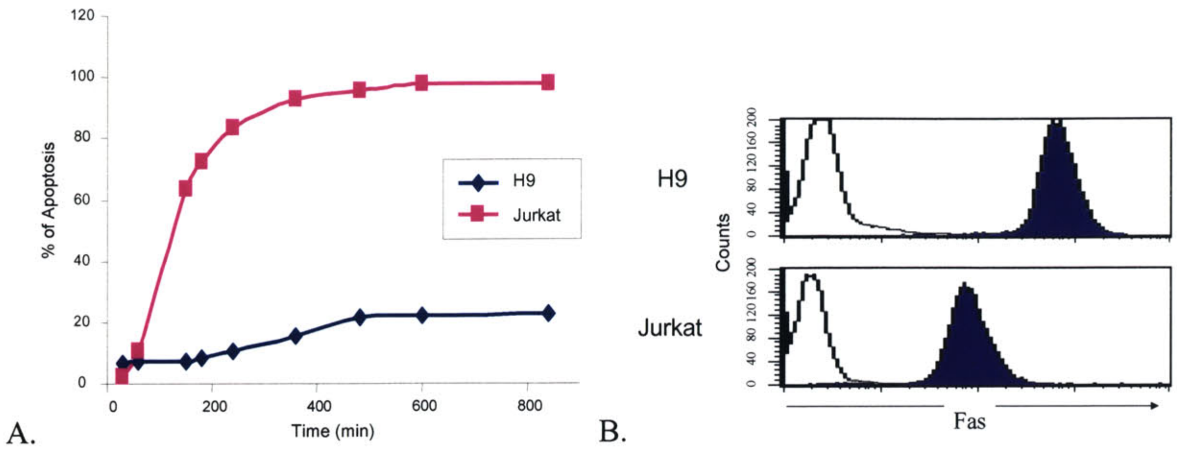


Figure 2.2 (courtesy of Dr. Fei Hua) A) Jurkat cells are more sensitive to 100ng/ml FasL-induced apoptosis than H9 cells B) Cells were labeled with fluorescent-conjugated anti-Fas and the number of Fas on the cell membrane compared by the fluorescence intensity. Unfilled peaks are negative control.

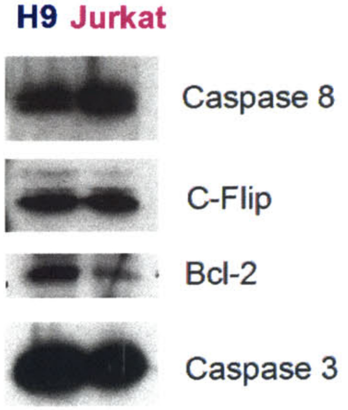


Figure 2.3 (Courtesy of Dr. Fei Hua) Jurkat cells have higher levels of caspase-8 but lower levels of Bcl-2 and caspase-3.

Chapter 3. Programs for Detailed Modeling of Cell Signaling Pathways

Introduction

The detailed approach to systems biology attempts to study the precise mechanisms of cellular behavior with more molecules and with greater resolution than has been previously attempted. This approach presents two challenges. First, with more molecules, the modeling begins to more closely reproduce the complexity of the cell. The cell has naturally evolved as a secure system with many seemingly redundant pathways as well as feedback control mechanisms. The behavior that results from these mechanisms is often difficult to predict. Second, at the most mechanistic and granular level, cellular behavior is a series of chemical reactions. Intuiting the behavior of molecules at the level of chemical reactions is also difficult. In order to assist in these challenges, a program that allows for modeling of cellular behavior at the level of chemical reactions is important.

Creating a program to model cellular behavior is difficult due to conflicting considerations. On one hand, computational modeling of cell pathways is still very new. As a result, standards are not yet established and the modeler will most likely want new functionality as the field develops. On the other hand, it is time consuming to recreate models when the current program proves inadequate. For these reasons, before creating a model, it is important to attempt to assess whether current programs are flexible enough to adjust to future developments yet complete enough that models will not have to be rewritten.

The specific purpose of this section of the project was to identify or create a program for constructing computational models of cell signaling pathways. An important requirement was that the program be intuitive. In addition, the created models needed to be self explanatory similar to data in a publication. After considering the requirements for a program and reviewing the state of the art, I determined that JDesigner satisfied all the requirements and was the most complete of the currently available programs.

Requirements for the Program

After discussion with members in the Van Parijs lab, I determined that the four most desirable properties of a cell modeling program are that it possesses a biologically intuitive interface, the ability to use different reaction representations, space to annotate reactions, and the capacity to optimize parameters using experimental data.

The following paragraphs discuss each requirement in greater detail.

Intuitive Interface

To facilitate the creation of computational models, an intuitive interface is important. Generally reactions within biological systems are represented using bubble diagrams similar to that shown in Figure 2.1. The figure provides some mechanistic insight into the system, showing for example, that FasL binds Fas as a trimer then recruits FADD and caspase 8. The figure also shows that caspase 8 can be inhibited by c-Flip. However, the diagram omits complex details such as that a high concentration of procaspase 8 bound to FADD induces autocatalytic cleavage activating caspase 8 and that c-Flip may inhibit caspase 8 activation through competition (Igney 2002). An interface similar to the bubble diagram representation, with some but not all detail, would provide the most intuitive representation of a biological system.

Different Reaction Representations

Past work has focused on the use of ordinary differential equations or stochastic simulations to mathematically represent chemical reactions (Bhalla 2002, Mendes 1998).

Ordinary differential equations reflect biological processes when reactants are abundant. Using differential equations, the reaction "A binds with B to form C reversibly,"

$$A+B \Leftrightarrow C$$

becomes the series of equations

$$\frac{d[C]}{dt} = k_1[A][B] - k_2[C]$$
$$\frac{d[A]}{dt} = k_2[C] - k_1[A][B]$$
$$\frac{d[B]}{dt} = k_2[C] - k_1[A][B]$$

The first equation, $\frac{d[C]}{dt} = k_1[A][B] - k_2[C]$, represents how the change in the concentration of C with respect to time, $\frac{d[C]}{dt}$, depends on the rate at which A binds to B and forms C, $k_1[A][B]$, minus the rate at which C degrades into A and B, $k_2[C]$. Similarly, the second and third equations represent the change in the concentration of A and B respectively. k_1 and k_2 are known as rate constants. A complete model using differential equations requires knowledge of all the chemical reactions, the rate constants, and the initial concentrations of each reactant.

Stochastic simulations use similar knowledge. However, stochastic simulations assume that the initial concentrations of reactants are so low that reactions are probabilistic rather than guaranteed. Instead of $k_1[A][B]$-$k_2[C]$, stochastic simulations use Fc, the flux of C. The probability that a reaction occurs to produce C is the flux of C over total flux, Fc/Ft. At each time step, one reaction is selected probabilistically and carried out to determine the concentrations of the reactants at the next time step (Kibby 1969).

Because each reaction representation has different strengths, an optimal program would offer both differential equations and stochastic simulations.

Space for Annotation

Researchers also need to store annotations for the various sources of information including the literature supporting each reaction and parameter. This is especially important when two sources in the literature give contradictory values (Hoffman et al. 2002).

Optimization

Finally, an ideal program would perform optimization to fit the computational model to experimental data. This functionality is especially important because parameter values can be unavailable, vague, or contradictory in the literature. In addition, feedback loops within a model can amplify error, yielding an almost-correct system that behaves incorrectly.

Mendes and Kell tested common optimization algorithms such as evolutionary programming, truncated Newton, and genetic algorithms. Not surprisingly, they found that no single method works best for all problems (Mendes 1998). Thus, an ideal program would provide several optimization algorithms for the user to try.

Review of Potential Modeling Programs

Similar projects in the past have used programs such as Matlab (Schoeberl 2002), Kinetikit (Bhalla 2002), and JDesigner (Hoffmann 2002). Of these three programs, Matlab and Kinetikit meet only some of the requirements while JDesigner meets all four of the previously established requirements.

Matlab is a commercial software package that provides a flexible environment for mathematical manipulation of data (Mathworks). As a result, a programmer can code any desired mathematical reaction representation including ordinary differential equations and stochastic simulations. As an added benefit, Matlab comes with several ordinary differential equation solvers. The company also offers several optimization packages although these were not purchased and explored in this study. Annotation can be added as comments in the code. However, Matlab was not created with the analysis of biological systems in mind. At best, the user can program an interpreter to convert a worksheet of biological equations into differential equations as shown in Figure 3.1 (Shoeberl 2002). This approach distracts from the biology and draws attention to the mathematical representation.

Kinetikit, also known as GENESIS, is an academic program specifically designed to provide a friendly biological interface (Bhalla 2002). However, Kinetikit does not support stochastic reactions because the creators believed that the technology to gather enough experimental information to justify a stochastic model did not yet exist. In addition, the program does not offer any optimization algorithms.

JDesigner is part of the Systems Biology Workbench project established at the California Institute of Technology (California Institute of Technology, Sauro). JDesigner offers a biologically intuitive interface as shown in Figure 3.2. Associated with each molecular reaction is mathematical representation. JDesigner specifically supports ordinary differential equations and stochastic simulations but mathematical equations of any format can also be typed in. JDesigner also allows for annotation of nodes and reactions. Although optimization is not implemented, the models can be saved in a standardized language, SBML, to be imported into a different package that does perform optimization. In particular, a program known as Gepasi (Mendes 1997) is also part of the Systems Biology Workbench and specializes in fitting algorithms. Gepasi offers both global algorithms such as evolutionary algorithms and simulated annealing as well as local algorithms such as gradient descent.

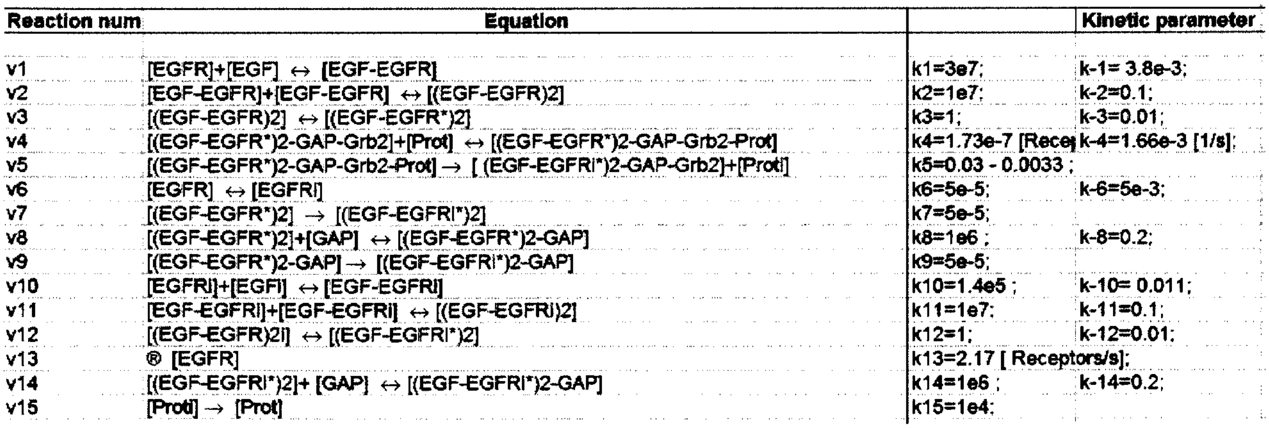
Some other programs explored are displayed in Table 3.1

Table 3.1 Summary of Potential Modeling Programs.

Program	Drawbacks
DBsolve, KineCyte	No longer supported
Virtual Cell	Could not install
ProcessDB	No documentation
Entelos LabBuilder	Proprietary
GNS	No mathematical backend
ModelMaker, ACSL, VisSim	Cumbersome interface
Berkeley Madonna, JSim	No support for annotation
STELLA, SCAMP, BioQuest, E-cell	No support for optimization

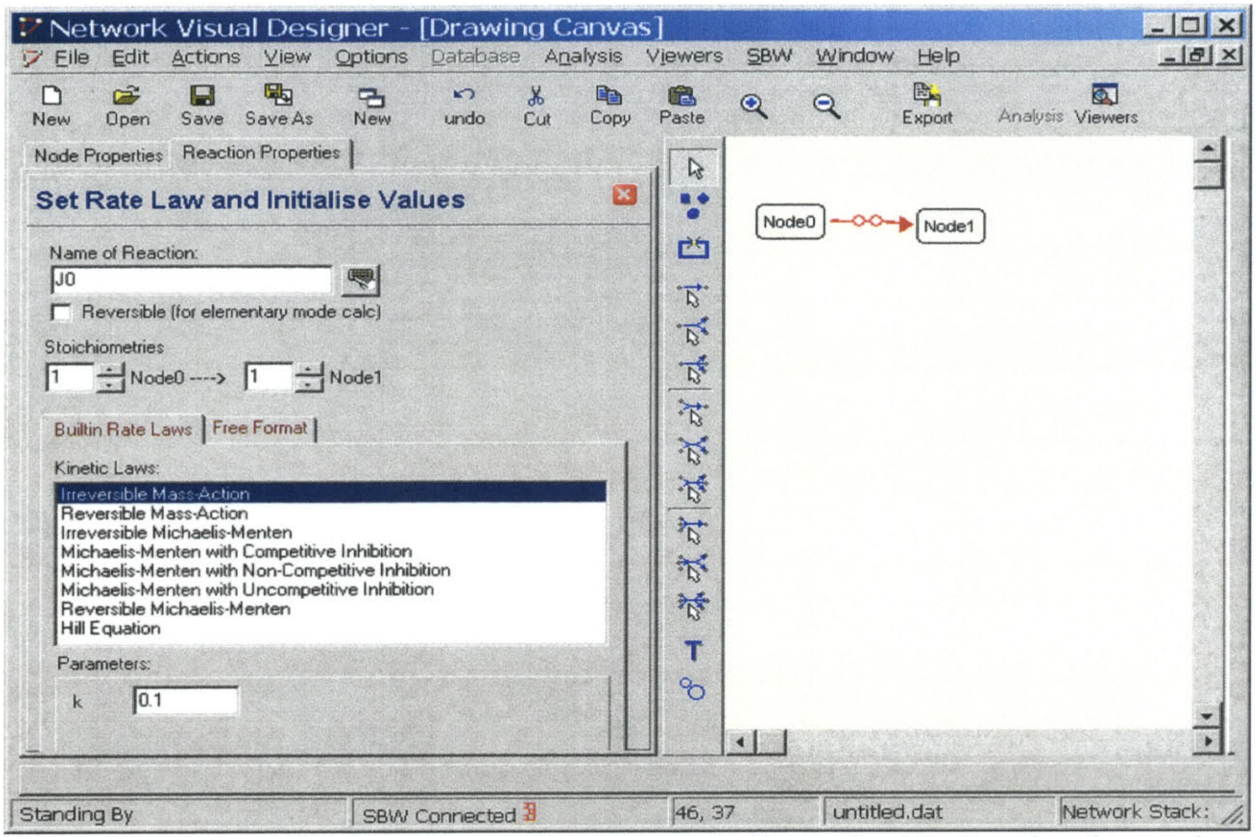
Discussion of JDesigner

Beyond the minimum requirements JDesigner offers some extra advantages as well as some drawbacks. Among the advantages are that JDesigner can display output in graphical or worksheet form. This feature allows the user to output the data to other data organization programs such as Excel. In addition, JDesigner is open source allowing a programmer to modify the program for specific needs. Among the drawbacks are that JDesigner can quickly become visually complex. Unlike the bubble diagram introduced in the "Intuitive Interface" section, JDesigner cannot represent a concept such as Flip inhibition of Caspase 8 recruitment without depicting the mechanism in dense detail as shown in Figure 3.3. In particular, Figure 3.3 displays all the permutations of three FADD molecules and three Flip molecules that could cause or inhibit cleavage of caspase 8. A related drawback is that JDesigner does not allow the depiction of reactions at different levels of abstraction. For example, if the exact mechanism of Flip mediated inhibition were not known, one might still want to include inhibition in the model as a boolean variable. With JDesigner the mixture of boolean and differential equations is not possible. Finally, JDesigner does not incorporate any mechanism for output and model management which can quickly become a problem. For example, the user will often change the initial concentration of one molecule in the model to observe the outcome. Without careful documentation, the user can easily become confused about which outputs correlate to which changes in initial concentration.



Reaction num	Equation		Kinetic parameter
v1	[EGFR]+[EGF] ↔ [EGF-EGFR]	k1=3e7;	k-1= 3.8e-3;
v2	[EGF-EGFR]+[EGF-EGFR] ↔ [(EGF-EGFR)2]	k2=1e7;	k-2=0.1;
v3	[(EGF-EGFR)2] ↔ [(EGF-EGFR*)2]	k3=1;	k-3=0.01;
v4	[(EGF-EGFR*)2-GAP-Grb2]+[Prot] ↔ [(EGF-EGFR*)2-GAP-Grb2-Prot]	k4=1.73e-7 [Rece	k-4=1.66e-3 [1/s];
v5	[(EGF-EGFR*)2-GAP-Grb2-Prot] → [(EGF-EGFRi*)2-GAP-Grb2]+[Proti]	k5=0.03 - 0.0033 ;	
v6	[EGFR] ↔ [EGFRi]	k6=5e-5;	k-6=5e-3;
v7	[(EGF-EGFR*)2] → [(EGF-EGFRi*)2]	k7=5e-5;	
v8	[(EGF-EGFR*)2]+[GAP] ↔ [(EGF-EGFR*)2-GAP]	k8=1e6 ;	k-8=0.2;
v9	[(EGF-EGFR*)2-GAP] → [(EGF-EGFRi*)2-GAP]	k9=5e-5;	
v10	[EGFRi]+[EGFi] ↔ [EGF-EGFRi]	k10=1.4e5 ;	k-10= 0.011;
v11	[EGF-EGFRi]+[EGF-EGFRi] ↔ [(EGF-EGFRi)2]	k11=1e7;	k-11=0.1;
v12	[(EGF-EGFR)2i] ↔ [(EGF-EGFRi*)2]	k12=1;	k-12=0.01;
v13	® [EGFR]	k13=2.17 [Receptors/s];	
v14	[(EGF-EGFRi*)2]+ [GAP] ↔ [(EGF-EGFRi*)2-GAP]	k14=1e6 ;	k-14=0.2;
v15	[Proti] → [Prot]	k15=1e4;	

Figure 3.1 Worksheet of biological equations. A program written in Matlab can convert these equations into differential equations.


Network Visual Designer - [Drawing Canvas]
File Edit Actions View Options Database Analysis Viewers SBW Window Help
New Open Save Save As New undo Cut Copy Paste Export Analysis Viewers
Node Properties Reaction Properties
Builtin Rate Laws Free Format
Node0
Node1

Set Rate Law and Initialise Values

Name of Reaction: J0

□ Reversible (for elementary mode calc)

Stoichiometries

1 Node0 ---> 1 Node1

Kinetic Laws:

- Irreversible Mass-Action
- Reversible Mass-Action
- Irreversible Michaelis-Menten
- Michaelis-Menten with Competitive Inhibition
- Michaelis-Menten with Non-Competitive Inhibition
- Michaelis-Menten with Uncompetitive Inhibition
- Reversible Michaelis-Menten
- Hill Equation

Parameters:

k 0.1

Standing By | SBW Connected | 46, 37 | untitled.dat | Network Stack:

Figure 3.2 User Interface of JDesigner. The right panel offers an intuitive interface for creating reactions. The left panel offers the option of built-in rate laws or free format rate laws to represent the reaction on the right. The small keyboard symbol beside the “Name of Reaction” user input box offers the user the option of adding annotation.

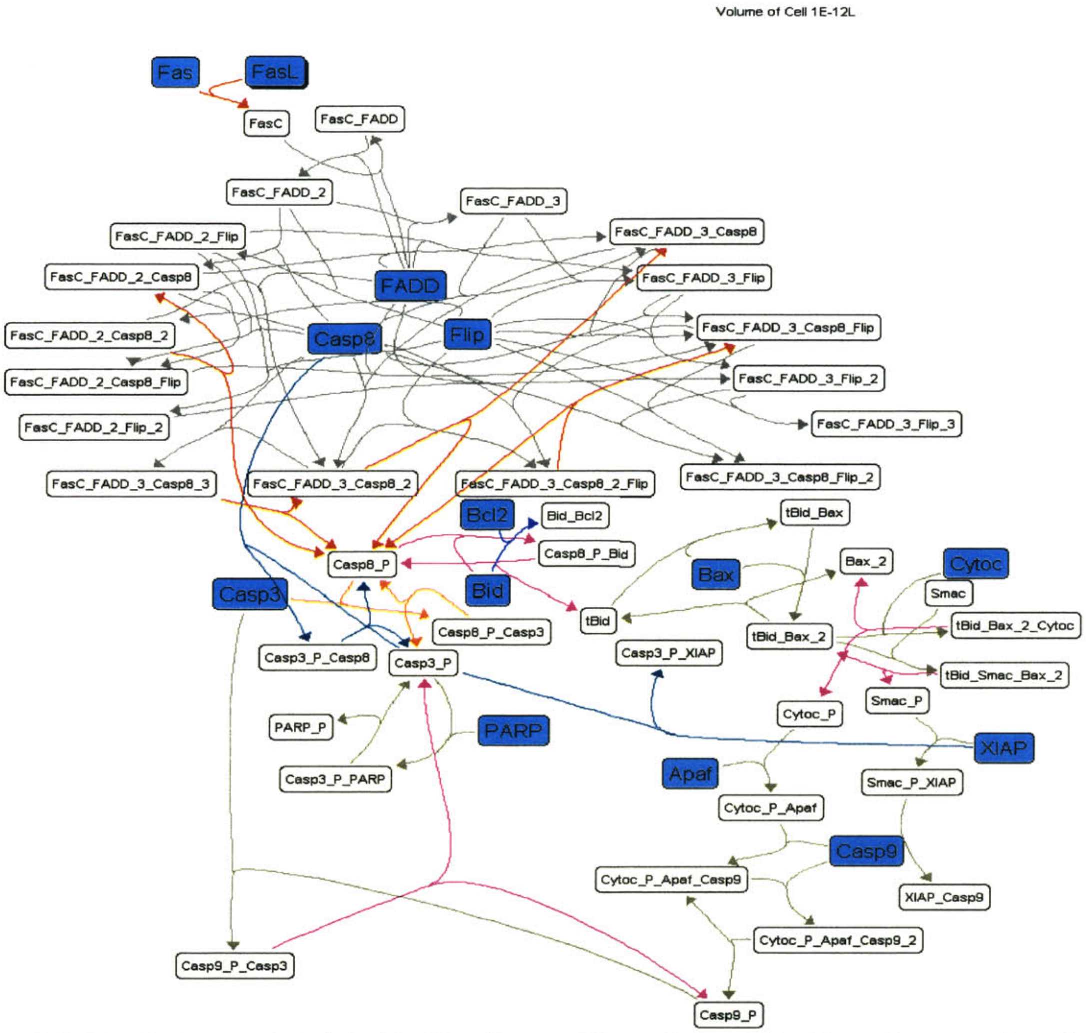


Figure 3.3 Fas Pathways Modeled in JDesigner. The red arrows indicate key reactions leading to the activation of caspase 8. The orange arrows indicate reactions in the direct pathway to cleavage of capase 3. The purple arrows indicate key reactions in the mitochondrial pathway. The blue arrow indicates inhibition of the mitochondrial pathway by Bcl-2. The green arrows indicate reactions that are still being tested. Important molecules are highlighted in light blue.

Chapter 4. Apoptosis Assays

Introduction

Apoptosis, programmed cell death, serves as an important and easily measured outcome of the interaction of multiple cellular cues in T cells. This study addresses two receptor mediated pathways that can result in T cell apoptosis. First, activation of Fas via Fas ligand binding will generally induce apoptosis in T cells. In addition, activation of the T cell receptor (TCR) can modulate these cell death signals. We are interested in the molecular interaction of the Fas and TCR pathways. When Fas and TCR are both activated, the extent of interaction between the pathways can be observed through the difference in the levels of apoptosis compared to Fas or TCR stimulation alone. However, because apoptosis is influenced by multiple cues, accurate measurement requires careful control of experimental conditions.

Among the cues that can influence apoptosis or apoptosis measurements are cell density, handling, and length of staining. In particular, some cells die due to overcrowding. Other cells require growth factors for survival, released in an autocrine manner or by nearby cells. Cells can also die due to the physical damage caused by rough handling. In addition to variables affecting cell death, there are variables affecting the measurement of cell death. The two most common stains for apoptotic cells, annexin-V and propidium iodide (PI), each identify different features of a dying cell and each have experimental limitations. Annexin-V is known to fade with time, while the PI dye can enter and stain live cells, albeit more slowly than dying cells, and may be toxic. To determine which of these parameters influence the accuracy of apoptosis experiments, the effects of different cell densities after 24, 48 and 96hrs, the effect of transferring and washing cells, and the effect of PI staining over time were studied.

Materials

The immortalized human T cell lines H9 [ATCC HTB-176] and Jurkat were grown with 5% CO_2 in C10 medium (420ml RPMI, 50ml fetal bovine serum [Gibco 20437-028], 5ml Penicillin-Streptomycin [Gibco 15140-122], 5ml L-glutamine 200mM [Gibco 25030-081], 5ml non-essential amino acid solution 10mM [Gibco 11140-050], 5ml sodium pyruvate 100mM [Gibco 11360-070], 5ml Hepes 1M [Gibco 15630-080], 5ml 5.5mM 2-Mercaptoethanol [Gibco 21985-023] sterile filtered through 0.22um filter). Propidium iodide was obtained from Roche Applied Science (Indianapolis, IN.). RPMI and phosphate buffered saline (PBS) was prepared by the media kitchen at the MIT Center for Cancer Research according to standard recipes.

FACScan (BD Biosciences, San Jose, CA) was used to perform flow cytometry.

Protocols

Counting Cells

Because the cells tend to settle to the bottom of the flask in which they are grown, the cells were well mixed before 20ul of cell solution was added to 80ul of Trypan Blue stain [Gibco 15250-061]. Four quadrants of the Hausser Dark-Line hemacytometer [Hausser Scientific, Horsham, PA] were counted excluding any dead cells which appear blue due to the uptake of Trypan Blue. The cell concentration was determined by the following equation: Cells/ml = total number of counted cells/4x10^4x5.

Cell Death Assay

On day 1, cells were diluted to a density of 0.3x10^6 cells/ml in pre-warmed C10 media. On day 2, the cells were counted and the concentration recorded. An aliquot of the cells was spun down at 12000rpm for 5min and resuspended in medium to a density of 5x10^6 cells/ml. The rest of the cells were kept in the original flask. The resuspended cells and media were added to each well in 3 96 well flat-bottom plates [Corning Inc., Corning, NY] as indicated in table 4.1. Column 5, rows 1, 2, and 3 contained cells that were taken directly from the original flask. This column served as a control for any necrosis that might occur as a result of resuspending the cells to a density of 5x10^6 cells. The starting time for incubation was noted and the plates were incubated at 37°c with 5% CO_2 for 24hrs for plate 1, 48hrs for plate 2, and 96hrs for plate 3.

Table 4.1 Cell dilution and media added to each well in a 96 well flat bottom plate.
Plate 1 (24hr), 2 (48hr), 3 (96hr)

Column	Row	ul of 5x10^6 cell solution	Media (ul)	Crowding Concentration (10^6 cells/ml)
1,2,3	1	4	196	0.1
	2	10	190	0.25
	3	20	180	0.5
	4	40	160	1
	5	120	80	3
	6	200	0	5
	7			
	8			

After 24, 48, and 96hrs, cells from the appropriate plate were moved to 96-well round-bottom plates [Corning Inc., Corning, NY]. To ensure complete transfer of all the cells, cells were pipetted vigorously in the well before transferring. The entire plate was spun down and the supernatant removed. At this point, the end time for incubation was recorded. 200ul of cold PBS was added to each well to remove color and debris. The plate was spun down again. Finally, 200ul of cold PBS was added and the cells were transferred to FACS tubes [Falcon 35-2052]. 200ul of cell culture was also transferred directly from the flask into 3 FACS tubes. These extra tubes served as a control to determine the difference in necrosis between the cells that had been subjected to all the transfer and wash steps as opposed to cells that had not.

Fluorescence activated cell sorting (FACS) was performed by adding 200ul of PI (5ug/ml) to each tube immediately before taking a FACS measurement and vortexing. 20,000 events considered to be live cells were collected for each sample. Samples were left for an hour then measured again to test for changes in measured apoptosis.

Discussion

Cell Density

A value of 10% cell death or 90% survival was considered acceptable in these experiments. Figure 4.1a shows that Jurkat cells exhibit a low level of apoptosis when cultured at densities of up to 1 million cells/ml 24hrs after the cells were originally transferred into the 96 well plate. On each subsequent day, the densities with acceptable levels of cell death dropped by half to 0.5 million cells/ml on the second day, and 0.25 million cells/ml on the third day.

H9 cells are larger cells than Jurkat cells (data not shown), and therefore are likely to be crowded at lower densities than Jurkat cells. Thus a shift of the cell death curves to the left, to lower densities, was expected. However, on the first and second day, the H9 cells behaved similarly to the Jurkat cells with a low level of apoptosis for up to 1 million cells/ml and 0.5 million cells/ml respectively (Figure 4.1b). By the third day even the lowest cell density of 0.1 million cells/ml showed cell death greater than 10%, suggesting that the effects of crowding were most pronounced at this time. One possible explanation for this phenomenon is that the H9 cells have a slower cell cycle than the Jurkat cells and the effects of growth and crowding do not appear until the third day.

Neither cell line seemed to suffer from the reduction in autocrine or paracrine growth factors likely to be associated with culturing cells at lower densities. If growth factors were important, the plot of cell death was expected to be a concave parabola, with high cell death at low densities due to the lack of cytokines, and high cell death at high densities due to crowding. Instead curves (Figure 4.1a, b) appear to be sigmoidal with increased cell death correlating with increased crowding. The lack of cell death at low densities may be explained by two possibilities. First, the densities tested were not low enough to induce cell death due to growth factor withdrawal. In other words, even at the lowest densities tested, the cells are producing enough growth factors to grow. Second, the Jurkat and H9 cells do not need cytokines because they are immortalized and obtain all necessary nutrients from the culture medium which is supplemented with serum proteins and other essential molecules. Experiments to distinguish these possibilities would involve culturing Jurkat and H9 cells in the presence of growth factor inhibitors or in medium with reduced levels of serum proteins.

These experiments suggest that apoptosis assays would be best performed at the lowest possible cell densities, because these are associated with lower levels of background cell death. However, the advantage of the lower level of cell death must be balanced with the requirements for performing FACS analysis. With less cells, it is more difficult to obtain a statistically significant number of measurements (typically 20,000 cells or events). Therefore, it was determined that performing experiments at 0.5 million cells/ml within 24hrs would provide the optimal balance between low levels of cell death and speed of FACS.

Processing Steps

Figure 4.2 compares the cells plated in column 5 of the the 96-well plates (cells which are at the same cell density as the cells in the flask) with the cells transferred directly from the flask into FACS tubes on the day of performing the FACS. Theoretically the cell density and the percent of cell death in these populations should be the same. However, the cells taken from the flask consistently have a lower percentage of cell death. This observation indicates that during the process of growing in a 96 well plate, being transferred, spun down, and washed, cell death increases, probably due to mechanical stress. The H9 cells appear to be particularly sensitive showing a 45% difference in cell death by day 3 (Figure 4.2b). However, for at least 24hrs, the difference between the well and flask populations is small enough to keep the percentage of cell death below 10%. From these results as well as the results for cell density, it was decided that cells would experience the least amount of stress and background apoptosis if experiments were conducted within 24hrs.

Staining Over Time

Figure 4.3 shows that an increase in staining with PI can occur after less than 1hr. All samples consistently showed an increase in staining and the staining could increase observed cell death by up to 18%. These results ran counter to the expectation that the PI might, like Annexin-V, fade over time and decrease observed cell death. One explanation for the increase in staining may be that prolonged exposure to PI can stain cells on the way to dying. In particular, PI crosses the permeable membrane of dead cells to stain the DNA. The intact membrane of live cells generally prevents the PI from reaching the DNA. However, cells on the way to dying may have slightly more permeable membranes that slowly allow the PI to cross. This may lead to an increase in stain with time. Another explanation is that the cells begin to die due to media withdrawal because the cells are left in PI without growth factors for one hour. Whether the dye began to stain live cells or the live cells began to die, it was decided that PI should be added immediately before performing FACS and FACS should be performed no more than 1hr after media withdrawal.

In conclusion, we established that apoptosis assays should be performed within 24hrs with an initial cell density of 0.5 million cells/ml to reduce background cell death. In addition, FACS analysis should be performed no more than 1hr after withdrawal from media and PI should be added immediately before performing FACS.

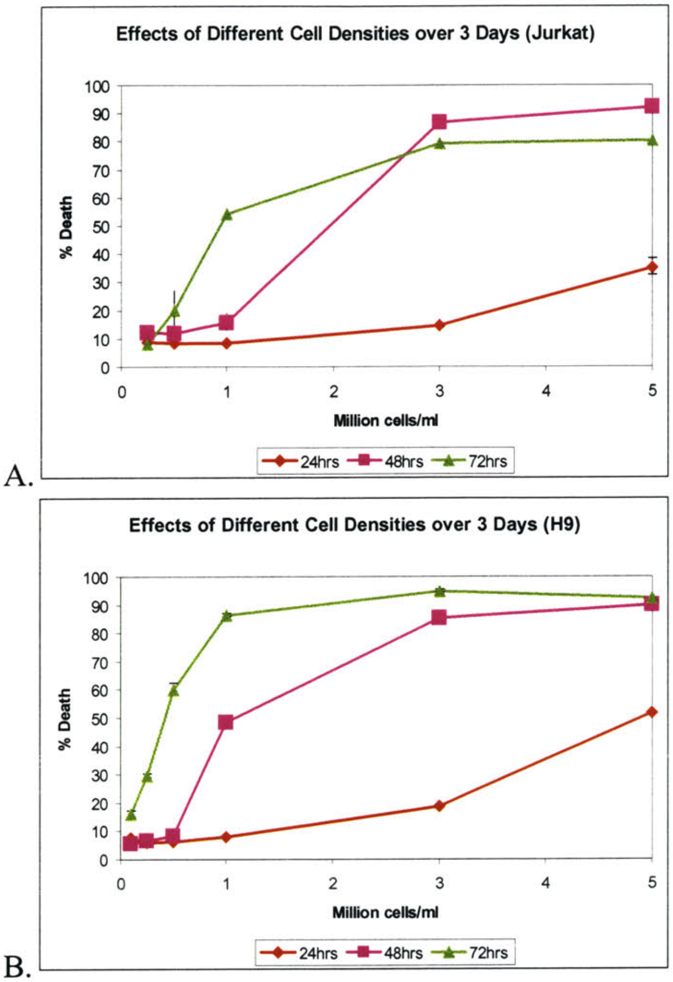


Figure 4.1 Apoptosis of Jurkat and H9 cells Initially Plated at Different Cell Densities over 3 Days. Cells were plated at different initial densities then measured for cell death by FACS analysis of propidium iodide staining after 24, 48 and 72 hrs. A) Jurkat cells are below 10% cell death for densities up to 1 million cells/ml after 1 day but only densities below 0.25 million cells/ml are below 10% cell death after 3 days. B) H9 cells are below 10% cell death for densities up to 1 million cells/ml after 1 day but even a density of 0.1 million cells/ml cannot prevent cell death greater than 10% after 3 days.

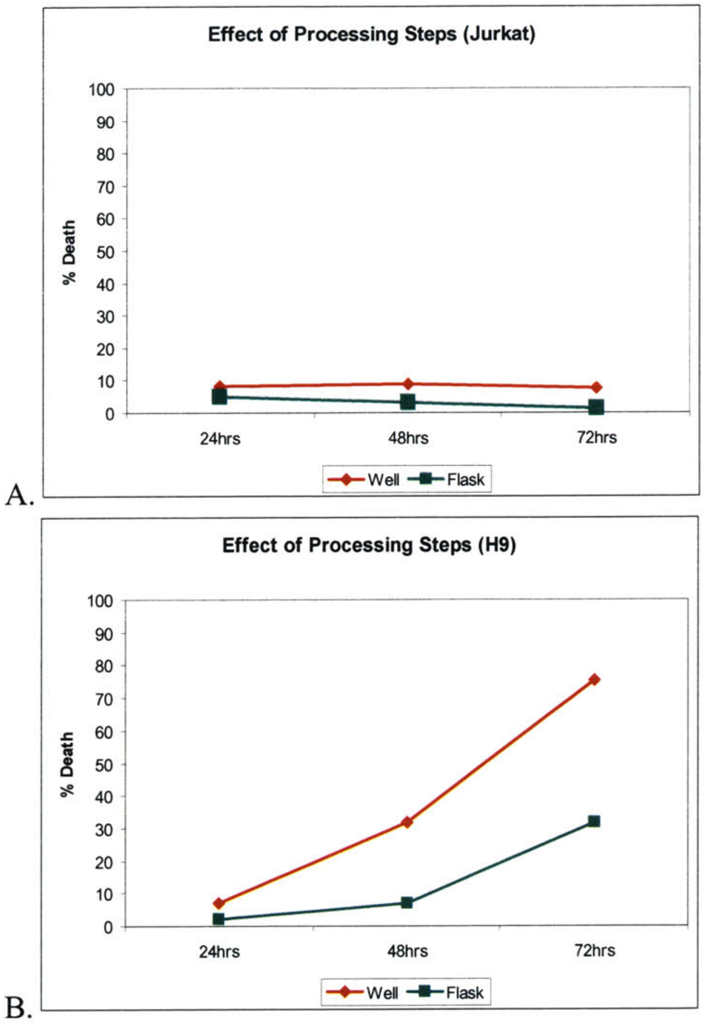


Figure 4.2 Effect of Processing Steps on Cell Death of Jurkat and H9 cells over 3 days. A) Cell death of Jurkat cells is affected by only 4-6% due to the growth conditions in the 96 well plate and the transfer and wash steps. B) H9 cells exhibit a great disparity between cells grown in the wells and subjected to the transfer and wash steps compared to cells in the flask after 48hrs.

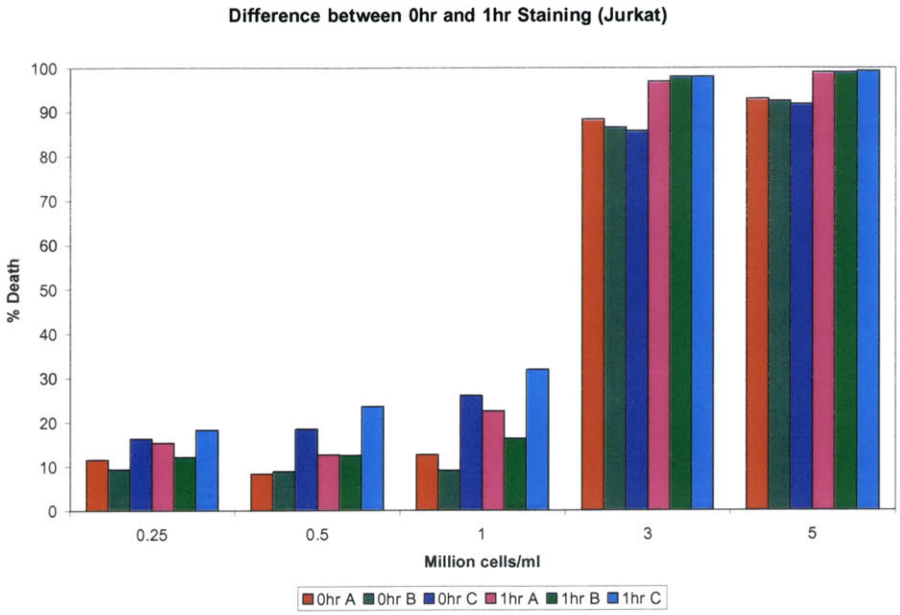


Figure 4.3 Difference between 0hr and 1hr Propidium Iodide Staining of Jurkat cells on a Representative Day (48hrs). The graphs show each triplicate A, B, and C for each cell density measured right after PI addition or 1hr after PI addition. The difference in staining ranges from 1-18%. Similar results were obtained for H9 cells.

Chapter 5. RNAi

Introduction

A cellular process known as RNA interference (RNAi) has recently been discovered and has been shown to be useful, experimentally, to efficiently suppress gene expression in mammalian cells. The RNAi pathway can be triggered by the introduction or expression of short sequences of double stranded RNA known as short interfering RNAs (siRNA) in cells. This induces the cleavage of complementary mRNA sequences, resulting in reduced stability of the mRNA and, consequently, a decrease in the level of its product, a protein, in the cell.

The mechanistic details of RNAi are still being investigated, but it appears that the double stranded siRNAs are incorporated into a complex known as the RNA-induced silencing complex (RISC). This complex unwinds the siRNA and is guided by one of the strands to the complementary mRNA sequence which is then cleaved (see McManus and Sharp (2002), Dykxhoorn et al. (2003), and Denli and Hannon (2003) for more detailed reviews).

There are several methods to introduce the siRNA into mammalian cells. The short hairpin method used by the Van Parijs laboratory as well as others, involves the use of vectors that express short hairpin RNAs (shRNAs) in cells. The shRNA consists of one strand of the siRNA (the sense strand), a loop sequence, and the other strand of the siRNA (the antisense strand) (Rubinson et al. 2003). The loop sequence is eventually cleaved by a protein, Dicer, leaving just the siRNA sequence.

Experimentally RNAi provides two advantages over other techniques to genetically manipulate mammalian cells. First, transfecting siRNAs is much less involved than knocking out a gene, reducing time and cost. Second, different siRNA sequences can repress genes to different extents. In theory this should allow cell behavior to be systematically observed at multiple gene expression levels. These experimental advantages are especially important for systems biology. Because RNAi is such a flexible and efficient technique, it can be used as a screening method to knockdown key genes and observe the cellular effects. In addition, multiple gene expression levels provide an opportunity to study the behavior of the cell with different starting states for a single gene.

One of the difficulties for the systematic use of RNAi is the selection of functional siRNA sequences. As mentioned above, different sequences can repress to different extents and many sequences are considered nonfunctional because they fail to silence gene expression altogether. Many researchers have been working to unravel the requirements to select functional siRNA sequence and their work is reviewed below. Automating this process provides two key benefits. First, a program will save the user time in gathering information to select a sequence. Second, a program can serve as a center for knowledge about siRNA selection so each user does not need to keep up-to-date with developments in the understanding of RNAi as long as the program is kept up-to-date. The specific purpose of this aim was to identify or create a program for predicting successful sequences.

Requirements for a Sequence Prediction Program

The study of RNAi is a recent field and is highly active. In order to incorporate new information and to handle the lack of standards, a sequence prediction program must be both extensible and flexible. For example, a study by Khvorova et al. (2003) elucidating an important criterion for successful siRNA sequences was published in October, just two months prior to the writing of this thesis. An extensible program should easily expand to consider this new criterion.

The flexibility of the program is also important. In particular, because the importance each selection criterion relative to others has not been studied thoroughly it is important that the program allow a scientist to perform the ranking manually if desired. In addition, the program should be flexible to features of the shRNA that are likely to change such as the motif used and the loop sequence (explained below).

Requirements for Functional siRNA Sequences

By examining large populations of functional and non-functional siRNAs, two groups have recently established criteria that statistically lead to the selection of better siRNAs. Khvorova et al. (2003) identified that the free energy, ie. the strength of the bonds, between the two siRNA strands had a significant impact on the efficiency of gene silencing. They determined that successful siRNA sequences had lower free energy on the 5' end of the antisense strand, the loop end in the shRNA, than on the 3' end, the free end in the shRNA (see Figure 5.1). Successful sequences also had a low free energy for nucleotides 9 to 14 of the antisense strand. These free energy characteristics are hypothesized to help the RISC unwind the siRNA. Pusch et al. (2003) showed that a single nucleotide mismatch between the siRNA sequence and the targeted mRNA also significantly reduced the effect of the siRNA, presumably by causing the guidance of the RISC to be less accurate. In addition, mismatched sequences may induce a different gene silencing pathway, known as miRNA which functions by attenuating translation.

Other criteria for selection siRNA sequences are not based on such strong experimental evidence but are generally included in selection algorithms. These include maintaininng the GC content of the sequence between 30% and 70% (Dykxhoorn 2003). The reasoning was that mRNA nucleotides G and C are known to bind more strongly than A and U. Thus a high GC content may affect RNAi by making unwinding by the RISC difficult, while low GC content might make duplexes unstable and reduce their lifespan. Many researchers also suggest running a Blast search, a search to find nontargeted genes with sequences similar to the siRNA, to reduce nonspecific knockdowns (McManus and Sharp 2002, Dykxhoorn 2003). Other restrictions are laboratory specific. For example, the Van Parijs laboratory uses siRNAs with the sequence motif AAGN20 where N represents any nucleotide and 20 represents 20 Ns in a row. Other laboratories use other motifs such as NAN19NN, NARN17YNN, and NANN17YNN where R represents A or G, and Y represents C or T (Dykxhoorn 2003). These restrictions are typically imposed by the specific vector system that is used. In addition, most laboratories use shRNA expression systems that utilize RNA polymerase III to transcribe the short RNA hairpin. This polymerase will terminate if it detects a stretch of 4 of more As in a row. Therefore the use of the pol III promoter requires a search to check that such a sequence is not present in the shRNA.

Review of Potential RNAi Sequence Prediction Programs

Several public programs exist for the prediction of siRNA sequences including siRNA Target Finder (Ambion), RNAi OligoRetriever (Ravi Sachidanandam Laboratory at Cold Spring Harbor Laboratory), and Hairpin siRNA selection Program (Biocomputing at Whitehead Institute).

siRNA Target Finder and RNAi OligoRetriever are both missing important criteria and flexibility. siRNA Target Finder allows the user to restrict GC content and avoid sequences with four As in a row. In addition, the user receives a link to perform a Blast search with the results. However, the program does not allow the user to avoid sequences targeting single nucleotide polymorphisms and does not allow the user to specify the sequence motif or the loop sequence for shRNAs. RNAi OligoRetriever provides the user with a choice of four common RNAi systems, which generally determines the motif, then returns siRNA sequences ranked according to an internal set of criteria.

In contrast, the Hairpin siRNA selection Program does include most of the currently known important criteria. The user inputs the accession number, which is a unique identifier for each gene in the NCBI database, Genbank, of the target gene and sequence motifs desired for the shRNA. The program outputs each potential siRNA sequence with criteria including GC content, single nucleotide polymorphisms, and Blast results. In addition, the program considers other often ignored criteria such as self alignments. Sequences with self alignments may fold over on themselves and create structures that prevent uptake by RISC. However, the program does not allow the user to avoid sequences with four As in a row and does not work for all genes that are in GenBank.

At the time of writing this thesis, none of the above programs considered 5' versus 3' free energy. In addition, none of the programs allowed an experienced user to rank the importance of the criteria. To solve these problems, a new RNAi sequence prediction program was created.

Program Design

The new RNAi sequence prediction program was specifically designed to incorporate extensibility and flexibility. For extensibility, the program was written with a main function that calls several subroutines that calculate the values for each criterion. The values calculated by each subroutine are explained below. Adding a new criterion simply involves writing another subroutine and adding a call to the subroutine in the main function. The code is well commented to make this task straightforward for programmers unfamiliar with the program.

To incorporate flexibility, the program was designed to gather the information for each criterion separately, then generate an output file for each criterion in Excel. In Excel, the user can maually sort the sequences based on the criteria that are deemed most important or relevant. Once the sequences are sorted, the user can run a macro to identify the four top-ranked, nonoverlapping sequences. This feature has been incorporated because overlapping sequences may cause the RISC to target the same region and interfere with each other. The number of top sequences to return is flexible and can be changed by changing the value for a variable, top_num_seq, defined at the beginning of the macro. In order to find the top nonoverlapping sequences, each sequence is considered a node and is weighted by the rank. For example, the first sequence has a weight of 1, and the second sequence has a weight of 2. The A* algorithm is then run to find a path of length top_num_seq with the least weight and nonoverlapping nodes. The main drawback of using the rank as the weight is illustrated by the example when sequence 1 is far superior to sequences 2 and 3 which are almost equal in quality. In other words, the rank does not reflect accurately the differences in the quality of the sequences. In the future, one might consider other weighting schemes such as a summation of the values for each criterion. Another macro formats the chosen sequences by adding sequences such as the loop for shRNAs. The sequences to be added are flexible and can be changed by editing the beginning of the macro.

The input for the program is a file, "Sequences.txt", listing the GenBank ID and annotation for the target gene. The output, as explained above, is an Excel file, "Output", listing potential sequences followed by the values for each criterion. This file consists of twelve columns:

Column 1: GenBank ID originally provided by the user

Column 2: Annotation originally provided by the user

Column 3: Potential siRNA sequence. Currently the program searches the target mRNA for AAGs then takes the next twenty nucleotides resulting in sequences with the motif AAGN20. This motif is currently hard coded but programming a subroutine to search for other motifs should not be difficult.

Column 4: Position of the region targeted by the potential siRNA sequence on the target mRNA. This information is used by the A* algorithm explained above.

Column 5: GC content of the potential siRNA sequence.

Column 6: Longest self alignment found within the potential sequence. This criterion is identical to the self alignment criterion used by the Hairpin siRNA selection Program and has a slightly misleading name. The subroutine searches for consecutive self alignments not gapped self alignments. For example, the program will return AGCT as the longest self alignment for the sequence AGCTACCGCGGAAGCT even though AGCT_CCG will align with TCGA_GGC with only one mismatch. Standard algorithms for determining gapped self alignments for sequences as short as 23 nucleotides are currently unavailable.

Column 7 and 8: Results of Blast for the potential siRNA sequence. Reporting the results of Blast presented two challenges. First, GenBank contains many redundancies such as a record for a gene and a record for the same gene transfected into a different organism. As a consequence, Blast results will show that the potential sequence aligns perfectly with several records in GenBank. The program currently assumes that all perfect alignments are actually alignments with the target gene and reports the first Blast result with an imperfect alignment. However, the assumption may be incorrect and may miss a perfect alignment with a non-target gene. Furthermore, the first imperfect alignment may just be a variant form of the target gene. Second, the time required to execute a Blast search is currently a bottleneck for the efficiency of the program. Both these problems may be solved by using a database with fewer redundant records such as UniGene. However, this also presents problems because GenBank is the most comprehensive database available and not all unique genes in GenBank are in UniGene.

Column 9: 5' versus 3' free energy on the antisense strand. The free energy is calculated using the values listed in the Erratum by Khvorova et al. (2003).

Column 10: 9 to 14 nucleotide free energy on the antisense strand. The free energy is calculated as above.

Column 11: Presence of single nucleotide polymorphisms in the targeted region

Column 12: Presence of a termination sequence, four As in a row, within the potential sequence.

The code for the program and macros written in Python and Visual Basic respectively are provided in Appendix A. Python was chosen as the programming language because of the functions provided by the Biopython Project, an open source project to develop Python tools for bioinformatics (Biopython). In particular, Biopython includes a useful interface to perform Blast searches. Excel was chosen as an output format due to the prevalence of Microsoft.

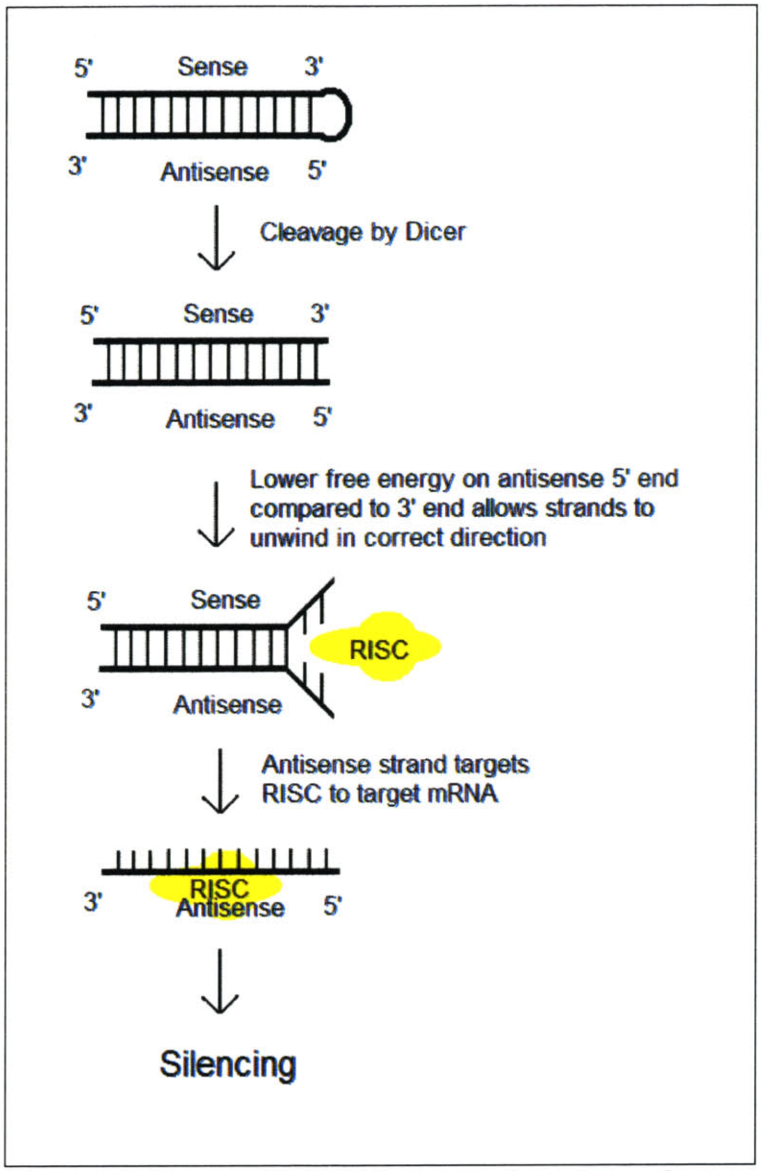

5'
Sense
3'
3'
Antisense
5'
Cleavage by Dicer
5'
Sense
3'
3'
Antisense
5'
Lower free energy on antisense 5' end compared to 3' end allows strands to unwind in correct direction
5'
Sense
RISC
3'
Antisense
Antisense strand targets RISC to target mRNA
RISC
3'
Antisense
5'
Silencing

Figure 5.1 Successful shRNAs tend to have lower free energy on the antisense 5' end compared to the 3' end.

Chapter 6. Application of JDesigner to Model Fas Signaling Pathways

As mentioned in Chapter 2, an important unanswered question about Fas signaling in T cells is how and why these cells adopt either type I or type II behavior. This question is being addressed by comparing Fas signaling in H9 cells (type I) and Jurkat cells (type II) experimentally. However, the complexity of Fas signaling pathways makes it difficult to perform a systematic and comprehensive analysis of this biological process in the laboratory. For this reason, we constructed a computational model to represent the current state of knowledge about the Fas signaling pathways with the goal of exploring type I and type II signaling behaviors in silico.

Results and Discussion

The architecture of the Fas signaling pathway that leads to apoptosis has been fairly well established using traditional molecular biology techniques (see Chapter 2). This suggested the possibility of creating a detailed computational model. Of the possible mathematical representations, ordinary differential equations were chosen because most molecules were considered to be abundant making a stochastic model unnecessarily complex. In addition, similar models of signaling pathways have used ordinary differential equations in the past (Hoffman et al. 2002, Schoeberl et al. 2002). All reactions were modeled as bimolecular based upon the reasoning that multi-molecular reactions can be represented by a series of bimolecular reactions at the most basic level. A comprehensive, deterministic model for the Fas signaling pathways was created by Dr. Fei Hua in Matlab, and subsequently translated into JDesigner by me (Figure 3.3).

The architecture of the model generally follows the structure of Fas signaling determined experimentally and detailed in chapter 2. However, a number of simplifications and assumptions were made.

First, binding of Fas and Fas ligand (FasL) to form the Fas complex (FasC) was modeled as a bimolecular reaction rather than as a stepwise association of each of three FasL with one of three Fas. This representation is based on the observation by Siegel et al. (2000) that the Fas receptor exists on the surface of cells as a preassociated trimer. It is worth noting that some groups, including Holler et al. (2003), suggest that two trimeric FasL may be required for signaling. In the future we will use our computational model to explore how these different ways of triggering Fas might affect signaling by this receptor.

Second, we assumed that caspases would intiate signaling as soon as two or more of these molecules were recruited to signaling complexes, such as the death inducing signaling complex, DISC, formed by Fas, FADD, and caspase-8, or the apoptosome formed by cytochrome c, Apaf-1, and caspase-9. This assumption is based on experiments by Chang and Yang (2000) that demonstrate that dimerization is sufficient to activate caspase 8. The apoptosome is known to be a complex of seven units of 1 Apaf, 1 cytochrome-c and 1 caspase-9 each (Shi 2002). Originally we considered modeling the formation of the apoptosome as the formation of each three molecule unit followed by the binding together of seven of these units. However, how this complex forms is still unknown and we believed such a representation would be misleading. Therefore we compromised by modeling the interaction as bimolecular.

Third, we assumed that signaling by BAX was initiated upon dimerization. This may be an oversimplification because Gross et al. (1998) have shown that forced dimerization of BAX results in apoptosis and caspase-3 and caspase-9 activity but no detectable release of cytochrome

c. In addition, Saito et al. (2000) report the release of cytochrome c with the oligomerization of between 2 to 4 BAX molecules. However, it is not clear that changing the requirement for BAX signaling from 2 to 4 molecules has an operational impact on the behavior of our model (data not shown).

Finally, the inhibition of the mitochondrial pathway by Bcl-2 was modeled as Bcl-2 binding and inhibiting Bid rather than Bcl-2 binding and inhibiting Bax. While both these mechanisms of action could be correct, no direct evidence exists to distinguish them, and it is again unclear that these different representations would modify the behavior of the model (Igney and Krammer 2002).

Despite the wealth of information available on Fas apoptosis signaling pathways, little quantitative information is available on the rates of the chemical reactions that occur following Fas binding. Information provided by Donepudi et al. (2003), of the K_d, the off rate, for caspase-8 dimerization served as a basis for reaction rates in the DISC (Figure 2.1). We also included reaction rates that have been used successfully in the computational model of a similar apoptosis signaling pathway, the Tumor Necrosis Factor alpha pathway(Birgit Schoeberl Personal Communication). The original estimates were then optimized within an order of magnitude using simulated annealing in Gepasi. Gepasi is a program in the Systems Biology Workbench that provides various optimization algorithms as mentioned in chapter 3. Simulated annealing is an algorithm that searches for a global minimum. In this case, the algorithm searched for rate constants that would minimize the square error of the model behavior compared to the training data. We used experimentally obtained data from Jurkat cells on the rate of cleavage of key signaling proteins, caspase 8, caspase 3, Bid, as training data (Fei Hua, unpublished data). The resulting fit is shown in Figure 6.1.

Once the model fit the training data, it was used to systematically explore the features of Fas signaling that might result in type I versus type II behavior. We anticipated that the model, in its original configuration, would represent the Fas signaling behavior of a type II cell because the data used to fit the model was derived from Jurkat (type II) cells. An important prediction of this hypothesis was that Bcl-2 overexpression should decrease caspase-3 cleaveage. Surprisingly, the model behaved as would be expected of a type I cell, with the level of Bcl-2 expression making little difference in the level of caspase 3 cleavage (Figure 6.2). To test that whether this was an artifact of the model, and whether we had inaccurately represented the mitochondrial component of the type II Fas signaling pathway, we tested whether Bid cleavage could be observed and whether it was subject to the levels of Bcl-2 expression in the cell (Figure 6.3). As expected for a type II cell, the levels of Bcl-2 in the cells affected the amount of Bid cleaved, indicating that our model accurately represented the initiation of mitochondrial signaling as seen in type II cells. We are currently creating laboratory experimental systems to over and underexpress Bcl-2 in Jurkat cells and will use the results generated in these experiments to alter and further optimize our model. More generally, we will continue to obtain experimental data that will allow us to solidify the model by clarifying the architecture of the signaling pathways and providing better estimates of rate constants for individual reactions involved in these pathways. We will also examine specific components of Fas signaling pathways, such as the possible existence and role of a positive feedback loop involving caspase3, predicated by Slee et al. (1999).

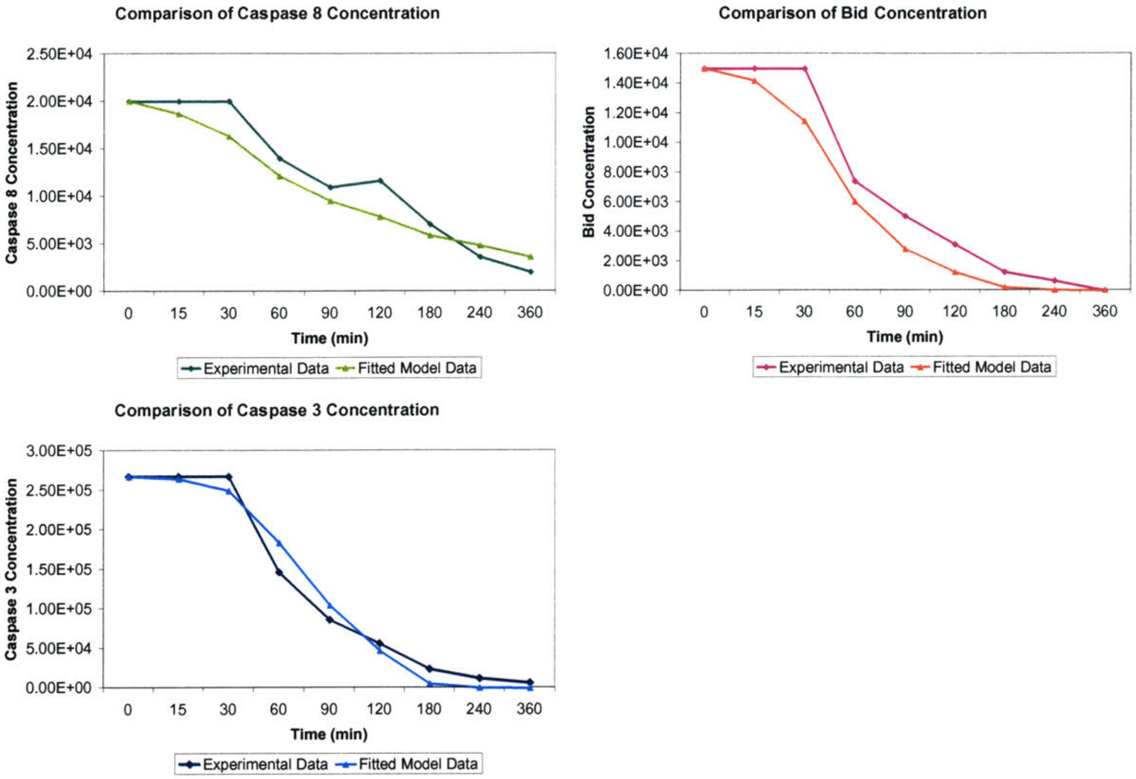


Figure 6.1 Results of Fitting the ODE Model to Experimental Data. Concentration is measured in terms of molecules per cell.

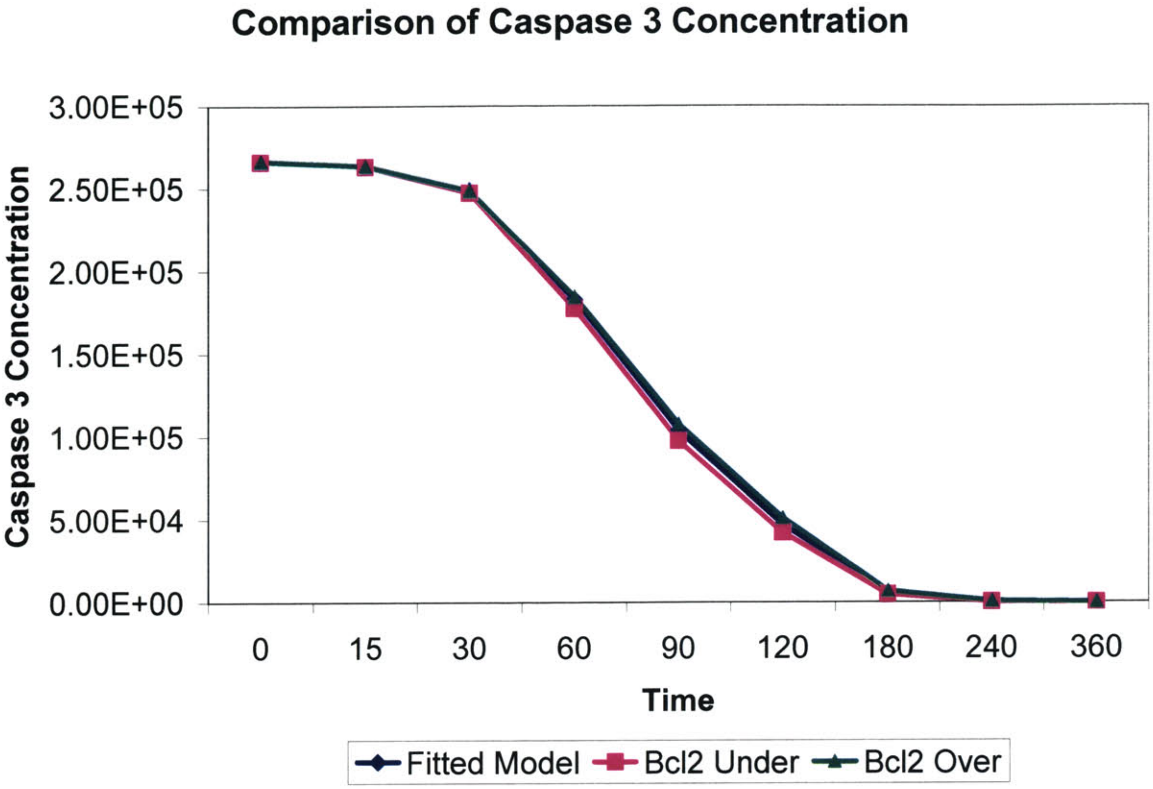


Figure 6.2 Results of Bcl-2 Underexpression and Overexpression in the Fitted ODE Model. The level of Bcl-2 makes little difference in the level of caspase-3 cleavage indicating the model behaves like a type I cell.

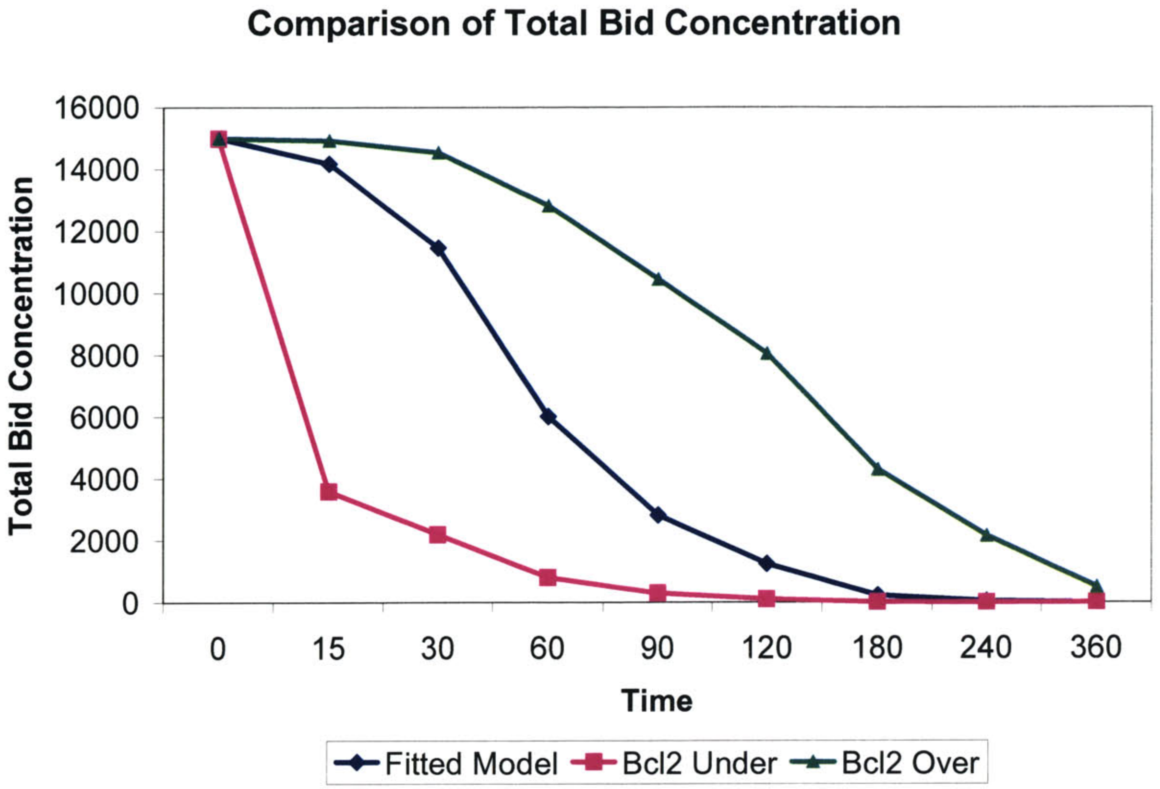


Figure 6.3 Results of Bcl-2 Underexpression and Overexpression in the Fitted ODE Model. The level of Bcl-2 changes the amount of Bid cleaved.

Chapter 7. Application of Apoptosis Assays to the Fas Signaling Pathways

Fas is often responsible for triggering signals that induce cells to undergo apoptosis. In T cells, signals from the TCR appear to modulate the function of Fas in complex and poorly understood ways. The optimized apoptosis assay was used to systematically screen for stimulation conditions where simultaneous engagement of both the TCR and Fas receptors would alter the sensitivity of T cells to apoptosis. .

Materials

rhsSuperFasL was purchased from Alexis Biochemicals, San Diego, CA. rhsSuperFasL is a soluble, recombinant form of human FasL crosslinked to imitate FasL in its trimerized form. Anti-CD3 purified mouse anti-human monoclonal antibody [555336] was purchased from BD Pharmingen, San Diego, CA.

Protocols

On day 1, cells were counted and diluted to a concentration less than 0.5 million cells/ml to ensure the cells would be healthy for the day of the experiment. On day 2, the cells were counted, spun down, and resuspended in C10 medium to a density of 1 million cells/ml. anti-CD3 and superFasL were used to stimulate the TCR and Fas respectively. These ligands were mixed as follows:

For soluble anti-CD3:
4ul of stock solution (10^6ng/ml) was added to 996ul of media for 1000ul of $4x10^3$ng/ml
9ul of $4x10^3$ng/ml solution was added to 891ul of media for 900ul of 40ng/ml
For FasL:
3ul of stock solution (10^5ng/ml) was added to 747ul of media for 750ul of 400ng/ml
8ul of 400ng/ml solution was added to 792ul of media for 800ul of 4ng/ml

Ligand solution and media were added to the wells of a 96-well flat bottom plate as indicated in Table 7.1. Finally, 100ul of cell solution was added to each well. The starting time was recorded and the plates were incubated at 37°c with 5% CO_2. After 24hrs, FACS analysis was performed to determine the relative levels of apoptosis.

Table 7.1 Ligand solutions and media added to the rows and columns of a 96 well plate

Fas (ng/ml) → anti-CD3	0 (50ul media)	1 (50ul 4ng/ml)	5 (2.5ul 400ng/ml and 47.5ul media)	10 (5ul 400ng/ml and 45ul media)	50 (25ul 400ng/ml and 25 media)
0 (50ul media)					
1 (5ul 40ng/ml and 45ul media)					
10 (50ul 40ng/ml)					
100 (5ul $4x10^3$ng/ml and 45ul PBS)					
1000 (50ul $4x10^3$ng/ml)					

Results and Discussion

The apoptosis assay can serve as a quick screen for conditions under which the interaction between Fas and TCR pathways may be greatest. The purpose of this segment of the thesis was to explore a range of TCR and Fas stimulation in order to find these interesting conditions.

In these experiments, both Jurkat and H9 cells were exposed to a wide concentrations of reagents to trigger the TCR (anti-CD3 antibody) or Fas receptor (superFasL). The TCR has a short intracellular domain and relies on associated molecules, CD3, to transmit signals. Anti-CD3 brings these molecules together thus mimicking physiological stimulation of TCR. Therefore the range of TCR stimulation was studied in terms of anti-CD3 concentration. Initially, these stimuli were applied individually. As shown in Figure 7.1a, anti-CD3 stimulation resulted in low but detectable levels of apoptosis for both cell types. The dose response of superFasL (Figure 7.1b) shows that Jurkat cells exhibit higher death at lower concentrations of superFasL than do H9 cells. This agrees with previous findings in our laboratory that Jurkat cells are more sensitive to Fas-mediated killing (Figure 2.2a). However, in these experiments we found that H9 cells eventually reach, at higher concentrations of FasL, levels of apoptosis close to those of Jurkat cells. This contradicts our previous findings (Figure 2.2a) where H9 cells were found to plateau at a lower level of apoptosis than Jurkat cells. The difference may be explained by the ten hour difference in the duration of the experiment, or by intrinsic variations in these experiments. Tests to distinguish these two possibilities are underway.

Because both Fas and TCR induce apoptosis in isolation, we predicted that adding both stimuli simultaneously would have an additive effect on apoptosis. As shown in Figure 7.2, we found that the effects of FasL dominate over the effects of anti-CD3 on apoptosis in general. Interestingly, H9 cells stimulated with anti-CD3 concentrations greater than 100ng/ml showed a small but statistically significant reduction in apoptosis for Fas concentrations greater than 10ng/ml. This protective effect is more readily observed when the background level of apoptosis induced by FasL alone is subtracted out from the level of apoptosis induced by both stimuli. Figure 7.3a also shows that anti-CD3 has a small protective effect in Jurkat cells for concentrations of FasL up to 5ng/ml. Interestingly, the dose response curve of Jurkat cells to

FasL plateaus soon after 5ng/ml. The dose response curve of H9s to FasL does not plateau in the range of the FasL concentrations studied. Notably, the protective effect of anti-CD3 in H9 cells also continues to increase. This suggests that anti-CD3 may provide some protection against Fas-mediated killing up to the concentration at which FasL reaches maximal apoptosis. This effect was seen in each triplicate as well as in the average of the triplicates (data not shown). However, these experiments need to be repeated.

In conclusion, these initial experiments indicate that Jurkat and H9 cells both show a low level of apoptosis in response to anti-CD3 but have differing sensitivities to superFasL. When both signals are provided simultaneously, the effect of FasL dominates the overall shape of the apoptosis curves. However, at levels of FasL that do not induce maximal apoptosis, costimulation with anti-CD3 can provide a small protective effect. This protective effect could not have been predicted from the response to each ligand separately and shows the utility of this apoptosis assay for an initial screen for interesting behaviors of signaling pathways using a systems biology approach. These experiments will be repeated to confirm the results obtained, and the intracellular signals that correlate with cellular outcomes identified and studied.

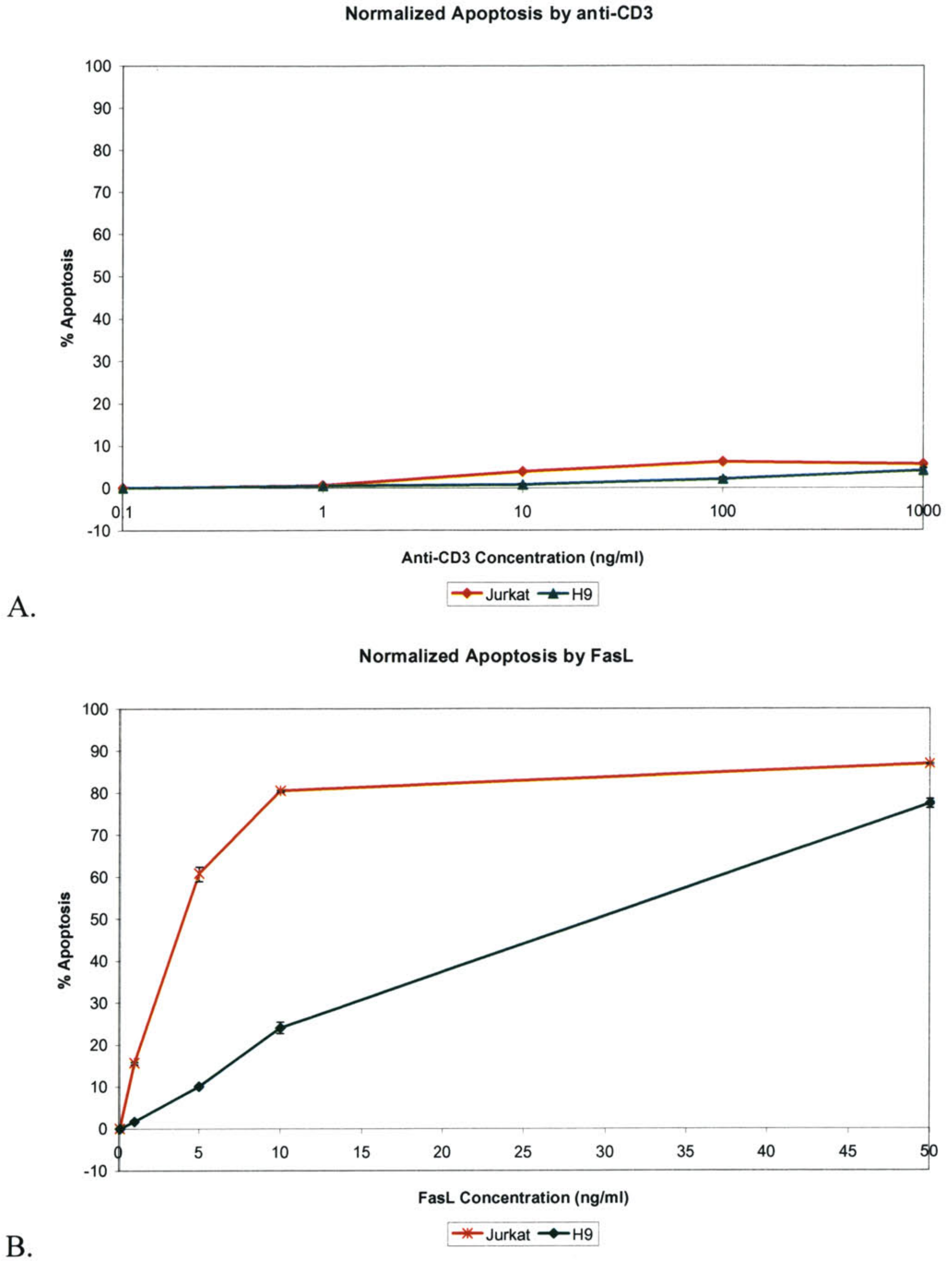


Figure 7.1 Dose response curves of Jurkat and H9 cells to anti-CD3 and superFasL. A) Jurkat and H9 cells both exhibit low but positive levels of apoptosis after anti-CD3 stimulation. B) FasL induces higher levels of apoptosis than anti-CD3 and Jurkat cells exhibit more death at lower concentrations than H9 cells.

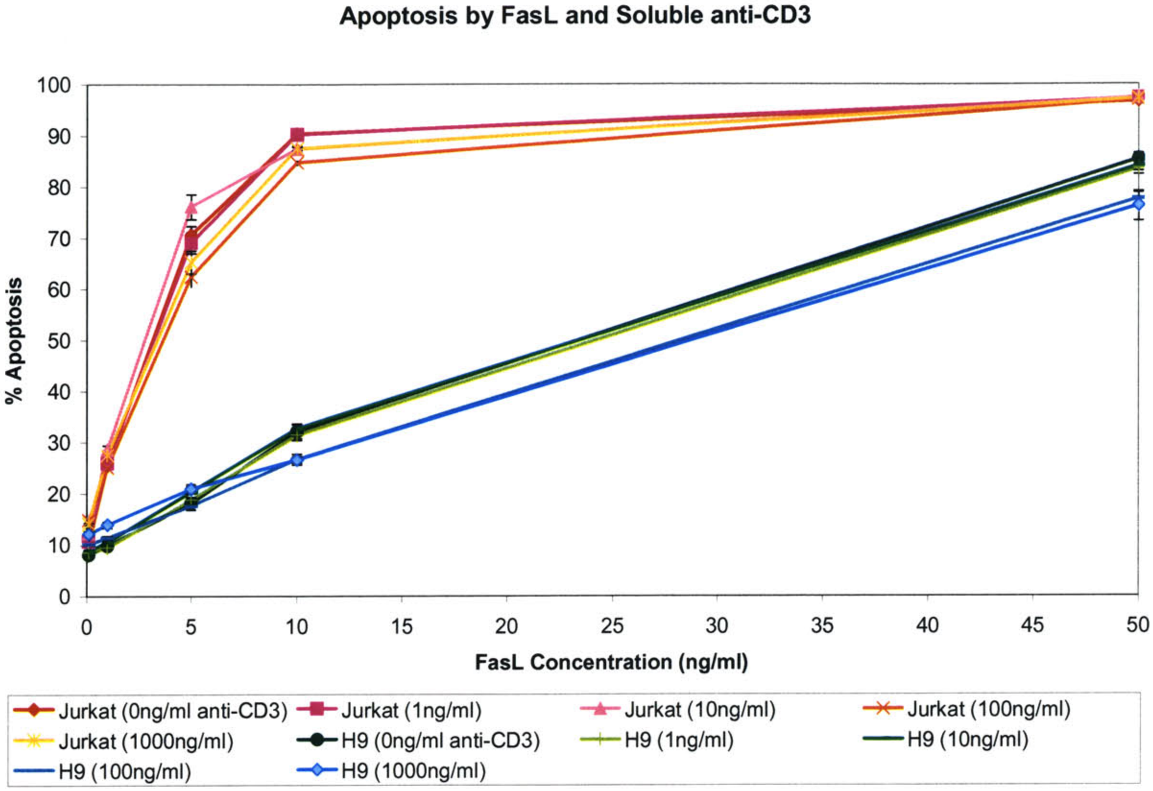


Figure 7.2 Costimulation of Jurkat and H9 cells with soluble anti-CD3 and superFasL. Each line represents a different concentration of anti-CD3. The influence of FasL dominates over any effect of anti-CD3.

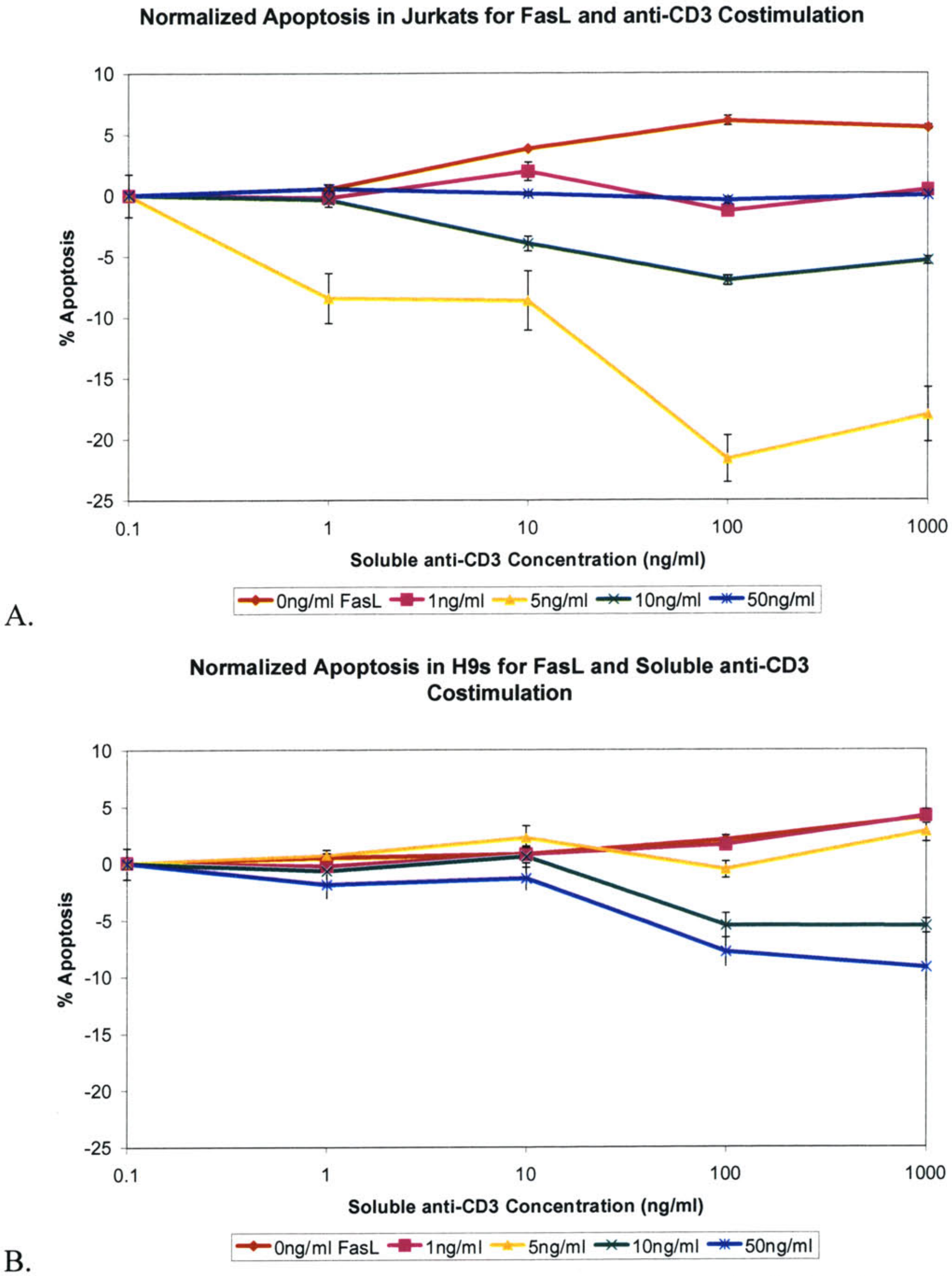


Figure 7.3 Subtle effects of costimulation with soluble anti-CD3 and superFasL. The dose response for FasL is subtracted out to reveal the subtle effects of the costimulation. Each line represents a different concentration of FasL. A) Jurkat cells show a reduction in apoptosis for Fas concentrations up to 5ng/ml. B) H9 cells show a slight but significant reduction in apoptosis for FasL concentrations greater than 10ng/ml and anti-CD3 concentrations greater than 100ng/ml.

Chapter 8. Application of the RNAi Program to the Fas Pathways

As explained in chapter 5, an RNAi sequence prediction program was created based on current knowledge about criteria important for a functional siRNA sequence. The purpose of this section of the thesis was to test the performance of this RNAi program at detecting successful shRNA sequences in the context of the Fas signaling pathway, with the ultimate goal of creating a suite of reagents to modulate gene expression and test predictions about the behavior of Fas signaling made by our computational model of this pathway.

Results and Discussion

The shRNA prediction program was used to predict functional sequences for three components of the Fas signaling pathway for which experimental data on the performance of shRNAs was available, namely mouse caspase-8, caspase-3, and caspase-9. The program took about three hours to run all three genes. The output of the program was sorted based first on the presence of a termination sequence, second on the 5' versus 3' antisense free energy difference, and third on the 9 to 14 nucleotide free energy. The results are shown in Tables 8.1, 8.2, and 8.3 respectively. Christopher Dillon generously provided the results for sequences already tested in the Van Parijs laboratory. Unsuccessful sequences are highlighted in red, successful sequences are highlighted in yellow.

All of the successful RNAi sequences exhibited the combination of characteristics considered important by the literature. In particular, the sequences had negative 5 versus 3 prime antisense free energy differences. This criteria was not met for 4 of the 5 unsuccessful caspase-8 sequences (Table 8.1), and 2 of the 4 unsuccessful caspase-9 sequences (Table 8.3). 2 of the 5 unsuccessful caspase-8 sequences also appeared to show high homology to other genes (Table 8.1). Upon further inspection, these homologies were actually against variants of the original gene. This result underscores the potential advantage of performing sequence comparisons in databases with fewer redundancies (see chapter 5).

The reason for the failure of the sequences with negative 5' versus 3' antisense free energy is much more difficult to determine. The one unsuccessful caspase-8 sequence with a negative 5' versus 3' antisense free energy (Table 8.1) has a slightly high Blast result with 19 out of 23 nucleotides identical to a gene of unknown function on chromosome 12 (caspase-8 is located on chromosome 1). While this might be predicted to increase nonspecific knockdown of genes, it should not necessarily decrease specific knockdown. All the criteria for the two unsuccessful caspase-9 sequences are within the range suggested by literature (Table 8.2). Thus, successful sequences met all the criteria but meeting all the criteria did not guarantee successful sequences. This suggests that the current set of selection criteria used does not provide sufficient predictive power in selecting shRNAs. As a consequence, it remains necessary to test at least four or five sequences to have a reasonable probability of identifying a functional one.

Future work on this program will involve improving the subroutine for Blast by performing sequence comparisons in a database with fewer redundancies. In addition, the subroutine for self alignments may be improved by a subroutine that considers gapped alignments. It is likely that an effective prediction program for selecting shRNAs will only be generated once more information becomes available about the features of successful sequences. It might be possible to develop a learning algorithm to identify such features.

Table 8.1 Sequences predicted for Caspase-8. The output of the program was sorted based first on the presence of a termination sequence, second on the 5 versus 3 prime antisense free energy difference, and third on the 9 to 14 nucleotide free energy. There were no successful sequences for caspase-8. Unsuccesful sequences are highlighted in red.

GenBank ID	Annot	AAGN20	Position	GC cont	Self align	Blast ide	Blast title	5 vs 3	9 to 14	SNP over	Termination
33859519	NM_0	AAGCGC	1179	47.826	GCGC	(17, 17)	>gb|AC13	-4.1	8.2	FALSE	FALSE
33859519	NM_0	AAGGGA	846	52.174	AAGG	(17, 17)	>gb|AC12	-2.4	8.2	FALSE	FALSE
33859519	NM_0	AAGGCA	440	39.13	TTTC	(22, 22)	>emb|AJ	-2.4	6.4	FALSE	FALSE
33859519	NM_0	AAGGCT	1169	56.522	GCGC	(16, 16)	>gb|AC01	-1.8	8.5	FALSE	FALSE
33859519	NM_0	AAGCTC	1696	47.826	AGCT	(21, 21)	>emb|AJ	-1.6	9.7	FALSE	FALSE
33859519	NM_0	AAGGCC	1015	56.522	CCCGGG	(16, 16)	>gb|AC12	-1.5	6.7	FALSE	FALSE
33859519	NM_0	AAGCCC	178	69.565	GAGGCC	(17, 17)	>gb|AF12	-1.5	9.7	FALSE	FALSE
33859519	NM_0	AAGGAA	1061	39.13	ACAG	(21, 21)	>gb|AF06	-1.5	6.1	FALSE	FALSE
33859519	NM_0	AAGTGG	1427	39.13	GTG	(17, 17)	>gb|AC12	-1.2	6.9	FALSE	FALSE
33859519	NM_0	AAGACC	323	56.522	GGC	(19, 19)	>emb|AL5	-1.2	9.7	FALSE	FALSE
33859519	NM_0	AAGGGC	424	52.174	TTG	(18, 18)	>gb|AC07	-0.9	8.5	FALSE	FALSE
33859519	NM_0	AAGACT	1199	43.478	AAG	(21, 21)	>gb|AC00	-0.7	6.6	TRUE	FALSE
33859519	NM_0	AAGTTC	340	43.478	AAG	(16, 16)	>gb|AC13	-0.7	8.5	FALSE	FALSE
33859519	NM_0	AAGAAC	964	47.826	AG	(18, 18)	>gb|AC00	-0.6	9.9	FALSE	FALSE
33859519	NM_0	AAGACC	1093	34.783	CCTT	(17, 17)	>gb|AC12	-0.4	8.2	FALSE	FALSE
33859519	NM_0	AAGGAG	1264	52.174	GGAGGC	(16, 16)	>gb|AC13	-0.3	8.1	FALSE	FALSE
33859519	NM_0	AAGAGA	915	43.478	GTGA	(17, 17)	>gb|AC11	-0.3	6.4	FALSE	FALSE
33859519	NM_0	AAGCAG	1660	56.522	AAG	(19, 19)	>gb|AC12	-0.3	8.2	FALSE	FALSE
33859519	NM_0	AAGGAA	842	52.174	AGG	(19, 20)	>gb|AC13	-0.3	6.7	FALSE	FALSE
33859519	NM_0	AAGCAG	373	56.522	CATC	(18, 18)	>gb|BC05	-0.1	7.8	FALSE	FALSE
33859519	NM_0	AAGATC	793	34.783	GATC	(16, 16)	>gb|AC12	0	6	FALSE	FALSE
33859519	NM_0	AAGGAA	1361	39.13	GGAAGT	(17, 17)	>gb|AC12	####	6.4	FALSE	FALSE
33859519	NM_0	AAGATG	1572	43.478	TGTC	(18, 18)	>emb|AL6	0.1	7	FALSE	FALSE
33859519	NM_0	AAGAAT	270	39.13	TGGA	(17, 17)	>gb|AC12	0.1	8.5	FALSE	FALSE
33859519	NM_0	AAGAAC	1192	39.13	AAG	(17, 17)	>ref|NM_	0.2	6.4	FALSE	FALSE
33859519	NM_0	AAGGAG	1583	39.13	GATG	(22, 22)	>dbj|AK07	0.2	6.6	FALSE	FALSE
33859519	NM_0	AAGTTT	950	26.087	GTT	(17, 17)	>gb|AC09	0.3	7.2	FALSE	FALSE
33859519	NM_0	AAGGGA	1654	56.522	GCA	(20, 20)	>gb|AC13	0.3	6.6	FALSE	FALSE
33859519	NM_0	AAGTGA	605	43.478	AATT	(17, 17)	>emb|AL5	0.3	6.7	FALSE	FALSE
33859519	NM_0	AAGAGC	778	56.522	GATC	(18, 18)	>gb|U342	0.4	9.7	FALSE	FALSE
33859519	NM_0	AAGAAG	602	43.478	AATT	(17, 17)	>emb|AL5	0.5	9.9	FALSE	FALSE
33859519	NM_0	AAGGGT	1237	52.174	CGT	(15, 15)	>ref|NM_	0.5	5.5	FALSE	FALSE
33859519	NM_0	AAGACA	923	47.826	GTGA	(17, 17)	>gb|AF06	0.6	6.4	FALSE	FALSE
33859519	NM_0	AAGAGG	1079	43.478	AGAG	(22, 22)	>gb|AF06	0.6	6.4	FALSE	FALSE
33859519	NM_0	AAGATC	810	39.13	GATC	(19, 19)	>gb|AC12	0.7	6.7	FALSE	FALSE
33859519	NM_0	AAGATG	674	43.478	AAG	(19, 20)	>ref|NM_	0.7	8.2	FALSE	FALSE
33859519	NM_0	AAGAAG	370	52.174	TCGA	(20, 20)	>gb|BC05	1.2	6.1	FALSE	FALSE
33859519	NM_0	AAGCTG	667	47.826	AGCT	(18, 18)	>ref|NM_	1.2	6.6	FALSE	FALSE
33859519	NM_0	AAGACG	1637	52.174	CG	(18, 18)	>gb|AC12	1.3	9.4	FALSE	FALSE
33859519	NM_0	AAGAAC	308	52.174	ACTG	(17, 17)	>emb|AL6	1.5	7.9	FALSE	FALSE
33859519	NM_0	AAGCAC	816	47.826	GC	(21, 21)	>emb|AJ	1.5	6.7	FALSE	FALSE
33859519	NM_0	AAGCCA	184	73.913	GAGGCC	(16, 16)	>emb|AL9	1.7	9.4	FALSE	FALSE
...	...	...	...	...	...	...	...	...	...	...	...
33859519	NM_0	AAGTAA	1365	43.478	AAGT	(17, 17)	>gb|AC12	2.1	8.2	FALSE	FALSE
33859519	NM_0	AAGAAG	1693	47.826	AAGAAG	(20, 20)	>gb|AC08	2.1	7	FALSE	FALSE

Table 8.2 Sequences predicted for Caspase-3. The successful sequence is highlighted in yellow.

GenBank I	Annot	AAGN20	Position	GC cont	Self align	Blast id	Blast title	5 vs 3	9 to 14	SNP over	Terminati
6753283	NM_0	AAGCCG	523	39.13	AAG	(18, 18)	>gb|U881	-4.5	6.7	FALSE	FALSE
6753283	NM_0	AAGGGC	434	39.13	ATTT	(18, 18)	>emb|AL6	-3.7	6.3	FALSE	FALSE
6753283	NM_0	AAGCTG	733	52.174	AGCT	(21, 21)	>gb|U637	-2.4	11.7	FALSE	FALSE
6753283	NM_0	AAGGTG	787	52.174	GAATTC	(15, 15)	>gb|AC12	-1.8	9.6	FALSE	FALSE
6753283	NM_0	AAGCCA	420	52.174	CCATGG	(19, 19)	>dbj|AK03	-1.8	6.9	FALSE	FALSE
6753283	NM_0	AAGCTG	953	47.826	AGCT	(17, 17)	>gb|AC10	-1.3	9.4	FALSE	FALSE
6753283	NM_0	AAGGAG	393	47.826	AGCT	(19, 19)	>gb|AC13	-1.2	7.3	FALSE	FALSE

Table 8.3 Sequences predicted for Caspase-9

GenBank I	Annot	AAGN20	Position	GC cont	Self align	Blast id	Blast title	5 vs 3	9 to 14	SNP over	Terminatio
31560478	NM_0	AAGGCC	1459	47.826	GGCC	(17, 17)	>gb|AC12	-4.8	6.1	FALSE	FALSE
31560478	NM_0	AAGGGC	1994	39.13	AAGG	(17, 17)	>gb|AC12	-4.2	7	FALSE	FALSE
31560478	NM_0	AAGGCA	1057	56.522	AAGG	(18, 18)	>emb|AL7	-3.6	8.5	FALSE	FALSE
31560478	NM_0	AAGGCC	1108	65.217	CCAAGG	(20, 21)	>gb|AC12	-3	12.4	FALSE	FALSE
31560478	NM_0	AAGCGG	302	65.217	GGC	(16, 16)	>emb|BX	-2.9	8.5	FALSE	FALSE
31560478	NM_0	AAGCCC	978	47.826	AGCT	(18, 18)	>emb|AL6	-2.8	6.7	FALSE	FALSE
31560478	NM_0	AAGGCC	1115	60.87	CCAGCT	(18, 18)	>ref|NM_	-2.8	9.1	FALSE	FALSE
31560478	NM_0	AAGGCA	277	56.522	AAG	(17, 17)	>gb|AC13	-2.7	9.7	FALSE	FALSE
31560478	NM_0	AAGCAG	321	47.826	GGATCC	(19, 19)	>gb|AC12	-2.1	8.2	FALSE	FALSE
31560478	NM_0	AAGCCA	495	52.174	AGA	(22, 23)	>gb|AC13	-1.5	6.4	FALSE	FALSE
31560478	NM_0	AAGCAG	313	56.522	GGATCC	(16, 16)	>gb|AC13	-1.5	8.2	FALSE	FALSE
31560478	NM_0	AAGGGA	1327	43.478	AAG	(17, 17)	>gb|AC12	-1.2	6.7	FALSE	FALSE
31560478	NM_0	AAGACC	1279	56.522	CTGCAG	(17, 17)	>gb|AC02	-0.9	7.6	FALSE	FALSE
31560478	NM_0	AAGATC	576	43.478	GATC	(18, 18)	>dbj|AP0	-0.7	7.9	FALSE	FALSE
31560478	NM_0	AAGCTG	1751	52.174	AGCT	(18, 18)	>gb|AC12	-0.3	6.4	FALSE	FALSE
31560478	NM_0	AAGTTT	1145	52.174	AA	(20, 20)	>gb|AC00	#####	8.5	FALSE	FALSE
31560478	NM_0	AAGGAG	396	47.826	AGCT	(18, 18)	>emb|AL6	0.3	6.1	FALSE	FALSE
31560478	NM_0	AAGTGG	1220	52.174	GTAC	(16, 16)	>gb|BC01	0.5	7.9	FALSE	FALSE
31560478	NM_0	AAGACC	1584	52.174	CCGG	(22, 23)	>dbj|AK08	0.5	8.2	FALSE	FALSE
31560478	NM_0	AAGAAC	756	47.826	AG	(21, 21)	>emb|AL9	0.6	6.4	FALSE	FALSE
31560478	NM_0	AAGTCC	1934	60.87	CCAGCT	(20, 21)	>gb|AC00	0.6	11.2	FALSE	FALSE
31560478	NM_0	AAGTTA	1984	39.13	AGAA	(16, 16)	>gb|AC00	0.8	8.2	FALSE	FALSE
31560478	NM_0	AAGAAG	1581	47.826	CCGG	(22, 23)	>dbj|AK08	1.4	7.6	FALSE	FALSE
31560478	NM_0	AAGTCG	94	65.217	GAGCTC	(16, 16)	>gb|AF18	1.5	7.9	FALSE	FALSE
31560478	NM_0	AAGAAC	1662	52.174	AGAA	(18, 18)	>gb|AC09	1.5	8.2	FALSE	FALSE
31560478	NM_0	AAGCTG	414	65.217	AGCT	(22, 23)	>dbj|AB01	1.5	9.6	FALSE	FALSE
31560478	NM_0	AAGAGC	472	65.217	AGAGCT	(17, 17)	>gb|BC01	1.5	10.9	FALSE	FALSE
31560478	NM_0	AAGCTC	984	52.174	CAGGCC	(19, 19)	>gb|AC01	1.8	6.9	FALSE	FALSE
31560478	NM_0	AAGCTC	1434	56.522	AGCTC	(18, 18)	>gb|AC12	1.8	9.7	FALSE	FALSE

Chapter 9. Summary of Contributions

In summary this thesis has created or explored three tools for a systems biology approach to studying apoptosis signaling in T cells. We have identified JDesigner as the optimum modeling program currently available and documented the current state of knowledge about Fas by creating an ordinary differential equation model. Using this model we have explored type I versus type II behavior and discovered that a type I model can fit the data from a type II cell (Jurkat cells). We have also established an optimum protocol for performing apoptosis assays with H9 and Jurkat cells. Using this protocol, we have found evidence to suggest interaction between the TCR and Fas signaling pathways. Finally, we have created a program for function siRNA sequence prediction. We tested this program and discovered that the currently known criteria for selecting siRNA sequences are not powerful enough to predict functional siRNA sequences though all functional sequences follow these criteria.

Future work will involve laboratory experiments to gather more data to optimize the computational Fas model, repetition of the apoptosis assays to confirm our initial findings, and rewriting of the subroutines in the siRNA program for faster Blast searches and more accurate self alignments.

References

Allan LA, et al. Nat Cell Biol. 2003 July; 5(7):647-54

Ambion. siRNA Target Finder. http://www.ambion.com/techlib/misc/siRNA_finder.html

Bhalla US. Methods Enzymol. 2002; 345: 3-23

Biocomputing at Whitehead Institute. Hairpin siRNA selection Program. http://jura.wi.mit.edu/pubint/http://iona.wi.mit.edu/siRNAext/

Biopython. http://biopython.org

California Institute of Technology. Systems Biology Workbench. http://www.cds.caltech.edu/erato

Chang HY, Yang X. Microbiol Mol Biol Rev. 2000 Dec; 64(4): 821-46

Denli AM, Hannon GJ. Trends Biochem Sci. 2003 Apr; 28(4): 196-201

Donepudi M, et al. Mol Cell. 2003 Feb; 11(2):543-9

Dykxhoorn DM, et al. Nat Rev Mol Cell Biol. 2003 Jun; 4(6): 457-67

Engles IH, et al. Oncogene 2000 Sep 21; 19(40):4563-73

Goldsby RA, et al. Immunology: Fifth Edition. W.H. Freeman and Company, New York 2003; 239-41

Gross A, et al. EMBO J. 1998 Jul 15; 17(14): 3878-3885

Hoffmann A, et al. Science 2002 Nov 8; 298(5596): 1241-5

Holler N, et al. Nat Immunol. 2000 Dec; 1(6): 489-95

Holler N, et al. Mol Cell Biol. 2003 Feb; 23(4):1428-40

Holmstrom TH, et al. EMBO J. 2000 Oct 16; 19(20):5418-28

Igney FH, Krammer PH. Nat Rev Cancer 2002 Apr; 2: 277-88

Jackson CE, et al. Am J Hum Genet. 1999 Apr; 64(4):1002-14

Kennedy et al. J Exp Med. 1999 Dec 20; 190(12):1891-6

Kibby MR. Nature 1969 Apr 19; 222(190): 298-9

Khvorova A, et al. Cell 2003 Oct 17; 115(2):209-16

Khvorova A, et al. Cell 2003 Nov 14; 115(4):505

Luo X, et al. Cell 1998 Aug 21; 94:481-90

MathWorks. Matlab. http://www.mathworks.com/products/matlab/

McManus MT, Sharp PA. Nat Rev Genet. 2002 Oct; 3(10): 737-46

Mendes P. Trends Biochem Sci. 1997 Sep; 22(9): 361-3

Mendes P, Kell D. Bioinformatics. 1998; 14(10): 869-83

Peter ME, Krammer PH. Cell Death Differ. 2003 Jan; 10(1):26-35

Pusch O, et al. Nucleic Acids Res. 2003 Nov 15; 31(22):64444-9

Ravi Sachidanandam Laboratory at Cold Spring Harbor Laboratory. RNAi OligoRetriever. http://katahdin.cshl.org:9331/RNAi/html/rnai.html

Rubinson DA, et al. Nat Genet. 2003 Mar; 33(3): 401-6

Saito M, et al. Nat Cell Biol. 2000 Aug; 2(8):553-555

Sauro H. JDesigner. http://www.cds.caltech.edu/hsauro/JDesigner.htm

Scaffidi C, et al. EMBO J. 1998 Mar 16; 17(6):1675-87

Scaffidi C, et al. J Biol Chem. 1999 Aug 6; 274(32):22532-8

Schoeberl B, et al. Nat Biotechnol. 2002 Apr; 20(4): 370-5

Shi Y. Structure (Camb). 2002 Mar; 10(3): 285-8

Siegel RM, et al. Science 2000 Jun 30; 288(5475):2354-7

Slee EA, et al. J Cell Biol. 1999 Jan 25; 144(2):281-92

Sun XM, et al. J Biol Chem. 2002 Mar 29; 277(13):11345-51

Wei MC, et al. Genes Dev. 2000 Aug 15; 14(16):2060-71

Acknowledgments

Last but not least I would like to acknowledge the following people:

Luk Van Parijs for his enthusiastic encouragement and advice throughout the Fas project.

Fei Hua for her help with every aspect of the Fas project from modeling to experimentation.

Herbert Sauro for his quick response to problems encountered with JDesigner.

Melissa Lambeth for her help with setting up and performing apoptosis assays.

Lucia Wille for her help in understanding TCR and for reviewing this thesis.

Christopher Dillon for providing siRNA sequence data and walking me back from lab almost every night.

Birgit Schoeberl for providing data for the computational Fas model.

Doug Lauffenburger for guidance in exploring the interaction between Fas and TCR.

LVP lab for providing constructive criticism.

Appendix A
Code for RNAi sequence prediction program

```
# Rayka Yokoo
# Program to find information important for creating RNAi sequences
# Output from this program can be fed into Excel and sorted for the
most
# important information
# A macro in Exel uses the A* algorithm to find the optimum non
overlapping
# set of RNA sequences
# Input--File, Sequences, in the same folder as this program with
# each GenBank ID and annotation on a separate line
# Output--Tab delimited file, Output, of AAGN20 sequences found in the
DNA for the
# GenBank IDs given along with extra information.
# Column 1-GenBank ID
# Column 2-Annotation
# Column 3-AAGN20 sequence found
# Column 4-Position
# Column 5-GC content
# Column 6-Self alignments
# Column 7-Blast similarity (identity)
# Column 8-Blast similarity (title)
# Column 9-5 antisense free energy
# Column 10-9 to 14 nt region (antisense) free energy
# Column 11-SNPs
# Column 12-Termination Seq

from Bio import SeqUtils
from Bio.Blast import NCBIWWW
from Bio import *
import difflib

# Main function that calls all others
# Requires that file, Sequences.txt, is in same folder as this program
# Output--File called Output
def main():
    file = open("D:/Rayka/Sequences.txt")
    gids = list()

    # Create a dictionary of GIDs to user annotation
    gidToAnnot = dict()
    for line in file:
        gidAndAnnot = line.split(" ",1) # Line should be GID followed
by annotation
        gid = gidAndAnnot[0]
        annotWn = gidAndAnnot[1]
        annot = annotWn.strip("\n") # Strip newline character
        gids.append(gid)
        gidToAnnot[gid] = annot

    # Find GenBank entry for GIDs
    # Create a dictionary of gids to seq, organism, and SNPs
    gidToSeqOrgSNP = dict()
    ncbi_dict = GenBank.NCBIDictionary(parser =
GenBank.FeatureParser())
    for gid in gids:
        print gid
        seqOrgSNP = list()
        # Sequence
        entry = ncbi_dict[gid]
        seq = entry.seq
        seqOrgSNP.append(seq)
        features = entry.features
        SNPs = list()
        for feature in features:
            # Organism
            if feature.type == "source":
                 sourceInfo = feature.qualifiers
                 orgInfo = sourceInfo['organism']
                 org = orgInfo[0]
            # SNPs
            if feature.type == "variation":
                SNPs.append(feature.location)
        seqOrgSNP.append(org)
        seqOrgSNP.append(SNPs)
        gidToSeqOrgSNP[gid] = seqOrgSNP


    # Prepare to write to Output
    out_file = open("Output", "w")
    out_file.write("GenBank ID\tAnnotation\tAAGN20
sequence\tPosition\tGC content\tSelf alignments\tBlast identity\tBlast
title\t5 vs 3 prime free energy\t9 to 14 nt region (antisense) free
energy\tSNP overlap\tTermination Sequence\n")

    # Create a dictionary of gids to a dictionary of AAGN20 seqs
    # Dictionary of AAGN20 strings maps AAGN20 seqs to list of
features
    # The list of features is in the same order as the columns
(Position,
    # GCContent, etc)
    gidToAAGDict = dict()
    for gid in gids:
        seqOrgSNP = gidToSeqOrgSNP[gid]
        seq = seqOrgSNP[0]
        print seq
        org = seqOrgSNP[1]
        print org
        SNPs = seqOrgSNP[2]
        print SNPs
        AAGSeqToFeatures = dict()

        # Find AAGN20 Sequences
        stringSeq = seq.tostring()
        AAGPos = stringSeq.find("AAG", 0)
        while AAGPos > -1:
            AAGEnd = AAGPos + 23
            AAGSeq = seq[AAGPos:AAGEnd]
            print AAGSeq
            # Create list of features
            AAGFeatures = list()
            # Annotation
            annot = gidToAnnot[gid]
            # Position
            AAGFeatures.append(AAGPos+1)
            # GC Content
            GCContent = SeqUtils.GC(AAGSeq)
            print GCContent
            AAGFeatures.append(GCContent)
            # Self alignments
            max_align = self_align(AAGSeq)
            print max_align
            AAGFeatures.append(max_align)
            # Blast similarity--NEED TO MAKE THIS FASTER. CONSIDER
LOCAL BLAST
            blastInfo = blast_sim(AAGSeq.tostring(), org)
            print blastInfo
            AAGFeatures.append(blastInfo)
            # 5 to 3prime free energy
            fiveTothree_energy = fiveTothree(AAGSeq)
            print fiveTothree_energy
            AAGFeatures.append(fiveTothree_energy)
            # 9 to 14 nt region (antisense) free energy
            nineTofourteen_energy = nineTofourteen(AAGSeq)
            print nineTofourteen_energy
            AAGFeatures.append(nineTofourteen_energy)
            # SNPs
            SNP_overlap = overlap(SNPs, AAGPos+1, AAGEnd+1)
            print SNP_overlap
            AAGFeatures.append(SNP_overlap)
            # Termination seq
            termination = termination_seq(AAGSeq)
            print termination
            AAGFeatures.append(termination)

            # Write to Output

out_file.write(gid+"\t"+annot+"\t"+AAGSeq.tostring()+"\t"+repr(AAGPos+
1)+"\t"+repr(GCContent)+"\t"+max_align.tostring()+"\t"+repr(blastInfo[
0])+"\t"+blastInfo[1]+"\t"+repr(fiveTothree_energy)+"\t"+repr(nineTofo
urteen_energy)+"\t"+repr(SNP_overlap)+"\t"+repr(termination)+"\n")

            # Prepare for next loop
            AAGSeqToFeatures[AAGSeq] = AAGFeatures
            AAGPos = stringSeq.find("AAG", AAGPos+3)
        gidToAAGDict[gid] = AAGSeqToFeatures
    out_file.close()
    print gidToAAGDict

# Column 6-Self alignments
# Find longest self alignment by finding longest match between
# given seq and antiparallel strand
# Returns--substring that self aligns in seq
def self_align(seq):
    antiparallel = SeqUtils.antiparallel(seq)
    matcher = difflib.SequenceMatcher(None, seq.tostring(),
antiparallel)
    matchIndices = matcher.find_longest_match(0, len(seq), 0,
len(antiparallel))
    alignBegin = matchIndices[0]
    alignEnd = alignBegin + matchIndices[2]
    return seq[alignBegin: alignEnd]

# Column 7-Blast similarity
# Input--sequence to be Blasted and organizm from which the sequence
originates
# Returns--Results of Blast in the form of a tuple with
# highest (number of identities/total aligned) and title of alignment
# Ignores sequences in which number of identities equals
# length of original sequence because these are most likely self
identities
def blast_sim(seq, org):
    # Word size of 7, Filter off and Expect Value 1000 suggested by
    # Blast website for searching short nucleotide sequences
    # CONSIDER USING UNIGENE
    blast_results = NCBIWWW.blast('blastn', 'nr', seq,
entrez_query=org, filter='off', expect='1000', word_size='7')
    blast_parser = NCBIWWW.BlastParser()
    blast_record = blast_parser.parse(blast_results)
    alignments = blast_record.alignments
    blastSimInfo = list()
    simInfoFound = False
    for alignment in alignments:
        hsps = alignment.hsps
        # First hsp usually has the most significance
        hsp = hsps[0]
        # Identity is tuple (number of identities/total aligned)
        # Ignore sequences with number of identities equal to
        # length of original sequence becuase these are most likely
self
        identity = hsp.identities
        if identity[0] != len(seq):
            blastSimInfo.append(identity)
            alignmentWnt = alignment.title
            alignmentWt = alignmentWnt.replace("\n", " ")
            alignment = alignmentWt.replace("\t", " ")
            blastSimInfo.append(alignment)
            simInfoFound = True
            break
    if not simInfoFound:
        blastSimInfo.append([0,0])
        blastSimInfo.append("None")
    return blastSimInfo

# Column 9-5 vs 3prime antisense free energy
# Free energy calculated based on Khvorova et al. Cell Vol. 115 209-
216 Oct. 2003
# Input--Sequence in all capitals
# Returns--difference in free energy between the 5 and 3 prime
antisense strand of
# the given sequence
def fiveTothree(seq):
    antiparallel = SeqUtils.antiparallel(seq)
    fivePrimeEnergy = free_energy_help(antiparallel[0:5])
    threePrimeEnergy = free_energy_help(antiparallel[-6:-1])
    return fivePrimeEnergy-threePrimeEnergy

# Column 10-9 to 14 nt region (antisense) free energy
# Free energy calculated based on Khvorova et al. Cell Vol. 115 209-
216 Oct. 2003
# Input--Sequence in all capitals
# Returns--free energy for the 9 to 14 nt region antisense of the
given sequence
def nineTofourteen(seq):
    antiparallel = SeqUtils.antiparallel(seq)
    nineTofourteen_energy = free_energy_help(antiparallel[9:14])
    return nineTofourteen_energy

# Define dictionary of free energy values
# Matrix based on Erratum for Khvorova et al. Cell Vol. 115 209-216
Oct. 2003
# Key is first nucleotide
# Values are in the order of A, C, G, T
free_energy_val = dict()
free_energy_val['A'] = [1.9, 1.3, 1.6, 1.5]
free_energy_val['C'] = [1.9, 3.1, 3.6, 1.6]
free_energy_val['G'] = [1.6, 3.1, 3.1, 1.3]
free_energy_val['T'] = [1.0, 1.6, 1.9, 1.9]

# Help routine for free_energy
# Computes the free energy of seq
def free_energy_help(seq):
    i = 0
    j = 1
    free_energy = 0
    while j < len(seq):
        firstChar = seq[i]
        secondChar = seq[j]
        free_energy_list = free_energy_val[firstChar]
        if secondChar == "A":
            free_energy = free_energy + free_energy_list[0]
        elif secondChar == "C":
            free_energy = free_energy + free_energy_list[1]
        elif secondChar == "G":
            free_energy = free_energy + free_energy_list[2]
        else:
            free_energy = free_energy + free_energy_list[3]
        i = i+1
        j = j+1
    return free_energy

# Column 11-SNPs
# Input--List of Bio.SeqFeature.FeatureLocations, sequence start
position, and
# end position
# Output--True iff any FeatureLocation overlaps sequence
def overlap(locations, seqStart, seqEnd):
    start = SeqFeature.ExactPosition(seqStart, extension = 0)
    end = SeqFeature.ExactPosition(seqEnd, extension = 0)
    overlap_exists = False
    for location in locations:
        loc_start = location.start
        loc_end = location.end
        if (start <= loc_start and loc_start <= end) or (start <=
loc_end and loc_end <= end) or (loc_start <= start and start <=
loc_end):
            overlap_exists = True
            break
    return overlap_exists

# Column 12-Termination Seq
# Input--Sequence in all capitals
# Output--True iff given sequence contains AAAA or TTTT
def termination_seq(seq):
    stringSeq = seq.tostring()
    fourA = stringSeq.find("AAAA")
    fourT = stringSeq.find("TTTT")
    termination = False
    if fourA != -1 or fourT != -1:
        termination = True
    return termination


main()
```

Macro to run A* Algorithm

```
'Rayka Yokoo
'Macro written to choose the top 4 sequences that do not overlap
'Requires that Book1 is open
'Output is written to Book1 Worksheet 2
```

```
Sub RNAi_Astar()

'Define number of sequences wanted
top_num_seq = 4

'Note workbook with siRNA sequences
original_workbook = ActiveWorkbook.Name

'Define column with accession numbers, annotationss, siRNA sequences,
'and empty column at the end
accession_column = 1
annot_column = 2
siRNA_column = 3
empty_column = 13

'Determine number of rows
row_number = 1
Cells(row_number, siRNA_column).Select
While Not IsEmpty(Selection)
row_number = row_number + 1
Cells(row_number, siRNA_column).Select
Wend

'Determine the optimum arrangement of siRNA sequences that don't
overlap
'Runs Astar algorithm
'Would like to memoize whether two sequences overlap but having
trouble creating fixed array

'Add all sequences as initial paths
'Column 1-path weight, Column 2-path
'Perform calculations in Sheet1 in Book1
number_sequences = row_number - 2
Workbooks("Book1").Activate
Worksheets(1).Select
Columns(1).ClearContents
Columns(2).ClearContents

number_paths = 0
'Start counting sequences at 0
For i = 0 To number_sequences - 1
Weight = i
Path = Join(Array(i), "")
'Rows start at 1
Cells(i + 1, 1).Value = Weight
Cells(i + 1, 2).Value = Path
number_paths = number_paths + 1
Next

'Pick the path with the least weight-row 1
Weight = Cells(1, 1).Value
Path = Cells(1, 2).Value
```

```
path_length = Len(Path)

'Extend until path length equals 4
path_found = True
While Not path_length = top_num_seq And path_found

'Extend
last_node = Val(Left(Path, 1))
'For each sequence ranked lower than the last sequence in the existing
path
'Check for overlap then add to the existing path
For i = last_node + 1 To number_sequences - 1
Overlap = False

'Check for overlap
For j = 1 To path_length
path_node = Val(Mid(Path, j, 1))
If overlapCalc(path_node, i, original_workbook) Then Overlap = True:
Exit For
Next

'If there is no overlap, add to existing path
If Not Overlap Then new_path = Join(Array(i, Path), ""): _
new_weight = i + Weight: _
Cells(number_paths + 1, 1).Value = new_weight: _
Cells(number_paths + 1, 2).Value = new_path: _
number_paths = number_paths + 1
Next

'Prepare variables for next loop
Cells(1, 1).Clear
Cells(1, 2).Clear
number_paths = number_paths - 1
ActiveSheet.Columns.Sort Key1:=Columns(1), Order1:=xlAscending,
Header:=xlNo
If IsEmpty(Cells(1, 1)) Then path_found = False _
Else Weight = Cells(1, 1).Value: _
Path = Cells(1, 2).Value: _
path_length = Len(Path)

Wend

'Output path to Book1 Worksheet 2 or display error message
Workbooks(original_workbook).Activate
total_num_sequences = 1
If path_found Then
For j = 1 To Len(Path)
Worksheets(1).Select
sequence_number = Val(Mid(Path, j, 1))
'Sequences enumerated starting at 0 but found in rows starting at 2
sequence = Cells(sequence_number + 2, siRNA_column).Value
accession_number = Cells(sequence_number + 2, accession_column).Value
```

```
Workbooks("Book1").Worksheets(2).Cells(total_num_sequences, 1) =
accession_number
Workbooks("Book1").Worksheets(2).Cells(total_num_sequences, 2) = j
Workbooks("Book1").Worksheets(2).Cells(total_num_sequences, 3) =
sequence
total_num_sequences = total_num_sequences + 1
Next
Else: MsgBox ("Cannot find non-overlapping sequences")
End If
End Sub



'Function returns true if two given sequences overlap, false otherwise
Function overlapCalc(seq1, seq2, data_workbook)
'Define position column
pos_column = 4
'Sequences enumerated starting at 0 but found in rows starting at 2
seq1_pos = Workbooks(data_workbook).Worksheets(1).Cells(seq1 + 2,
pos_column).Value
seq2_pos = Workbooks(data_workbook).Worksheets(1).Cells(seq2 + 2,
pos_column).Value
If Abs(seq1_pos - seq2_pos) < 23 Then overlapCalc = True Else
overlapCalc = False

End Function
```

Macro to Create shRNA Sequence

```
'Rayka Yokoo
'Macro written to add
'preT, hairpin, complementary, terminator, postC and sticky end
sequences
'to each RNAi sequence in a worksheet
Sub RNAi_tran()

'Define preT, hairpin, terminator, postC, sticky end sequence
preT = "T"
hairpin = "TTCAAGAGA"
terminator = "TTTTTT"
postC = "C"
stickyEnd = "TCGA"

'Looks like barcode isnt needed
'First row, first column should contain the next barcode sequence
row_number = 1
Cells(row_number, 3).Select
'barcode = Selection.Value
'row_number = row_number + 1
'Cells(row_number, 1).Select

'Run through column of RNAi sequences adding preT, hairpin,
complementary,
'terminator, barcode, and postC sequences
While Not IsEmpty(Selection)
origRNAiseq = Selection.Value
RNAiseq = Right(origRNAiseq, 21)
RNAiseq = UCase(RNAiseq)
complement = seqComplement(RNAiseq)
RNAcomponents = Array(preT, RNAiseq, hairpin, complement, terminator,
postC)
RNAicomplete = Join(RNAcomponents, "")
Cells(row_number, 4).Select
ActiveCell.Value = RNAicomplete

'Find reverse complement of complete RNAi sequence and add sticky end
RNAiComplement = seqComplement(RNAicomplete)
revRNAicomponents = Array(stickyEnd, RNAiComplement)
revRNAiComplete = Join(revRNAicomponents, "")
Cells(row_number, 5).Select
ActiveCell.Value = revRNAiComplete

'Increment barcode and row number for next loop
'barcode = barcodeCounter(barcode)
row_number = row_number + 1
Cells(row_number, 3).Select
Wend

'Output next barcode for the user
'ActiveCell.Value = barcode
End Sub

'Function returns the reverse complement of the sequence given
Function seqComplement(seq)

'Reverse Sequence
reverseSeq = StrReverse(seq)

'Find complement fo each character in sequence
For counter = 1 To Len(reverseSeq)
Select Case Mid(reverseSeq, counter, 1)
Case "A"
reverseSeq = Mid(reverseSeq, 1, counter - 1) + Replace(reverseSeq,
"A", "T", counter, 1)
Case "T"
reverseSeq = Mid(reverseSeq, 1, counter - 1) + Replace(reverseSeq,
"T", "A", counter, 1)
Case "C"
reverseSeq = Mid(reverseSeq, 1, counter - 1) + Replace(reverseSeq,
"C", "G", counter, 1)
Case "G"
reverseSeq = Mid(reverseSeq, 1, counter - 1) + Replace(reverseSeq,
"G", "C", counter, 1)
End Select
Next
```

```
'Return result
seqComplement = reverseSeq
End Function

'Function returns the next barcode
'The barcode counts up in the order of A, T, C, then G
Function barcodeCounter(code)

'Reverse code for simplicity
reverseCode = StrReverse(code)

'Find letter that needs to be incremented and increment
For counter = 1 To Len(reverseCode)
Select Case Mid(reverseCode, counter, 1)

'Letter with the lowest value-increment to T
Case "A"
reverseCode = Mid(reverseCode, 1, counter - 1) + Replace(reverseCode,
"A", "T", counter, 1)
Exit For

'Increment to C
Case "T"
reverseCode = Mid(reverseCode, 1, counter - 1) + Replace(reverseCode,
"T", "C", counter, 1)
Exit For

Case "C"
reverseCode = Mid(reverseCode, 1, counter - 1) + Replace(reverseCode,
"C", "G", counter, 1)
Exit For

'Letter with the highest value-set to A and run through loop again
Case "G"
reverseCode = Mid(reverseCode, 1, counter - 1) + Replace(reverseCode,
"G", "A", counter, 1)
End Select
Next

'Return result
barcodeCounter = StrReverse(reverseCode)
End Function
```