Abstract

Computer simulations have been used in biology for a long time, but it is only recently that simulations of a whole cell at a time have been considered, due to the enormous complexity and computation costs. However, the method is without doubt very promising, and may soon become computationally feasible thanks to the rapid development of computer hardware.

There are several possible uses of whole-cell computer simulations. A trivial use is to verify that the underlying model behaves as its real-world counterpart in different environments. Provided the model is good enough, simulation can also be used to predict the outcome of experiments. This way, it may be possible to simulate some experiments that simply cannot be carried out on real cells.

The goal of the Smartcell project is to provide a framework where a biologist can construct cell models and perform simulations on them in an intuitive way. The intended user group requires that the communication with the user be in biological terms, and that the physics and computer science parts of modelling and simulation be hidden inside the framework.

This work is the first part of the Smartcell project, preparing the ground for further developments. The focus lies on modelling issues and simulation mechanisms, the major part being a comprehensive description of the notation to be used for cell models. In most cases the alternative design choices are presented, and a motivation given for the actual choice. An effective simulation algorithm that traces out the time evolution of the model is also included, as well as a description of an intermediate format used by the simulation engine, and a mechanism for automatically converting a Smartcell model to the intermediate format. Finally, there is a sample model written in the Smartcell modelling format that demonstrates several modelling techniques.

Populärvetenskaplig sammanfattning

Datorsimuleringar har använts länge inom biologin, men det är först på senare tid som man har börjat undersöka möjligheterna att simulera en hel cell på en gång. Å ena sidan är det en enorm uppgift, å andra sidan blir datorerna bättre hela tiden. Och oavsett hur användbara simuleringarna visar sig vara, är det intressant att undersöka vad som är viktigt för att en cellmodell ska bli realistisk.

Cellsimuleringar har flera tänkbara användningsområden. Man kan t ex testa att den modell av cellen man har beter sig som en riktig cell, eller se vad som händer när man utsätter den för olika behandlingar. På det sättet kan man göra experiment som inte låter sig göras på riktiga celler.

Smartcell är ett projekt som går ut på att konstruera ett ramverk för cellmodellering och -simulering. Tanken är att biologer ska kunna använda ramverket för att enkelt kunna konstruera och testa modeller, utan att behöva bekymra sig alltför mycket om de bakomliggande fysiska detaljerna. Därför krävs det att de begrepp som används vid modellkonstruktion och simulering känns naturliga för en biolog.

Detta examensarbete är det första förberedande steget i Smartcellprojektet. Det som tas upp är huvudsakligen vad som bör finnas med i en cellmodell och hur man ska kunna beskriva det i ett formellt språk, dvs på ett sätt som kan tolkas maskinellt på ett entydigt sätt. Dessutom beskrivs en metod för att utföra simuleringsförsök på en sådan modell. Slutligen presenteras en exempelmodell som visar hur några vanliga mekanismer i cellen kan modelleras.

Contents

1 Introduction

The cell is the fundamental unit of life. A cell consists of a small volume, separated from its surroundings by a cell membrane and sometimes also by a cell wall. The inside of the cell is a viscous fluid, filled with molecules ranging in size from small ions to large biopolymers, as well as larger structures made from these building blocks. The living cell is never at rest; it is constantly interacting with its surroundings and using the energy gained from metabolism to grow and propagate (with the exception of specialised cells in a multicellular organism). Even though it may seem simple, the function of a cell can be very complex indeed, and the understanding of the cell is of enormous importance to biology.

Recently, several new methods have been developed that allow researchers to study the cell more closely. Several genomes have been completely sequenced, including those of *E coli*, *A thaliana* and *S cerevisiae* (baker's yeast), and many more are currently in progress. Extensive yeast-two-hybrid screenings are being carried out to find protein–protein interactions (see e g Uetz *et al*, 2000). Marker proteins such as GFP (green fluorescent protein) have been fused to other proteins to see where in the cell they are located. Microarray assays generate vast amounts of gene expression data. What can we do with all this information? One approach is to reconstruct the function of the cell by means of a model. Models can be used for testing hypotheses, but they can also be seen as a way of organising experimental data in a more manageable and understandable structure. Although it may seem optimistic to include everything that is known today in one enormous model, maybe this goal could at least be partially reached in five or ten years.

To do this we need the right tools. This paper describes the first step towards such a tool, called Smartcell. More specifically, by means of its model description format, Smartcell can be used for constructing models of the cell. Smartcell also provides a simulation engine, with the help of which the behaviour of the model can be investigated.

1.1 Modelling and Simulation in Biology

The use of computers as a modelling and simulation tool in biology is certainly not new. The first efforts, dating back to the late 1950s, were programs specially written to simulate a certain feature of the cell (see review by Garfinkel, 1981). The next step, a natural one as computers became more common in the labs, was the development of more flexible and user friendly programs. A number of programs—see Mendes (1997) for a list of some representative ones—have been developed, that can be used to investigate the properties of biochemical pathways. These programs all use differential equations to describe the system. There is also a freely available program called StochSim that uses stochastic kinetics, that has been used to successfully model chemotaxis in bacteria (Morton-Firth, 1998).

Gene networks are another target for modelling and simulation. Several approaches are described in the reviews by McAdams and Arkin (1998) and Smolen *et al* (2000). In the most basic model, each gene is represented by a boolean switch that can be either ON or OFF. The state of the switch depends on the state of other genes, and the whole network is updated synchronously

in discrete time steps. Another alternative is to allow the activation range of the gene to be continuous and describe the interactions between genes as differential equations. Time delays can be introduced to make the model more physically accurate. It has been shown that the delays due to diffusion and biopolymer synthesis are necessary for the correct function of certain gene circuits (McAdams and Shapiro, 1995).

Recently, a new category of tools has appeared, that aims at the modelling and simulation of a whole cell at once. To my knowledge, two such projects exist so far. The Virtual Cell (Schaff and Loew, 1999) is one of them. It has a distributed architecture where the model editor, implemented as a java applet, executes on a client computer, but the actual computations are carried out on a simulation server. The Virtual Cell aims at very large, integrative models, typically of eukaryotic cells. The spatial distribution of species is taken into account, and it has a compartment structure for the transport of species between organelles. All processes are modelled as differential equations. Gene expression is not explicitly considered, but it can of course also be modelled using differential equations. The Virtual Cell has been used to model a calcium wave in a neuronal cell. The other project is called E-Cell (Tomita *et al*, 1999). The group behind this project has built an impressive model of *Mycoplasma genitalis*, including 127 genes. The model uses differential equations for the chemical kinetics. The paper also mentions the use of stochastic gene expression, but at the same time claims that differential equations are used for the same purpose. It is not clear how this is done. The relations between species are defined in a rules file. The Virtual Cell and E-Cell both come close to the aim of Smartcell, so they are the standards by which the usability and performance of Smartcell should be compared.

1.2 The Aim of This Project

The aim of this project is to design and build a solid framework for cell modelling and simulation. Models constructed using the framework should be able to include all important features of the cell, at the desired level of detail. It should be possible to partition the model into modules, so that modules can be shared with the research community. Finally, the framework must be easy to use.

A new model description format has been developed with these requirements in mind, based on XML (Extensible Markup Language). The intention was to keep the model description format as independent as possible from the rest of the framework; previous model formats have often been tightly connected to their respective software, making the model non-portable. This independence has been maintained as far as possible; however, some features of the framework, such as the underlying kinetics model, necessarily shine through.

Contrary to most of the previously mentioned modelling and simulation software, Smartcell is based on a stochastic kinetics model. We hope that, in addition to being more physically correct, this choice will decrease the simulation times for large models. The reason for this is that we expect many species to be localised to parts of the cell. In a stochastic model, this reduces the state space and reduces the number of events that are enabled at any time. The result is a smaller model and a faster simulation.

2 Cell Modelling and Simulation

2.1 The Model

A *model* is an abstraction of a system; a collection of hypotheses and facts that describe the behaviour of the system under certain circumstances. (My definition, adapted from Schaff and Loew, 1999).

A model can be constructed in several ways, depending on the purpose of modelling and on the a priori knowledge about the system. As described in Heinrich *et al* (1977) and Morton-Firth (1998), there are two extremes of model-building strategies that can be clearly distinguished, called the integrative and the reductionist strategy respectively. An integrative model tries to be as complete as possible and to reproduce as many experimental observations as possible. It will be difficult to extract any information from such a model in itself, as it is almost as complex as the real system. A reductionist model, on the other hand, aims to be as simple as possible. It will necessarily focus on the features considered most important and ignore the rest of the system. Such a model may have important pedagogical properties, but it will be useless for predicting the behaviour of the real system. Because of its intended use, a Smartcell model is typically of the integrative type.

Another design choice that must be made is the semantic level of the model. A high semantic level, in this case, means that the model description makes sense to a biologist; the terms that are used to define model features are the same words used by the biologist to describe real cells. If, on the other hand, the model is described using an abstract mathematical notation, the terms used when constructing the model carry little biological meaning and so have a low semantic level. The Smartcell model description format is designed to be at a high semantic level, although some mathematical notation is sometimes required, e g for reaction rates. Apart from making it more easy to use for the intended user group, a high semantic level also has obvious advantages when it comes to the interpretation of simulation results. However, the high semantic level is not suitable for simulation, and so a more compact format (the "core model format") is used internally in the simulation engine. The conversion to this format is done automatically, as will be described later.

2.1.1 The Model Description Format

There are several ways of making a machine-readable model representation. A survey was made of the formats used by present modelling/simulation packages. Unfortunately, most packages use their own proprietary format. This solution is not very flexible and it also requires a special parser. A better idea is to use a format that is already in use, but to my knowledge there are no suitable formats available. Schaff and Loew (1999) suggest that their model description format be used as a standard, but it has not been made publicly available, and it is probably too specific to be useful for this purpose anyway. The E-Cell rule file format (Tomita *et al*, 1999) also seems to specific (and not very user friendly).

Another approach that has been tried is the puritan object oriented way (Stoffers *et al*, 1992). Here, the model is defined in source code that uses base classes for enzymes, reactions, etc. This code is then interpreted as is, or com-

piled into a runnable simulation engine. This approach asks a lot from the user, and therefore is not well suited to our needs.

Finally, there is the possibility of using a general-purpose markup language, such as XML (Extensible Markup Language). XML is already being used for databases, web pages, mathematics and for other purposes more related to biology, including bioinformatics. XML documents are text files, containing plain text and markup tags just like HTML documents. The markup tags define the semantic meaning of the text they contain. Here is an example, showing how the text of a book can be written in a fictious "book markup language", together with the XML tags that add some meta-information such as the book title and the division into chapters:

```
<book author="William Gibson" title="Neuromancer">
  <chapter>... first chapter ...</chapter>
  <chapter>... second chapter ...</chapter>
</book>
```

Apart from the fact that model descriptions written in text-based markup can be expected to be very large, this choice has none of the drawbacks of the other formats. It also has the advantage of being human-readable and -writable.

In this case, the XML document describes part of a biological model, and so a set of markup tags or elements that can be used for this purpose is needed. Such a set of elements, together with the rules that define how they may be combined, is called an XML vocabulary and is formally described in a document type definition (DTD). The DTD for Smartcell model description files can be found in appendix A, together with a sample model that is intended to shed some light on the rather abstract XML element descriptions that follow.

An important feature is that a model can be split into several files, or modules. One reason for this is that it makes the model more manageable. Second, modules can be shared with the research community. Third, it makes it possible to "plug in" only those modules that are believed to be of interest for a particular simulation.

2.1.2 Presenting the Modelling Target

This work focuses on the prokaryotic cell. There are several reasons for this: compared to a eukaryotic cell, the prokaryotic cell is small, it has a simpler structure (no nucleus, no mitochondria, no microtubules etc) and a simpler function as well (e g no splicing). More specifically, the primary target of simulation is the bacterium *Escherichia coli*, which is a popular model system and therefore well characterised (see Neidhardt *et al*, 1990, for a comprehensive presentation).

Escherichia coli (or *E coli* for short) is a gram-negative, rod-shaped bacterium which is typically about 1.5 μm long and has a cross-section diameter of about 1 μm (figure 1). The interior of the cell, the cytosol, is covered by a membrane called the cell membrane or cytoplasmic membrane. This membrane is in turn surrounded by another membrane called the outer membrane, a special feature of gram-negative bacteria. The membranes restrict the passage of macromolecules and some small molecules. The space between the cell membrane and the outer membrane is called the periplasmatic space. The solute composition of the periplasmatic space is very different from that on either side of

the enclosing membranes. Outermost there is a cell wall to protect the cell and maintain its shape.

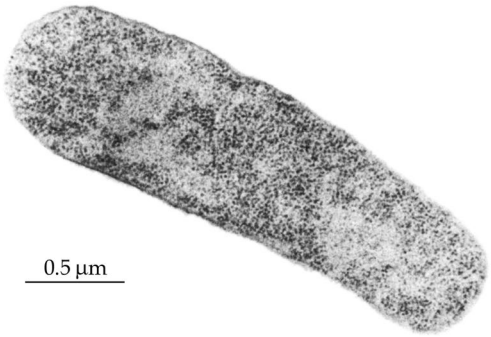

Figure 1: Micrograph of an *E coli* cell (from Neidhardt *et al*, 1990). The black dots seen in the figure are ribosomes, and the regions without ribosomes are nucleoids.

An *E coli* cell is usually equipped with flagella and pili on its outside. Flagella are bundles of fibres that are used for cell movement. Pili are rod-shaped organelles that are used for adhesion. Some pili, called sex pili, can be used to transfer DNA from one cell to another; others allow bacteria to stick to surfaces.

The prokaryotic genome is densely packed into a nucleoid (see figure 1), and is usually referred to as the chromosome even though it is not organised into the complex chromatin structures known from eukaryotes. Contrary to the eukaryotic cell, the prokaryotic cell has no organelles bounded by membranes; all of the processes going on in the cell share the same space. But this lack of membranes does not imply a lack of organisation. Indeed, the high solute concentration in the cytosol makes it very viscous, like a thick gel. Thus, it is possible for things to stay in place even without the support of membranes.

Table 1 shows the constituents of an average *E coli* cell. Naturally, these figures must not be taken too seriously, but they do give an idea of the proportions.

2.1.3 Cell Modelling

In Smartcell terms, the molecular species in a model are referred to as *entities*. Entities are referred to by name in the model, and each entity must have a unique name and be declared using one of the entity-declaring XML elements. The reason for having several entity-declaring elements is that they carry different semantic meanings. The three elements are *species*, *complex*, and *dna*. Species is the basic type, used for most entities. Complex-type entities are entities that are put together from other entities. Finally, the dna-type entities have some special features such as binding sites that reflect the role of DNA in the cell. The declaration of an entity also involves the declaration of a number of properties, such as molecular weight and ionic valence. These properties are common to all instances (molecules) of the entity type.

Another important term that must be defined is *process*. A process makes things happen in a cell, and always involves at least one entity. Some examples

of processes (see figure 2) are the creation or destruction of entities, chemical reactions, dimerisation, diffusion, and active transport. There are several XML elements for declaring processes as well, including the *process*, *diffusion*, *complex-formation*, and *multistep-process* elements.

This definition does not include processes that operate on the geometry of the model rather than the entities. These processes, called *shape-distorting processes*, will be discussed later.

2.1.4 Selecting a Kinetics Model

What options are there when it comes to the representation of the entities and their interactions? At the highest level of detail, one could take into account the motions of individual molecules, or even atoms, and let them interact according to quantum mechanical principles. This *microscopic* representation may be suitable for theoretical studies of small systems, but it is useless for modelling things as large as a cell.

In a model employing *mesoscopic* kinetics, each entity is represented by a copy number, i e the number of copies inside some (possibly infinitely large) volume element. Nothing is known about the distribution of the particles inside the volume element. Interactions between entities are defined as stochastic events that operate on the copy numbers. Stated in mathematical terms, the amount of species i in the volume element V at time t is described by a discrete copy number $X_i(t)$. There is also a set of state-altering events $\{e_j\}$, and for each event there is a "probability per unit time" p_j, which may be a function of the state $\{X_i\}$.

The most popular approach so far is that of *macroscopic* kinetics. The entities are represented by concentrations, and the interactions between them by differential equations. As a result, models that use macroscopic kinetics are de-

Table 1: Constituents of an average *E coli* cell (from Neidhardt *et al*, 1990).

Compound	**mass/fg**	**relative molecular mass/u**	**number of molecules**	**kinds of molecules**
Protein	155.0	40,000	2,360,000	1,050
rRNA	48.0	1,000,000	18,700	1
tRNA	8.6	25,000	205,000	60
mRNA	2.4	1,000,000	1,380	400
DNA	9.0	2,500,000,000	2.13	1
Lipid	26.0	705	22,000,000	4
Lipopolysaccharide	10.0	4346	1,200,000	1
Murein	7.0	$(904)_n$	1	1
Glycogen	7.0	1,000,000	4,360	1
Soluble pool[1]	8.0			
Inorganic ions	3.0			
Water	6,700			
Total	9,500			

[1]building blocks, metabolites and vitamins

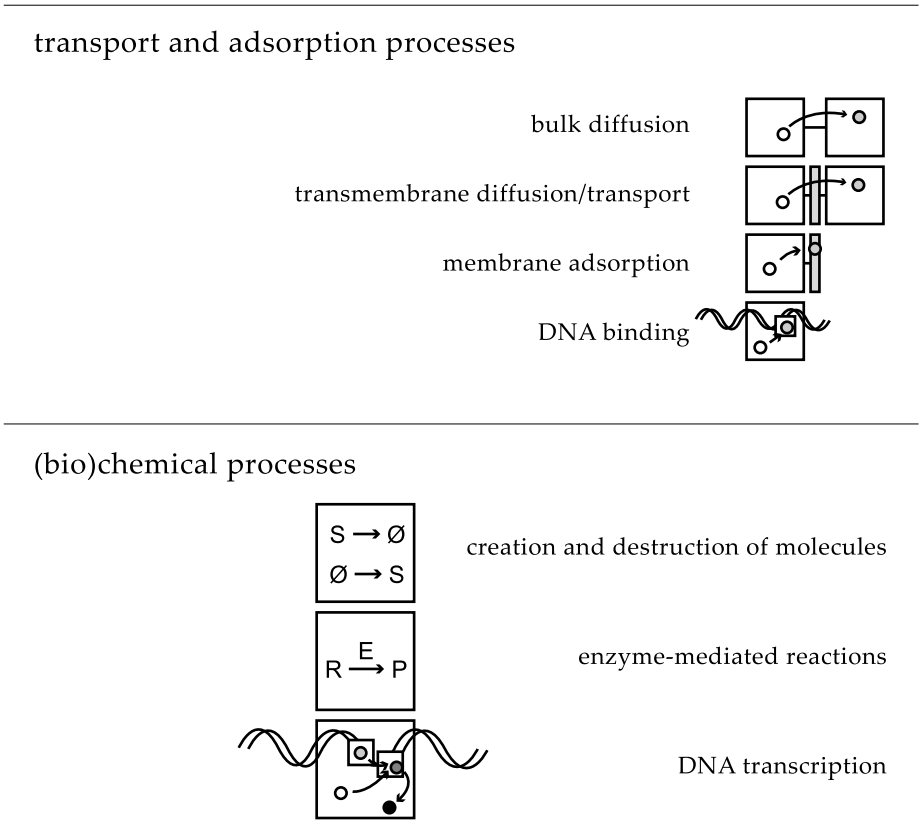

Figure 2: Some examples of processes. The boxes represent sites and the circles represent entities.

terministic. Mathematically, each species i in the volume element V at time t is represented by a concentration $C_i(t)$, which is real and non-negative. A system of differential equations $\{\dot{C}_j = f_j(C_1, C_2, \ldots, C_N, t)\}$ defines the time evolution of the state $\{C_i\}$.

The popularity of the macroscopic kinetics model comes as no surprise, because it does have some major advantages. Concentrations are what people are working with in the lab. The differential equation formulation is the one commonly used in the lab as well as in the literature. The differential equations are well-known and there are standard methods available for solving them. But there is a catch: the assumptions made in the macroscopic model are not valid when concentrations get very low, which may cause the model to break down. So what works fine for large volumes may not work for cells, where extremely low concentrations are common (Halling, 1989). Even though this has been known for a long time, many people are still unaware of it (see discussion and references in Paulsson, 2000, p 11). The only assumption made by the mesoscopic model is that nonreactive collisions occur far more often than reactive collisions (Gillespie, 1976), so that the molecules are uniformly distributed in V and their energies remain Bolzmann distributed. This assumption should be

valid at all times for cell biological models.

The mesoscopic kinetics model is the one chosen for Smartcell. If possible, it would be interesting to try a hybrid model where a macroscopic representation is used for the more abundant entities. This could give a significant performance improvement, but at the same time lead to intricate compatibility problems in the cases where the two paradigms make contact.

2.1.5 Form and Function

The cell is not just any volume element; it has a shape, and the distribution of entities therein is crucial to certain processes (see Shapiro and Losick, 2000, for a review). For example, in cell division, the establishment of a concentration gradient is necessary for the formation of a septum at the correct place (Lutkenhaus and Addinall, 1997). Taking the spatial distribution of entities into account may seem to contradict the very nature of mesoscopic kinetics, where the point is not to know where the entities are. However, this obstacle can be overcome by dividing the volume element into several, smaller volume elements. In this way a coarse spatial distribution is obtained, and the mesoscopic kinetics model remains valid as long as the volume elements are sufficiently large.

Consider the usual approach in macroscopic kinetics models. The finite difference method—probably the most common method—uses a computational mesh, where the mesh points approximate infinitely small volume elements. When the mesh is regular and rectangular, the corresponding system of difference equations can be written as a matrix with certain symmetry properties. The same starting point can be used in the mesoscopic case: we use a mesh to divide space into volume elements. But then we take the volume elements to be the vertices of a graph, called the geometry graph, and add edges that connect adjacent volume elements. A further development is to allow the vertices not only to represent volume elements, but also structural elements of lower dimensionality. The semantic meaning of the edges—that of physical proximity—remains unchanged. This way complex features such as the intricate transport system of the cell and membrane diffusion can be described in a simple and elegant way. The vertices of the geometry graph will hereafter be referred to as *sites* of dimensionality 3 (volume element or solid), 2 (boundary), 1 (path), and 0 (point), respectively.

For simplicity, all solid sites are taken to be cubic and of identical size. This of course implies that all boundaries are squares and have the same size as well. The side of a solid site is called the *lattice unit* and denoted by λ. All paths are also discretised into path segment sites in a similar fashion; each path segment resides in one solid or boundary site and has length λ. The direction of the path may be of importance and therefore must be defined.

The construction of a model at the level of individual volume elements is a difficult and tedious task. However, the Smartcell model description format defines the model at a higher level, and the conversion to individual volume elements is taken care of automatically, as will be shown later.

2.1.6 Describing the Model Geometry

The question of how to describe the model geometry can be rephrased as how to specify a geometry graph by means of XML elements. Most users will probably find that "rod-shaped" is a better description of the *E Coli* geometry than a detailed list of sites and connecting edges. On the other hand, some model geometries may ve so specific that no high-level description (e g rod-shaped) will do.

The solution to this dilemma is to use a low-level description format, and supply some pre-defined model geometry modules with the framework. This leaves us with the question of how to define an arbitrary geometry graph using XML elements.

But first the important term *region* must be defined. When building a model, one frequently needs to refer to a set of sites collectively. This is accomplished through the introduction of the *region*. Mathematically, the model geometry is represented by an undirected graph $G = \langle S, E \rangle$ where S is the set of sites (vertices) and E is the set of edges connecting the sites. A region $R \subseteq S$ is a set of sites. Regions may be defined as part of the model geometry, and they can also be constructed from existing regions by means of the set operations union ($A \cup B$), intersection ($A \cap B$), difference ($A \backslash B$), and complement (A^c). There are also two special regions called *empty* ($= \emptyset$), and *universe* ($= S$). A region where all sites have the same dimensionality (solid, boundary, path, or point) is called *homogeneous*.

In its current version, the Smartcell DTD does not contain any elements for defining the geometry graph. However, one way of doing this that I call the lattice operation method will be presented here. Although not able to define really arbitrary geometry graphs, it is able to describe most, if not all, physically relevant geometry graphs.

The lattice operation method works much like a pixel-based paint programme. The method starts with the introduction of a *lattice*; a three-dimensional scaffold where volume elements (solid-type sites) can be attached. A two-dimensional model can be built by letting the lattice have unit thickness in one dimension.

The lattice starts out empty, but sites may be added and assigned to regions by means of *operations*. An operation is given as

```
site-operation target-region [ region-operation destination-region ]
```

where the site-operation is one of

add, adds new sites, replacing existing ones

add-if-empty, adds new sites only to empty lattice points, does not modify already habitated lattice points

replace, replaces existing sites

edit, no adding or removing of sites; just selects existing sites for a region operation

remove, removes existing sites

The target region may be a reference to an existing region, but as there are no such regions at all in the beginning except the trivial regions, there must also be *geometric* ways of referencing sites. These are:

block, all lattice points with centres within a block of specified centre and size

ellipsoid, all lattice points with centres within an ellipsoid of specified centre and size

border, a border, one lattice unit thick, around a specified region

The last part of the operation specifies an optional region operation to be carried out on the sites identified by the site operation and target region. Valid region operations are:

add-to-region, the sites are added to the destination region

remove-from-region, the sites are removed from the destination region

With the solid sites in place we can start adding other geometry features as well. There are a number of operations available for this purpose:

interconnect, takes one region as parameter; connects all neighbour pairs for which both neighbours belong to the region.

boundary, takes three regions A, B, and C as parameters; adds boundary sites between every region A-region B neighbour pair and assigns them to region C.

path, takes a list of point coordinates and a name as parameters. The path is discretised to fit the lattice and broken into path segment sites, which are assigned to a region named after the path.

point, takes a name and a set of coordinates as parameters; adds a point site to the geomery graph and assigns it to the region with the specified name.

It should be stressed that this is just one way of defining the model geometry; there may be other alternatives better suited for the needs of the Smartcell framework. We shall finish this section with an example of how to describe a simple coliform model geometry with this notation:

```
lattice unit( .2 ) size( 2.2, 2.2, 2.2 )
```

An initialisation operation that was not described above. The lattice unit (the distance between two lattice points) is taken to be .2 μm, and the size of the lattice to be 2.2 μm in each spatial dimension. The lattice is centered at (0, 0, 0).

```
add-if-empty ellipsoid( (0, 0, 0), (2, 1, 1) ) add-to-region "cytosol"
```

Create sites at the mesh points within an ellipsoid with centre at origo and length 2 μm in the x direction and 1 μm in the y and z directions, then assign the sites to the cytosol region

```
interconnect "cytosol"
```

Connect neighbour sites within the cytosol.

```
add-if-empty border( "cytosol" ) add-to-region "surroundings"
```

Create a border around the cytosol and name it "surroundings".

```
interconnect "surroundings"
```

Connect neighbour sites within the surroundings. Note that some surroundings sites may still be unconnected due to the jagginess of the discretised ellipsoid's edge.

```
boundary "cytosol", "surroundings", "cell_membrane"
```

Add a boundary site between each cytosol-surroundings pair and add it to the cell_membrane region.

```
path (.7, 0, 0), (0, .35, 0), (-.7, 0, 0), (0, -.35, 0), (.7, 0, 0) "chromosome"
```

Add a path called chromosome that lies in the $z = 0$ plane. This is the last operation; we now have a crude but functional model geometry.

2.1.7 Describing Entities

The entities and processes, together with the model geometry, are the basic building blocks of a Smartcell model. This section describes how the entities are described using the Smartcell model description format, and the following section addresses the description of processes.

Entities can be declared using the XML elements *species*, *complex* and *dna*. The species element is the most basic one: it declares the existence of a chemical species with a name, a relative molar mass, and an ionic valence.

The *complex* XML element declares an entity that is put together from a number of other entities. The constituents are identified by their names, and association and dissociation rates can be given for a region using the *complex-formation* XML element. Several complex-formation elements may be required if there are several ways to put together the complex, or if the rates differ in different regions.

The *dna* element is used to declare DNA species, which are quite special. Primarily, they have the ability to bind several kinds of ligands (transcription factors, RNA polymerases, etc) at specific positions (e g promoters and response elements). Relevant positions along the DNA species are declared using *dna-site* elements, which relate the positions to the length of the DNA molecule. The dna-site elements can be organised into more manageable parts using the *dna-section* element.

Diffusion elements can be included in the entity-declaring elements, to specify the diffusion properties of the entity within a region. The attributes of the diffusion element are *region*, *type* and D. The type attribute may be *bulk* (diffusion between connected solid sites belonging to the region), *boundary* (diffusion between connected boundary sites belonging to the region) or *trans-boundary* (diffusion between solid sites on either side of a boundary site, where the boundary site belongs to the region). The diffusion constant is given in $\mu\text{m}^2\ \text{s}^{-1}$. For semipermeable membranes, the permeability constant P should be given instead of D.

Localisation, i e, the restriction of entities to a part of the model, is an important phenomenon in cell biology, and should be incorporated into the model. One way of doing this is to explicitly allow the presence of an entity in parts of the model. Another option is to find the possible locations implicitly from the initial state and diffusion processes. Smartcell uses the latter method, so there is no need to supply any localisation information here.

2.1.8 Describing Processes

Processes are described using the *process* XML element. As can be seen from the example processes in figure 2, the precise definition of a process may require a lot of information. Not only is the species of the participating entities of importance, but so is the environment. Therefore, the first thing that must be defined is the local geometry where the process takes place. The local geometry is defined by one or more *process-site* XML elements, each corresponding to one site of the geometry graph (and to a box in the figure). The entities that take part in the process are declared inside their respective process sites. A special *connected-to* XML element connects the process sites to each other, which is the same as requiring that the corresponding sites of the geometry graph be connected.

The entities involved in a process fall into three categories: *reactants*, *products* and *effectors*, and are declared within the process-site elements using XML elements with the same names. Reactants are entities that are consumed by the process and products are entities that are created by the process. Effectors are entities, such as enzymes or transcription factors, that take part in the process without being consumed. The declaration of reactants and products involve the name of the entity and a multiplicity, which defaults to unity. For effectors there is also a type attribute, which declares the role of the effector. The default effector type is cofactor, which will make the rate of the process proportional to the amount of effector—this is the usual case with enzymes, for example. The other effector types, require-gt, require-lt, require-eq and require-neq, define that the process is enabled if and only if the amount of effector is greater than, less than, equal to or not equal to the given multiplicity, respectively. These effector types will be used mostly for transcription events, where the activation state of a gene may depend on the absence or presence of effectors at the promoter and enhancer regions.

A critical parameter of a process is the rate at which it proceeds. The rate of a chemical reaction (or some other process) is usually given either by a rate constant k or by means of a rate law. These terms originate in the macroscopic domain, but we will show that the rate constant can and will be used for our mesoscopic models as well.

For an *elementary reaction*, the velocity v_μ of a chemical reaction R_μ is given by

$$v_\mu = k_\mu \prod_{i \in R \cup C} C_i^{m_i}, \tag{1}$$

where R and C are the reactants and cofactors of R_μ. C_i is the concentration and m_i the multiplicity (number of molecules involved in the reaction) of species i. Most chemical reactions occur in a sequence of elementary steps involving just one or two molecules or ions (Atkins, 1994, p 882).

Rate laws, on the other hand, compress a series of elementary reactions into one summary reaction and a rate law. The rate law gives the overall velocity v as a function of the concentrations of the participating entities. As an example, consider the well-known Michaelis-Menten law of enzyme activity. The model, which is the simplest one that accounts for the kinetic properties of many enzymes, is

$$E + S \underset{k2}{\overset{k_1}{\rightleftharpoons}} ES \overset{k_3}{\rightarrow} E + P \tag{2}$$

where E is the enzyme, S is the substrate (reactant) and P is the product. By assuming that the system is in a *steady state*, and that the *enzyme is saturated* with substrate, the velocity can be shown (see Stryer, 1995, p 192) to be

$$v = v_{\text{max}} \frac{[S]}{[S] + K_M}, \tag{3}$$

where

$$v_{\text{max}} = k_3[E_{\text{total}}] \text{ and } K_M = \frac{k_2 + k_3}{k1}. \tag{4}$$

Here $[\cdot]$ denotes a concentration. This rate law often works better than one would expect, considering the assumptions made. They will seldom be valid in a living cell, though—a cell in a steady state is by necessity dead. The Smartcell framework handles elementary reactions only, but in most cases it will be possible to separate the rate laws into elementary reactions with little effort.

The rate constant k of a process is declared using the *rate* attribute. If the process is reversible, the reverse rate can also be declared using the *reverse-rate* attribute (one must be very careful with effectors of the limiting kind in reversible reactions, as the reverse process is formed by changing all reactants to products and vice versa. The effectors are left as they are, which may lead to unexpected behaviour from the limiting effectors). The rate is given as a numerical constant or an expression, and there is no way of attaching a unit to it in the model description file. Therefore, there may be no doubt about which unit is used.

Let us consider the subject of *units* for a moment. The unit of preference for concentrations is the Molar (M), where 1 M = 1 mol per dm^3. The concentrations of solutes in a cell ranges from a few nanomoles (corresponding to a single molecule) to tens of millimolars. Entities bound to a membrane (residing in a boundary site) can be quantified by a surface density, and entities bound to a fiber (in a path segment site) by a path density. For reasons of consistency, all concentrations are given in mol per dm^d, where d is the dimensionality (i e, the number of spatial dimensions). When there are several entities from different sites involved in a process, the units tend to get quite complex. This should not occur frequently, though.

A multistep process is a special kind of process, declared by the *multistep-process* XML element. It is an adaptation of a feature presented in the paper by Gibson and Bruck (2000), that is likely to give an enormous performance improvement for certain kinds of processes. The processes of interest are chains of irreversible, monomolecular processes, such as transcription and translation. Consider a chain of n reactions, each with the same reaction parameter c. Each reaction turns one reactant molecule S_i into one product molecule S_{i+1},

which is at the same time the reactant of the next reaction. If none of the intermediate products is of interest, the whole chain of processes can be reduced to one process, turning S_0 directly into S_n. The multistep-process element is used in the same way as the process element with the following exceptions: there must be exactly one reactant entity and one product entity, there is an attribute n that defines the number of steps in the process, and there is no reverse-rate attribute.

There is one class of processes yet to be described: the shape-distorting processes. These processes incorporate the mechanisms that affect the model geometry into the model itself. Cell growth, cell fission and DNA diffusion are examples of such mechanisms. The question is how to specify such a process; it is out of scope for this work, but should be considered in the future. One possibility is specify rules on the lattice level, like cellular automata, where the rules may involve entities in the affected sites.

As I see it, this implicit formulation where the outcome is not necessarily known is the most interesting one, because it captures the underlying physics albeit in a simplified way. However, it will probably be quite difficult to define the rules. Another alternative is to explicitly specify the changes to the geometry that happen in a sequence, possibly with requirements that must be fulfilled before the next step can take place. This should not be too difficult to implement as a first step, and may give a very accurate description of e g cell growth and fission.

2.2 Simulation

Simulation is the process of performing experiments on a model. Using simulation techniques instead of actually performing the experiments can save time and money; in addition there are experiments that are impossible to carry out in reality, that may be possible to simulate. The validity of the model is crucial; if the model is invalid, the simulation results will also be invalid. This is sometimes referred to as the GIGO principle: garbage in yields garbage out.

A model that accurately describes a system can be used to test a hypothesis concerning the system, using e g the well-known chi-square test. The analogy with real experimental work continues into the data analysis domain: the number of trajectories (simulation runs) required to reach a certain level of confidence is of course equal to the number of real experiments it would take to reach the same confidence level.

Smartcell models are models of living cells. What kind of results can be gained from simulations on such a model? When building a model, or tuning its parameters, one is frequently interested in the concentration or spatial distribution of entities. To find them, one can set up a dummy experiment, where the cell is simply "living" under normal conditions, while the appropriate measurements are made. Other simulations that require a working model are the measurement of the relaxation time following a perturbation, or to see how the modification of one model parameter affects other parts of the model.

2.2.1 Simulation Parameters

A real, wet experiment in cell biology is usually carried out as follows: a system is set up by growing cells under controlled conditions, then the system is disturbed in some way, e g by moving the cells to a new growth medium. Observations are made by taking samples from the cell culture and performing measurements on them. In some cases it may also be possible to perform the measurements *in vivo*. Eventually, conclusions are drawn from the observations. A Smartcell simulation run closely mimics this procedure, as it requires the definition of an *initial state*, an optional set of *constraints* and a set of *observations*. These things are all defined by XML elements with the same names, as will be described next.

The initial state describes "what is where and how much" at the start of the simulation run. The *initial-state* element defines the concentration of an entity in a homogeneous region. Several initial-state elements may be required if the concentration varies throughout the model. The concentration (in mol per dm^{-d}, where d is the dimensionality of the region) is given as a numerical expression contained within the initial-state element, and may be a function of spatial coordinates.

Constraints regulate the amount of a species after the start of the simulation. Constraints should be used with care, as they can lead to unphysical results if misused. But, when used in a sensible way, constraints can be a powerful tool. This is another advantage of simulation, as some constraints may be very difficult or even impossible to enforce on the real system. Constraints are defined in the same way as initial states, but as functions of time. A time interval $[t_{\mathrm{start}}, t_{\mathrm{end}})$ when the constraint is active can also be specified; if none is given, it defaults to $[0, \infty)$.

A special XML element is used for specifying how DNA entities are mapped to paths; it is called *dna-mapping* and takes an entity name and a path as attributes. One can consider this element as a special case of the constraint element.

Observing, or measuring, the presence of an entity is of course much easier in the simulated environment than in the real system, as the exact state of the simulated system is always known. There are two kinds of observations defined in the Smartcell framework. They may not seem very useful as they are; indeed, they are quite low-level, and so a separate data analysis tool will prove useful (yet another analogy to real experiments). The observation types are described in table 2. The output format for the simulation engine is not defined here, but it would make sense to use an XML vocabulary for the output as well.

Table 2: Basic observation types; their parameters and outputs.

Observation Type	**Parameters**	**Output**
snapshot	time t (s), homogeneous region R, entity i	table of $\langle \mathbf{x}_s, C_i(t) \rangle$, one entry for each site $s \in R$.
time-series	start time t_{start} (s), sampling interval t_i (s), end time t_{end} (s) (optional, defaults to ∞), homogeneous region R, entity i	table of $\langle t, \int_R C_i(t) \rangle$

2.2.2 The Core Model

As previously described, the Smartcell model description format is on a high semantic level by design. Although this is helpful when a model is constructed, it makes it difficult to use the model for simulation purposes. The *core model format*, a more abstract model description format, is used internally in the framework. The real advantage of this format is that there is an automatic procedure for generating a core model from a Smartcell model and a set of simulation parameters! This section describes the core model format in detail, and thereafter moves on to the mechanisms involved in the conversion.

The elements of a core model are:

slots $\Sigma = (s_1, s_2, \ldots, s_N)$. Each slot holds the (discrete) copy number of an (entity, site) pair.

parameters $P = (p_1, p_2, \ldots, p_M)$. The parameters are like slots, but real-valued. They hold values such as the current time and temperature. The slots and the parameters together make up the state of the system.

events $E = \{e_1, e_2, \ldots, e_L\}$. The events describe the dynamics of the model. Each event $e = \langle A, a, n \rangle$ consists of an action vector A, a probability per unit time a, and a number of steps n. The action vector represents the change made to the system when the event is executed, so that $\Sigma(t+\tau) = \Sigma(t) + A$. The probability per unit time a may depend on the state. The

number of steps is usually one. When $n \neq 1$, the event is carried out in n steps, each with probability a per unit time (compare to the multistep processes). In this case, a must be constant in order for the event to be physically valid.

constraints $C = \{c_1, c_2, \ldots, c_K\}$, each constraint $c = \langle n, \alpha, T \rangle$ constraining slot number n or parameter number $n-N$ to the expression α during the time interval $T = [t_{\mathrm{start}}, t_{\mathrm{end}})$.

Contrary to the Smartcell model, the core model also includes initial state and contraint information.

The conversion from a Smartcell model, a geometry graph, and a set of simulation parameters into a core model is done as follows. Please note that this conversion is valid for a static model only, i e, a model without shape-distorting processes. A non-static model would require a regeneration of the core model each time the geometry changes (which is a complex operation, but not impossible).

A final note before we start with the details of the conversion process: it is important to keep references from the core model back into the source model. The evaluation of data must be done in the context of the source model rather than the core model; without these references it will not be possible to make a meaningful interpretation of the simulation results.

To start with, the DNA sites must be added to the geometry graph. The mapping of dna species onto paths are specified using dna-mapping elements; for each such element, the DNA sites of the DNA species are projected on the path and added to the geometry graph as point sites. A special kind of site, called a DNA pseudosite, is also added and then connected to all point sites.

Entities of all types (species, complex, and dna) are enumerated, and so are all sites in the geometry graph. Now, a trivial slot mapping is to let each (entity, site) pair correspond to a slot. The trivial slot mapping produces a dense map matrix. However, in most cases, an entity will only be found in a small number of sites, making the actual mapping matrix very sparse. A method must be developed to find out which slots will actually be used (i e, possibly nonzero), and remove the rest from the model. A further optimisation is to also remove the slots that necessarily remain constant throughout the simulation run, and handle them separately.

Parameter-type simulation parameters are split into one parameter and one constraint in the core model.

Diffusion specifications and complex formations are converted into normal processes in an obvious way, except for the rates of diffusion processes which need a bit more work. This conversion procedure is described in appendix B. Reversible processes are split into two irreversible processes, also in an obvious way. This leaves us with a number of processes and multistep-processes to convert into events.

In general, one process corresponds to several events. Consider a process which involves several entities residing in different process sites. Each mapping of the process sites onto the geometry graph that conforms to the connectivity of the process sites corresponds to a separate event. With this mapping established, the action vector can be compiled from the set of reactants and products. An algorithm for the mapping has been constructed and is given in appendix C.

The Smartcell model description format uses the rate attribute k to define the rate of a process. The core model, on the other hand, uses a "reaction probability per unit time" a for the same purpose. This is where the stochastic nature of the core model shows. Although conceptually similar, the two are very different on one point: k is defined with respect to concentrations and is therefore independent of volume. The reaction probability is defined as a function of discrete copy numbers, and therefore changes with volume (unless the process is unimolecular). As will be shown, the reaction probability can be deduced from the rate, but first the reaction probability must be defined.

In a stochastic model the deterministic concept of "reaction rate" is meaningless; its stochastic counterpart is the reaction parameter c, or "reaction probability per unit time". The formal definition, given by Gillespie (1976), is:

> $c_\mu \delta t \equiv$ average probability, to first order in δt, that a particular combination of R_μ reactant molecules will react accordingly in the next time interval δt.

This definition of c gives an average probability per unit time for each particular combination of reactant molecules R and cofactors C. By multiplying c with the number of available combinations h, given by

$$h = \prod_{i \in R \cup C} \binom{X_i}{m_i}, \tag{5}$$

we get the propensity per unit time a. Here, X_i is the copy number of entity i, and m_i is the multiplicity of the same entity. The limiting effectors—i e, those effectors that are not of the cofactor type—are also included in h as unit step or impulse factors, assuming the value 1 when the requirement is fulfilled and 0 otherwise.

The reaction probability can be deduced from the reaction rate as follows (adapted from Gillespie, 1976): the average reaction rate of a stochastic process is

$$\langle a \rangle = c \langle h \rangle \ [\mathrm{s}^{-1}] \tag{6}$$

where $\langle \cdot \rangle$ denotes an average. The rate constant k is conventionally defined as the average reaction rate per unit volume (or area), divided by the product of the average densities of the reactants; using the stochastic average reaction rate and adjusting for the units we get

$$k = \frac{\langle a \rangle}{N_A \lambda^d \prod_{i \in I} \langle C_i \rangle^{m_i}} = c \cdot \frac{\langle h \rangle}{N_A \lambda^d \prod_{i \in I} \langle C_i \rangle^{m_i}} \tag{7}$$

which can be rearranged to

$$\begin{aligned} c &= k \cdot \frac{N_A \lambda^d \prod_{i \in R \cup C} \langle C_i \rangle^{m_i}}{\langle h \rangle} = k \cdot \frac{N_A \lambda^d \prod_{i \in R \cup C} \langle \frac{X_i}{N_A \lambda^{d_i}} \rangle^{m_i}}{\langle \prod_{i \in R \cup C} \binom{X_i}{m_i} \rangle} \\ &\approx k \cdot \frac{N_A \lambda^d \prod_{i \in R \cup C} \langle \frac{X_i}{N_A \lambda^{d_i}} \rangle^{m_i}}{\langle \prod_{i \in R \cup C} \frac{X_i{}^{m_i}}{m_i!} \rangle} \approx k N_A \lambda^d \prod_{i \in R \cup C} \frac{m_i!}{(N_A \lambda^{d_i})^{m_i}} \\ &= k \cdot N_A^{1 - \sum m_i} \lambda^{d - \sum d_i m_i} \prod m_i! \end{aligned} \tag{8}$$

In this expression, λ is the lattice unit, N_A is Avogadro's number, and d is the dimensionality of the product sites (this must be identical for all products, or else the rate is not well defined). d_i is the dimensionality of the site where entity i resides.

2.2.3 Time Evolution of a Stochastic System

Now, with the core model in place, we can start making predictions. The time evolution of the system can be found by setting up and solving the corresponding master equation. This is the same as a transformation of the core model into a system of coupled differential equations in stochastic variables (see Gillespie, 1976, for a summary of the master equation formalism). The solution of the master equation contains all information there is about the system, but this approach is typically impossible to use as the computing time required for the solution explodes with increasing model size.

Fortunately, there are other methods. Whereas solving the master equation can be seen as "finding all trajectories through state space at once", it is also possible (and usually more computationally feasible) to find a number of trajectories, one at a time, and make an estimation. The method is simple: start from an initial state, then trace out the trajectory stepwise by choosing the next state according to calculated transition probabilities. The properties of the system can be estimated from a set of trajectories starting from the same initial state. This method also has its drawbacks; one is the large number of trajectories that may be required for a good average estimation. There is also the probability that some unusual states are never reached and therefore do not contribute to the averages.

Stated in a formal way, the simulation algorithm in its most basic form is as follows (Gillespie's First Reaction Method, 1976, reformulated by Gibson and Bruck, 2000):

1. initialise the state vector and set $t \leftarrow 0$
2. calculate the propensity function (probability per unit time) a_i for each event i
3. for each event i, generate a putative time τ_i according to an exponential distribution with parameter a_i
4. let μ be the event whose putative time τ_μ is least
5. update the slots to reflect the execution of event μ and let $t \leftarrow t + \tau_\mu$
6. go to step 2

Two things can be noted here: first, the algorithm does not handle multistep processes. Second, it draws the putative time from a negative exponential distribution. We will continue by motivating the use of the negative exponential distribution and, at the same time, find a suitable distribution for multistep processes.

The transition from propensity per unit time a to putative execution time τ is given by the following argument (from Lukkien *et al*, 1998): Assuming that there is only a single enabled event—this is not a restriction as all events

are independent—with propensity a per unit time, and let τ be the time that it occurs. By definition, the probability of the system staying in the initial state for another interval δt is $(1 - a\delta t)$. Thus

$$P(\tau \geq t + \delta t) = P(\tau \geq t)(1 - a\delta t). \tag{9}$$

Rewriting and taking the limit $\delta t \to 0$ gives

$$\frac{dP(\tau \geq t)}{\delta t} = -aP(\tau \geq t). \tag{10}$$

Integration yields

$$P(\tau \geq t) = \exp(-at). \tag{11}$$

Thus it can be seen that the putative time of execution τ has a negative exponential distribution with parameter a. Furthermore, this result can be generalised for multistep processes. It follows immediately from the definition of the gamma distribution that the putative time of execution for a multistep process with n steps is distributed as $\Gamma(n, a)$. When n is one, this reduces to a negative exponential distribution. We therefore replace step 3 with "for each event i, generate a putative time τ_i according to $\Gamma(n, a)$".

Gibson and Bruck (2000) also introduce a number of optimisations to the algorithm, resulting in what they call the "Next Reaction Method". These modifications are:

- recalculate a_i only when necessary. This is accomplished by maintaining, for each slot, a list of dependent events. When an event is executed, the events affected by the modified slots are marked as "dirty"; their propensities will be recalculated before the next event is selected.

- introduce absolute time—set $\tau \leftarrow t + \tau$ in step 3, and $t \leftarrow \tau$ in step 5. Then store the τ_is as well, not just the a_is. Thanks to absolute time, these entities need only be recalculated when a_i changes or when the event is executed.

- use an indexed priority queue for storing the τ_is. This guarantees that the updating of the queue will be efficient.

Gibson and Bruck also describe a way of reusing random numbers. They show that, under certain circumstances, the putative execution times can be updated by means of a scaling procedure instead of by another random number generation. This does not seem to be a good idea to me, though. The scaling procedure can be costly in itself, and there is also the danger of large round-off errors.

The multistep event concept also comes from the same article. To handle multistep events, we must add to the algorithm a priority queue Q_i for each multistep event i, where the putative times of the events that are on their way are stored. When the reactant slot is increased, a new putative time is generated (as mentioned above) and put into the queue, and when the reactant slot decreases (possibly, but not necessarily, due to the execution of the event), the first element of the queue—i e, the element with smallest τ—is removed from the queue.

The stochastic simulation concept requires a source of good random numbers. This means that if a generator of pseudorandom numbers is used, as is customary on digital computers, it must have high randomness (no obvious bias) and long period (all pseudorandom number generators have finite periods, i e, the sequence starts repeating sooner or later). A good choice for Smartcell is a pseudorandom number generator called Mersenne Twister (Matsumoto and Nishimura, 1998) that has a period of $2^{19937} - 1$. It is also fast—another critical property.

3 Results

This report describes the planning and construction of the Smartcell modelling and simulation framework. The framework is unique due to its combination of localisation and diffusion phenomena with mesoscopic kinetics. It also incorporates most mechanisms of cell function in an elegant and consequent way. Not only the biological part, i e, the features of the cell, have been considered during the design of the framework, but also the physical and computational aspects of modelling and simulation. The result is a flexible framework with a well-designed model description format and a simulation algorithm with high expected performance.

The Smartcell project was initiated by Luis Serrano, who had been outlining a simulation tool that would be helpful in genetic engineering. With this tool, it should be possible to perform simulation experiments on a model of a cell, to see what would happen if the expression of a gene were up- or down-regulated. The cell model should be assembled from modules, preferably downloaded from public databases. Such a tool would of course not only be of use in the field of genetic engineering, but would be helpful in several research areas.

The construction of a framework of this complexity simply cannot be accomplished by one person in six months' time. Instead, this work is to be seen as the initial phase of a larger project. The first step was to write down the ideas, trying to break down the project into parts, and finding one or a few parts that could be implemented and tested separately. This was not so easy, as it turned out—the parts were simply too intertwined. Instead, the goal was set as the formulation of a theoretical foundation for the framework; the model description format, the intermediate format and automatic conversion method, and the simulation algorithm. The initial time plan was revised to include more theory and less programming. Many problems have been solved, others have been stated explicitly and formally, and yet others still remain to find and solve. But these are left as an exercise to the reader.

The near future directions of the project, as I see it, are as follows:

implementation. A simple implementation of the described algorithms in C++ or Java should not be too difficult to achieve. Only when this exists will we know how the algorithms perform.

modelling concepts. There are particularly two modelling concepts that need more attention: the geometry description and the shape-distorting processes.

optimisations. Thanks to its modular organisation, the basis of the framework is a good target for optimisations. For example, it should be possible to modify the simulation algorithm for parallel execution in a multicomputer cluster. In general it will not be possible to partition the core model into independent parts, but an "optimist" parallelisation might perform well, provided the dependency between parts is small.

extensions. A model editor with a graphical user interface will probably be of great use for the model builders. An automatic database import tool would also be nice, for importing e g DNA species from the genome databases or enzyme kinetics parameters from enzyme databases.

model construction and validation. This is the most important part: the whole-cell simulation concept must be tested in itself! Is it at all possible to create an *in silico* cell? What is the typical execution time and accuracy of a simulation run? This we cannot know until a model has been written and the simulation engine is up and running.

Acknowledgements

Many thanks to Luis Serrano (EMBL) for making the project possible, Johan Elvnert (TTA) for financing the project, Erik Zeitler and Emmanuel Lacroix (EMBL) for discussions, Maria Andér (UTH/EMBL) for endless questions, Mats Gustafsson (Signals and Systems Group, Uppsala University) for being the formal examiner, and Matti Nikkola (Department of Biology Education, Uppsala University) for coordinating the diploma work.

4 References

Atkins P W (1994). *Physical Chemistry, 5th ed*. Oxford University Press, Oxford

Garfinkel D (1981). Computer modeling of metabolic pathways. *TiBS* **6**:69–71

Gibson M A and Bruck J (2000). Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *J Phys Chem* **104**:1876–1889

Gillespie D T (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J Comput Phys* **22**:403–434

Halling P J (1989). Do the laws of chemistry apply to living cells? *TiBS* **14**:317–318

Heinrich R, Rapoport S M and Rapoport T A (1977). Metabolic Regulation and Mathematical Models. *Prog Biophys Molec Biol* **32**:1–82

Lakshminarayanaiah N (1984). *Equations of Membrane Biophysics*. Academic Press, Inc.

Lukkien J J, Segers J P L, Hilbers P A J, Gelten R J and Jansen A P J (1998). Efficient Monte Carlo methods for the simulation of catalytic surface reactions. *Phys Rev E* **53**:2598–2610

Lutkenhaus J and Addinall S G (1997). Bacterial Cell Division and the Z Ring. *Annu Rev Biochem* **66**:93–116

Matsumoto M and Nishimura T (1998). Mersenne Twister: A 623-dimensionally equidistruibuted uniform pseudorandom number generator. *ACM TOMACS* **8**:3–30

McAdams H H and Arkin A (1998). Simulation of prokaryotic genetic circuits. *Annu Rev Biophys Biomol Struct* **27**:199–224

McAdams H H and Shapiro L (1995). Circuit Simulation of Genetic Networks. *Science* **269**:650–656

Mendes P (1997). Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3. *TiBS* **22**:361–363

Morton-Firth C J (1998). *Stochastic simulation of cell signalling pathways*. Ph D thesis, Cambridge

Neidhardt F C, Ingraham J L and Schaechter M (1990). *Physiology of the Bacterial Cell, A Molecular Approach*. Sinacor Associates, Inc. Sunderland, Massachusetts, USA

Paulsson J (2000). *The Stochastic Nature of Intracellular Control Circuits*. Ph D thesis, Uppsala

Schaff J and Loew L M (1999). The Virtual Cell. *Pacific Symposium on Biocomputing* **4**:228–239

Schaff J, Fink C C, Slepchenko B, Carson J H and Loew L M (1997). A General Computational Framework for Modeling Cellular Structure and Function. *Biophys J* **73**:1135–1146

Shapiro L and Losick R (2000). Dynamic Spatial Regulation in the Bacterial Cell. *Cell* **100**:89–98

Smolen P, Baxter D A and Byrne J H (2000). Modeling Transcriptional Control in Gene Networks—Methods, Recent Results, and Future Directions. *Bull Math Biol* **62**:247–292

Stoffers H J, Sonnhammer E L L, Blommestijn G J F, Raat N J H and Westerhoff H V (1992). METASIM: object-oriented modelling of cell regulation. *CABIOS* **8**:443–449

Stryer L (1995). *Biochemistry, 4th ed*. W H Freeman and Company, New York

Stundzia A B and Lumsden C J (1996). Stochastic Simulation of Coupled Reaction-Diffusion Processes. *J Comput Phys* **127**:196–207

Tomita M, Hashimoto K, Takahashi K, Shimizu T S, Matsuzaki Y, Miyoshi F, Saito K, Tanida S, Yugi K, Venter J C and Hutchison C A (1999). E-CELL: software environment for whole-cell simulation. *Bioinformatics* **15**:72–84

Uetz P, Giot L, Cagney G, Mansfield T A, Judson R S, Knight J R, Lockshon D, Narayan V, Srinivasan M, Pochart P, Qureshi-Emili A, Li Y, Godwin B, Conover D, Kalbfleisch T, Vijayadamodar G, Yang M, Johnston M, Fields S and Rothberg J M (2000). A comprehensive analysis of protein/protein interactions in Saccharomyces cerevisiae. *Nature* **403**:623–627

A Model Description Format Reference

A Smartcell model consists of one or more module files. A module file is an XML document that conforms to the Smartcell DTD.

An XML file is a text document where the text is structured into *elements*. The elements carry a semantic meaning, which may differ between different kinds of documents. The element boundaries are identified by tags; a sequence of characters delimited by angle brackets. The similarity with HTML is not just in the name (Extensible Markup Language vs HyperText Markup Language); the documents also tend to look similar even though XML documents may describe things that have little or nothing in common with world wide web pages.

XML is far stricter than HTML when it comes to formal requirements. For a text file to be a valid XML document, there is an overall structure that must be followed, and the naming and nesting of tags is strictly regulated. It is also possible to reference a DTD (Document Type Definition) from the document, thereby adding an additional set of requirements on the attributes, order, and nesting of elements. Below is the DTD used for Smartcell module files. Each element type is provided with a description and an example.

A.1 The Smartcell DTD

```
<!--
Smartcell Module Document Type Definition, version 1.1.3
Copyright (C) Anders Kaplan 2001

See http://www.w3.org/XML/ for a description of XML and DTDs.

Smartcell model description files must use this DTD and have "module" as their
root element. There are some additional requirements on model description files
as well; see the documentation for a description.

Note: Most model features must be given a unique name. Any valid XML string is a
valid name and the names are always case sensitive. This means that Atp, atp and
ATP are three different names.
-->


<!-- ********************************************************************************

module element

******************************************************************************** -->


<!-- module
The root element of all module files. Modules may also contain sub-modules.

Example:
  <module name="E coli: chemotaxis">
  ...
  </module>
-->
<!ELEMENT module ANY>
<!ATTLIST module
  name CDATA #REQUIRED
>


<!-- ********************************************************************************

description elements

******************************************************************************** -->
```

```
<!-- description
Virtually all model features can be given a description using this element; by
convention, the description (if there is one) should be the first contained
element. The descriptions are intended for model builders, e g for documenting
why something was implemented in a certain way, and their content is purely
informational.

The elements typically encountered in descriptions are reference, authors, date,
dbxref, etc. It is also perfectly legal to use HTML in the description.

Example:
  <description>Description element.</description>
-->
<!ELEMENT description ANY>


<!-- reference
A reference to an article, a book, a web page etc.

Example:
  <reference>
    <authors>Gillespie D T</authors>
    <title>Exact Stochastic Simulation of Coupled Chemical Reactions</title>
    <publication>J Phys Chem (1976);81:2340-2361</publication>
  </reference>
-->
<!ELEMENT reference (authors, title, publication*)>
<!ELEMENT authors (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT publication (#PCDATA)>


<!-- date
A date in zero-padded yyyy-mm-dd format.

Example:
  <date>2000-12-09</date>
-->
<!ELEMENT date (#PCDATA)>


<!-- database (cross) reference
A reference to an object in a database. The database tag indicates the database
(for example, swissprot or embl). The unique id element indicates the key for
the object in the database, which should be the natural key for the object in
the database. For the protein and DNA databases, these are the accession numbers
of the sequences.

Example:
  <dbxref>
    <xref_db>swissprot</xref_db>
    <xref_db_id>P09651</xref_db_id>
  </dbxref>
-->
<!ELEMENT dbxref (xref_db, xref_db_id?)>
<!ELEMENT xref_db (#PCDATA)>
<!ELEMENT xref_db_id (#PCDATA)>


<!-- ************************************************************************

entities

************************************************************************ -->


<!-- species
The most basic kind of entity: a chemical species.

Attributes:
* ionic_valence: the charge of the particle; unit: electron charges
* M: relative molecular mass; unit: u (the unified atomic mass unit)
```

```
The diffusivity of the entity may be specified by including diffusion elements.

Examples:
  <species name="ATP" />
  <species name="Cl-" ionic_valence="-1" M="35.45" />
-->
<!ELEMENT species (description?, diffusion*)>
<!ATTLIST species
  name CDATA #REQUIRED
  ionic_valence CDATA #IMPLIED
  M CDATA #IMPLIED
>


<!-- complex
A complex of two or more entities. The complex is treated as a new kind of
entity, so its components cannot participate in processes individually.

The ionic_valence and M attributes are the same as for the species element.

Association and dissociation rate constants can be specified for various regions
by including complex-formation elements. See the description of the process
element for a discussion of the units used.

The diffusivity of the entity may be specified by including diffusion elements.

Example:
  <complex name="TTWAA">
    <description>Tar-Tar-CheW-CheA-CheA complex</description>

    <complex-formation region="universe" type="region" rate="6.40e-1"
        reverse-rate="1e6">
      <complex-component name="TTW" />
      <complex-component name="WAA" />
    </complex-formation>

    <complex-formation region="universe" type="region" rate="1.12e-1"
        reverse-rate="1e6">
      <complex-component name="TTWW" />
      <complex-component name="AA" />
    </complex-formation>
  </complex>
-->
<!ELEMENT complex (
  description?,
  (diffusion|complex-formation)*
)>
<!ATTLIST complex
  name CDATA #REQUIRED
  ionic_valence CDATA #IMPLIED
  M CDATA #IMPLIED
>

<!ELEMENT complex-formation (
  description?,
  complex-component,
  complex-component+
)>
<!ATTLIST complex-formation
  region CDATA #REQUIRED
  type (region|boundary|path|point) #REQUIRED
  rate CDATA #REQUIRED
  reverse-rate CDATA "0"
>

<!ELEMENT complex-component (description?)>
<!ATTLIST complex-component
  name CDATA #REQUIRED
  multiplicity CDATA "1"
>


<!-- dna
```

```
A DNA species.

The presence of relevant regions (such as genes, operators, promoters, etc)
along the DNA molecule is marked out using dna-site elements. The dna-site
elements may be organised using dna-section elements. The dna-sections are
informational only and carry no information in themselves.

The diffusivity of a dna entity may be specified by including diffusion
elements, but they will be ignored if the entity is mapped to a path.

Attributes:
* name: the name of the DNA species or dna-site
* size: the size of the DNA species; unit: base pairs
* shape: shape of the DNA species; circular or linear
* start, end: the position of the dna-site along the molecule; unit: base pairs
* strand: the strand (sense or antisense)

Example:
  <dna name="chromosome" size="82727" shape="circular">
    <dna-section>
      <dna-site name="PDHC:E1b" start="426" end="916" />
    </dna-section>
  </dna>
-->
<!ELEMENT dna (description?, (dna-section|dna-site)*, diffusion*)>
<!ATTLIST dna
  name CDATA #REQUIRED
  size CDATA #REQUIRED
  shape (circular|linear) #REQUIRED
>

<!ELEMENT dna-section (description?, (dna-section|dna-site)*)>

<!ELEMENT dna-site (description?)>
<!ATTLIST dna-site
  name CDATA #REQUIRED
  start CDATA #REQUIRED
  end CDATA #IMPLIED
  strand (sense|antisense) "sense"
>


<!-- diffusion
The diffusion element can be used to specify the diffusion constant D of the
species for some region. The diffusion types are:
* bulk: diffusion between two connected solid sites belonging to the region
* boundary: diffusion between two connected boundary sites belonging to the
  region
* path: diffusion between two connected path segments belonging to the region
* transboundary: diffusion between solid sites on either side of the boundary,
  for boundary sites belonging to the region. If the membrane is semipermeable,
  the permeability constant P should be given instead of D.

The unit of the diffusion constant D is um^2 s^-1.

Example:
  <species name="Ca2+">
    <description>Diffusion of Ca2+ ions</description>
    <diffusion region="universe" type="bulk" D="10.0" />
  </species>
-->
<!ELEMENT diffusion (description?)>
<!ATTLIST diffusion
  region CDATA #REQUIRED
  type (bulk|boundary|path|transboundary) #REQUIRED
  D CDATA #REQUIRED
>


<!-- ************************************************************************

processes

************************************************************************ -->
```

```
<!-- process
The process element defines a process involving one or more entities.

The entities may be located at the same site or at different sites, as specified
by the process-site elements. A process site is a site of a certain type,
connected to other process sites as specified by connected-to elements. The name
attribute of the connected-to element must match the name of another process-
site belonging to the same process, unless the process-site is a DNA site.
Participating entities are declared within the process-sites using reactant,
product and effector elements.

A DNA site may also be connected to its parent DNA entity, in which case it must
have the same path mapping as any other DNA sites connected to the same DNA
entity. This feature is useful when there are several copies of a gene, e g when
modelling plasmids. By tying the DNA sites to the host plasmid, all DNA sites
are guaranteed to be located on the same plasmid.

The rate of the process is given by the rate constant; a reverse rate can also
be given for reversible processes. The units used for the entity
concentrations are mol/L for entities in solution, mol/dmˆ2 for surface-bound
entities, mol/dm for path-bound entities, and mol for entities at point sites.
The unit of the rate follows from these choices (it is typically one of the
standard ones: M, 1/s, M/s, etc).

There are two kinds of effectors: cofactors and limiting effectors. A cofactor
is treated as a reactant that is not consumed by the process; it is the type
to use for catalysts. The limiting effectors enable or disable the process,
depending on the amount of the effector.

Examples:
  <process name="pyruvate -> acetyl CoA" rate="0.07" >
    <process-site name="1" region="universe" type="solid">
      <reactant name="pyruvate" />
      <effector name="pyruvate dehydrogenase complex" type="cofactor" />
      <product name="actetyl CoA" />
    </process-site>
  </process>
-->
<!ELEMENT process (description?, process-site+)>
<!ATTLIST process
  name CDATA #REQUIRED
  rate CDATA #REQUIRED
  reverse-rate CDATA #IMPLIED
>

<!ELEMENT process-site ((reactant|effector|product)*, connected-to*)>
<!ATTLIST process-site
  name CDATA #REQUIRED
  region CDATA #REQUIRED
  type (solid|boundary|path|point) #REQUIRED
>

<!ELEMENT connected-to EMPTY>
<!ATTLIST connected-to
  name CDATA #REQUIRED
  direction (positive|negative) #IMPLIED
>

<!ELEMENT reactant EMPTY>
<!ATTLIST reactant
  name CDATA #REQUIRED
  multiplicity CDATA "1"
>

<!ELEMENT effector EMPTY>
<!ATTLIST effector
  name CDATA #REQUIRED
  type (cofactor|require-gt|require-lt|require-eq|require-neq) "cofactor"
  multiplicity CDATA "1"
>

<!ELEMENT product EMPTY>
```

```
<!ATTLIST product
  name CDATA #REQUIRED
  multiplicity CDATA "1"
>


<!-- multistep-process
A multistep-process is a special kind of process that compresses a chain of
irreversible, monoreactant steps, each of which having the same rate, into one
process. This kind of element will typically be used for modelling transcription
and translation processes.

Attributes:
* name: the name of the multistep-process
* rate: reaction rate; unit: s^-1
* steps: number of steps

There are also some additional requirements on the involved entities:
* there must be exactly one reactant and one product entity, each with unit
  multiplicity. The reactant and product must be declared in process site of the
  same dimensionality.
* any effectors must be of the limiting kind (no cofactors)

Example:
  <multistep-process name="transcription:p1" rate=".033" steps="300">
    <description>Transcription of the 300-bp gene p1</description>

    <process-site name="promoter" region="Pp1" type="point">
      <reactant name="RNAPolII" />
    </process-site>

    <process-site name="terminator" region="Tp1" type="point">
      <product name="RNAPolII/mRNA-p1" />
    </process-site>
  </multistep-process>
-->
<!ELEMENT multistep-process (description?, process-site+)>
<!ATTLIST multistep-process
  name CDATA #REQUIRED
  rate CDATA #REQUIRED
  steps CDATA #REQUIRED
>


<!-- ************************************************************************

simulation parameters

************************************************************************ -->


<!-- initial-amount
An element used to define the amount of an entity at the beginning of the
simulation.

Attributes:
* entity: the name of the entity
* region, type: the sites that should be assigned an initial amount

The element contains an expression for a concentration that is given in M,
mol/dm^2, mol/dm, or mol for solid regions, boundary regions, paths, and point
sites, respectively.

If the initial state of a site is defined by two initial-amount elements, the
last definition is used.

Example (three mM of ATP throughout all solid sites in the model):
  <initial-amount entity="ATP" region="universe" type="solid">
    3e-3
  </initial-amount>
-->
<!ELEMENT initial-amount (#PCDATA|description)*>
<!ATTLIST initial-amount
  entity CDATA #REQUIRED
```

```
  region CDATA #REQUIRED
  type (solid|boundary|path|point) #REQUIRED
>


<!-- dna-mapping
Maps one DNA molecule to a path. The mapping remains for all time.

Attributes:
* entity: the name of the entity (must be of dna type)
* path: the path to which the dna is mapped.

Example:
  <dna-mapping entity="e coli genome" path="chromosome" />
-->
<!ELEMENT dna-mapping EMPTY>
<!ATTLIST dna-mapping
  entity CDATA #REQUIRED
  path CDATA #REQUIRED
>


<!-- constraint
This element can be used to define a constraint on the amount of an entity in
some region as a given function of time.

Attributes:
* entity: the name of the entity
* region, type: the sites where the amount should be constrained
* start-time, end-time: the time interval when the constraint should be active,
  defaults to [0, infinity). Unit: s

The element contains an expression for a concentration that is given in M,
mol/dm^2, mol/dm, or mol for solid regions, boundary regions, paths, and point
sites, respectively.

There cannot be two active constraints for the same entity at any site.

Example:
  <constraint entity="glucose" region="extracellular" type="solid" end-time="2">
    <description>linear glucose gradient from t=0 to t=2 s</description>
    0.05*t
  </constraint>

  <constraint entity="glucose" region="extracellular" type="solid"
      start-time="2">
    <description>constant glucose concentration for t >= 2 s</description>
    0.1
  </constraint>
-->
<!ELEMENT constraint (#PCDATA|description)*>
<!ATTLIST constraint
  entity CDATA #REQUIRED
  region CDATA #REQUIRED
  type (solid|boundary|path|point) #REQUIRED
  start-time CDATA "0"
  end-time CDATA #IMPLIED
>


<!-- parameter
This element is used to specify the value of a parameter.

Attributes:
* name: the name of the parameter
* start-time, end-time: the time interval where the parameter is defined,
  defaults to [0, infinity). Unit: s

The element contains an expression that specifies the value of the parameter as
a function of time.

The time intervals of two specifications of the same parameter may not overlap.

Example:
```

```
  <parameter name="lu">
    <description>Defining the lattice unit as a function of time to simulate
    linear growth</description>
    .1+0.005*t
  </parameter>
-->
<!ELEMENT parameter (#PCDATA|description)*>
<!ATTLIST parameter
  name CDATA #REQUIRED
  start-time CDATA "0"
  end-time CDATA #IMPLIED
>


<!-- snapshot
This element defines that an observation should be made. A snapshot is an
observation that shows the spatial distribution of an entity at a certain time.

Attributes:
* name: the name of the observation
* entity: the name of the entity
* region, type: the sites to observe. Note: the "any" type is only valid for
  absolute amount observations
* absolute-amount: specifies whether absolute amounts (unit: copy number) should
  be output instead of the normal concentrations (unit: mol, mol/dm^2, mol/dm,
  or M).
* time: the time when the observation should be made; unit: s

Example:
  <snapshot name="Initial distribution of ATP in the cytosol" entity="ATP"
    region="cytosol" type="solid" time="0" />
-->
<!ELEMENT snapshot EMPTY>
<!ATTLIST snapshot
  name CDATA #REQUIRED
  entity CDATA #REQUIRED
  region CDATA #REQUIRED
  type (solid|boundary|path|point|any) #REQUIRED
  absolute-amount (yes|no) "no"
  time CDATA #REQUIRED
>


<!-- time-series
This element defines that an observation should be made. A time-series is a
repeated measurement of the total amount of a species in a region.

Attributes:
* name: the name of the observation
* entity: the name of the entity
* region, type: the sites to observe. Note: the "any" type is only valid for
  absolute amount observations
* absolute-amount: specifies whether absolute amounts (unit: copy number) should
  be output instead of the normal concentrations (unit: mol, mol/dm^2, mol/dm,
  or M).
* start-time: the time when the observation should be started, defaults to 0.
  Unit: s
* interval: time between measurements; unit: s
* end-time: the time when the observation is finished, defaults to infinity.
  Unit: s

Example:
  <time-series name="Total ATP concentration in the cytosol" entity="ATP"
    region="cytosol" type="solid" start-time="0" interval="1" />
-->
<!ELEMENT time-series EMPTY>
<!ATTLIST time-series
  name CDATA #REQUIRED
  entity CDATA #REQUIRED
  region CDATA #REQUIRED
  type (solid|boundary|path|point) #REQUIRED
  absolute-amount (yes|no) "no"
  start-time CDATA "0"
  interval CDATA #REQUIRED
```

```
  end-time CDATA #IMPLIED
>


<!-- End of DTD -->
```

A.2 Order of Elements

The order of the elements in a model is not arbitrary. The geometry of the model should be specified first. Next comes the mechanism part, i e the entities and processes. Finally the simulation parameters are given: an initial state, constraints, and observations. Ordering the elements this way, preferably in separate files, is good practice as it makes the model construction easier to read and less error prone.

A.3 Additional Specifications

initial amounts. The initial amounts of all entities must be known for all sites. If no such declaration is made for a (site, entity) pair, a default value of zero is used.

the region attribute. Frequently, elements have a region attribute, specifying the set of sites for which the element applies. The region attribute can name an existing region, or it may specify a new region, constructed from existing regions by means of set operators.

The set operators are `+`, `*`, `-` and `!` for union, intersection, difference, and complement, respectively. Parentheses can also be used if necessary, and if a region name includes one of the special characters `+*-()%` or whitespace, it can be escaped by prefixing it with a percent sign (%).

expressions. Expressions, rather than numerical constants, are handled by several elements and attributes, including the rate attribute of the process element and the initial-amount element. Expressions are specified as is customary in computer programmes using the operators `+-*/()^`, the functions are `exp`, `log`, `sin`, `cos`, `sqrt` (square root), `step` (the Heaviside step function), and `delta` (the Kronecker delta function).

The variables, parameters and constants that can be used are:

- pi (π),
- NA (Avogadro's number/mol^{-1}),
- t (current time/s),
- lu (lattice unit/μm),
- T (absolute temperature/K), and
- r1, r2, and r3 (spatial coordinates of the site/μm).

A.4 Sample Model

This is a partial implementation of the bacteriophage lambda lysis/lysogeny switch model, described by Arkin, Ross and McAdams (1998). It demonstrates many concepts of Smartcell model building, and also points out the contrast

between a general modelling framework and a simulation engine written for a certain model. As will be seen, a number of tricks are required to re-formulate the rather specific model of Arkin *et al* into the more general Smartcell model description format.

A.4.1 Files and Modelling Basics

As is customary, the model is described in one file and the simulation parameters in another file. Each file starts with a header that identifies it as being an XML document:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE module SYSTEM "smartcell.dtd">
```

The first line declares that the document conforms to XML version 1, and that the file is an 8-bit text file (so that it can be interpreted correctly; XML always uses Unicode internally). The meaning of the second line is that the root element of the document is *module*, and that the definition of this element type is given in the Smartcell DTD (which, incidentally, resides in a file called smartcell.dtd). The root element of an XML file is the element that contains all other XML elements in the file; the XML standard states that there must be exactly one root element in each document.

As expected, the file that contains the model definition starts with a module declaration (the root element) and a description as follows:

```
<module name="Phage Lambda Lysis/Lysogeny Decision Switch">
  <description>
    A partial implementation of the model described in <reference><authors>
    Arkin A, Ross J and McAdams H H</authors><title>Stochastic Kinetic Analysis
    of Developmental Pathway Bifurcation in Phage Lambda-Infected Escherichia
    coli Cells</title><publication>Genetics 149:1633-1648 (1998)</publication>
    </reference>.

    The original model does not include any localisation/diffusion information,
    so a single, cell-size volume element is suitable. Other model
    geometries can of course also be used, but then one must take care to
    include diffusion specifications for the entities.
  </description>
```

The name of the module should be informative, and it is good practice to further describe the contents of a module with a description element. XML comments, special elements that start with "`<!--`" and end with "`-->`", can also be used, but their use is not recommended as they belong to the XML file rather than to the model. Description elements, on the other hand, can be attached to almost all kinds of elements, and do form part of the model. The description element always follows immediately after the start tag of the element it describes, by convention.

A.4.2 Geometry

As has already been mentioned, there is not much that can be done about the model geometry yet. Still, given the possibility to choose, a geometry consisting of a single volume element would be the one most appropriate for this model, because that is what was used in the original model by Arkin *et al*.

A.4.3 Chemical Reactions

The next part of the model is the mechanism part, which defines how entities interact. It starts out by specifying a number of standard chemical reactions taking place in the cytosol; a few representative ones are

- CI degradation:

$$\mathrm{CI} \xrightarrow{k} \emptyset; k = 0.0007 s^{-1}$$

- CI dimerisation:

$$2\mathrm{CI} \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{CI}_2; k_1 = 0.05 M^{-1} s^{-1}, k_{-1} = 0.5 s^{-1}$$

- CII degradation assisted by the protease P1:

$$\mathrm{CII} + \mathrm{P1} \underset{k2}{\overset{k_1}{\rightleftharpoons}} \mathrm{P1} - \mathrm{CII} \xrightarrow{k_3} \mathrm{P1};$$

$$k_1 = 0.01 M^{-1} s^{-1}, k_2 = 0.01 s^{-1}, k_3 = 0.002 s^{-1}.$$

These reactions are easily described using *process* elements, but before doing so the chemical species must be declared using the *species* element. One would normally specify the diffusion parameters of the entities at the same time, but this is not done here as the model does not include diffusion.

The dimer is declared using the complex element, where the dimerisation process is incorporated as a complex-formation element.

As can be seen, the CII degradation process is described as a series of elementary steps; potentially a very accurate model. When this level of accuracy is not required, it may be possible to replace the three reactions with a single reaction having CII as reactant and P1 as a cofactor. This simplification is safe as long as one of the forward reactions is much faster than the other, and the intermediate product is not interesting from a modelling point of view (as may be the case in certain types of inhibition).

```
<species name="CI" />
<process name="CI degradation" rate="0.0007">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="CI" />
  </process-site>
</process>

<complex name="CI2">
  <complex-formation region="cytosol" type="solid" rate="0.05"
    reverse-rate="0.5">
    <complex-component name="CI" multiplicity="2" />
  </complex-formation>
</complex>

<species name="CII" />
<species name="P1" />
<complex name="CII-P1">
  <complex-formation region="cytosol" type="solid" rate="0.01"
    reverse-rate="0.01">
    <complex-component name="CII" />
    <complex-component name="P1" />
  </complex-formation>
</complex>
<process name="P1-assisted CII degradation step 2" rate="0.002">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="CII-P1" />
  </process-site>
</process>
```

A.4.4 Transcription

A large part of the model is concerned with the transcription machinery and its regulation. The transcription part is therefore declared in a sub-module to make the model description file easier to read:

```
<module name="Transcription">
```

Ignoring the DNA of the host cell—and most of the phage DNA as well—we can describe the interesting part of the phage DNA as follows:

```
<dna name="phage lambda genome" size="48502" shape="circular">
  <description>The interesting part of phage DNA (see figure 1 in the paper by
    Arkin et al). Positional data taken from the database entry EMBL:LAMCG.
    The mRNA feature table entries are used for promoter and terminator
    positions, and misc_signal entries for the operator sites.
  </description>
  <dna-section>
    <dna-site name="TL2" start="33100" strand="antisense" />
    <dna-site name="cIII" start="33299" end="33463" strand="antisense" />
    <dna-site name="TL1" start="33494" strand="antisense" />
    <dna-site name="TL1.end" start="33494" strand="antisense" />
    <dna-site name="n" start="35037" end="35438" strand="antisense" />
    <dna-site name="NutL" start="35518" end="35534" strand="antisense" />
    <dna-site name="OL1" start="35591" end="35607" strand="antisense" />
    <dna-site name="OL2" start="35615" end="35631" strand="antisense" />
    <dna-site name="PL" start="35582" strand="antisense" />
    <dna-site name="cI" start="37227" end="37940" strand="antisense" />
    <dna-site name="PRM" start="37940" strand="antisense" />
    <dna-site name="OR3" start="37951" end="37967" strand="sense" />
    <dna-site name="OR2" start="37974" end="37990" strand="sense" />
    <dna-site name="OR1" start="37998" end="38014" strand="sense" />
    <dna-site name="PR" start="38023" strand="sense" />
    <dna-site name="cro" start="38041" end="38241" strand="sense" />
    <dna-site name="NutR" start="38265" end="38281" strand="sense" />
    <dna-site name="PRE" start="38343" strand="antisense" />
    <dna-site name="OE1" start="38372" strand="antisense" />
    <dna-site name="OE2" start="38373" strand="antisense" />
    <dna-site name="TR1" start="38370" strand="sense" />
    <dna-site name="cII" start="38360" end="38653" strand="sense" />
    <dna-site name="o" start="38686" end="39585" strand="sense" />
    <dna-site name="p" start="39582" end="40283" strand="sense" />
    <dna-site name="TR2" start="40624" strand="sense" />
  </dna-section>
</dna>
```

There is one special site in this list that is not present in either the original model or the database entry. It is named TL1.end, and its purpose will be explained later.

The genes in the lysis/lysogeny decision switch are regulated in several ways. Transcription initiation depends on ligands binding to the operator sites located close to the promoters. The original model takes on a statistical, thermodynamic perspective on ligand binding; each operator/ligand state is assigned a free energy, and the transcription initiation rate from each state is weighed by the probability of the system being in the specific state. This gives a total rate of transcription initiation as a function of ligand concentrations. It is obvious that this construct can not be transferred directly to a Smartcell model. To overcome this, each ligand binding process is addressed individually for each operator state.

The sample processes below show part of the transcription initiation at the P_{RE} promoter (see figure 3). Although the actual inititiation step is the closing of the RNAP complex, the RNAP has to bind first, and CII is also involved.

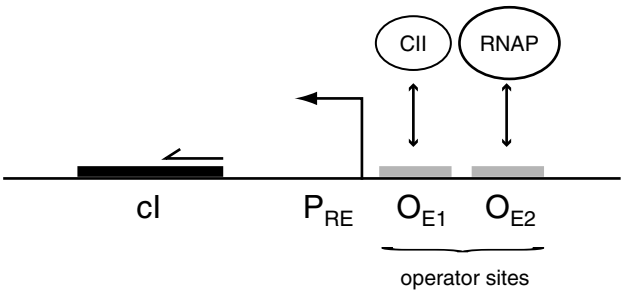

Figure 3: The P_{RE} promoter region. Each operator site may bind zero or one molecule of a certain species, so there are four distinct operator states.

The quotient between association rate and dissociation rate is related to the change in free energy as

$$\frac{k_1}{k_{-1}} = K = e^{-\Delta G/RT}, \tag{12}$$

which means that we can find the value of k given k_{-1}, ΔG, and T. Consider the case of CII binding to O_{E1}. According to Arkin *et al*, the change in free energy associated with this event is -9.7 kcal/mol when O_{E2} is unoccupied, and -11.6 kcal/mol when O_{E2} binds RNAP. Thus, in the former case, and under the assumption that $T = 298K$,

$$K = \exp(-\Delta G/RT) \approx 1.30 \cdot 10^7. \tag{13}$$

By assuming the largest of the rates to be 10^3 s^{-1}, we get $k_1 = 10^3$ s^{-1} and $k_{-1} = 0.77 \cdot 10^{-5}$ s^{-1}. This particular process is given below in Smartcell notation (the other association/dissociation pairs are similar). The process involves three process sites: the cytosol, the O_{E1} operator site, and the O_{E2} operator site. In the binding process, one CII molecule moves from the cytosol site to the O_{E1} site, which means that the two sites must be connected. Only one CII molecule can bind to the O_{E1} site at any time, so the process is allowed only when the number of CII molecules at O_{E1} is zero. This process also depends on the absence of RNAP at O_{E2}; the case where RNAP is present has another rate and is described by another process.

```
<process name="CII binding to OE1, OE2 empty" rate="1e3">
  <process-site name="cyt" region="cytosol" type="solid">
    <connected-to name="o1" />
    <reactant name="CII" />
  </process-site>
  <process-site name="o1" region="OE1" type="point">
    <connected-to name="phage lambda genome" />
    <effector name="CII" type="require-eq" multiplicity="0" />
    <product name="CII" />
  </process-site>
  <process-site name="o2" region="OE2" type="point">
    <connected-to name="phage lambda genome" />
    <effector name="RNAP" type="require-eq" multiplicity="0" />
  </process-site>
</process>
```

```
<process name="CII releasing from OE1, OE2 empty" rate=".77e-5">
  <process-site name="cyt" region="cytosol" type="solid">
    <connected-to name="o1" />
    <product name="CII" />
  </process-site>
  <process-site name="o1" region="OE1" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="CII" />
  </process-site>
  <process-site name="o2" region="OE2" type="point">
    <connected-to name="phage lambda genome" />
    <effector name="RNAP" type="require-eq" multiplicity="0" />
  </process-site>
</process>
```

We now turn to transcription elongation. In its most basic form, transcription elongation is typically described using the multistep process element. The following model fragment describes the transcription starting at the P_L promoter (see figure 4). This promoter is situated next to a 2483 nucleotides long open reading frame, and is chosen as an example because it has some interesting properties. Each elongation step can be written

$$\mathrm{RNA}(n) \xrightarrow{k} \mathrm{RNA}(n+1),$$

where $k = 30\ \mathrm{s}^{-1}$ is the transcription rate.

```
<multistep-process name="PL transcription" rate="30" steps="2483">
  <process-site name="PL" region="PL" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAPc" />
  </process-site>
  <process-site name="TL2" region="TL2" type="point">
    <connected-to name="phage lambda genome" />
    <product name="RNAP-mRNA(PL,TL2)" />
  </process-site>
</multistep-process>
```

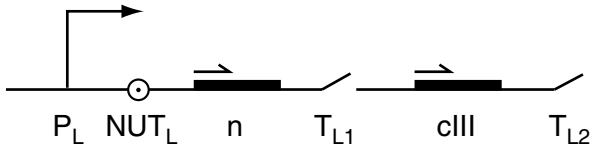

Figure 4: The P_L promoter and the following open reading frame. Transcription may be terminated at T_{L1} or T_{L2}, leading to a monocistronic or a polycistronic transcript, respectively. NUT_L is an antitermination site that is not included in this example.

Assuming that the RNA polymerase and the mRNA detach from the termination site more or less instantaneously (the assumption is expressed through the choice of reaction rate—there may be more accurate figures available in the literature):

```
<process name="TL2 termination" rate="1e9">
  <process-site name="TL2" region="TL2" type="point">
    <reactant name="RNAP-mRNA(PL,TL2)" />
  </process-site>
  <process-site name="cyt" region="cytosol" type="solid">
```

```
    <connected-to name="TL2" />
    <product name="RNAP" />
    <product name="mRNA(PL,TL2)" />
  </process-site>
</process>
```

The P_L ORF adds both termination and antitermination to the basic mode of operation. To include termination, the process is split into two: transcription upstream and downstream from the termination site. There are two possible outcomes from the RNAP-mRNA complex at the termination site: elongation may continue, resulting in a complete mRNA, or the premature mRNA may be released. As in the original model, transcription also slows down at the termination site.

```
<multistep-process name="PL transcription, part 1" rate="30"
  steps="2089">
  <process-site name="PL" region="PL" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAPc" />
  </process-site>
  <process-site name="TL1" region="TL1" type="point">
    <connected-to name="phage lambda genome" />
    <product name="RNAP-mRNA(PL,TL1)" />
  </process-site>
</multistep-process>

<process name="TL1 termination" rate="25">
  <process-site name="TL1" region="TL1" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAP-mRNA(PL,TL1)" />
  </process-site>
  <process-site name="cyt" region="cytosol" type="solid">
    <connected-to name="TL1" />
    <product name="RNAP" />
    <product name="mRNA(PL,TL1)" />
  </process-site>
</process>

<process name="TL1 elongation" rate="5">
  <process-site name="TL1" region="TL1" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAP-mRNA(PL,TL1)" />
  </process-site>
  <process-site name="TL1.end" region="TL1.end" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAP-mRNA(PL,TL1)" />
  </process-site>
</process>

<multistep-process name="PL transcription, part 2" rate="30" steps="394">
  <process-site name="TL1.end" region="TL1.end" type="point">
    <connected-to name="phage lambda genome" />
    <reactant name="RNAP-mRNA(PL,TL1)" />
  </process-site>
  <process-site name="TL2" region="TL2" type="point">
    <connected-to name="phage lambda genome" />
    <product name="RNAP-mRNA(PL,TL2)" />
  </process-site>
</multistep-process>
```

As a matter of fact, the transcription of this particular gene is even more complex than this, because it also involves an antitermination process: the RNAP-mRNA complex may be modified at the NUT_L site to become "immunised" against premature termination at T_{L1}. The implementation of this is not shown here, but is easily put together using the above concepts. We can therefore end the transcription submodule:

```
</module>
```

A.4.5 Translation

The translation of messenger RNA starts immediately after the transcription start, as the ribosomes attach to the free end of mRNA. In the model of Arkin *et al*, the ribosomes compete for the free end of mRNA with RNAse E. The reaction rates are related in such a way that an average of ten protein molecules are synthesised from each transcript, before the transcript is degraded by the RNAse.

It is difficult to implement this exact timing in a Smartcell model. With the transcription model described above, there is no way of knowing when a free end is available. The approach taken here is to allow the transcription process to finish before letting any ribosomes or RNAse molecules attach. To compensate for lost translation time, it is then assumed that half of the mRNA has already been translated when the ribosome attaches. This approximation certainly has its drawbacks (for example, ribosome crowding is totally ignored) and it may even be totally unacceptable for some applications. There is also a loss of semantics in the model. On the other hand, it does get the averages right and also the skewed time distributions characteristic of gene expression (McAdams and Arkin, 1997). The competition between translation initiation and mRNA degradation can be formulated as

```
<process name="mRNA(PL) translation initiation" rate="0.002">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="ribosome" />
    <effector name="mRNA(PL)" />
    <product name="ribosome-mRNA(PL)" />
  </process-site>
</process>

<process name="mRNA(PL) degradation" rate="0.03">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="mRNA(PL)" />
    <effector name="RNAse E" />
  </process-site>
</process>
```

As the rate of the second process is specified in a peculiar way in the original model, it was assumed that the free concentration of RNAse E is 3 nM in the cytosol. One can then compare the velocities of the two reactions; the velocity of the translation initiation reaction should be ten times that of the degradation reaction, as the average number of transcripts from one mRNA molecule is ten.

Note that the mRNA molecule is removed as soon as the RNAse attaches; this correctly models the function of the system, but may yield strange mRNA concentration measurements.

The last part of the translation process is the translation elongation, modelled as a multistep process, and the (almost instantaneous) dissociation of the ribosome from the nascent protein. The rate is 100 amino acids or 33 codons per second:

```
<multistep-process name="mRNA(PL) translation elongation" rate="33" steps="66">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="ribosome-mRNA(PL)" />
    <product name="ribosome-N" />
  </process-site>
</multistep-process>

<process name"ribosome-N dissociation" rate="1e9">
  <process-site name="cyt" region="cytosol" type="solid">
    <reactant name="ribosome-N" />
```

```
    <product name="ribosome" />
    <product name="N" />
  </process-site>
</process>
```

Now this particular mRNA is in fact polycistronic – the gene coding for CIII is on the same mRNA transcript. According to Lewin (2000; p 128) the ribosomes do not translate the coding regions of a polycistronic mRNA in sequence, but release and assemble anew at each start codon. This means that the translation of the genes on the same mRNA can be treated as separate processes and one needs only add another multistep process for the CIII translation (not shown here).

This is the end of the model description section, and so we turn to the simulation parameters module.

A.4.6 Cell Growth by Parameter Variation

Cell growth is a fundamental issue for an organism whose life cycle lasts for only about 35 minutes. The rapid volume expansion means that dilution may be more important than degradation in regulatory circuits.

In the Arkin *et al* model, the volume increases linearly from 10^{-15} L to the double volume in 35 minutes. The ideal way of modelling cell growth in the Smartcell environment would be to add more volume elements as time proceeds, but unfortunately this is not yet possible. Instead, one can maintain a fixed number of volume elements but increase their size with time. As the geometry model consists of a single cubic volume element, the side of which is the lattice unit, the lattice unit λ should start out at 1 μm in order to obtain the required volume. As the volume is proportional to λ^3, the expression for the lattice unit looks as follows:

```
<parameter name="lu">1 + 0.020*t^(1/3)</parameter>
```

A.4.7 Initial Amounts and Constraints

All of the phage lambda gene products start out at zero concentration. Therefore it is not necessary to add initial-amount elements for them, but doing so is good practice:

```
<initial-amount name="Cro" region="universe" type="solid">
  0
</initial-amount>
```

The initial concentrations for all householding proteins were given in the original paper, so they are also easily added. However in some cases the choice must be made whether the concentration should be constrained. Consider RNAP (RNA polymerase) as an example: even though it is temporarily immobilised by certain processes, the effect of the fluctuations is negligible. One can therefore put a time-invariant constraint on the concentration at 30 nM, thereby simplifying the model and reducing execution time:

```
<constraint name="RNAP" region="cytosol" type="solid">
  30e-9
</constraint>
```

The final example on initial amounts is the MOI (multiplicity of infection), defined as the number of post-infection phage particles in a specific cell. The MOI is represented in the model by the number of copies of phage DNA. Because each DNA molecule should be mapped to a separate path, a number of paths will have to be defined for MOI larger than one. This example shows how a single copy of phage DNA is mapped to an existing path called *chromosome*.

```
<dna-mapping entity="phage lambda genome" region="chromosome" />
```

It is assumed that infection takes place "early enough" in the cell cycle so that the phage DNA is present at the start of the cell cycle.

A.4.8 Simulation Output

The interesting outputs from the simulation are the amounts of Cro and CI dimers as functions of time. This information is gathered in the form of two time series (the interval between samples was arbitrarily chosen to be one second):

```
<time-series name="Cro2(t)"
  entity="Cro2" region="cytosol" type="solid" interval="1" />

<time-series name="CI2(t)"
  entity="CI2" region="cytosol" type="solid" interval="1" />
```

That concludes this example. Although not complete and testable, it should give a hint at what can be done and what can not be done with the Smartcell framework.

B Mesoscopic Diffusion

In a macroscopic model, the diffusion of an entity in solution is easily expressed as a differential equation, and added to the system of coupled differential equations that describe the interactions between species. Diffusion across boundaries or along paths is handled in much the same way. This is not quite the case for stochastic kinetics. In this appendix a new method for introducing diffusion into a mesoscopic model is presented.

To start with, let us take a look at the macroscopic diffusion equation. In a macroscopic model, the concentration C of some species is assumed to be continuous in space. Consider a small volume element δV which is centred at $\mathbf{r}$. There are several factors contributing to the flux of matter in and out of this volume element. Let $S(\mathbf{r})$ denote the amount of species that is created in δV per unit time and volume (S may of course assume negative values as well). We also introduce the flux, denoted by $\mathbf{j}(\mathbf{r})$. These quantities are related by a continuity relation,

$$\iiint_{\delta V} \frac{\partial C}{\partial t}\, dV = \iiint_{\delta V} S dV - \iint_{\partial \delta V} \mathbf{j} \cdot d\mathbf{S} \tag{14}$$

Applying Gauss' theorem yields

$$\iiint_{\delta V} \frac{\partial C}{\partial t}\, dV = \iiint_{\delta V} S dV - \iiint_{\delta V} \nabla \cdot \mathbf{j}\, dV \tag{15}$$

In the limit of vanishingly small δV, the equation becomes exact in the integrands and reads $\partial C/\partial t = S - \nabla \cdot \mathbf{j}$. The factors contributing to the flux are:

diffusion. Fick's law states that the diffusion flux $\mathbf{j}_D = -D\nabla C$. Here, D is the diffusion constant and C is the concentration.

ion drift due to electrostatic fields. Ionic species have a drift speed $\mathbf{s}$, which gives rise to an ion flux $\mathbf{j}_E = -\mathbf{s}C$. The drift speed is related to the electric field by the relation $\mathbf{s} = u\mathbf{E}$, and the ion mobility u is in turn given by the Einstein relation $u = zDe/k_BT$. Here, z is the ionic valence, e is the elementary charge, k_B is Bolzmann's constant and T is the absolute temperature. This leads to the equation $\mathbf{j}_E = -zDeC\mathbf{E}/k_BT$. See Atkins (1994) for a more detailed descriptions of these relations.

convection due temperature gradients, stirring, etc. The convective flux is given by $\mathbf{j}_C = -\mathbf{v}C$, where $\mathbf{v}$ is the velocity field.

By superposition we arrive at an expression for $\mathbf{j}_i$ that, through the introduction of the constants a_i and $\mathbf{b}_i$, can be written

$$\mathbf{j}_i = -a\nabla C - \mathbf{b}C. \tag{16}$$

The mesoscopic flux is neither space-continuous nor time-continuous. Rather, it is a stochastic event; a transient movement of entities between neighbour sites. What is the propensity of such an event? We focus on one of the sites, say u (see figure 5), and try to find the escape probability ε_{uv}, defined as

$\varepsilon_{uv}\delta t \equiv$ average probability, to first order in δt, that a particle will escape from u to v in the next time interval δt.

The one-to-one correspondence between the escape probability ε_{uv} and the reaction parameter c ensures that the two kinds of stochastic events can be safely mixed. The flux of interest is a pure outflux, averaged over the whole surface and also averaged over each particle. The character of the boundary influences the flux as well. An infinitely thin, permeable boundary is assumed to behave as if there were no boundary at all. A permeable boundary with thickness, on the other hand, is assumed to slow down the diffusion by a linear factor P/D, just as in the macroscopic case (see Lakshminarayanaiah, 1984, pp 45-46).

Assuming that the particles are uniformly distributed throughout the volume element, that their energies have the Bolzmann distribution and that the constants a and $\mathbf{b}$ are known, it should be possible to calculate ε_{uv} exactly. This turns out to be a difficult task, however. Stundzia and Lumsden (1996) describe such a calculation for the special case where $\mathbf{b} = 0$. Their approach is to introduce diffusion terms into the chemical master equation and solve the equation using Green's functions. The following calculation uses some techniques from Stundzia and Lumsden's work, but is less formal. On the other hand, it does not require infinite series expansion of the Greens functions for large times.

The net flux from u to v can be written as

$$\begin{aligned} j_{uv} &= \iint_\sigma \mathbf{j} \cdot d\mathbf{S} \\ &= \iint_\sigma (-a\nabla C - \mathbf{b}C) \cdot d\mathbf{S} \\ &\approx \lambda^2 \langle -a\nabla C - \mathbf{b}C \rangle_\sigma \cdot \hat{\mathbf{n}} \end{aligned} \tag{17}$$

where the averaging approximation is justified by the fact that the boundary area is small. Here, $\hat{\mathbf{n}}$ is the unit outward normal from u. The concentration gradient on σ can be approximated by

$$\langle \nabla C \rangle_\sigma \approx \frac{C_v - C_u}{\lambda}\, \hat{\mathbf{n}}. \tag{18}$$

Likewise, the average concentration on σ can be taken as the average of the concentrations in u and v:

$$\langle C \rangle_\sigma \approx \frac{1}{2}(C_u + C_v). \tag{19}$$

This leaves us with the equation

$$j_{uv} \approx a\lambda(C_u - C_v) - \frac{1}{2}\lambda^2(\mathbf{b} \cdot \hat{\mathbf{n}})(C_u + C_v). \tag{20}$$

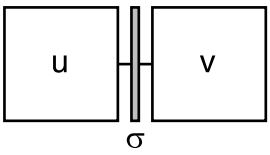

Figure 5: Two solid sites u and v, separated by a boundary σ.

Back in the mesoscopic model, each particle in u has the probability ε_{uv} of escaping to v in the next unit time interval. These events are independent, so an average of $X_u\varepsilon_{uv}$ particles will escape per time unit. The reverse process takes place simultaneously in v, leading to an average net flux $X_u\varepsilon_{uv} - X_v\varepsilon_{vu}$ from u to v. For reasons of consistency, we require that this average net flux be equal to the j_{uv} calculated above. Therefore, by identification,

$$\begin{aligned} \varepsilon_{uv} &= \frac{C_u}{X_u}(a\lambda - \frac{1}{2}\lambda^2(\mathbf{b}\cdot\hat{\mathbf{n}})) \\ &= \frac{a}{N_A\lambda^2} - \frac{\mathbf{b}\cdot\hat{\mathbf{n}}}{2N_A\lambda} \end{aligned} \tag{21}$$

It should be noted that $\hat{\mathbf{n}}$ is the outward unit normal relative to each volume element, so that $\hat{\mathbf{n}}_v = -\hat{\mathbf{n}}_u$.

A similar calculation for the two-dimensional case gives exactly the same result. With this expression, a diffusion movement will be present even if the concentrations in the neighbour sites are equal, as opposed to the situation in deterministic models. The stochastic model comes closer to the thermodynamic character of diffusion—including interesting fluctuation phenomena—but it may do so at the cost of a higher computational load.

If the external fields are very strong, ε_{uv} may become negative. To prevent this unphysical condition, the lattice unit must be chosen small enough so that at all times

$$\lambda \leq \frac{2a}{|\mathbf{b}\cdot\hat{\mathbf{n}}|}. \tag{22}$$

However, it is very unlikely that any external fields will be included into the model. The electrostatic interaction will be very weak in the solvent, due to the screening effect of the water (according to Luis Serrano). But a strong electric potential can be maintained across membranes, so it could be useful in this case. A model of a calcium wave in a neuronal cell, that includes a membrane potential, is described in Schaff *et al* (1997).

C Subgraph Mapping

This algorithm takes as inputs

- the geometry graph $G = \langle S, E \rangle$, where $S = \{s_1, s_2, \ldots, s_N\}$ is the set of vertices (sites) and $E = \{(i, j)\}$ is the set of edges. Because the graph is undirected, $(i, j) \in E \implies (j, i) \in E$.
- the local geometry graph $G' = \langle S', E' \rangle$, that is to be mapped onto G. The edges of the local geometry graph are similar to the edges of the geometry graph, but its vertices are subsets of S: $S' = \{s'_1, s'_2, \ldots, s'_M\}, s'_i \subseteq S$.

and outputs

- all mappings $M = \{m_1, m_2, \ldots, m_M\}$ for which $m_i \in s'_i$ and $(i, j) \in E' \implies (m_i, m_j) \in E$.

Stated less formally, a mapping is constructed by selecting one site from the allowed region of each process site, so that the connections between process sites also exist in the geometry graph.

Figure 6 shows an outline of the algorithm, which is given in detail below.

1. Set all m_i to null and clear the state stack.
2. Select any i for which m_i is null. To improve performance, make the selection so that the number of edges in E' involving i is maximised.
3. Find C, the set of candidates for m_i :
   (a) Set $C = s'_i$.
   (b) Remove all non-null m_j from C (no single site may be mapped to two process sites).
   (c) For each non-null m_j :
   if $(i, j) \in E'$
   set $C \leftarrow C \cup \{s | (s, m_j) \in E\}$.
   (If the process site is connected to another process site j that is already mapped, then all sites in C that are not connected to m_j are removed).
4. If $C = \emptyset$, go to step 8.
5. Select one item c from C, set $C \leftarrow C$ $\{c\}$, and push (i, C) onto the state stack.
6. If there are any null entries left in M, go to step 2.
7. Output the mapping. Go to step 8.
8. If the state stack is empty, terminate.
9. Pop (i, C) from the state stack and set m_i to null. Go to step 4.

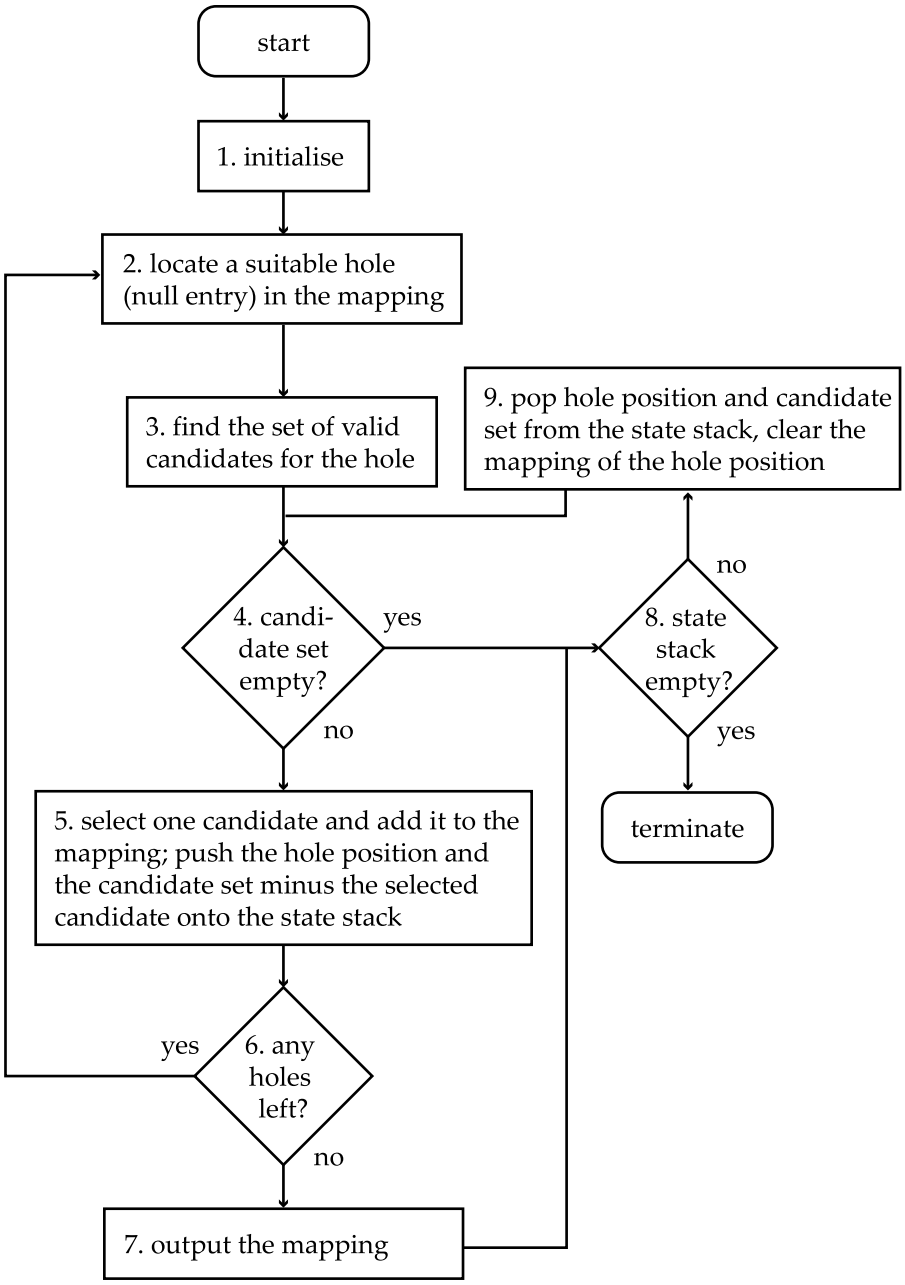

Figure 6: An overview of the subgraph mapping algorithm. The numbers are the same as in the textual description.