CHAPTER 2

Representation of biochemical networks

2.1 Coupled chemical reactions

As was illustrated in the first chapter, a powerful and flexible way to specify a model is to simply write down a list of reactions corresponding to the system of interest. Note, however, that the reactions themselves specify only the qualitative structure of a model and must be augmented with additional information before they can be used to carry out a dynamic simulation on a computer. The model is completed by specifying the *rate* of every reaction, together with initial amounts of each reacting species.

Reconsider the auto-regulation example from Section 1.5.7:

$$\begin{aligned} g + P_2 &\rightleftharpoons g \cdot P_2 \\ g &\longrightarrow g + r \\ r &\longrightarrow r + P \\ 2P &\rightleftharpoons P_2 \\ r \longrightarrow \emptyset, &\qquad P \longrightarrow \emptyset. \end{aligned}$$

Although only six reactions are listed, there are actually eight, as two are reversible. Each of those eight reactions must have a rate law associated with it. We will defer a complete discussion of rate laws until Chapter 6. For now, it is sufficient to know that the rate laws quantify the propensity of particular reactions to take place and are likely to depend on the *current* amounts of available reactants. In addition there must be an *initial* amount for each of the five chemical species involved: $g \cdot P_2$, g, r, P, and P_2. Given the reactions, the rate laws, and the initial amounts (together with some assumptions regarding the underlying kinetics, which are generally not regarded as part of the model), the model is specified and can in principle be simulated dynamically on a computer.

The problem is that even this short list of reactions is hard to understand on its own, whereas the simple biologist's diagram (Figure 1.7) is not sufficiently detailed and explicit to completely define the model. What is needed is something between the biologist's diagram and the list of reactions.

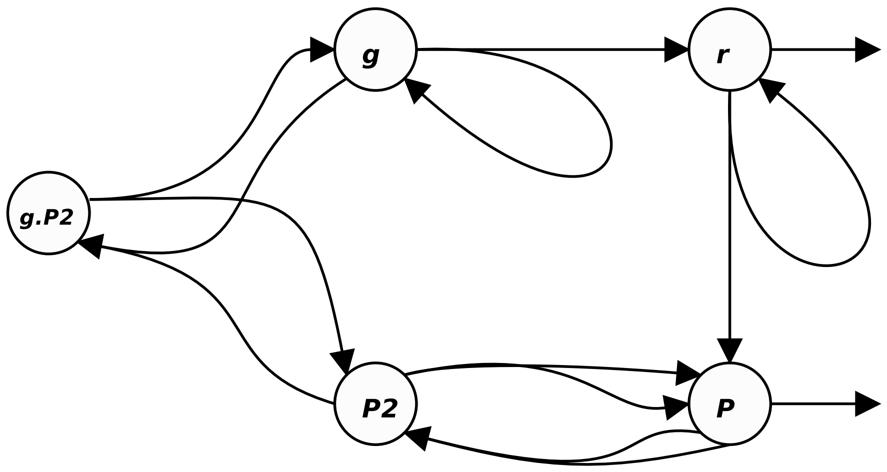

Figure 2.1 *A simple graph of the auto-regulatory reaction network.*

2.2 Graphical representations

2.2.1 Introduction

One way to begin to understand a reaction network is to display it as a pathway diagram of some description. The diagram in Figure 2.1 is similar to that used by some biological model-building tools such as CellDesigner.*

Such a diagram is easier to understand than the reaction list, yet it contains the same amount of information and hence could be used to generate a reaction list. Note that auto-regulation by its very nature implies a 'loop' in the reaction network, which is very obvious and explicit in the associated diagram. One possible problem with diagrams such as this, however, is the fact that it can sometimes be hard to distinguish which are the reactants and which are the products in any given reaction (particularly in large complex networks), and this can make it difficult to understand the flow of species through the network. Also, the presence of 'branching' and 'looping' arcs makes them slightly unnatural to work with directly in a mathematical sense.

Such problems are easily overcome by formalising the notion of pathway diagrams using the concept of a *graph* (here we mean the mathematical notion of a graph, not the idea of the graph of a function), where each *node* represents either a chemical species or a reaction, and arcs are used to indicate reaction pathways. In order to make this explicit, some elementary graph theoretic notation is helpful.

2.2.2 Graph theory

Definition 2.1 *A* directed graph *or* digraph, *$\mathcal{G}$ is a tuple (V, E), where $V = \{v_1, \ldots, v_n\}$ is a set of* nodes *(or* vertices*) and $E = \{(v_i, v_j) | v_i, v_j \in V,\ v_i \rightarrow v_j\}$ is a set of directed* edges *(or* arcs*), where we use the notation $v_i \rightarrow v_j$ if and only if there is a directed edge from node v_i to v_j.*

* Software web links tend to go out of date rather quickly, so a regularly updated list is available from the book's web page. See the links for Chapter 2.

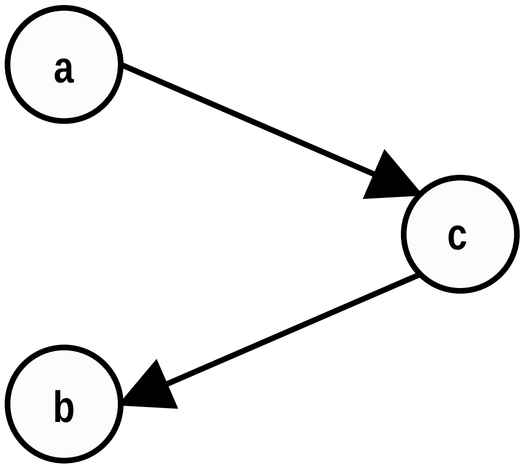

Figure 2.2 *A simple digraph.*

So the graph shown in Figure 2.2 has mathematical representation

$$\mathcal{G} = (\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}\ ,\ \{(\mathsf{a}, \mathsf{c}), (\mathsf{c}, \mathsf{b})\}).$$

Definition 2.2 *A graph is described as* simple *if there do not exist edges of the form* (v_i, v_i) *and there are no repeated edges. A* bipartite *graph is a simple graph where the nodes are partitioned into two distinct subsets* V_1 *and* V_2 *(so that* $V = V_1 \cup V_2$ *and* $V_1 \cap V_2 = \emptyset$*) such that there are no arcs joining nodes from the same subset.*

Referring back to the previous example, the partition $V_1 = \{\mathsf{a}, \mathsf{b}\},\ V_2 = \{\mathsf{c}\}$ gives a bipartite graph ($\mathcal{G}$ is said to be bipartite over the partition), and the partition $V_1 = \{\mathsf{a}\},\ V_2 = \{\mathsf{b}, \mathsf{c}\}$ does not (as the edge (c, b) would then be forbidden). A *weighted* graph is a graph which has (typically positive) numerical values associated with each edge.

2.2.3 Reaction graphs

It turns out that it is very natural to represent sets of coupled chemical reactions using weighted bipartite digraphs where the nodes are partitioned into two sets representing the species and reactions. An arc from a species node to a reaction node indicates that the species is a reactant for that reaction, and an arc from a reaction node to a species node indicates that the species is a product of the reaction. The weights associated with the arcs represent the stoichiometries associated with the reactants and products. There is a very strong correspondence between reaction graphs modelled this way, and the theory of Petri nets, which are used extensively in computing science for a range of modelling problems. A particular advantage of Petri net theory is that it is especially well suited to the discrete-event stochastic simulation models this book is mainly concerned with. It is therefore helpful to have a basic familiarity with Petri nets and their application to biological modelling.

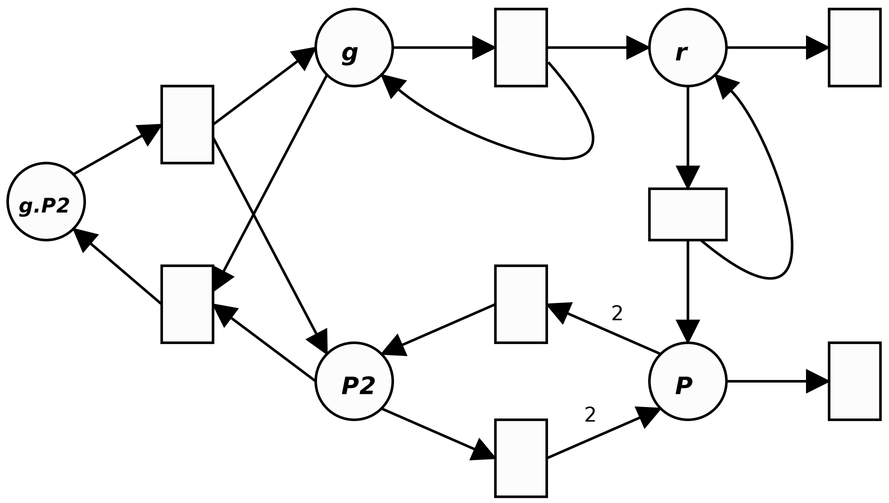

Figure 2.3 *A Petri net for the auto-regulatory reaction network.*

2.3 Petri nets

2.3.1 Introduction

Petri nets are a mathematical framework for systems modelling together with an associated graphical representation. Goss and Peccoud (1998) were among the first to use stochastic Petri nets for biological modelling. More recent reviews of the use of Petri nets for biological modelling include Pinney et al. (2003), Hardy and Robillard (2004), and Chaouiya (2007). A brief introduction to key elements of Petri net theory will be outlined here — see Reisig (1985) and Murata (1989) for further details.

By way of an informal introduction, the Petri net corresponding to the network shown in Figure 2.1 is shown in Figure 2.3. The rectangular boxes in Figure 2.3 represent individual reactions. The arcs into each box denote reactants, and the arcs out of each box denote products. Numbers on arcs denote the *weight* of the arc (unnumbered arcs are assumed to have a weight of 1). The weights represent reaction stoichiometries. The Petri net graph is only a slight refinement of the basic graph considered earlier, but it is easier to comprehend visually and more convenient to deal with mathematically (it is a bipartite graph). It is easy to see how to work through the graph and enumerate the full list of chemical reactions.

Traditionally each place (species) node of a place/transition (P/T) Petri net has an integer number of 'tokens' associated with it, representing the abundance of that 'species'. This fits in particularly well with the discrete stochastic molecular kinetics models we will consider in more detail later. Here, the number of tokens at a given node may be interpreted as the number of molecules of that species in the model at a given time (Figure 2.4). The collection of all token numbers at any given point in time is known as the current *marking* of the net (which corresponds here to the *state* of the reaction system). The Petri net shows what happens when particular transitions 'fire' (reactions occur). For example, in the above state, if two reactions occur, one a repression binding and the other a translation, the new Petri net will be as given in

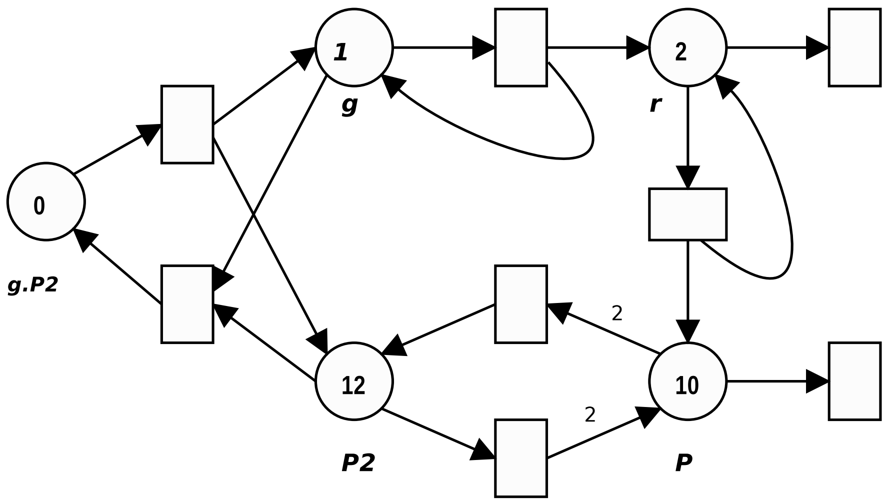

Figure 2.4 *A Petri net labelled with tokens.*

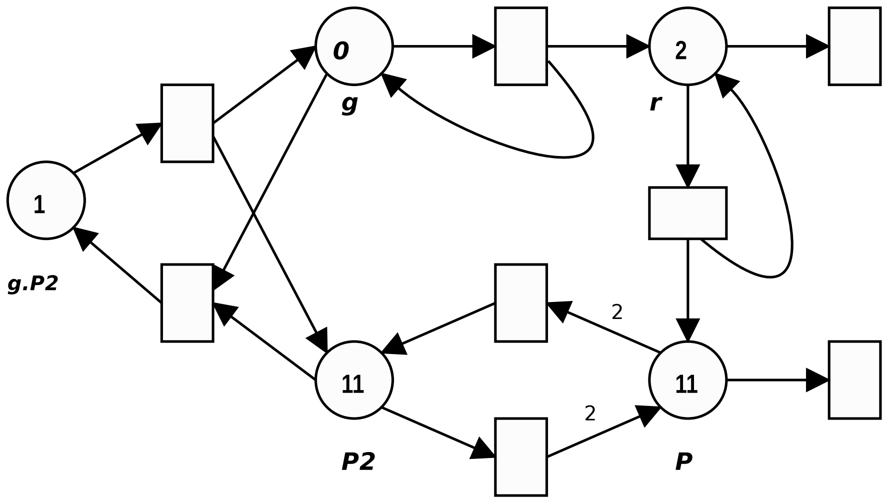

Figure 2.5 *A Petri net with new numbers of tokens after reactions have taken place.*

Figure 2.5. This is because the repression binding has the effect of increasing $g \cdot P_2$ by 1, and decreasing g and P_2 by 1, and the translation has the effect of increasing P by 1 (as r is both increased and decreased by 1, the net effect is that it remains unchanged). So the old and new Petri net markings can be written as

Species	No. tokens
$g \cdot P_2$	0
g	1
r	2
P	10
P_2	12

and

Species	No. tokens
$g \cdot P_2$	1
g	0
r	2
P	11
P_2	11

Table 2.1 *The auto-regulatory system displayed in tabular (matrix) form (zero stoichiometries omitted for clarity)*

	Reactants (Pre)					Products ($Post$)				
Species	$g \cdot P_2$	g	r	P	P_2	$g \cdot P_2$	g	r	P	P_2
Repression		1			1	1				
Reverse repression	1						1			1
Transcription		1					1	1		
Translation			1					1	1	
Dimerisation				2						1
Dissociation					1				2	
mRNA degradation			1							
Protein degradation				1						

respectively. A transition (reaction) can only fire (take place) if there are sufficiently many tokens (molecules) associated with each input place (reactant species). We are now in a position to consider Petri nets more formally.

2.3.2 *Petri net formalism and matrix representations*

Definition 2.3 *A Petri net, N, is an n-tuple $(P, T, Pre, Post, M)$, where $P = \{p_1, \ldots, p_u\}$, $(u > 0)$ is a finite set of* places, *$T = \{t_1, \ldots, t_v\}$, $(v > 0)$ is a finite set of* transitions, *and $P \cap T = \emptyset$. Pre is a $v \times u$ integer matrix containing the weights of the arcs going from places to transitions (the (i, j)th element of this matrix is the weight of the arc going from place j to transition i), and $Post$ is a $v \times u$ integer matrix containing the weights of arcs from transitions to places (the (i, j)th element of this matrix is the weight of the arc going from transition i to place j).*[†] *Note that Pre and $Post$ will both typically be* sparse *matrices.*[‡] *M is a u-dimensional integer vector representing the current* marking *of the net (i.e. the current state of the system).*

The initial marking of the net is typically denoted M_0. Note that the form of Pre and $Post$ ensure that arcs only exist between nodes of different types, so the resulting network is a bipartite graph. A particular transition, t_i can only fire if $M_j \geq Pre_{ij},\ j = 1, \ldots, u$.

In order to make this concrete, let us now write out the reaction list for Figure 2.3 in the form of a table, shown in Table 2.1. We can then use this to give a formal Petri net specification of the system.

[†] Non-existent arcs have a weight of zero.

[‡] A *sparse* matrix is a matrix consisting mainly of zeros. There are special algorithms for working with sparse matrices that are much more efficient than working with the full matrix directly. It is hard to be precise about how sparse a matrix has to be before it is worth treating as a sparse matrix, but for an $n \times n$ matrix, if the number of non-zero elements is closer to order n than order n^2, it is likely to be worthwhile using sparse matrix algorithms.

Table 2.2 *Table representing the overall effect of each transition (reaction) on the marking (state) of the network*

Species	$g \cdot P_2$	g	r	P	P_2
Repression	1	−1	0	0	−1
Reverse repression	−1	1	0	0	1
Transcription	0	0	1	0	0
Translation	0	0	0	1	0
Dimerisation	0	0	0	−2	1
Dissociation	0	0	0	2	−1
mRNA degradation	0	0	−1	0	0
Protein degradation	0	0	0	−1	0

$$N = (P, T, Pre, Post, M),\ P = \begin{pmatrix} g \cdot P_2 \\ g \\ r \\ P \\ P_2 \end{pmatrix},\ T = \begin{pmatrix} \text{Repression} \\ \text{Reverse repression} \\ \text{Transcription} \\ \text{Translation} \\ \text{Dimerisation} \\ \text{Dissociation} \\ \text{mRNA degradation} \\ \text{Protein degradation} \end{pmatrix}$$

$$Pre = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},\ Post = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},\ M = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 10 \\ 12 \end{pmatrix}.$$

Now when a particular transition (reaction) occurs (given by a particular row of Table 2.1), the numbers of tokens associated with each place (species) will decrease according to the numbers on the LHS (Pre) and increase according to the numbers on the RHS ($Post$). So, it is the *difference* between the RHS and the LHS that is important for calculating the change in state associated with a given transition (or reaction). We can write this matrix out in the form of a table, shown in Table 2.2, or more formally as a matrix

$$A = Post - Pre = \begin{pmatrix} 1 & -1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix}.$$

This 'net effect' matrix, A, is of fundamental importance in the theory and application of Petri nets and chemical reaction networks. Unfortunately there is no agreed standard notation for this matrix within either the Petri net or biochemical network literature. Within the Petri net literature, it is usually referred to as the *incidence matrix* and is often denoted by the letter A, C, or I.[§] Within the biochemical network field, it is usually referred to as the *reaction* or *stoichiometry matrix* and often denoted by the letter A or S. As if this were not confusing enough, the matrix is often (but by no means always) defined to be the *transpose* of the matrix we have called A,[¶] as this is often more convenient to work with. Suffice to say that care needs to be taken when exploring and interpreting the wider literature in this area. For clarity and convenience throughout this book, A (the reaction matrix) will represent the matrix as we have already defined it, and S (the stoichiometry matrix), will be used to denote its transpose.

Definition 2.4 *The* reaction matrix

$$A = Post - Pre$$

is the $v \times u$-dimensional matrix whose rows represent the effect of individual transitions (reactions) on the marking (state) of the network. Similarly, the stoichiometry matrix

$$S = A^{\mathsf{T}}$$

is the $u \times v$-dimensional matrix whose columns represent the effect of individual transitions on the marking of the network.[||]

However, it must be emphasised that this is not a universally adopted notation.

Now suppose that we have some reaction events. For example, suppose we have one repression binding reaction and one translation reaction. We could write this list of reactions in a table as

[§] I is a particularly bad choice, as this is typically used in linear algebra to denote the *identity matrix*, which has '1's along the diagonal and '0's elsewhere.

[¶] The transpose of a matrix is the matrix obtained by interchanging the rows and columns of a matrix. So in this case it would be a matrix where the rows represented places (species) and the columns represented transitions (reactions).

[||] Here and elsewhere, the notation $^{\mathsf{T}}$ is used to denote the transpose of a vector or matrix.

Reaction	No. transitions
Repression	1
Reverse repression	0
Transcription	0
Translation	1
Dimerisation	0
Dissociation	0
mRNA degradation	0
Protein degradation	0

or more neatly as a vector

$$r = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Note that in order to save space, column vectors such as r are sometimes written as the transpose of row vectors, e.g., $r = (1, 0, 0, 1, 0, 0, 0, 0)^{\mathsf{T}}$. We can now use matrix algebra to update the marking (state) of the network.

Proposition 2.1 *If r represents the transitions that have taken place subsequent to the marking M, the new marking $\tilde{M}$ is related to the old marking via the matrix equation*[**]

$$\tilde{M} = M + Sr. \tag{2.1}$$

Note that this equation (2.1) is of fundamental importance both for the mathematical analysis of Petri nets and biochemical networks and also for the development of simulation and inference algorithms. We will use this equation extensively in a variety of different ways throughout the book. Before we justify it, it is probably helpful to see a simple and direct application of it in practice.

In the context of our example, we can compute the new marking from the old

[**] Note that in matrix equations, addition is defined element-wise, but multiplication is defined in a special way. The product of the $n \times m$ matrix A and the $m \times p$ matrix B is the $n \times p$ matrix whose (i, j)th element is $\sum_{k=1}^{m} a_{ik} b_{kj}$. A vector is treated as a matrix with one column.

marking as

$$\begin{aligned}
\tilde{M} &= M + Sr \\
&= M + A^{\mathsf{T}} r \\
&= \begin{pmatrix} 0 \\ 1 \\ 2 \\ 10 \\ 12 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -2 & 1 \\ 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 1 \\ 2 \\ 10 \\ 12 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 2 & 0 & -1 \\ -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 1 \\ 2 \\ 10 \\ 12 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \\ -1 \end{pmatrix} \\
&= \begin{pmatrix} 1 \\ 0 \\ 2 \\ 11 \\ 11 \end{pmatrix}.
\end{aligned}$$

We can see that (2.1) appears to have worked in this particular example, but we need to establish why it is true in general.

Proof. The ith row of A represents the effect on the marking of the ith reaction. The same is true of the ith column of S, which we denote s^i. Clearly r_i is the number of type i reactions that take place, so the change in marking, $\tilde{M} - M$, is given by

$$\begin{aligned}
\tilde{M} - M &= s^1 r_1 + \cdots + s^v r_v \\
&= \sum_{i=1}^{v} s^i r_i
\end{aligned}$$

$$\begin{aligned}
&= \sum_{i=1}^{v} S e_i r_i \\
&= S \sum_{i=1}^{v} e_i r_i \\
&= Sr,
\end{aligned}$$

where e_i is the ith unit v-vector.[††]
□

2.3.3 Network invariants and conservation laws

There are two Petri net concepts that are of particular relevance to biochemical networks: P- and T-invariants.

Definition 2.5 *A* P-invariant *(sometimes referred to in the literature as an* S-invariant*) is a non-zero u-vector y that is a solution to the matrix equation $Ay = 0$. That is, y is any non-zero vector in the null-space of A.*[‡‡]

The null-space of A therefore characterises the set of P-invariants. These P-invariants are interesting because they correspond to *conservation laws* of the network.

In the example we have been studying, it is clear that the vector $y = (1, 1, 0, 0, 0)^\mathsf{T}$ is a P-invariant (as $Ay = 0$). This vector corresponds to the fairly obvious conservation law

$$g \cdot P_2 + g = \text{Constant}.$$

That is, the total number of copies of the gene does not change. It is true in general that if y is a P-invariant then the linear combination of states, $y^\mathsf{T} M$, is conserved by the reaction network. To see why this works, we can evaluate the current linear combination by computing $y^\mathsf{T} M$, where M is the current marking. Similarly, the value of the linear combination when the marking is $\tilde{M}$ is $y^\mathsf{T}\tilde{M}$. So the change in the linear combination is

$$\begin{aligned}
y^\mathsf{T}\tilde{M} - y^\mathsf{T}M &= y^\mathsf{T}(\tilde{M} - M) \\
&= y^\mathsf{T} S r \\
&= (S^\mathsf{T} y)^\mathsf{T} r \\
&= (Ay)^\mathsf{T} r \\
&= 0,
\end{aligned}$$

where the second line follows from (2.1) and the last line follows from the fact that y is a P-invariant.[§§]

[††] The ith unit vector, e_i, is the vector with a 1 in the ith position and zeros elsewhere. Multiplying a matrix by e_i has the effect of picking out the ith column.

[‡‡] The null-space of a matrix (sometimes known as the kernel) is defined to be the set of all vectors that get mapped to zero by the matrix.

[§§] We also used the fact that for arbitrary (conformable) matrices A and B, we have $(AB)^\mathsf{T} = B^\mathsf{T}A^\mathsf{T}$.

Definition 2.6 *A T-*invariant *is a non-zero, non-negative (integer-valued) v-vector x that is a solution to the matrix equation $Sx = 0$. That is, x is in the null-space of S.*

These invariants are of interest because they correspond to sequences of transitions (reactions) that return the system to its original marking (state). This is clear immediately from (2.1). If the primary concern is continuous deterministic modelling, then any non-negative solution is of interest. However, if the main interest is in discrete stochastic models of biochemical networks, only non-negative integer vector solutions correspond to sequences of transitions (reactions) that can actually take place. Hence, we will typically want to restrict our attention to these.

In the example we have been considering, it is easily seen that the vectors $x = (1, 1, 0, 0, 0, 0, 0, 0)^{\mathsf{T}}$ and $\tilde{x} = (0, 0, 0, 0, 1, 1, 0, 0)^{\mathsf{T}}$ are both T-invariants of our network. The first corresponds to a repression binding and its reverse reaction, and the second corresponds to a dimerisation and corresponding dissociation. However, not all T-invariants are associated with reversible reactions.

Although it is trivial to verify whether a given vector is a P- or T-invariant, it is perhaps less obvious how to systematically find such invariants and classify the set of all such invariants for a given Petri net. In fact, the singular value decomposition (SVD) is a classic matrix algorithm (Golub and Van Loan, 1996) that completely characterises the null-space of a matrix and its transpose and hence helps considerably in this task. However, if we restrict our attention to positive integer solutions, there is still more work to be done even once we have the SVD. We will revisit this issue in Chapter 11 when the need to find invariants becomes more pressing.

Before leaving the topic of invariants, it is worth exploring the relationship between the number of (linearly independent) P- and T-invariants. The *column-rank* of a matrix is the dimension of the image-space of the matrix (the space spanned by the columns of the matrix). The *row-rank* is the column-rank of the transpose of the matrix. It is a well-known result from linear algebra that the row and column ranks are the same, and so we can refer unambiguously to the *rank* of a matrix. By the *rank-nullity theorem*, the dimension of the image-space and null-space must sum to the dimension of the space being operated on, which is the number of columns of the matrix. So, if we fix on S, which has dimension $u \times v$, suppose the rank of the matrix is k. Let the dimension of the null-space of S be t and the dimension of the null-space of $A(= S^{\mathsf{T}})$ be p. Then we have $k + p = u$ and $k + t = v$. This leads immediately to the following result.

Proposition 2.2 *The number of linearly independent P-invariants, p, and the number of linearly independent T-invariants, t, are related by*

$$t - p = v - u. \tag{2.2}$$

In the context of our example, we have $u = 5$ and $v = 8$. As we have found a P-invariant, we know that $p \geq 1$. Now using (2.2) we can deduce that $t \geq 4$. So there are at least four linearly independent T-invariants for this network, and we have so far found only two.

2.3.4 Reachability

Another Petri net concept of considerable relevance is that of *reachability*.

Definition 2.7 *A marking $\tilde{M}$ is* reachable *from marking M if there exists a finite sequence of transitions leading from M to $\tilde{M}$.*

If such a sequence of transitions is summarised in the v-vector r, it is clear that r will be a non-negative integer solution to (2.1). However, it is important to note that the converse does not follow: the existence of a non-negative integer solution to (2.1) does not guarantee the reachability of $\tilde{M}$ from M. This is because the markings have to be non-negative. A transition cannot occur unless the number of tokens at each place is at least that required for the transition to take place. It can happen that there exists a set of non-negative integer transitions between two valid markings M and $\tilde{M}$, but all possible sequences corresponding to this set are impossible.

This issue is best illustrated with an example. Suppose we have the reaction network

$$\begin{aligned} 2X &\longrightarrow X_2 \\ X_2 &\longrightarrow 2X \\ X_2 &\longrightarrow X_2 + Y. \end{aligned}$$

So X can dimerise and dissociate, and dimers of X somehow catalyse the production of Y. If we formulate this as a Petri net with $P = (X, X_2, Y)$, and the transitions in the above order, we get the stoichiometry matrix

$$S = \begin{pmatrix} -2 & 2 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

If we begin by thinking about getting from initial marking $M = (2, 0, 0)^\mathsf{T}$ to marking $\tilde{M} = (2, 0, 1)^\mathsf{T}$, we see that it is possible and can be achieved with the sequence of transitions t_1, t_3, and t_2, giving reaction vector $r = (1, 1, 1)^\mathsf{T}$. We can now ask if marking $\tilde{M} = (1, 0, 1)^\mathsf{T}$ is reachable from $M = (1, 0, 0)^\mathsf{T}$. If we look for a non-negative integer solution to (2.1), we again find that $r = (1, 1, 1)^\mathsf{T}$ is appropriate, as $\tilde{M} - M$ is the same in both scenarios. However, this does not correspond to any legal sequence of transitions, as *no* transitions are legal from the initial marking $M = (1, 0, 0)^\mathsf{T}$. In fact, $r = (0, 0, 1)^\mathsf{T}$ is another solution, since $(1, 1, 0)^\mathsf{T}$ is a T-invariant. This solution is forbidden despite the fact that firing of t_3 will not cause the marking to go negative.

This is a useful warning that although the matrix representation of Petri net (and biochemical network) theory is powerful and attractive, it is not a complete characterisation — discrete-event systems analysis is a delicate matter, and there is much that systems biologists interested in discrete stochastic models can learn from the Petri net literature.

2.4 Stochastic process algebras

Stochastic process algebras are another way of representing reactive systems as collections of concurrent and independent ‘agents’ or ‘processes’ which interact with other processes via ‘reaction channels’. There are many variations on this approach; here we consider briefly the two methods that are most commonly used in the context of modelling biochemical networks.

2.4.1 Stochastic π-calculus

The usual approach to modelling biochemical networks with π-calculus is to model each molecule in the system as an independent ‘agent’ or ‘process’ which interacts with other processes via reaction channels. The stochastic π-calculus is a simple description of how processes are transformed, and in interaction with what other processes. This is most easily explained in the context of a very simple example. We will use the Lotka–Volterra example from Section 1.6:

$$\begin{aligned} Y_1 &\longrightarrow 2Y_1 \\ Y_1 + Y_2 &\longrightarrow 2Y_2 \\ Y_2 &\longrightarrow \emptyset. \end{aligned}$$

In its most basic form, the π-calculus requires all processes to pairwise interact, which corresponds to chemical reactions involving exactly two species. Here this can be accomplished by adding in dummy species D_1 and D_2 whose levels are unaffected by the reactions as follows:

$$\begin{aligned} D_1 + Y_1 &\xrightarrow{r_1} D_1 + 2Y_1 \\ Y_1 + Y_2 &\xrightarrow{r_2} 2Y_2 \\ D_2 + Y_2 &\xrightarrow{r_3} D_2. \end{aligned}$$

This is now in the form of a reaction system involving four species (processes) and three reaction channels (labelled r_1, r_2, r_3) which can be converted directly into the π-calculus. For each of the four species in the reaction system, we consider the reactions it can participate in, and write out a corresponding process rule. There are several variations on the precise notation used, but typically the resulting system will be presented in something like the following way:

$$\begin{aligned} D_1 &= ?r_1.D_1 \\ Y_1 &= !r_1.(Y_1|Y_1) + ?r_2.\emptyset \\ Y_2 &= !r_2.(Y_2|Y_2) + !r_3.\emptyset \\ D_2 &= ?r_3.D_2. \end{aligned}$$

The first line says that D_1 can only participate in reaction r_1, and that when it does, it is transformed to itself. The ? means that it is an ‘input’ for channel r_1, and as such can only interact with processes that ‘output’ on r_1. Note that the terms ‘input’ and ‘output’ are *not* synonymous with ‘reactants’ and ‘products’. They are just terms to identify the respective ends of the pairwise interaction between (reacting) processes.

The second line says that Y_1 can participate in either r_1 (as an output) or r_2 (as an input). If it participates in r_1, the result will be two copies of Y_1, and if it participates in r_2, it will simply disappear. The third and fourth lines are similar. Together the rules represent a description of the reaction system equivalent to the list of biochemical reactions.

In the context of the small networks considered in this text, it is difficult to see the advantage of this representation over the basic reaction list or corresponding Petri net representation, but advocates claim that the π-calculus scales much better than methods based on lists of reactions in the context of larger, more complex models. See Phillips and Cardelli (2007) and Blossey et al. (2006) for further details of this modelling approach.

2.4.2 *Bio-PEPA*

PEPA is a stochastic process algebra closely related to the π-calculus, aimed at analysis and evaluation of concurrent systems. Bio-PEPA is an extension specifically targeted at the modelling of biochemical reaction networks. In many ways, it is an attempt to formalise reaction models as described in SBML (discussed in the next section). Bio-PEPA adopts the 'species as processes' abstraction model, similarly to the Petri-net approach. In Bio-PEPA, the Lotka–Volterra model previously discussed could be represented as follows:

$$\begin{aligned} Y_1 &\overset{\text{def}}{=} (r_1,1)\downarrow Y_1 + (r_1,2)\uparrow Y_1 + (r_2,1)\downarrow Y_1 \\ Y_2 &\overset{\text{def}}{=} (r_2,1)\downarrow Y_2 + (r_2,2)\uparrow Y_2 + (r_3,1)\downarrow Y_2 \end{aligned}$$

$$Y_1(y_{1,0}) \underset{\{r_2\}}{\bowtie} Y_2(y_{2,0}).$$

The first line defines the transformations affecting species Y_1. The first term: $(r_1, 1)\downarrow Y_1$ means that Y_1 participates as a reactant (hence $\downarrow$) in reaction r_1, where it has stoichiometry 1. In the second term, $\uparrow$ is used to flag a product (which has stoichiometry 2 in reaction r_1). Just as for π-calculus, $+$ is used to delimit choices. The second line describes the transformations affecting Y_2. Line 3 completes the definition of the model, and describes the synchronisation requirements. Here species Y_1 and Y_2 must be synchronised for a reaction of type r_2. The values $y_{1,0}$ and $y_{2,0}$ represent the initial amounts (or concentrations) of Y_1 and Y_2, respectively, and hence can be used to initialise the state of the model.

Again, in the context of simple biochemical network models, it is difficult to see any great advantage of this approach over the stochastic Petri net representation. The advantages of Bio-PEPA over the stochastic π-calculus is that it more closely matches the structure of biochemical network models. For example, it explicitly encodes stoichiometry, allows synchronisations involving more than two reactants, and also allows the use of general rate laws (not discussed here). For further information about this approach to network modelling, see Ciocchetta and Hillston (2009) and references therein.

2.5 Systems Biology Markup Language (SBML)

Different representations of biochemical networks are useful for different purposes. Graphical representations (including Petri nets) are useful both for visualisation and analysis, and matrix representations (and other formal representations, such as process algebras) are useful for mathematical and computational analysis. The Systems Biology Markup Language (SBML), described in Hucka et al. (2003), is a way of representing biochemical networks that is intended to be convenient for computer software to generate and parse, thereby enabling communication of biochemical network models between disparate modelling and simulation tools. It is essentially an eXtensible Markup Language (XML) encoding (DuCharme, 1999) of the reaction list, together with the additional information required for quantitative modelling and simulation. It is intended to be independent of particular kinetic theories and should be as appropriate for discrete stochastic models as for continuous deterministic ones. The existence of a widely adopted exchange format for systems biology models makes it possible to develop a web-based community resource for archiving and sharing of models. The BioModels database, described in Le Novère et al. (2006), is just such a resource, and contains quantitative models covering an range of organisms and cellular processes, mainly encoded using SBML.

We will concentrate here on the version of SBML known as Level 3 (version 1, core), as this is the most popular variant of the recent specifications (at the time of writing) and contains sufficient features for the biochemical network models considered in this book. Further details regarding SBML, including the specification and XML Schema, can be obtained from the SBML.org website. Note that SBML should perhaps not be regarded as an alternative to other representations, but simply as an electronic format which could in principle be used in conjunction with any of the representations we have considered (though it corresponds most closely to the Petri net representation). Also note that it is not intended that SBML models should be generated and manipulated ‘by hand’ using a text editor, but rather by software tools which present to the user a more human-oriented representation. It is also worth bearing in mind that SBML continues to evolve. At the time of writing, SBML Level 3 (version 2, core) is the current specification, and various packages are available which extend the functionality and feature coverage of SBML in various ways. SBML (Levels 2 and 3) encodes all mathematical formulae using MathML (an XML encoding for mathematical notation) rather than as strings containing algebraic expressions. SBML Level 2 can also be used for encoding the discrete stochastic models covered in this text; this was discussed in detail in the first edition of this book (Wilkinson, 2006). SBML Level 1 is not sufficiently expressive to unambiguously encode discrete stochastic models, and should now be considered obsolete.

2.5.1 Basic document structure

An SBML (Level 3) model consists of lists of *functions*, *units*, *compartments*, *species*, *parameters*, *initial assignments*, *rules*, *constraints*, *reactions*, and *events*. Each of these lists is optional. We will concentrate here on units, compartments, species, parameters, and reactions, as these are sufficient for adequately describing most simple discrete stochastic models. This basic structure is encoded in SBML as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
    level="3" version="1">
  <model id="MyBiochemicalNetwork" name="My␣biochemical␣
      network" substanceUnits="item" timeUnits="second"
      volumeUnits="litre" extentUnits="item">
    <listOfUnitDefinitions>
     ...
    </listOfUnitDefinitions>
    <listOfCompartments>
     ...
    </listOfCompartments>
    <listOfSpecies>
     ...
    </listOfSpecies>
    <listOfParameters>
     ...
    </listOfParameters>
    <listOfReactions>
     ...
    </listOfReactions>
  </model>
</sbml>
```

Note that as part of the model declaration it is possible to declare default units for the model. This is a new feature in Level 3. The declaration given is likely to be appropriate for many discrete stochastic models, and means that in many cases a separate `<listOfUnitDefinitions>` section of the model will not be required. However, this is only the case if the model uses built-in SBML units, otherwise a units section will still be necessary. The `extentUnits` are used to define the units of the kinetic laws (to be discussed later), which have units of `extentUnits` per `timeUnits`.

2.5.2 Units

The (optional) units list allows definition and redefinition of the units used by the model. Discrete stochastic models encoded in SBML Level 2 often contained the following units declaration.

```
    <listOfUnitDefinitions>
      <unitDefinition id="substance">
        <listOfUnits>
          <unit kind="item"/>
        </listOfUnits>
      </unitDefinition>
    </listOfUnitDefinitions>
```

This declaration had the effect of changing the default substance units from the default value (*mole*) to *item*. The effect of this is that subsequent (unqualified) specifications of (and references to) amounts will be assumed to be in the unit of *item*. That is, amounts are interpreted as numbers of molecules rather than the default of numbers of moles. There are no default units in SBML Level 3, but this construct can still be used in an SBML Level 3 document if the model declaration includes `substanceUnits="substance"` rather than `substanceUnits="item"`. Although this is no longer necessary, it is likely to be a commonly encountered declaration in models that have been converted to SBML Level 3 from SBML Level 2. Units turn out to be a rather delicate issue. We will examine units in some detail later in the book when we look at kinetics and rate laws. For further details on using units with SBML, see the specification document which includes an example of a model encoded with discrete stochastic simulation in mind. Note that many simulators in current use ignore the units section of the SBML document. In practice, this means that many deterministic simulators will assume units of mole, and most stochastic simulators will assume units of item, irrespective of the content of this section. However, it is important to ensure that models are encoded correctly so that they are not misinterpreted in the future.

2.5.3 *Compartments*

The compartment list simply states the compartments in the model. So for a model with two compartments, the declaration might be as follows.

```
<listOfCompartments>
  <compartment id="Cell" spatialDimensions="3" size="1"/>
  <compartment id="Nucleus" spatialDimensions="3" size="1"/>
</listOfCompartments>
```

A simulatable model must have at least one compartment, and each compartment should be given an *id*. You may also specify the number of spatial dimensions the compartment has (typically 3), and a size (or volume) in the current model size units appropriate for the number of spatial dimensions (`volumeUnits` for compartments with 3 spatial dimensions).

2.5.4 *Species*

The species list simply states all species in the model. So, for the auto-regulatory network model we have been considering throughout this chapter, these could be declared using

```
<listOfSpecies>
  <species id="Gene" compartment="Cell" initialAmount="10"
       hasOnlySubstanceUnits="true"/>
  <species id="P2Gene" name="P2.Gene" compartment="Cell"
      initialAmount="0" hasOnlySubstanceUnits="true"/>
  <species id="Rna" compartment="Cell" initialAmount="0"
      hasOnlySubstanceUnits="true"/>
  <species id="P" compartment="Cell" initialAmount="0"
      hasOnlySubstanceUnits="true"/>
  <species id="P2" compartment="Cell" initialAmount="0"
      hasOnlySubstanceUnits="true"/>
</listOfSpecies>
```

There are several things worth pointing out about this declaration. First, the initial amounts are assumed to be in the model substance units, unless explicit units are specified (see the specification for how to do this). Also note that each species is declared with the attribute `hasOnlySubstanceUnits`. This has the effect of ensuring that wherever the species is referred to elsewhere in the model (for example, in rate laws), it will be interpreted as an amount (in the appropriate substance units), and not a concentration (substance per size). Most stochastic simulators will make this assumption anyway, but it is important to encode models correctly so that they will not be misinterpreted by tools which correctly interpret the specification. Each species also declares the compartment in which it resides. For species using substance units, this is largely a model annotation, but for species with concentration units, it provides a link to the size information needed to calculate the concentration.

2.5.5 Parameters

The parameters section can be used to declare names for numeric values to be used in algebraic formulae. They are most often used to declare rate constants for the kinetic laws of biochemical reactions, but can be used for other variables as well. An example parameter section might be as follows.

```
<listOfParameters>
 <parameter id="k1" value="0.01"/>
 <parameter id="k2" value="0.1"/>
</listOfParameters>
```

Parameters defined here are ‘global’ to the whole model. In contrast, any parameters defined in the context of a particular reaction will be local to the kinetic law for that reaction only.

2.5.6 Reactions

The reaction list consists of a list of reactions. A reaction in turn consists of a list of reactants, a list of products, and a rate law. A reaction may also declare ‘modifier’ species — species that are not created or destroyed by the reaction but figure in the rate law. For example, consider the following encoding of a dimerisation reaction.

```
      <reaction id="Dimerisation" reversible="false">
        <listOfReactants>
          <speciesReference species="P" stoichiometry="2"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="P2" stoichiometry="1"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci> k4 </ci>
              <cn> 0.5 </cn>
              <ci> P </ci>
              <apply>
                <minus/>
                <ci> P </ci>
                <cn type="integer"> 1 </cn>
              </apply>
            </apply>
          </math>
          <listOfLocalParameters>
            <localParameter id="k4" value="1"/>
          </listOfLocalParameters>
        </kineticLaw>
      </reaction>
```

There are several things to note about this declaration. One thing that perhaps appears strange is that the reaction is declared to be not reversible when we know that dimerisation is typically a reversible process. However, reactions should only be flagged as reversible when the associated kinetic law represents the combined effects of both forward and backward reactions. It turns out that while this is fine for continuous deterministic models, there is no satisfactory way to do this for discrete stochastic models. As a result, when developing a model for discrete stochastic simulation, the forward and backward reactions must be specified separately (and declared not reversible) along with their separate kinetic laws. We will examine the meaning and specification of reaction rates and kinetic laws later in this book. The specification of reactants and products and their associated stoichiometries is fairly self-explanatory. The kinetic law itself is a MathML (W3C, 2000) encoding of the simple algebraic formula `k4*0.5*P*(P-1)`. Rate laws will be discussed in more detail later, but it is important to know that the units of this law are of the form *substance / time*, using the model substance and time units. The kinetic law given above uses a local parameter `k4`. This constant will always be used in the formula of the kinetic law, masking any global parameter of the same name. To use a global parameter called `k4`, the entire section

```
<listOfParameters>
 <parameter name="k4" value="1"/>
</listOfParameters>
```

should be removed from the kinetic law. Kinetic laws can use a mixture of global and local parameters. Any reference to a compartment will be replaced by the size (volume) of the compartment. A list of reactions should be included in the SBML file between `<listOfReactions>` and `</listOfReactions>` tags.

2.5.7 The full SBML model

The various model components are embedded into the basic model structure. For completeness, Appendix A.1 lists a full SBML model for the simple auto-regulatory network we have been considering. Note that as this model uses locally specified parameters rather than globally defined parameters, there is no `<listOfParameters>` section in the model definition. This model can also be downloaded from the book's website.

2.6 SBML-shorthand

2.6.1 Introduction

SBML has become the *lingua franca* for electronic representation and exchange of models of interest in systems biology. Dozens of different software tools provide SBML support to varying degrees, many providing both SBML import and export provisions, and some using SBML as their native format. However, while SBML is a good format for computers to parse and generate, its verbosity, pedantic syntax, and low signal-to-noise ratio make it rather inconvenient for humans to read and write. I have found it helpful to develop a shorthand notation for SBML that is much easier for humans to read and write, and can easily be 'compiled' into SBML for subsequent import into other SBML-aware software tools. The notation can be used as a partial substitute for the numerous GUI-based model-building tools that are widely used for systems biology model development. An additional advantage of the notation is that it is much more suitable than raw SBML for presentation in a book such as this (because it is more concise and readable). Many of the examples discussed in subsequent chapters will be presented using the shorthand notation, so it is worth presenting the essential details here. Here we describe a particular version of the shorthand notation, known as 3.1.1. A compiler for translating the shorthand notation into full SBML is freely available; see Appendix B.2 for details.

2.6.2 Basic structure

The description format is plain ASCII text. The suggested file extension is `.mod`, but this is not required. All whitespace other than carriage returns is insignificant (unless it is contained within a quoted 'name' element). Carriage returns are significant. The description is case-sensitive. Blank lines are ignored. The comment character is # — all text from a # to the end of the line is ignored.

The model description must begin with the characters **`@model:`**`3.1.1=` (the 3.1.1 corresponds to the version number of the specification). The text following the = on the first line is the model identification string (ID). An optional model name may also be specified, following the ID, enclosed in double quotes. This model declaration line may be followed by an optional additional line declaring model units. For example, the shorthand text

```
@model:3.1.1=AutoRegulatoryNetwork "Auto-regulatory network"
 s=item, t=second, v=litre, e=item
```

will be translated to

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
    level="3" version="1">
  <model id="AutoRegulatoryNetwork" name="Auto-regulatory␣
      network" substanceUnits="item" timeUnits="second"
      volumeUnits="litre" extentUnits="item">
```

The model is completed with the specification of up to seven additional sections: **`@units`**, **`@compartments`**, **`@species`**, **`@parameters`**, **`@rules`**, **`@reactions`**, and **`@events`**, each corresponding to an analogous SBML section. Here we concentrate on the five sections, **`@units`**, **`@compartments`**, **`@species`**, **`@parameters`**, and **`@reactions`**, corresponding to the SBML sections, `<listOfUnitDefinitions>`, `<listOfCompartments>`, `<listOfSpecies>`, `<listOfParameters>`, and `<listOfReactions>`, respectively. The sections must occur in the stated order. Sections are optional, but if present, may not be empty. These are the only sections covered by this specification.

2.6.3 Units

The format of the individual sections will be explained mainly by example. The following SBML-shorthand

```
@units
 substance=item
 mmls=mole:s=-3; litre:e=-1; second:e=-1
```

would be translated to

```
    <listOfUnitDefinitions>
      <unitDefinition id="substance">
        <listOfUnits>
          <unit kind="item" exponent="1" scale="0" multiplier=
              "1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="mmls">
        <listOfUnits>
          <unit kind="mole" exponent="1" scale="-3" multiplier
              ="1"/>
          <unit kind="litre" exponent="-1" scale="0"
              multiplier="1"/>
          <unit kind="second" exponent="-1" scale="0"
              multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
    </listOfUnitDefinitions>
```

The unit attributes `exponent`, `multiplier`, and `scale` are denoted by the letters `e`, `m`, and `s`, respectively. Note that because there is no way to refer to units elsewhere in

SBML-shorthand, the only function for this section is to redefine model-wide units such as `substance` and `size` (named in the model declaration).

2.6.4 Compartments

The following SBML-shorthand

```
@compartments
 Cell=1
 Nucleus=0.1
 Cyto=0.8 "Cytoplasm"
 Mito "Mitochondria"
```

would be translated to

```
    <listOfCompartments>
      <compartment id="Cell" size="1"/>
      <compartment id="Nucleus" size="0.1"/>
      <compartment id="Cyto" name="Cytoplasm" size="0.8"/>
      <compartment id="Mito" name="Mitochondria"/>
    </listOfCompartments>
```

Note that if a `name` attribute is to be specified, it should be specified at the end of the line in double quotes. This is true for other SBML elements too.

2.6.5 Species

The following shorthand

```
@species
 cell:Gene = 10b "The Gene"
 cell:P2=0
 cell:S1=100 s
 cell:[S2]=20 sc
 cell:[S3]=1000 bc
 mito:S4=0 b
```

would be translated to

```
    <listOfSpecies>
      <species id="Gene" name="The␣Gene" compartment="cell"
               initialAmount="10" boundaryCondition="true"/>
      <species id="P2" compartment="cell" initialAmount="0"/>
      <species id="S1" compartment="cell" initialAmount="100"
               hasOnlySubstanceUnits="true"/>
      <species id="S2" compartment="cell"
               initialConcentration="20"
               hasOnlySubstanceUnits="true" constant="true"/>
      <species id="S3" compartment="cell"
               initialConcentration="1000"
               boundaryCondition="true" constant="true"/>
      <species id="S4" compartment="mito" initialAmount="0"
               boundaryCondition="true"/>
    </listOfSpecies>
```

Compartments are compulsory. An `initialConcentration` (as opposed to an `initialAmount`) is flagged by enclosing the species `id` in brackets. The boolean attributes `hasOnlySubstanceUnits`, `boundaryCondition`, and `constant` can be set to `true` by appending the letters `s`, `b`, and `c`, respectively. The order of the flags is not important.

2.6.6 Parameters

The section

```
@parameters
 k1=1
 k2=10
```

would be translated to

```
  <listOfParameters>
   <parameter name="k1" value="1"/>
   <parameter name="k2" value="10"/>
  </listOfParameters>
```

2.6.7 Reactions

Each reaction is specified by exactly two or three lines of text. The first line declares the reaction name and whether the reaction is reversible (**@rr**= for reversible and **@r**= otherwise). The second line specifies the reaction itself using a fairly standard notation. The (optional) third line specifies the full rate law for the kinetics. If local parameters are used, they should be declared on the same line in a comma-separated list (separated from the rate law using a `:`).

So for example,

```
@reactions
@r=RepressionBinding "Repression Binding"
 Gene + 2P -> P2Gene
 k2*Gene
@rr=Reverse
 P2Gene -> Gene+2P
 k1r*P2Gene : k1r=1,k2=3
@r=NoKL
 Harry->Jim
@r=Test
 Fred -> Fred2
 k4*Fred : k4=1
```

would translate to

```
<listOfReactions>
  <reaction id="RepressionBinding" name="Repression␣
      Binding" reversible="false">
    <listOfReactants>
      <speciesReference species="Gene" stoichiometry="1"/>
      <speciesReference species="P" stoichiometry="2"/>
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="P2Gene" stoichiometry="1"
          />
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> k2 </ci>
          <ci> Gene </ci>
        </apply>
      </math>
    </kineticLaw>
  </reaction>
  <reaction id="Reverse" reversible="true">
    <listOfReactants>
      <speciesReference species="P2Gene" stoichiometry="1"
          />
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="Gene" stoichiometry="1"/>
      <speciesReference species="P" stoichiometry="2"/>
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci> k1r </ci>
          <ci> P2Gene </ci>
        </apply>
      </math>
      <listOfLocalParameters>
        <localParameter id="k1r" value="1"/>
        <localParameter id="k2" value="3"/>
      </listOfLocalParameters>
    </kineticLaw>
  </reaction>
  <reaction id="NoKL" reversible="false">
    <listOfReactants>
      <speciesReference species="Harry" stoichiometry="1"/
          >
    </listOfReactants>
```

```
        <listOfProducts>
          <speciesReference species="Jim" stoichiometry="1"/>
        </listOfProducts>
      </reaction>
      <reaction id="Test" reversible="false">
        <listOfReactants>
          <speciesReference species="Fred" stoichiometry="1"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="Fred2" stoichiometry="1"/
              >
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci> k4 </ci>
              <ci> Fred </ci>
            </apply>
          </math>
          <listOfLocalParameters>
            <localParameter id="k4" value="1"/>
          </listOfLocalParameters>
        </kineticLaw>
      </reaction>
    </listOfReactions>
```

For information on the other SBML-shorthand sections, see the specification document on the SBML-shorthand website.

2.6.8 Example

The auto-regulatory network whose SBML is given in Appendix A.1 can be represented in SBML-shorthand in the following way.

```
@model:3.1.1=AutoRegulatoryNetwork "Auto-regulatory network"
 s=item, t=second, v=litre, e=item
@compartments
 Cell
@species
 Cell:Gene=10 s
 Cell:P2Gene=0 s "P2.Gene"
 Cell:Rna=0 s
 Cell:P=0 s
 Cell:P2=0 s
@reactions
@r=RepressionBinding "Repression binding"
 Gene+P2 -> P2Gene
 k1*Gene*P2 : k1=1
@r=ReverseRepressionBinding "Reverse repression binding"
```

```
 P2Gene -> Gene+P2
 k1r*P2Gene : k1r=10
@r=Transcription
 Gene -> Gene+Rna
 k2*Gene : k2=0.01
@r=Translation
 Rna -> Rna+P
 k3*Rna : k3=10
@r=Dimerisation
 2P -> P2
 k4*0.5*P*(P-1) : k4=1
@r=Dissociation
 P2 -> 2P
 k4r*P2 : k4r=1
@r=RnaDegradation "RNA Degradation"
 Rna ->
 k5*Rna : k5=0.1
@r=ProteinDegradation "Protein degradation"
 P ->
 k6*P : k6=0.01
```

2.7 Exercises

1. Go to the book's website* and follow the Chapter 2 links to explore the world of SBML and SBML-aware software tools. In particular, download and read the SBML Level 3 specification document.
2. Consider this simple model of Michaelis–Menten enzyme kinetics

$$\begin{aligned} S + E &\longrightarrow SE \\ SE &\longrightarrow S + E \\ SE &\longrightarrow P + E. \end{aligned}$$

   (a) Represent this reaction network graphically using a Petri net–style diagram.
   (b) Represent it mathematically as a Petri net, $N = (P, T, Pre, Post, M)$, assuming that there are currently 100 molecules of substrate S, 20 molecules of the enzyme E, and no molecules of the substrate-enzyme complex SE or the product P.
   (c) Calculate the reaction and stoichiometry matrices A and S.
   (d) If the first reaction occurs 20 times, the second 10, and the last 5, what will be the new state of the system?
   (e) Can you find a different set of transitions that will lead to the same state?
   (f) Can you identify any P- or T-invariants for this system?
   (g) Write the model using SBML-shorthand (do not attempt to specify any kinetic laws yet).

* URL: `https://github.com/darrenjw/smfsb`

(h) Hand-translate the SBML-shorthand into SBML, then validate it using the on-line validation tool at the the SBML.org website.

(i) Download and install the SBML-shorthand compiler and use it to translate SBML-shorthand into SBML.

(j) Download and install some SBML-aware model-building tools. Try loading your valid SBML model into the tools, and also try building it from scratch (at the time of writing, COPASI and CellDesigner are popular tools — there should be links to these from the SBML.org website).

2.8 Further reading

Murata (1989) provides a good tutorial introduction to the general theory of P/T Petri nets and Haas (2002) is the definitive guide to stochastic Petri nets. Pinney et al. (2003) and Hardy and Robillard (2004) give introductions to the use of Petri nets in systems biology, and Goss and Peccoud (1998) explore the use of stochastic Petri nets (SPNs) for discrete-event simulation of stochastic kinetic models. Also see Chaouiya (2007) for a more recent review, and Koch et al. (2010) for a recent comprehensive guide to the use of Petri nets in systems biology modelling.

Information on SBML and all things related can be found at the SBML.org website. In particular, background papers on SBML, the various SBML specifications, SBML models, tools for model validation and visualisation, and other software supporting SBML can all be found there.

See Phillips and Cardelli (2007) and Blossey et al. (2006) for further information regarding stochastic π-calculus, and Ciocchetta and Hillston (2009) for an overview of the Bio-PEPA stochastic process algebra.