An executable stochastic model for Fatty Acid metabolism

Argyris Zardilis

Department of Applied Mathematics and Theoretical Physics

University of Cambridge

This dissertation is submitted for the degree of

Master of Philosophy

St Catharine's College August 2014

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Argyris Zardilis
August 2014

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Relatively recent technological advances which led to an accumulation of a wealth of data that have made Biology as a discipline shift some of its efforts from understanding individual components to understanding systems of interacting components. A systems level understanding of biological system is really important as it is the distributed information processing that happens at the systems level that gives rise to phenotype and the macroscopic behaviour of cells to sustain life. Biology has not traditionally used any formal methods but as the systems under investigation grew in size some more formal approaches were needed. System Biology is a relatively new field that tries to fill the gap by bringing practises from traditionally more formal disciplines like Mathematics, Physics, and Computer Science into the study of biological systems [11].

In the area of metabolism we have gone from a study of the properties of single enzymes and metabolites to the study of the behaviour of entire metabolic pathways and even entire genome-wide metabolisms of mammalian and model organisms. This accumulated knowledge about the interconnections in and between metabolic pathways is deposited in large online databases that integrate informations from various sources. While these diagrams are a rich source of information since metabolism is a highly dynamic process sometimes a more dynamic picture is needed. Mathematical techniques like Flux Balance analysis originating from Dynamical Systems Theory are particularly popular in the analysis of the flows in and out of metabolic pathways.

We have recently been able to pinpoint the biochemical reasons behind different diseases in the loss of balance between anabolic and catabolic activities in cells. These activities are known to be tightly regulated to avoid excess production and accumulation of metabolites which can be the reasons behind observed disorders. The products and intermediaries of lipid metabolism are central in many metabolic disorders and diseases. While Flux Balance Analysis is of great importance in the study of large metabolic networks it is not so suited in studying more local probabilistic metabolic processes and their crucial regulatory mechanisms.

Here we propose an alternative reaction-centric and stochastic view of metabolic systems and lipid metabolism in particular which we think will be an important tool in the mechanistic characterisation of crucial local metabolic processes and their regulatory mechanisms. We also propose the formal language of Petri Nets, which has been successfully used before in biochemical networks, as a potential tool to capture this alternative reaction-centric view of lipid metabolism. The goals and the work done towards this project were therefore two-fold: Firstly create a reaction-centric model of the Fatty Acid(FA) synthesis/elongation process which is an important part of lipid metabolism and secondly assess the formal modelling language of Petri Nets as a potential tool to capture this view. Since part of the work done was about modelling methodology and since we noticed a more general trend towards executable formal models in biology [7] we also present here an alternative version of the model in the process algebra of pi-calculus in an attempt to contrast the two formal modeling approaches: net models and process algebra models.

In this introduction I start by introducing the currently most popular methodology for the description of metabolic pathways(FBA), outline the biological importance of lipid metabolism and the motivation for a reaction-centric view, and finally I try to place Petri Nets and pi-calculus(the two methodologies used) in the spectrum of formal distributed computing modelling language world.

1.1 Dynamical systems theory and Flux Balance Analysis

Systems Biology attempts to master the complexity and understand metabolic pathways and any biological system by usually contructing models. There are two types of models based on their construction method. Bottom-up models are constructed by extracting biological knowledge from experimental datasets. This category includes for example the diagrammatic means of capturing the interconnected structure of metabolic pathwayw. Top-down models on the other hand start from a detailed knowledge of the system to get a mechanistic model of the system [23].

Early attemps to capture the interactions and dependencies between components were made with bottom-up models which resulted in large diagrams that capture the conversion processes happending inside metabolic pathways. While these diagrams are important and are a useful represntation of our knowledge about metabolism they lack the dynamic nature which is crucial in a systems level understanding of a system which should not only include the connections between the components but also how these components interact over time, how their behaviour is controlled, and how they respond to external stimuli [13].

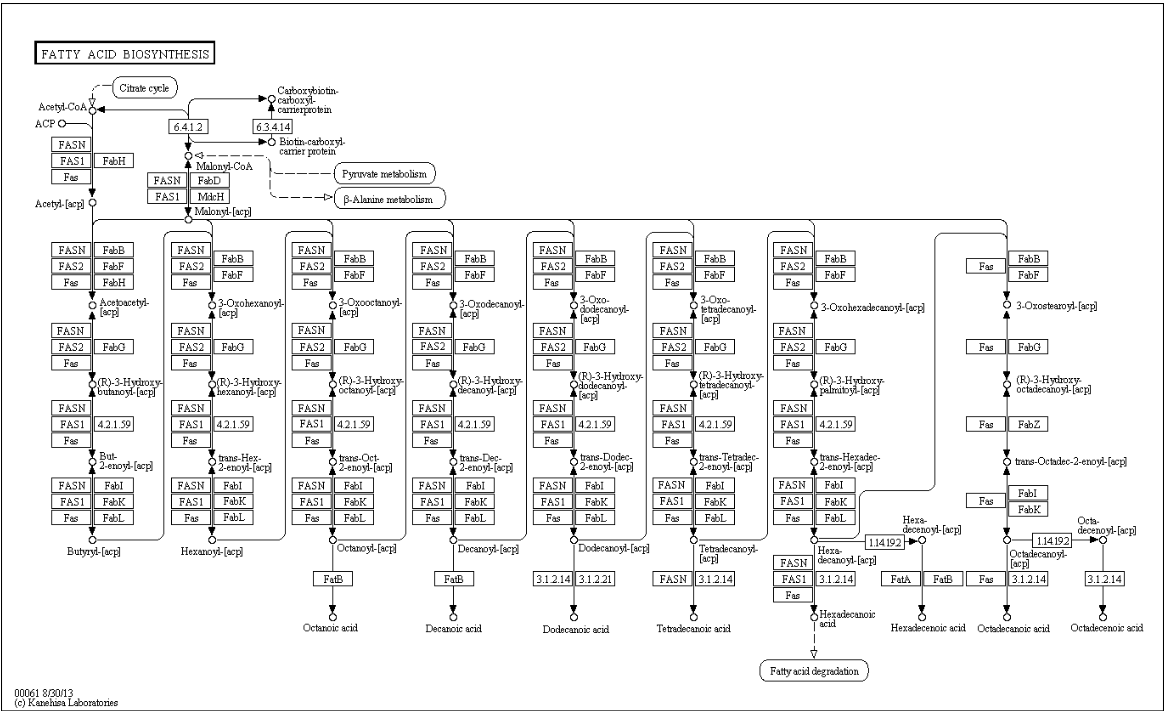


Fig. 1.1 Fatty acid synthesis pathway captured by the standard diagrammatic language used to capture interactions between components in a biological system. This was taken from KEGG a database which contains current knowledge about the working of these systems(from experiments for example) integrated with data from multiple sources.

This need to look into dynamic behaviour led to Dynamical System theory which based on its success in Physics has found its way in many other fields. Dynamical system theory captures the relationship between continuous quantities in difference-differential equations which describe the evolution of state variables in terms of changes in other state variables and even themselves thus capturing the interaction element. Since the time of Newton and classical mechanics ,where these ideas originated, the toolbox of dynamical systems theory has grown to include techniques for qualitative understanding of system without the need to solve them either numerically or analytically.

In biochemistry each species in the system is represented by its concentration leading to a differential equation for each species. The rate of change of the concentration of a species is described in terms of in flows- increasing the rate of change(production)- and out flows - decreasing the rate of change(degradation). These flows can be dependent on the concentrations of other variables(species) which participate in the same biochemical reactions. Consider for example a system with n components which we can group into the global state of the system $\mathbf{x} = (x_1, x_2, \dots x_n)$. All the reactions that take place in the system change, in discrete levels, the numbers of molecules of the species:

$$\mathbf{x} \xrightarrow{r_1(\mathbf{x})} \mathbf{x} + \boldsymbol{\delta_1}$$
$$\mathbf{x} \xrightarrow{r_2(\mathbf{x})} \mathbf{x} + \boldsymbol{\delta_2}$$
$$\vdots$$
$$\mathbf{x} \xrightarrow{r_m(\mathbf{x})} \mathbf{x} + \boldsymbol{\delta_m}$$

So for every reaction we have one such rule and δ_k are the discrete levels by which the species change for every reaction. Each reaction has an associated and possible state dependent rate $r_k(x)$ which is the expected number of times it takes place in a time unit. The in-flows are the positive deltas and out-flows the negative ones. The differential equation describing the evolution of the average numbers of species i is then the sum of the in and out flows over all reactions in the system:

$$\frac{dx_i}{dt} = \sum_m r_m(\mathbf{x})\delta_{mi}$$

Usually this formulation of the problem leads to integration with a numerical ODE solver of the above system to get the dynamic behaviour of the system. The usual problem mathematical biologists face is that the reaction rate function(the $r_k(\mathbf{x})$s) contain parameter constants that are not usually known.

In the contex of metabolic systems however there is a mathematical technique however, called Flux Balance Analysis, that is used to compute these flows without explicit knowledge of these parameters by making some assumptions about the system to simplify the problem [18]. If we assume that that metabolites cannot accumulate within the cell and their levels are more or less balanced then the system is at steady-state. The steady-state assumption means that the average positive flux-defined as the sum of all the in-flows- and the average negative flux-defined as the sum of all the out-flows- are equal which in turn means that the rate of change for every species , defined as the sum of all flows(negative and positive), is equal to 0. Solving for the fluxes leads to a linear equation for each species. For a more concise notation all the δ_{nm} are packaged into a matrix $\mathbf{S}$, traditionally called the Stoichiometric matrix, and the fluxes in a vector $\mathbf{r}$. Then the problem becomes solving the system $\mathbf{Sr} = \mathbf{0}$ to calculate the fluxes $\mathbf{r}$ through our system of interest. The only problem with this formulation of the problem is that the system is underdetermined since usually the number of reactions is higher than the number of species. To solve this the solution space is constrained with the use of an objective function which leads to a linear programming problem formulation which is easily solvable. An small contrived example to illustrate the above process is given in Figure 1.2.

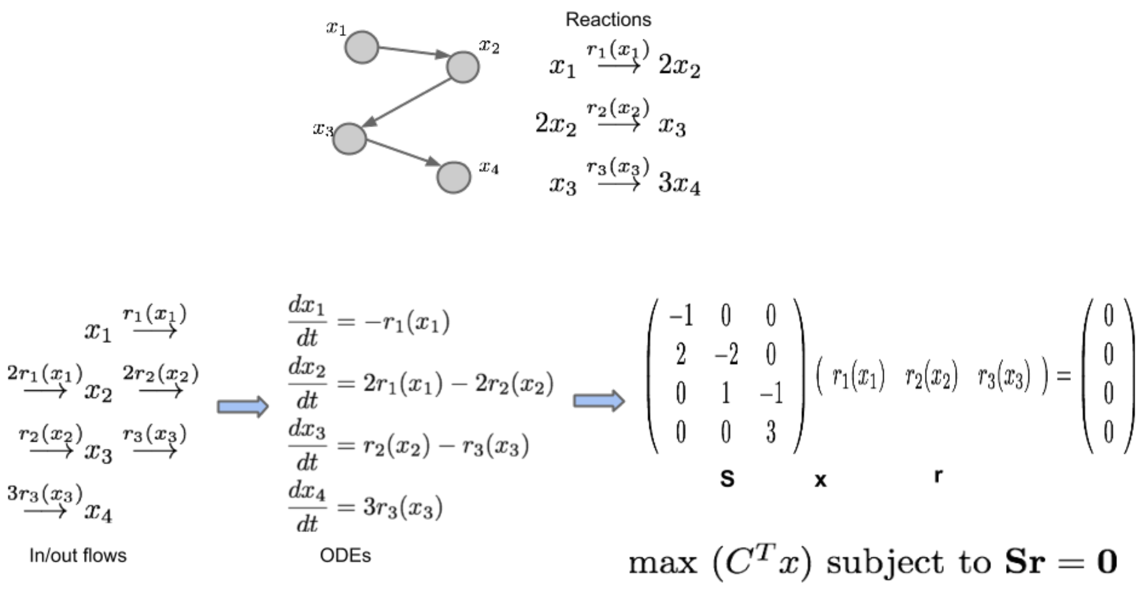

Reactions
$x_1 \xrightarrow{r_1(x_1)} 2x_2$
$2x_2 \xrightarrow{r_2(x_2)} x_3$
$x_3 \xrightarrow{r_3(x_3)} 3x_4$
In/out flows
ODEs
$\frac{dx_1}{dt} = -r_1(x_1)$
$\frac{dx_2}{dt} = 2r_1(x_1) - 2r_2(x_2)$
$\frac{dx_3}{dt} = r_2(x_2) - r_3(x_3)$
$\frac{dx_4}{dt} = 3r_3(x_3)$
S
x
r
$\max\ (C^T x)$ subject to $\mathbf{Sr} = \mathbf{0}$

Fig. 1.2 A small example how starting from a bottom-up model like a pathway from KEGG we can go on to apply FBA and calculate the fluxes through our pathway

Flux balance analysis is a very powerful technique since it allows us to overcome the problem of the rate function parameters and calculate the fluxes through the system in a computationally efficient way. It is not without its limitations though. For large networks of components finding a suitable objective function to used to transform to solve the under-determined problem can be difficult. More importantly and since it relies on the average behaviour as described by the differential equations it is not able to capture stochastic, non-deterministic behaviour.

1.2 Challenge of lipid metabolism

Lipid metabolism is the set of all the anabolic and catabolic processes that involve lipid products which serve mainly as energy stores or in the form of lipid bilayers as membranes. The starting point of lipid metabolism are the Fatty Acids which act as building blocks for more complex lipids like triacylglycerols. Lipids are produced by the organism to serve as energy stores in the well-fed state when there is an excess of carbohydrates and no immediate energy requirements. Any excess glucose molecules are converted to lipids as follows:

When the glucogen stores are full, glucose molecules will find the way through the glycolysis pathway blocked at the level of phosphofructokinase so they take a diversion through the Pentose Phosphate pathway and then join the glycolytic processes at a later stage bypassing the block. Then they continue down the normal route to Acetyl-CoA and the Krebs cycle in the mitochondria. Here since the immediate energy requirements are low Acetyl-CoA only makes it to the first stage of the Krebs cycle producing Citrate instead of continuing through the cycle and then to the Respiratory chain to produce energy. Since Citrate cannot proceed any further in the cycle its levels accumulate and at some point it diffuses from the mitochondrion into the cytosol where it is converted to Acetyl-CoA again to serve as a precursor to Fatty Acid Synthesis which required the NADPH produced by the Pentose Phosphate pathway. Once Fatty Acid are synthesised they can go on to form more complex lipid products like TAGs or get further modified by the Fatty Acid elongation pathway or by adding bonds to their CH tails. The beta-oxidation pathway in mitochondria breaks down Fatty Acids into Acetyl-CoA which is then fed into the Krebs cycle [22].

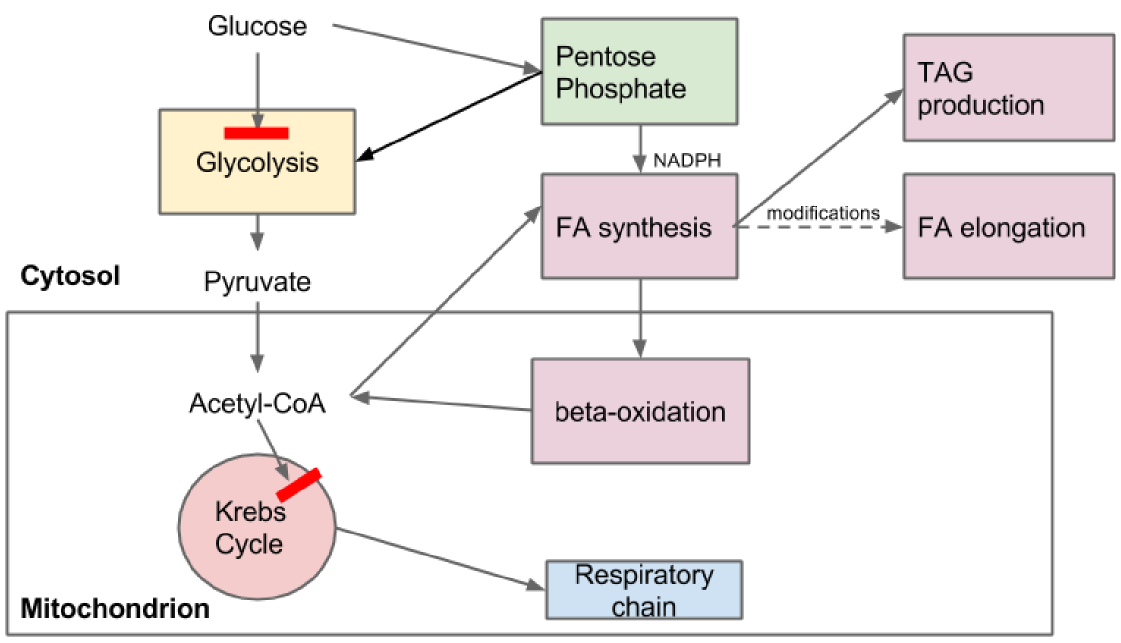


Fig. 1.3 A diagram outlining the main pathways involved in lipid metabolism.

All the above production processes have their equivalent catabolic processes and a balance between the two is essential for metabolic homeostasis in organisms. This balance is regulated by within and between pathway control signals which also integrate signals from the environment. In diseases like type I Diabetes this balance is disrupted because of insulin deficiency which breaks the regulatory signals. Fatty Acids and lipid metabolism pathways play a key role in this chain of disruptions. Also, free Fatty acids are associated with obesity and type II diabetes [3].

We notice from the above brief overview and the pictorial description of the interconnected pathways(Figure 1.3) that in metabolism and lipid metabolism products flow through the network with an iterative chain-like conversion process and that at several points there are probabilistic decisions to be made for the next step in the chain. These decisions are tightly regulated by signals from other pathways and the environment and disruption of these signals is observed in diseases like Diabetes. It is important therefore to have a mechanistic explanation of the 'trip' through the metabolic chain and the non-deterministic 'decision-making' that happens and its regulation. We believe that in order to get this mechanistic characterisation we would benefit from a stochastic reaction-centric view of lipid metabolism to give a more local perspective into the iterative conversion processes and their stochastic nature which stems from the probabilistic decision-making process. In this study we focus on the Fatty Acid Biosynthesis pathway which is an important part of lipid metabolism and it exhibits the characeteristics that are suited to a reaction-centric analysis: iterative behaviour with non-deterministic decisions and regulation mechanisms that relate it to other pathways.

A biochemical system, like the one given in the previous section, consisting of a number of reactions that alter the integer numbers of molecules of the species describes a stochastic process and a Continuous Time Markov Chain in particular. The differential equations we used as a basis for FBA are a just deterministic view of the average behaviour. The dynamics of the system can be set in motion inside a computer using the Gillespie exact simulation algorithm which creates sample time-series [9]. Statistical information like the strenth of the fluctuations (normalised variance) can be computed by collecting a lot of these traces.

In this study we propose Petri Nets as an alternative language to capture this reaction-centric view which we think is more expressive than the standard Markov processes formulation. While Petri Nets and their evolution through their formal operational semantics also describe a CTMC and are thus equivalent in power with the standard Markov jump processes we believe that they posses some characteristics(see next section and Methods) that makes them useful in the analysis of the reaction-centric view of lipid metabolism. Language is important not only because a syntax is needed to define our models in but also because the notation we use is our tool of thought and the use of the correct language for thinking about a problem can enhance our understadnding and ultimately help us solving it [12]. That is why we have more than one high-level programming language despite the fact that all of them are Turing complete and therefore have the ability to express all computations.

In the next section we try to place Petri Nets, the main modelling language used, and pi-calculus, the alternative used as a way to contrast net and process algebras models, in the spectra of formal modelling methodologies for distributed computation.

1.3 Computational models in Biology

Computer Science, although a fundamentally different discipline than Biology, has undergone a similar transformation in its focus from single information processing entities to systems of interacting entities(see the Internet, clusters). The term computation at the distributed level is now broader; it not only describes mere calculation but it also includes the interaction between components. This change in the term computation can be seen through the models we use to capture the notion of computation. At the early days of computing(even before actual electronic computers) the models of computation included lambda-calculus and Turing Machines. These early formalisms capture computation differently, Turing Machine as global state transitions in an Abstract Machine and lambda-calculus as reduction rules in a calculus, but they are effectively equivalent in expressive power(see Church-Turing thesis). As computer systems grew there was an interest for similar minimalistic languages to express distributed computation. Some formalisms that were invented during that period were Milner [15] Calculus of Communicating Systems, Lafont [14] Interaction nets, Milner et al. [16] pi-calculus, and Petri Nets [17].

The analogy between distributed computer systems and biochemical systems was made explicit by Berry and Boudol [2] and the Chemical Abstract Machine. The modelling langugages used for distributed computation are applicalbe in biological systems since they both include a high number of concurrent components with dependencies introducing non-deterministic behaviour. Fontana and Buss [8] went as far as to propose pi-calculus as an alternative to dynamical systems theory since he considered a calculus of objects that can change and their interactions more appropriate for describing biochemical systems than a calculus of continuous interacting quantities. Priami et al. [21] went on to use a stochastic version of pi-calculus to describe a biochemical systems and from then on other Computer Science inspired formalisms came up like Cardelli [4] with Brane-calculi, Danos and Laneve [6] with kappa, and Ciocchetta and Hillston [5] with Bio-PEPA.

Petri Nets lean towards the automata and Turing Machines style of computation because their operational semantics are in terms of state transitions from a distributed global state. They have a very intuitive graphical notation and concurrency, non-determinism, and the causal independencies between events is inherent in their structure. They also have a very natural correspondence to biochemical reactions and in their basic form have been used mainly to capture qualitative properties of biochemical systems [1]. In this study we use the stochastic version of Petri Nets which explicitly models time to describe and quantitatively analyse the reaction-centric view of the FA elongation and synthesis process.

Process algebras, which pi-calculus is an example of, on the other hand define syntax to specify the behaviour and state transitions of the individual components of the system independently therefore reactions and compositionality are not explicitly captured. In this study I particularly use Stochastic Pi Machine (SPiM) which is a programming language derived from stochastic pi-calculus but extended to include operational semantics for execution on an Abstract Machine and also a formal graphical notation [19, 20]. The fact that this variant shares characteristics from both net models and process algebras approached makes it an interesting case-study.

1.4 Outline of work

To summarise the main aim of this study was to assess the applicability of the Petri net formal language to capture a reaction-centric view of lipid metabolism by producing a model of the Fatty Acid biosyntesis/elongation process. For completeness we also created a stochastic pi-calculus version of the model to contrast the two methodologies as part of the more general turn towards executable formal models in Biology. An extended version of the basic process is also given including part of its regulatory mechanisms.

In chapter 2 an overview of the methods used in modelling the process is given, namely Petri Nets and stochastic pi-calculus. In section 3 the basic and extended model are presented along with their tuning with available metabolomics data. Finally in section **??**

Chapter 2

Methods

2.1 Data sources

2.2 Petri Nets

2.2.1 Syntax and Semantics of Basic nets

A Petri Net is a 4-tuple $(P, T, pre, post)$ defined as:

- a set of *places* (or conditions) P

- a set of *transitions* (or events) T

- a *preconditions map pre* $: T \rightarrow \mathbf{m}P$ which assigns a multiset of places $pre(t)$ to each transition $t \in T$

- a *postconditions map post* $: T \rightarrow \mathbf{m}P$ which assigns a multiset of places $post(t)$ to each transition $t \in T$

where $\mathbf{m}P$ is the space of multisets over P with a multiset over P defined as function $f : P \rightarrow \mathbb{N}$. The state of the system, called a marking, is again a multiset $\mathscr{M}$ over P. We can think of the marking as the distributed global state of the system. It is also common when defining a Petri Net to give the initial marking of the system usually written as $\mathscr{M}_0$.

The operational semantics of Petri Nets is defined in terms of changes in the global state through the action of the transitions T of the PN:

$$\mathscr{M} \xrightarrow{t} \mathscr{M}'$$

Here when a transition(event) t occurs it changes the state of the system from marking $\mathscr{M}$ to marking $\mathscr{M}'$. Unlike automata and Turing Machines however a transition does not occur from a single global state but instead it only affects part of the state:

$$\mathscr{M} \xrightarrow{t} \mathscr{M}' \text{ iff } pre(t) \leq \mathscr{M} \text{ and } \mathscr{M}' = \mathscr{M} - pre(t) + post(t)$$

For 2 multisets f and g over set X, $f \leq g$ is defined as $f \leq g \iff \forall x \in X f_x \leq g_x$. So a transition t is said to be 'enabled' if its preconditions are sufficiently marked($pre(t) \leq \mathscr{M}$) and when it 'fires' it changes the marking in the places in its vicinity, namely the set of places $\{p \mid pre(t)_p \neq 0$ or $post(t)_p \neq 0\}$($\mathscr{M}' = \mathscr{M} - pre(t) + post(t)$). The *post* and *pre* condition maps define the causal independence between transitions and add the ability of Petri Nets to model concurrency and non-determinism. Two (or more) transitions are concurrent if they do not share any preconditions. Non-determinism is added through dependencies of transitions which share preconditions. In that case a race condition is created because the dependent transitions compete at their shared preconditions with only one of them being able to fire at each step.

This formal definition of transitions leads to an algorithm for the executions of a PN as follows:

1. Initialise the net with $(P, T, pre, post)$ and set state $\mathscr{M} = \mathscr{M}_0$
2. Find enabled transitions, $enabled \subseteq T, \forall t \in T$ if $pre(t) \leq M$
3. Choose transition t from set of enabled at random
4. Update state according to $\mathscr{M} = \mathscr{M} - pre(t) + post(t)$
5. Repeat steps 2-4 until there are no more enabled transitions or an external stop condition is met(e.g. max number of steps)

The main strength of Petri Nets though lies on their very intuitive and widely used graphical representation. Petri Nets are represented as bipartite graphs with two sets of nodes, the places P and transitions T which are denoted by circles and rectangles respectively. A weighted directed edge with weight $n > 0$ is added between place p and transition t in the direction $p \rightarrow t$ if $pre(t)_p = n$. A weighted directed edge with weight $n > 0$ is added between transition t and place p in the direction $t \rightarrow p$ if $post(t)_p = n$. The markings are denoted as dots (tokens) in the circles depicting the places. So for example consider the net with $P = \{p1, p2, p3\}$, $T = \{t1\}$, $pre = \{(t1, \{(p1, 2), (p2, 1), (p3, 0)\})\}$, $post = \{(t1, \{(p1, 0), (p2, 0), (p3, 2)\})\}$, and marking $\{(p1, 3), (p2, 1), (p3, 1)\}$. The graphical depiction of this net would be the one shown in Figure 2.1. We could then also define the pre-places of a transition t as all the places p such that $pre(t)_p > 0$ and the post-places as all the places p such that $post(t)_p > 0$.

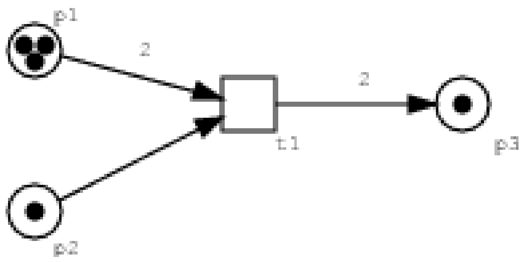


Fig. 2.1 Small basic Petri Net example

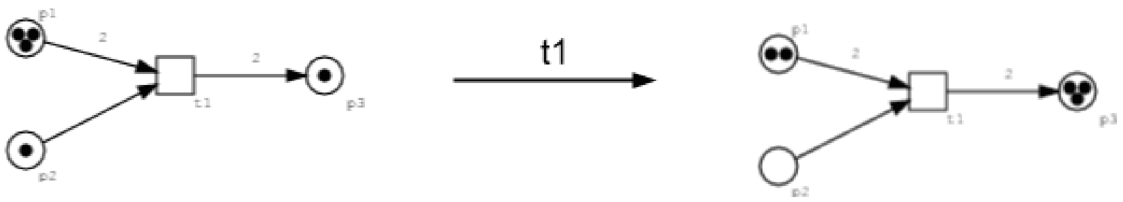


Fig. 2.2 The token game for basic Petri Nets.

The operational semantics of Petri Nets can also be defined very naturally in this graphical notation. When a transition t fires $pre(t)_p$ tokens are consumed from all the pre-places p and $post(t)_p$ tokens are added to all the post-places p. For the net given above(Figure 2.1) when transition $t1$ fires the 2 tokens are removed from $p1$ and 1 from $p2$ and 2 are added to $p3$(see Figure 2.2).The execution of the net can be seen as the flow of tokens through the net as transitions fire at each step according to the execution algorithm given above. This graphical view of the execution is called the 'token game' and it is very useful for gaining an insight into the dynamic behaviour of the system being modelled.

2.2.2 Stochastic Petri Net extension

Stochastic Petri Nets are an extension to the original Petri Net formalism to explicitly model time. Notice that even though the there is an ordering of transitions implied by their execution according to the algorithm given in the previous section the concept of time is not modelled explicitly in the basic Petri Net formalism. In order to model time explicitly Stochastic Petri Nets (SPNs) introduce a waiting time associated with each transition t, defined as random variable X_t distributed exponentially with potentially marking-dependent rate $\lambda_t(pre\_places(t))$ where pre_places is defined as before. Formally a SPN is a 5-tuple

$(P, T, pre, post, \nu)$ with everything defined as before with the addition of the map $\nu : T \rightarrow H$ where H is the set of all hazard functions $H = \{h_t \mid h_t : \mathbb{N}^{|pre(t)|} \rightarrow \mathbb{R}\}$. Hazard function is the typical name used in stochastic processes for the function giving the rate of the exponential time variable. In this case the hazard function h_t gives λ_t and the domain of h_t we will restrict to only the marking of the pre-places of t. This wait time associated with each transition changes the execution algorithm for Petri Nets; now when more than one transition is enabled a wait time is sampled from all of them and the one with the least waiting time gets to fire:

1. Initialise the net with $(P, T, pre, post, \nu)$, set state $\mathscr{M} = \mathscr{M}_0$, and $time = 0$
2. Find enabled transitions, $enabled \subseteq T, \forall t \in T$ if $pre(t) \leq M$
3. For each enabled transition t sample a wait time from an exponential distribution with rate $\lambda_t(pre\_places(t))$. Pick the transition t_i with least wait time to fire.
4. Proceed time $time = time + \tau_i$
5. Update state according to $\mathscr{M} = \mathscr{M} - pre(t_i) + post(t_i)$
6. Repeat steps 2-4 until there are no more enabled transitions or an external stop condition is met(e.g. max number of steps)

With this new variant of Petri Nets we maintain the non-determinism through the competition of enabled transitions with shared preconditions but by changing their rates we can favour some transitions over others. This Stochastic Petri Net formulation gives rise to Continuous Time Markov Chain and the execution algorithm defined above is more or less the Gillespie algorithm which is an exact algorithm for simulation of Markov jump processes. It is this Petri Net variant that we will use to model our system of interest, the Fatty Acid synthesis/elongation pathway. In the next section we describe the correspondence between a biochemical network and especially a metabolic pathway to a Stochastic Petri Net and some examples of previous uses of SPNs in biology/biochemistry.

2.2.3 Application to Biochemistry

The correspondence between a biochemical system and a Stochastic Petri Net is as follows: the chemical species of the system become places, reactions become transitions and their rates the rates of the transitions, and the stoichiometry of the reactions is represented in the *pre* and *post* maps. This is a very natural correspondence between chemical reactions and Petri Nets. In fact the standard notation for chemical notation is itself a process algebra albeit not a very formal one. So for example consider a made-up reaction that takes 2 molecules of A and 3 molecules of B and produces 5 molecules of C:

$$A + B \xrightarrow{r_1} 2C$$

This is an event, a reaction with rate k, which tells us how the state of the system changes but although implicitly we know the conditions for it to take place they are not formally captured by the informal process algebra of chemical reactions as defined by the standard arrow notation. A Petri Net representation of this reaction however captures the reaction more formally:

2.3 Stochastic Pi-calculus

This is going to contain a vey brief introduction to stochasic pi-calculus.

Chapter 3

Work

The aim of this study was to capture the iterative elongation process of even-chain saturated Fatty Acids. Fatty Acids are carboxylic acids with long hydrocarbon side groups (Figure 3.1) and they are usually characterised by the number of carbons in these side-groups, for example a FA with a chain of 6 carbons is usually denoted as C_6. The side-chains of FAs grow by successive C_2 concatenations. This concatenation process takes places at the FA biosynthesis pathway in the cytosol for chain lengths up to C_{18}. Further elongation takes places at the FA elongation pathway in the Endoplasmatic Reticulum(ER).

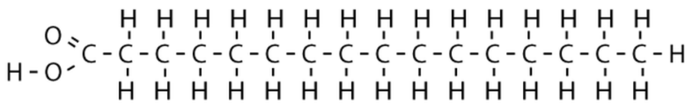


Fig. 3.1 The structure of Fatty Acids(FA) which consists of a long hydrocarbon side-group. The length of this side-group charactersises the FA.

In this study we are interested in the elongation process in its entirety so our models capture the combined effect of the two pathways responsible for it, FA biosynthesis *and* FA elongation. In this section I first introduce the basic model and the assumptions made to simplify it, the tuning of that basic model based on FA measurements, a version of the model in stochastic pi-calculus and finally the extension of the model with the introduction of some control mechanisms.

3.1 Basic model

3.1.1 Petri Net implementation

FA biosynthesis starts with Acetyl-CoA which is being converted to Malonyl-CoA. A reaction between Malonyl-CoA and Acetyl-CoA starts off the first C_4 FA product. After that there are successive C_2 concatenations to elongate the FA product with each elongation step , that requires 4 reactions, requiring another Malonyl-CoA(Figure 3.2). The full pathway from KEGG can be seen in Figure 3.3. The elongation pathway located in ER is similar in nature but the intermediaries are CoAs instead of ACPs(Figure 3.4).

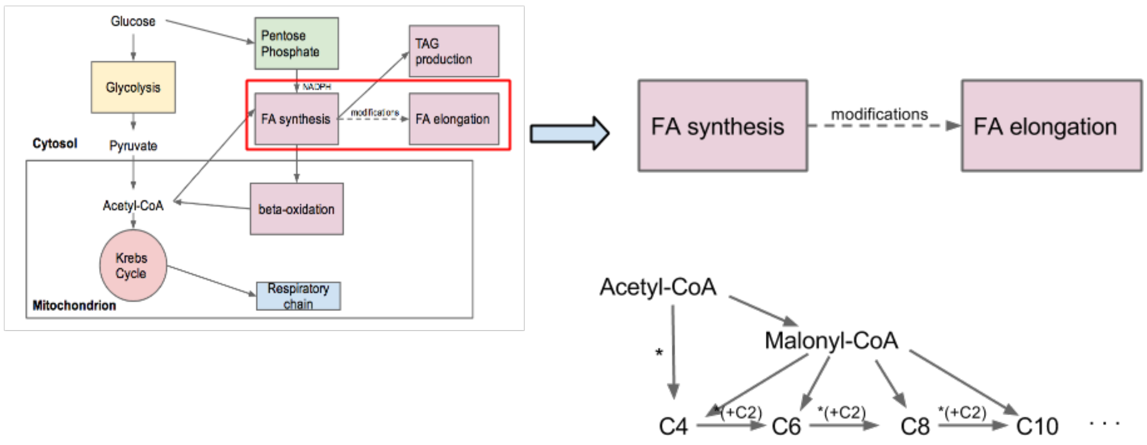


Fig. 3.2 FA metabolism in relation to other pathways.

Because of the natural correspondence between biochemical reactions as they are found in the KEGG static model to Petri Net model constructs a straight translation between the two is straightforward. All the pathway reactions become transitions with pre-places the reactants and post-places the products. Here we made our first assumptions by omitting the enzymes from the reactions. Information about the enzymes is not so important for our purposes since we are interested in the numbers of molecules of the metabolites in the system. Information about the enzymes can be incorporated in the transition rates. The net can be animated using its formal operational semantics and as transitions/reactions occur tokens/metabolites move through the net. The FA products are represented as sinks, places that are not pre-places to any transitions, so once a token reaches one of these sinks it is trapped. That way the probabilistic iterative nature of the process is capture completely, an intermediate of the process can either remain at that length or go on to form longer FAs. The sinks are therefore the outputs of the stochastic process with the inputs being the places that do not act as post-places for any transitions. Starting with some finite number of molecules

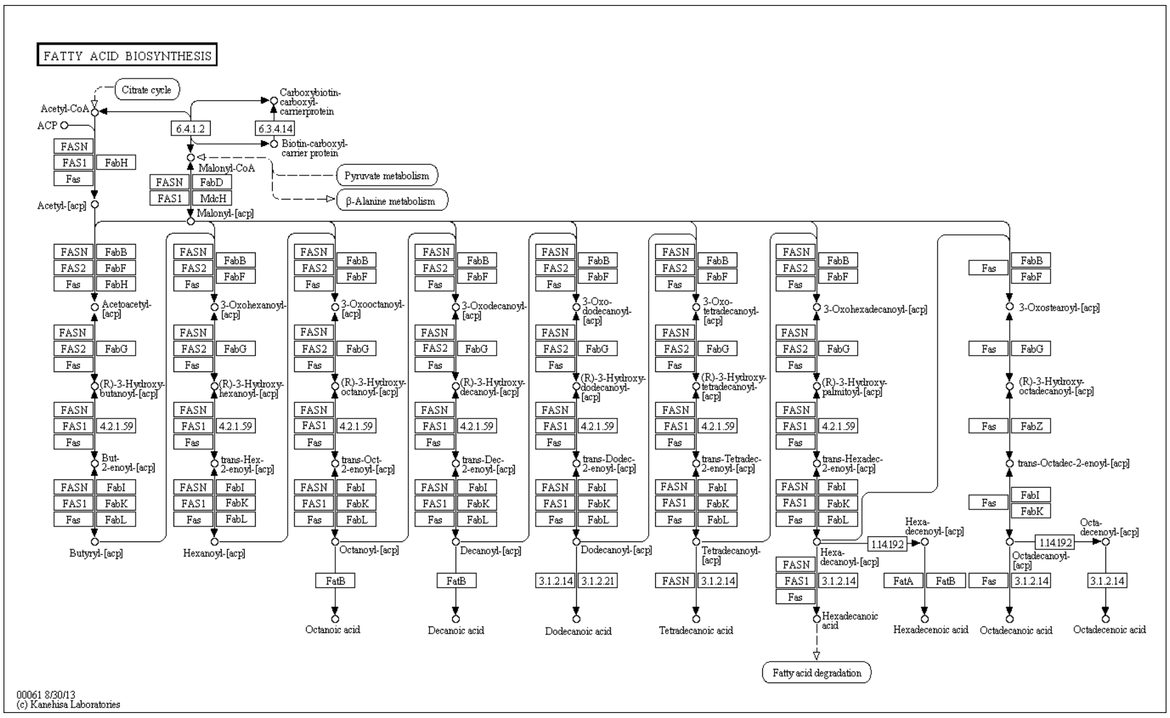


Fig. 3.3 FA biosynthesis in the cytosol. Notice the successive concatenations. Each concatenation takes 4 reactions.

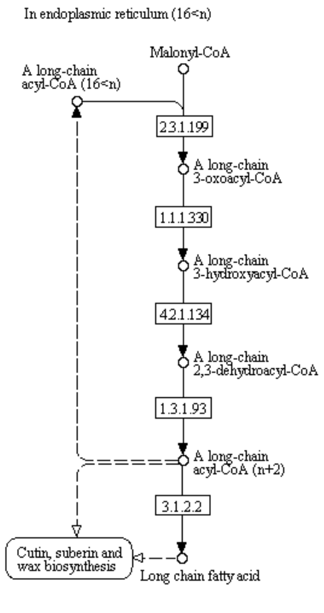

In endoplasmic reticulum (16<n)
Malonyl-CoA
A long-chain acyl-CoA (16<n)
2.3.1.199
A long-chain 3-oxoacyl-CoA
1.1.1.330
A long-chain 3-hydroxyacyl-CoA
4.2.1.134
A long-chain 2,3-dehydroacyl-CoA
1.3.1.93
A long-chain acyl-CoA (n+2)
3.1.2.2
Cutin, suberin and wax biosynthesis
Long chain fatty acid

Fig. 3.4 General form of FA elongation in ER. Notice that the intermediate products are CoAs instead of ACPs as in biosynthesis.

at the inputs, these will be consumed throughout the process until we reach a dead-state where no further transitions are enabled. The translation from the KEGG pathway model to the Petri Net model was done manually with the SNOOPY [10]tool which allows you to draw a net and play the token game for basic nets to observe its behaviour.

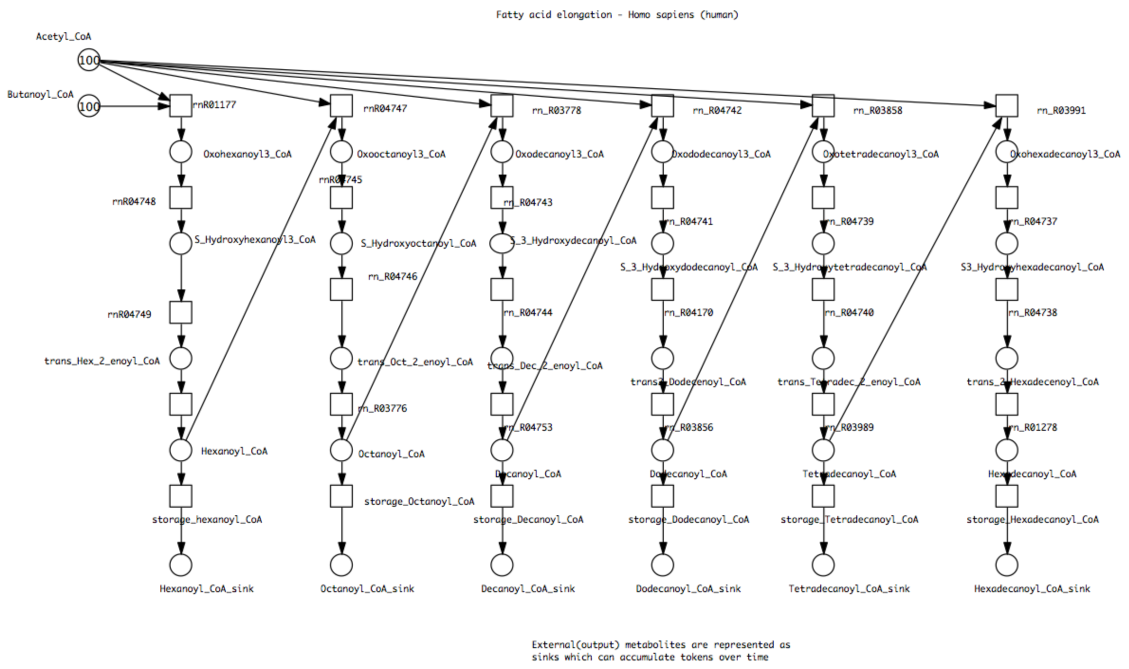


Fig. 3.5

A direct translation from KEGG is certainly useful for capturing a low-level biochemical view of the system. The model can offer an even more fine-control view of the system than the model presented in Figure 3.5 by including the enzymes in the reactions and information about their affinity to the reaction substrates. However including too many details in the model can sometimes hide the true aspect of the system the modeller is trying to understand and analyse. Here we are interested in the probabilistic, non-deterministic nature of the iterative FA elongation process which happens as this series of C_2 concatenation steps in FA elongation and biosynthesis pathways in ER and cytosol respectively. The outputs we are particularly interested to see are the numbers and proportions of FAs at different lengths. In Petri Nets model language these are the number and more importantly the proportions of molecules/tokens that end up in the sink places at the stop of the model execution. Because the process is inherently iterative and probabilistic these numbers will be different at the end of each model execution. This is in contrast with most dynamic modelling approaches as we are not interested in the time traces but rather only at the end result of this stochastic process. We can think of the Petri Net as a magic box governed by some probabilities(that we can tune) where we throw some inputs and according to these probabilities some output will come out at the other end in the form of proportions of FAs at different lengths. The probabilities are just the control parameters that guide the operation of the magic box.

Having outlined our modelling goals in the previous section we can now make some assumptions that will guide our design of the simplified synthesis/elongation model that will be our basic model throughout this study. First of all the model will capture the combined effect of both the pathways. The two pathways are in different compartments of the cell, one in cytosol and one in ER, but we will ignore the transportation of FAs from cytosol to ER and we will assume that the two process are sequential so a C_{18} for example can be elongated directly to a C_{20} while in reality it would have to first be transported to the ER. Since we also take this view of the net model as a control box turning input signal through a series of steps to an output signal we can safely squeeze the four step reaction chain needed for each C_2 concatenation step to a single step reaction and at the same time consider only the forward direction reactions. Also, since we are no longer consider exact chemical reactions the reactions rate functions will just be constant values and we will treat those more as probabilities that will govern the non-deterministic decisions during the execution of the model that transforms the input signal to output proportions of FAs. Finally I also decided to stop the elongation process at C_{22} and that the sinks will only include products C_{12} - C_{22}. FAs with lengths 4-10 will be included but no output will be accumulated there. These decisions are a bit arbitrary but they are ultimately tied with the output real datasets I had available for tuning the model(see subsequent sections).

To build the basic model then that follows from the goals and this assumptions was built manually with SNOOPY [10]. This basic model that will be the basis for our work in this study can be seen in Figure 3.6. The model is similar in nature to the more full model presented previously but the concatenation steps are represented as single transitions and the elongation process goes up to C_{22} because the elongation process in the ER is also included. Starting simple I also started from the inputs Malonyl-CoA and Acetyl-CoA whereas in reality the only precursor of FA biosynthesis is Acetyl-CoA and the 2 input points of the pathway are products coming from Acetyl-CoA. The model as presented in Figure 3.6 starts with inputs: 300 Malonyl-CoA and 100 Acetyl-CoA molecules. The model can then be set in motion with SNOOPY until it reaches a dead state which basically corresponds to the point where the system runs out of input metabolites to 'fuel' any further reactions. The output of the system is the number of tokens accumulated at each of the sinks or FAs of different lengths.

Observing the execution pattern of the basic net model we notice an execution pattern

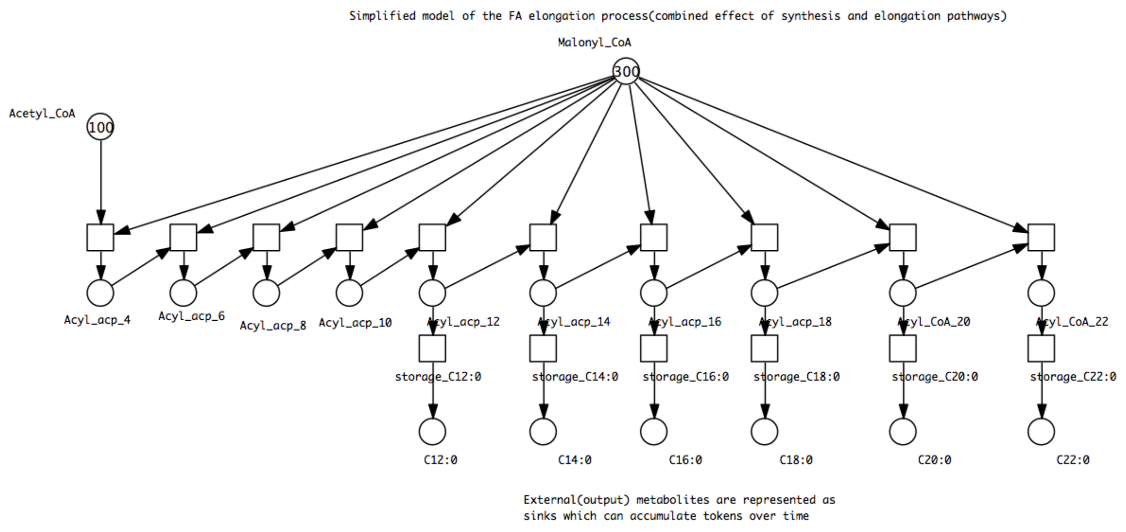


Fig. 3.6 Petri Net implementation of the basic model as described in the main text

which leads to an elegant view of the system as a series of binary probabilistic decisions or Bernoulli trials. The way the inputs of the model are set up, at the start of the model execution only the initial transition producing the first product C_4 is enabled. If we consider that after this transition fires (since its the only one enabled) it will only fire again after its initial product has reached a sink then at each point only one token is travelling through the net. This token goes through an iteration of binary decisions: stay at current length *or* continue to form a longer FA. More formally as the token gets transformed to an intermediate product there are only two transitions enabled: the transition taking it to the next longer intermediate and the transition taking to be stored at its current length(Figure 3.7). Let these two transitions be t_1 and t_2 respectively. According to the operational semantics of Stochatic Petri Nets(see Methods section) a wait time is sampled for each of the enabled transitions from a negative exponential distribution with rate the value of the rate function of the transition. In this case the rate function of the two transitions are constants(see assumptions) λ_{t_1} and λ_{t_2}. Since we know that one of the reactions will fire we can say that the firing probabilities of the two transitions or the probabilities of the token's decisions are:

$$P(staying) = P(t_1) = \frac{\lambda_{t_1}}{\lambda_{t_1} + \lambda_{t_2}}$$
$$P(continuing) = P(t_2) = \frac{\lambda_{t_2}}{\lambda_{t_1} + \lambda_{t_2}} = 1 - P(t_1)$$

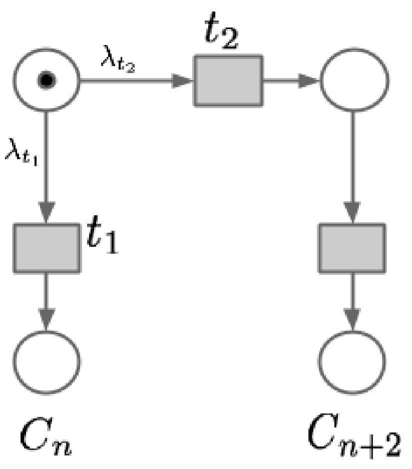


Fig. 3.7 Binary decision at a particular point in the execution.

In other words this is a Bernoulli trial with probability of success $p = P(t_1)$. Since we only care about the outputs at the end of the execution and not the timeframe of the process from start to finish then only the ratio of these two successive transitions rates is important and not their absolute numbers. Then the entire journey of a token through the network from the initial C_4 product until it reaches a sink and gets stored can be thought of a series of Bernoulli trials(Figure 3.8). Then the entire stochastic process can be thought of a sequence of successive realisations of this series of trials with tokens travelling through the net one after the other with each one going through the series of decisions or Bernoulli trials. The initial assumption that only one token travels through the net at each time can in fact be dropped since the output of the net only depends on the ratios of the pair of transitions representing each binary decision. I just chose to have it because this leads to the nice picture of the successive realisations of chains of Bernoulli trials which is intuitive for illustration purposes. It is also interesting to note how this binary decision corresponds to a conflict between two events that share pre-conditions which create non-determinsim through a race condition between the two depdendent events. This 'conflict' which represents this decision process is captured inherently in the structure and semantics of Petri Nets.

Each realisation of the stochastic process described by the Petri Net model which is itself a series of successive realisations of series of Bernoulli trials will have different outputs. While SNOOPY is a great tool for the visual side of things, creating the model and animating them, in order to be able to execute the model many times to get output profiles we needed a programmatic solution. I wrote code in Python that can read a model, execute

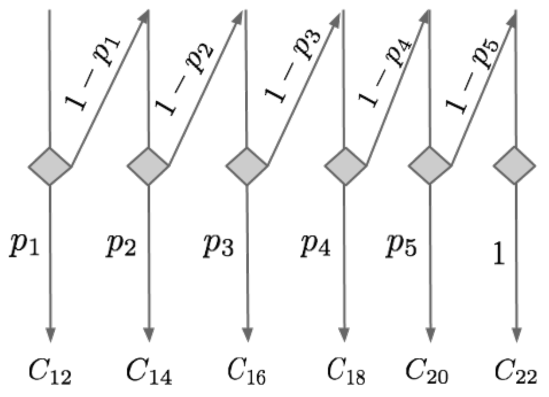
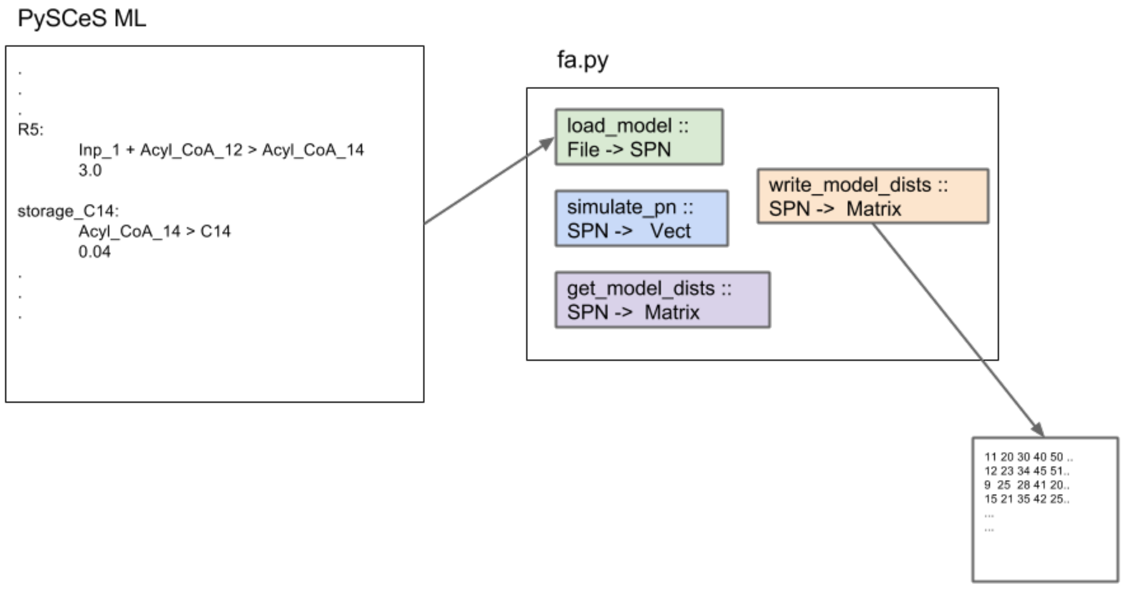


Fig. 3.8 The entire stochastic process a series of Bernoulli trials.

it once or many times, and write out the results. For the description of the model I used the PySCeS model description language for biochemical systems which is aimed at the Python language. The reactions are defined in the standard chemical notation and since this corresponds exactly to Petri Net transitions it was easy to load the description into an appropriate Petri Net representation(Figure 3.9).



```
.
.
.
R5:
        Inp_1 + Acyl_CoA_12 > Acyl_CoA_14
        3.0

storage_C14:
        Acyl_CoA_14 > C14
        0.04
.
.
.
```

```
11 20 30 40 50 ..
12 23 34 45 51..
9  25  28 41 20..
15 21 35 42 25..
...
...
```

Fig. 3.9 Structure of the code and workflow

3.1.2 Model parameters

In the previous section I described the building of the basic model based on the goals and assumptions of the study. The elongation aspect that we are interested in was reduced down to a sequence of successive series of Bernoulli trials that determine the length that a FA will stop its elongation at. The interesting parameters of the model, the ones that will affect the output, are the success probabilities for the stay-continue decisions for FA intermediaries with lengths $[C_{12}, C_{22}]$ since they are ones that have sinks. In this section two methods for the identification of these parameters for tuning the model so that its execution is biologically valid are presented. They both rely on experimental data from an MS study that contains relative MS spectra intensities for a number of lipid products extracted from cells. The aim of the study for which the experiment was conducted was to identify the effect of some treatment in lipid products so it included 90 samples from control and treatment (at different levels) individuals. Here we are only interested in creating a null model so only the 30 samples from the control individuals in the experiment were retained for the purposes of parameter tuning.

The outputs of the net model are integer number of tokens accumulated at each of the sinks or FA of different lengths while the experimental data are relative intensities which are continuous numbers. In order to be able to compare the two for the tuning of the parameters the experimental data were transformed from relative intensities to relative absolute numbers. The transormation of a sample of relative intensities $\mathbf{s} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ became a transformed sample $\mathbf{ts} = \{ts_i | ts_i = \lfloor s_i / min(\mathbf{s}) \rfloor \}$. Since we are only interested in the relative proportions of the outputs going from relative intensities to relative numbers should not affect the outcome. The data can then be seen as different realisations of the stochastic process described by the real model that acts in nature. The attempt is then to tune our model to be as close to the real model as possible by trying to tune the proportions of the outputs of our model to match the proportions of the outputs by the real model represented in the experimental data. This is classical Machine Learning formulation of the inverse problem(Figure 3.10). In this section I first present a method to get point estimates and then profiles for the success probabilities parameters of the series of Bernoulli trials representing the chain of decisions for a token travelling through the net.

Point estimates

The data more formally are a matrix D with an element D_{ik} being the number of tokens accumulated in the k-th FA in the i-th sample or realisation of the process. Since we have multiple realisations of Bernoulli trials the success probabilities can also be though of as

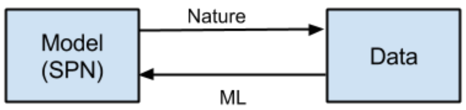


Fig. 3.10 ML inverse problem formulation

success probabilities in binomial distributions. The Maximum Likelihood point estimator for binomial success probabilities is given for the *i*-th decision corresponding to the *i*-th FA,

$$\hat{p}_i = \frac{k_i}{n_i}$$

, where k_i is the number of successes and n_i is the number of trials. If we extract information about the number of successes and number of trials for each binary decision in the process from the data that are different realisation of the process then we can straightforwardly compute the success probabilities we are after.

If we a take a specific sample (a row from data matrix D) then the number of successes for an FA are simply just the number of tokens of that FA in the sample since those are the tokens that when faced with the decision of whether to get stored as that FA or continue to form longer FAs they chose the former. Success and failure are arbitrarily assigned to staying and continuing respectively. The number of trials is the number of all the tokens that reached that FA intermediary, both the ones that stayed *and* the ones that continued. Since this is an iterative process the ones that continued are the number of tokens that made it to the sinks after the FA(the ones on the right as we look at the Petri Net model picture, Figure **??**). To illustrate this consider a sample which could be a row from our data matrix D: $\{C_{12} = 5, C_{14} = 10, C_{16} = 34, C_{18} = 23, C_{20} = 3, C_{22} = 5\}$. The ML estimator for the *i*-th FA product in that sample will be:

$$\hat{p}_i = \frac{C_i}{\sum_{n \geq i} C_n}$$

The success probabilities p_1 to p_6 for the above sample, with p_1 corresponding to the first

Parameter	Value
p_1	0.0003000656
p_2	0.0400323889
p_3	0.8235217126
p_4	0.9888339514
p_5	0.778597786
p_6	1.0

binary decision and C_{12}, p_s to the second decision and C_{14} and so on, are:

$$\begin{aligned}
\hat{p_1} &= 5/(5+10+34+23+3+5) = 0.0625\\
\hat{p_2} &= 10/(10+34+23+3+5) = 0.1333\\
\hat{p_3} &= 34/(34+23+3+5) = 0.523\\
\hat{p_4} &= 23/(23+3+5) = 0.742\\
\hat{p_5} &= 3/(3+5) = 0.375\\
\hat{p_6} &= 1
\end{aligned}$$

This is for one sample but since all the samples are from the same population(controls) we can combine the successes/trials data from all samples and generalise the ML success probability estimation for the *i*-th FA products as:

$$\hat{p_i} = \frac{\sum_k D_{ki}}{\sum_k \sum_{n\geq i} D_{kn}}$$

We assume that the columns of data matrix D are ordered according to the order of FAs in the system.

Computing the success probabilities from the real experimental dataset can be seen in Table **??**.

Since we do not care about the timeframe of the process but rather at the output at the end the relative proportions of the transitions participating in a binary decision are what is important therefore we can use the calculated success probabilities as the rates of the transitions related to the binary decision. So for example since $\hat{p_3} \approx 0.82$ then the rates of the transitions corresponding to the binary decision for the 3rd FA, C_{16}, will be: rate of transition for C_{16} storage (success) 0.82 and rate of transition for C_{18} formation (failure) $0.18(1-P(success))$.

Profiles

In the previous section I derived point estimates for the success probabilities guiding the binary decisions taking place during the elongation process. I did that by considering the success probabilities that maximise the likelihood function $L(D|p_i)$ for the i-th FA and data D being the number of successes and number of trials. In this section I will present a likelihood function for the entire process which comes naturally by thinking of the process as before as a sequence of series of Bernoulli trials. I then use this likelihood function to get the profiles of the parameters of interest.

The data from one of the samples can be seen as one realisation of the stochastic process with the numbers of tokens at each FA being the outputs of the series of Bernoulli trials that make up one realisation of the entire process. Each series of Bernoulli trials describes the journey of one token through the net which ends when it finds a sink. If for example the token ends up at the third sink that means that the outcome of the series of Bernoulli trials was: fail, fail, success. Therefore the data consisting of the number of tokens that ended up at each FA gives us a history of the series of trials that happened during that realisation of the process. Consider the following sample $\{C_{12} = 5, C_{14} = 10, C_{16} = 34, C_{18} = 23, C_{20} = 3, C_{22} = 5\}$ which is one realisation of the process from the start until reaching a dead-state. The numbers tell us the 'fate' of each of the tokens that went through the net during that realisation of the process, 5 tokens made the decision to stay at the first trial- [success]- , 10 tokens made a decision to continue on the first decision and then to stay on the second decision- [fail, sucess]-, 34 tokens continued on the first two decision and stayed on the third- [fail, fail, success], and so on. We can therefore calculate the probabilities for each FA product sinks as follows:

$$\begin{aligned}
P(C_{12}) &= p_1 \\
P(C_{14}) &= (1-p_1)p_2 \\
P(C_{16}) &= (1-p_1)(1-p_2)p_3 \\
P(C_{18}) &= (1-p_1)(1-p_2)(1-p_3)p_4 \\
P(C_{20}) &= (1-p_1)(1-p_2)(1-p_3)(1-p_4)p_5 \\
P(C_{22}) &= (1-p_1)(1-p_2)(1-p_3)(1-p_4)(1-p_5)p_6
\end{aligned} \tag{3.1}$$

The entire process can then be seen as a multinomial draw with 6 outcomes, $\{C_{12}, C_{14}, C_{16}, C_{18}, C_{20}, C_{22}\}$, with their respective probabilities given above and with the number of trials being the number of tokens that ended up in sinks for that particular realisation of the process. This gives the likelihood function of the data given the success probabilities as probability mass function of the multinomial distribution:

$$L(D|\theta) = f(C_{12},\ldots,C_{22};n;P(C_{12}),\ldots,P(C_{22})\}) =$$

Using the likelihood function for the entire process we can find the posterior of the initial success probabilities p_1 to p_6,

$$P(\{p_i\}|D) \sim L(D|\{p_i\})P(\{p_i\})$$

where the data D is simply the outputs of one realisation of the process or the counts at each sink and $P(\{p_i\})$ are the priors of the parameters. If we assume uniformative priors the above parameter posterior simply becomes: $P(\{p_i\}|D) \sim L(D|\{p_i\})$. Since the parameters are not exactly the parameters of the multinomial but they are only related through Equations 3.1 an analytic solutions is not very attractive but since it is still a small problem we can resort to a simple simulation scheme to sample from the required posterior. I used the standard Metropolis-Hastings MCMC method to sample from the posterior as given above starting from a random point in parameter space and a symmetric normal proposal distribution with mean the previous accepted guess and an arbitrary chosen variance. The MCMC traces and the histograms of the samples obtained from the posteriors can be seen in Figure.

3.1.3 A stochastic π-calculus implementation

In this section I present a stochastic pi-calculus version of the model which is specifically written in its SPiM variant which extends the traditional pi-calculus first by introducing explicitly time which is needed for quantitive analysis of biochemical networks and also by adding operational semantics in terms of an Abstract Machine and a graphical representation language. The SPiM code is given for the processes making up the system to introduce further some of the concepts of the SPiM language.

```
let Acetyl_CoA() =
(
do delay@mal_rate; Malonyl_CoA()
or delay@acyl_rate; Acyl_CoA()
)
and Malonyl_CoA() =
(
do !form6; ()
or !form8; ()
or !form10; ()
or !form12; ()
or !form14; ()
or !form16; ()
)
and Acyl_CoA() = ?form6; Acyl_CoA_6()
and Acyl_CoA_6() = ?form8; Acyl_CoA_8()
and Acyl_CoA_8() = ?form10; Acyl_CoA_10()
and Acyl_CoA_10() = ?form12; Acyl_CoA_12()
and Acyl_CoA_12() =
(
do ?form14; Acyl_CoA_14()
or delay@store_rate_12; C12()
)
and Acyl_CoA_14() =
(
do ?form16; Acyl_CoA_16()
or delay@store_rate_14; C14()
)
and Acyl_CoA_16() =
(
do ?form18; Acyl_CoA_18()
or delay@store_rate_16; C16()
)
and Acyl_CoA_18() =
(
do ?form20; Acyl_CoA_20()
or delay@store_rate_18; C18()
)
and Acyl_CoA_20() =
(
do ?form22; Acyl_CoA_22()
or delay@store_rate_20; C20()
)
and Acyl_CoA_22() = delay@0.0001; C22()
and C12() = ()
and C14() = ()
and C16() = ()
and C18() = ()
and C20() = ()
and C22() = ()

run(200 of Acetyl_CoA())
```

3.2 Extended model of FA synthesis

3.2.1 Petri Net implementation

3.2.2 Model Parameters

Chapter 4

Discussion

References

[1] Baldan, P., Cocco, N., Marin, A., and Simeoni, M. (2010). Petri nets for modelling metabolic pathways: a survey. *Natural Computing*, 9(4):955–989.

[2] Berry, G. and Boudol, G. (1989). The chemical abstract machine. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 81–94. ACM.

[3] Boden, G. and Shulman, G. (2002). Free fatty acids in obesity and type 2 diabetes: defining their role in the development of insulin resistance and β-cell dysfunction. *European journal of clinical investigation*, 32(s3):14–23.

[4] Cardelli, L. (2005). Brane calculi. In *Computational methods in systems biology*, pages 257–278. Springer.

[5] Ciocchetta, F. and Hillston, J. (2009). Bio-pepa: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33):3065–3084.

[6] Danos, V. and Laneve, C. (2004). Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110.

[7] Fisher, J. and Henzinger, T. A. (2007). Executable cell biology. *Nature biotechnology*, 25(11):1239–1249.

[8] Fontana, W. and Buss, L. W. (1996). *The barrier of objects: From dynamical systems to bounded organizations*. Citeseer.

[9] Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361.

[10] Heiner, M., Herajy, M., Liu, F., Rohr, C., and Schwarick, M. (2012). Snoopy–a unifying petri net tool. In *Application and Theory of Petri Nets*, pages 398–407. Springer.

[11] Ideker, T., Galitski, T., and Hood, L. (2001). A new approach to decoding life: systems biology. *Annual review of genomics and human genetics*, 2(1):343–372.

[12] Iverson, K. E. (2007). Notation as a tool of thought. *ACM SIGAPL APL Quote Quad*, 35(1-2):2–31.

[13] Kitano, H. (2002). Systems biology: a brief overview. *Science*, 295(5560):1662–1664.

[14] Lafont, Y. (1989). Interaction nets. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 95–108. ACM.

[15] Milner, R. (1980). A calculus of communicating systems.

[16] Milner, R., Parrow, J., and Walker, D. (1992). A calculus of mobile processes, ii. *Information and computation*, 100(1):41–77.

[17] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

[18] Orth, J. D., Thiele, I., and Palsson, B. Ø. (2010). What is flux balance analysis? *Nature biotechnology*, 28(3):245–248.

[19] Phillips, A. and Cardelli, L. (2007). Efficient, correct simulation of biological processes in the stochastic pi-calculus. *Computational Methods in Systems Biology*, 4695:184–199.

[20] Phillips, A., Cardelli, L., and Castagna, G. (2006). A graphical representation for biological processes in the stochastic pi-calculus. *Transactions in Computational Systems Biology*, 4230:123–152.

[21] Priami, C., Regev, A., Shapiro, E., and Silverman, W. (2001). Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information processing letters*, 80(1):25–31.

[22] Salway, J. G. (2013). *Metabolism at a Glance*. John Wiley & Sons.

[23] Schneider, H.-C. and Klabunde, T. (2013). Understanding drugs and diseases by systems biology? *Bioorganic & medicinal chemistry letters*, 23(5):1168–1176.

Appendix A

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.