INAUGURAL-DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von

Diplom-Biologin Ute Platzer

aus Hamburg

Tag der mündlichen Prüfung: .....

Simulation of Genetic Networks in Multicellular Organisms

Gutachter: Prof. Dr. Jeremy C. Smith

Prof. Dr. Werner Buselmaier

Contents

Part I.

Introduction

This part explains the motivation for this work and gives a short overview of the status of current research in the area of genetic networks, as well as an introduction to the main model organisms used in the simulation: *Caenorhabditis elegans*.

1. Overview and Motivation

Recent methods for large-scale acquisition of biological data have led to an enormous collection of information on gene sequences, gene expression, and protein structures. However, this has not led to the expected concurrent increase in knowledge. Therefore, advanced methods for the interpretation of data are required, not only on the molecular scale, but also on a higher level of whole cells and even whole organisms. The scientific field that has focused on this is called *systems biology*. The aim of systems biology is to understand how whole organisms work and how their molecular compounds interact to make up a living being.

One approach taken is to simulate an organism. The ulterior motive is that if a computer algorithm is found that reproduces a biological process, then scientists can look at the algorithm, manipulate it, and gain some insight on how the biological mechanism works.

A lot of work has already been done on simulation of singlecellular organisms such as bacteria or yeast. Even some multicellular models have been created and simulated successfully. One of these is a model of the *Drosophila* embryo by Bodnar (1997). He suggests

> [...] that enough data is available to do the same for *Caenorhabditis elegans* embryogenesis [...]

At the time I started to work on my thesis in the division Medical and Biological Informatics at the German Cancer Research Center in Heidelberg, the software framework CELL-O was being developed there. It was then used for three-dimensional simulation of cells. It was specialised on simulation of *C. elegans*, but could easily be extended to other cellular systems.

Therefore I decided to focus on the simulation of embryonic development of *C. elegans* and to develop a new method for the simulation of genetic regulatory networks in multicellular organisms. After studying current literature on regulatory interactions in the early embryo, my head was stuffed with genes and interactions, and I realised that a tool was needed that allowed to store the data in a comprehensive way. Thus I developed a graphical editor for genetic networks. After implementing the simulation algorithms I decided that GENE-O-MATIC, as the application was called by now, might be a valuable tool for any experimental biologist, and thus extra attention was paid to the graphical user interface, to make it most convenient and easy to use.

I applied the program to build a model of blastomere fate specification in *C. elegans* (consisting of a genetic network plus cells), and to simulate its behaviour, i.e. the expression and activity of the genes and cell movement and division over time. Because there is not only a lot of research going on about *C. elegans*, but also on other model organisms, I wanted to demonstrate the applicability of GENE-O-MATIC to other systems, which was done by simulating *Arabidopsis thaliana*.

The results of both the *C. elegans* and *Arabidopsis* simulations show very good agreement with experimental data. In addition, modifications to the *C. elegans* network reproduce the

effects seen in animals with the corresponding mutations. In some cases, assumptions had to be made to compensate for gaps in the data, which led to the proposal of experiments that could be performed to gain more information. These findings show that GENE-O-MATIC is a powerful aid to model building, experimental design, and hypothesis testing in biology.

1.1. Outline

The first part of the thesis gives an introduction to the subject. The state of the art in genetic network analysis and simulation is presented, and some approaches to modelling biological systems are described to compare them later to the approach taken in this thesis. Biological facts about the modelled organisms are given to facilitate the understanding of the results for non-biologists.

The second part of the thesis portrays the software used and implemented during the thesis. Extensions to the existing software framework are explained, and the implementation of the new application GENE-O-MATIC is described. The last chapter in this part delineates the details of the simulation algorithms.

The results achieved with the simulation software are depicted in the next part. Different models and their simulation results are presented and compared to experimental data.

In the final part, the results and their implications for experimentalists are discussed. The benefits and drawbacks of the chosen simulation method are contrasted and prospects for the future given.

The appendix contains additional information and tables. The user manual explains how to use GENE-O-MATIC. A short overview of UML aids in understanding the diagrams in chapters 5 and 6. The tables summarise biological data about *C. elegans* embroynic development.

2. Genetic Networks and Biological Simulation

The term *genetic network* denotes a set of regulatory entities and their interactions (cf. figure 2.1). It is most often used on the level of transcriptional regulation (thus the word *genetic*), but may also include translational or even protein activity regulation. In contrast to metabolic networks, which describe the influence enzymes have on chemical reactions, genetic networks describe the influence one molecule has on another. Genetic networks may include DNA sequences such as genes, promoters, and other regulatory sequences, mRNAs, and all types of proteins. As a collective term for the elements of a genetic network, here the term *biological information carrier* (BIC) is used. Often the term *gene* is used, but this is misleading because it is not always clear if a real gene or the collective term for gene, mRNA, or protein is meant. Therefore I prefer to use the more general term BIC instead. Arrows between the BICs in the network depict interactions. Usually there are two types of interactions: activating (painted with an arrowhead) and inhibiting (with a crossbar).

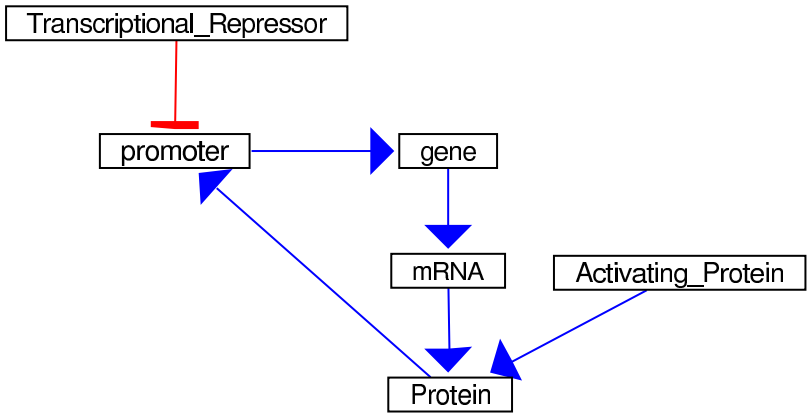

Figure 2.1.: A genetic network may contain promoters and other regulatory sequences, genes, mRNAs, and proteins (jointly called *biological information carriers* or BICs). Arrows between the elements depict how the elements influence each other. Here, blue arrows with arrowheads indicate a positive influence, and red arrows with crossbars indicate inhibitory influence.

2.1. Biological Relevance of Regulatory Networks

For a long time the complexity of biological experiments forced molecular biologists to focus their research on single proteins, or linear signalling cascades. The simultaneous experimental analysis of whole networks became feasible only with the invention of the microarray technique. Besides, it was assumed that different signalling pathways did not crosstalk, because nobody could imagine how that should work. However, newer results show that there are indeed cases where pathways interact and influence each other, and it becomes apparent that this is not only an exception, but may well be the rule. Therefore a holistic view on biological processes becomes more and more important.

The first genetic networks were simple paintings made by experimentalists. Their aim was to give an overview of a regulatory process. Now, the acquisition of large amounts of data by means of cDNA arrays, RNAi screens, yeast one- and two-hybrid screens, and co-immunoprecipitation, for example, makes larger, automatically-generated paintings necessary. Still, the fully automated elucidation of regulatory interactions from experimental data is not possible yet, and the significance of the interpretation of large datasets by computational methods such as clustering is arguable. Furthermore, there are processes such as embryonic development which are simply not accessible to the microarray technique because protein expression in single cells needs to be assessed. Therefore it is often required to compile a genetic network by hand, after collecting data gathered in hundreds or thousands of experiments. The effort repays, however, because computational or mathematical analysis of genetic networks can give new insights into the regulatory process.

2.2. Mathematical Models of Genetic Networks

In most multicellular organisms, each cell in an organism contains identical genetic information. But different cells use this information in different ways. The use of genetic information can be regulated at three levels: DNA (transcription), mRNA (splicing and translation), and protein (activity and degradation).

Several computer programs have been developed to analyze genetic networks and to identify stable states, which correspond to differentiated cell types. These programs present a static view of the differentiation process because they identify the stable states but do not describe the process of differentiation. Furthermore they do not take into account cell-cell interactions which are an important means of relaying fate specification signals. Nonetheless they can give valuable information on the soundness of a network, for example if the calculated stable states do not match the observed cell types, something must be wrong or at least still missing. For a recent review on modelling and simulation of genetic networks, see (de Jong, 2002).

Drosophila is the first organism for which a complex developmental model based on genetic data has been published (Bodnar, 1997; Bodnar and Bradley, 2001). Bodnar (1997) suggests that there is already enough data to perform similar simulations for other organisms, and specifically names *C. elegans*.

2.3. Simulation of Biological Processes

The comparison of simulation results with reality can give important information about the correctness of the underlying model. On the one hand, models that do not have the necessary level of precision, or those where some elements are missing or mis-placed will never produce the expected results in a simulation. On the other hand, if the simulated results are totally consistent with reality, this does not necessarily mean that the model adequately represents reality, too. Often many different models can lead to the same simulation results, and only one of them — if at all — is correct. Therefore, simulation can help to falsify a model, but can never be used to prove a model's correctness. Nonetheless, the more simulations of a model lead to predicted results, the higher the probability that the model is indeed correct. This is especially useful in biology, where modifications can be made to the model in similar manner as to the real system, for example knockouts, mutations, or cell ablation. Conversely, if a model shows high reliability, it can be used to make predictions on the outcome of experiments. Therefore simulation — *in silico* experiments — can be a helpful supplement and extension to biological *in vivo* and *in vitro* experiments.

2.4. Continuous *versus* Discrete Simulation

There are two basically different approaches to the simulation of biological or any other systems. One is the continuous approach. All variables (time, positions, concentrations, etc.) are real numbers. This approach is well-suited for models where a lot of quantitative data is available as an input to the model. Examples include models of chemical or enzymatic reactions and diffusion processes. Continuous models are often formulated as a set of differential equations, which relate the variables to their rates of change. Numerical integration of the differential equations allows to make predictions — to simulate — the behaviour of the system.

The other approach is the discrete one. Here, the variables can take only discrete, "quantised" values. The state of the system does not change continuously, but only at specific points in time, or in regular intervals. A special type of discrete simulation is the *event-based* simulation. Simulated entities (cells or molecules, for instance) are not treated as homogeneous suspensions, but as individual objects. The objects in the simulation generate events, such as cell division or transcription of a gene. These events of course do not take place continuously, but after defined time steps. Discrete simulation is suitable for modelling systems where the overall process is well understood, but little is known about the quantitative details.

2.5. Qualitative *versus* Quantitative Models

Independent of the type of simulation (continuous or discrete), a choice has to be made concerning the level of abstraction. A quantitative model is directly based on quantitative experimental data. It offers high precision and a very detailed representation of reality. The disadvantages of quantitative models are that precise data are required to build the model in the first place, and that the system must be described by means of mathematical formula that make use

of the data. For chemical and physical systems, both conditions can be satisfied easily, and also for biochemical systems. For more complex processes in biology, such as development, there are too often neither formulas nor quantitative data available.

In contrast, qualitative models represent a high level of abstraction. No exact values for parameters are required, the model describes only *if* something happens and not *how much* of it happens. Many people argue that qualitative descriptions are superficial, but one must understand the basic properties of the system before a more detailed model can be created. Genetic networks in their original form (as drawings) are predestined to be modelled qualitatively, as the arrows do not offer quantitative information on the interactions.

3. The Nematode *Caenorhabditis elegans*

Caenorhabditis elegans is a small nematode. In contrast to many of its relatives, it is free-living and not parasitic. It inhabits the soil and feeds on bacteria. The worm reproduces by egg-laying. The embryos hatch from the eggs after several hours and undergo four so-called larval stages before they become fertile adults. The adult worm has a length of about 1 mm. The egg is approximately 50 μm long. There are two sexes, hermaphrodite and male. During embryonic development, both sexes develop almost identically. Reproductive organs are formed mainly during the larval stages. Development inside the egg takes about 14 hours in total. The following larval development takes another 50 hours. Adults have a lifespan of about two weeks (Wood, 1988, p. 1f.).

3.1. *C. elegans* as a Model Organism

One of the particularities of *C. elegans* is its invariant cell lineage. In every individual worm the pattern of cell divisions is identical. This is the reason for the invariant cell number, too: the adult hermaphrodite has 959 cells, 558 of which are formed inside the egg. The adult male consists of 1031 cells, 560 of which are formed in the egg. Together with the fact that the worm is transparent during all developmental stages — even the egg shell is transparent — this makes *C. elegans* an ideal object of study. Already at the end of the 19th century, biologists showed an interest in nematodes as model organisms for their research (see for example Boveri, 1888). The worm's breakthrough came when Sydney Brenner decided to use *C. elegans* as a model for developmental studies and genetic experiments (Brenner, 1973, 1974; Sulston and Brenner, 1974).

The good observability of the worm led to the accumulation of data, both on molecular and cellular level. The whole cell lineage of the worm has been analysed and written down, first manually by Sulston and colleagues (1977,1983), and later by Schnabel *et al.* (1997) who used a 4D microscope to record cell positions and divisions semi-automatically.

Since the invention of RNA interference (RNAi) it was possible to specifically inhibit single genes or mRNA molecules without complex genetic modifications. In the RNAi technique, a mixture of double-stranded RNAs is injected into the cells and by an unknown mechanism interferes with translation (Fire *et al.*, 1998).

Because hermaphrodites can fertilize themselves, strains are easy to maintain and mutations and knockouts easier to obtain than in organisms which cannot reproduce without mating.

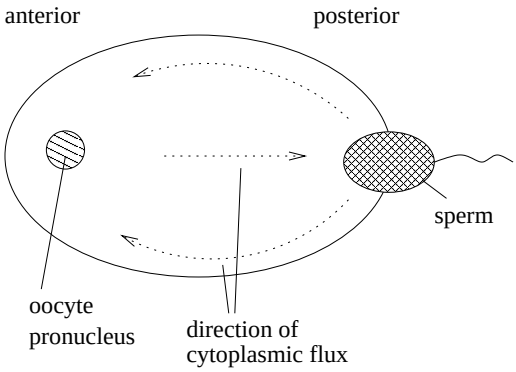

Figure 3.1.: The fertilization of the oocyte. The sperm enters the oocyte on one pole, which becomes the posterior. The oocyte pronucleus occupies the anterior region of the egg, opposite of the sperm entry point. Directions of cytoplasmic flux are indicated by arrows.

3.2. Steps in Worm Development

Development of the embryo of *Caenorhabditis elegans* can be divided into several distinct phases. The first is fertilization of the egg and the generation of polarity. After that, the fate of the blastomeres, or embryonic founder cells, is specified. This is done under control of mRNAs and proteins that were deposited in the egg by the mother. In the next step, the maternal proteins activate a group of embryonic transcription factors. Therefore, a transition from maternally to embryonically transcribed genes takes place . The embryonic genes thus activated are called organ identity genes. They activate organ-specific structural genes and thus initiate morphogenesis, which is the last step of embryonic development.

Blastomere fate specification will be treated in detail in chapter 8. The other steps are summarized here.

3.2.1. Establishment of Polarity

The oocyte of the worm is oval-shaped and its contents are symmetrically distributed (in contrast to *Drosophila*, for instance, where maternal mRNAs are asymmetrically distributed in the egg even before fertilisation). The oocyte moves along the gonad until it enters the spermathecum where it is met by the sperm. Because the oocyte leaves the gonad tube with one pole first, the sperm enters at that pole rather than somewhere in the middle of the oocyte (Goldstein and Hird, 1996). Sperm entry introduces an asymmetry into the oocyte. The pole with the sperm becomes the posterior end of the zygote and the worm. By mechanisms not fully understood, a cytoplasmic flux is initiated, where cortical cytoplasm flows to the anterior and in the centre of the oocyte it flows posteriorly (see figure 3.1). Those components of the cytoplasm that are to be located posterior are enriched in the posterior end of the egg, probably by fixing to the posterior cell cortex (Mello *et al.*, 1992). Transport along actin filaments or microtubules plays a role in the establishment of polarity in the egg, too. This is supported by experiments that

show that RNAi of actin-binding proteins abolishes polarity. The degradation of mislocated posterior molecules in the anterior part of the zygote also contributes to the asymmetry.

Once polarity has been established, it needs to be maintained and transduced during later cell divisions. Many genes have been identified that show an asymmetric distribution in the fertilized egg and in the early blastomeres, and whose deletion or RNAi leads to an abolishment of the asymmetric distribution of other asymmetrically localized proteins and thereby to a mis-specification of the fate of the embryonic founder cells and later blastomeres.

Complex interactions regulate the distribution of these asymmetrically localized proteins, and it is not yet fully understood what the primary actors in these interactions are. A very important complex is formed by PAR-3, PAR-6 and PKC-3 (Hung and Kemphues, 1999; Joberty *et al.*, 2000; Tabuse *et al.*, 1998). The MEX proteins seem to be mediators that transduce polarity from the fertilized egg to later stages (Crittenden *et al.*, 1997; Draper *et al.*, 1996; Guedes and Priess, 1996; Huang *et al.*, 2002; Mello *et al.*, 1992; Schubert *et al.*, 2000). They regulate the asymmetric distribution of the blastomere fate specification genes.

3.2.2. Determination of Organ Identity

After the fate of the blastomeres has been established, the next step is the determination of organ identity. In this step, all cells which are precursors of a specific organ — independent of their clonal origin — start to express organ identity genes. These genes do not convey any information about the cell or tissue type yet. They merely mark the cell as belonging to a specific organ. Also, they activate regulatory genes, which are organ-specific and which then specify the tissue type.

The process of organ identity determination is more difficult to investigate than blastomere fate specification, because more cells are involved and the cells are smaller. Therefore not so much information is available. The processes best studied include the specification of endoderm or intestine, hypodermis, and pharynx.

3.2.3. Morphogenesis

Once the cells know their fate, structural genes are activated to implement it. The cells change their shape. They loose the typical spherical shape of the early blastomeres and take on long (e.g. neurons) or flat (e.g. hypodermis) shapes as required. The cells also take up their function. The embryo starts to twist as a result of uncontrolled muscle activity, for instance. Because of this, semi-automated recording of the cell lineage or cell positions is no longer possible. Some morphogenic events, for example vulval development, take place only after hatching. Nonetheless, the vulva is one of the best studied organs in the worm.

Part II.

Materials and Methods

This part is concerned with software. It lists software that supports development or is required as an external part of the software I developed (*materials*, see chapter 4). The other chapters are concerned with the *methods*: software developed by me and my colleagues to address specific scientific questions. Chapter 5 describes the software framework which was already existent in the group and its extensions in which I took part actively. My own project is described in chapters 6 and 7. Chapter 6 concentrates on the “static” part, the network editor, while chapter 7 explains the “dynamic” simulation algorithms.

4. Software

4.1. Software Used to Support Development

This chapter describes the software that was used for the development of GENE-O-MATIC.

The programming language is JAVA, in the form of the Java 2 Standard Edition Software Development Kit (J2SE SDK) Version 1.4.2 from Sun Microsystems (Sun Microsystems, 2003a).

The IDE (integrated development environment, the software used to develop software) we used was IntelliJ Idea, version 1.3 (JetBrains, 2003).

To compile the source code, and to create binary distributions and JAVADOC documentation, the build tool Ant, version 1.6 (Apache, 2003) was used, which is itself written in JAVA. Compared to "make", another build tool, Ant is much easier to configure as the build file is in XML format. Compilation is much faster if done with Ant. Besides, Ant is fully integrated into the IntelliJ Idea development environment. The compiler we used is Jikes (IBM, 2002b), which is much faster than the javac compiler from Sun.

To manage the source code, track changes, and allow multiple users to work at the code concurrently, the version control system CVS (Concurrent Versions System) was used (Cederqvist, 1993).

4.2. JAVA Libraries Used Within CELL-O

The following additional software libraries were used in CELL-O. They are (or were, for JGL) available free of charge via the internet.

Java Help version 1.1.3 is used for the online help in GENE-O-MATIC.
URL: http://java.sun.com/products/javahelp/

Java 3D version 1.3.1 is used for the 3D display of the cells during simulation.
URL: http://java.sun.com/products/java-media/3D/

Java Media Framework (JMF) version 2.1.1 provides methods to create movies of cell simulations.
URL: http://java.sun.com/products/java-media/jmf/

JAMA (Java Matrix Package) version 1.0.1 contains mathematical methods for matrix calculation methods. It is applied for the calculation of Voronoi cell shapes and genetic network simulation.
URL: http://math.nist.gov/javanumerics/jama/

J-Sim version 0.2 – only one class is used that provides a stream of random numbers for the simulation.
URL: http://javasim.ncl.ac.uk/

JGL Libraries version 3.1.0 provides special lists and sets which are used in the event management in the simulation.
URL: http://www.recursionsw.com/products/jgl/jgl.asp

NCSA Portfolio version 1.2 is a collection of utility objects to use with Java 3D. It is used for the 3D cell simulation.
URL: http://www.ncsa.uiuc.edu/ srp/Java3D/portfolio/

Orbital version 1.1, a library that offers different methods for the treatment and analysis of logical formulas, is used for the interpretation of Boolean formulas in the genetic networks.
URL: http://functologic.com

VisAD is a JAVA component library for interactive and collaborative visualization and analysis of numerical data. Its methods to calculate the Delaunay triangulation for a set of points are used in CELL-O.
URL: http://www.ssec.wisc.edu/ billh/visad.html

Electric XML version 7.1 provides utilities to read and write XML files.
URL: http://www.themindelectric.com/exml/

5. The CELL-O Framework

The CELL-O framework was developed in the Division Medical and Biological Informatics at the German Cancer Research Center. Its initial application was the simulation of *C. elegans* embryos using a three-dimensional cell model (Gumbel *et al.*, 2000; Gumbel, 2001). The structure is shown in figure 5.1. The framework is organised in three layers, where the lowest contains the most general, application-independent code. The middle layer contains biology-related classes[1], and the top layer contains application-specific code. The framework provides basic classes for setting up a simulation and a hierarchy of classes which represent simulated objects. Also, there is a hierarchy of classes that represent containers for the simulated objects, the tissue or "World" that is simulated. Specialized classes for specific simulation purposes, for example for the worm, were developed.

The application that mainly uses the original part of the framework is called CELL-O-SIM. It uses the 3D cell model to simulate cell migration in the *C. elegans* embryo (Gumbel, 2001). Another application is ALES, "A Lineage Evaluation Software" that can be used to detect critical cell divisions where subtrees with different cell fate determinants are created (Braun, 2003; Braun *et al.*, 2003). GENE-O-MATIC, the application described in this thesis, is the third application in the framework.

The extensions to the framework listed in the next sections represent a joint initiative of all PhD and diploma students in our group because they are required as a basis for the different applications.

5.1. Cell Data Management

Ongoing development has led to the design of an interface[1] hierarchy for dealing with cell lineages and administrating cell data. These classes and interfaces are located in the package `dkfz.celldata`. Their relationship is portrayed in figure 5.2. The interfaces represent basic properties of cells. The most general interface is the `CellIdentifier`. It provides a method to get an ID value for the cell. This ID represents the cell's position in the cell lineage (see figure 5.3). It is an array of Boolean values (`true` and `false`). The first cell in the lineage has an empty ID. For each division, `false` or `true` is appended to the lineage ID of the mother for the left or right daughter, respectively. In the worm, the left daughter in the lineage tree represents the anterior, left, or dorsal daughter of a division, while the right cell in the tree corresponds to the posterior, right, or ventral daughter cell.

If there is more information available about a cell than just the ID, the interface `CellDescriptor` can be implemented. The `getType` method can be used to determine the cell's

[1] For an explanation of the terms "class" and "interface", see appendix B.

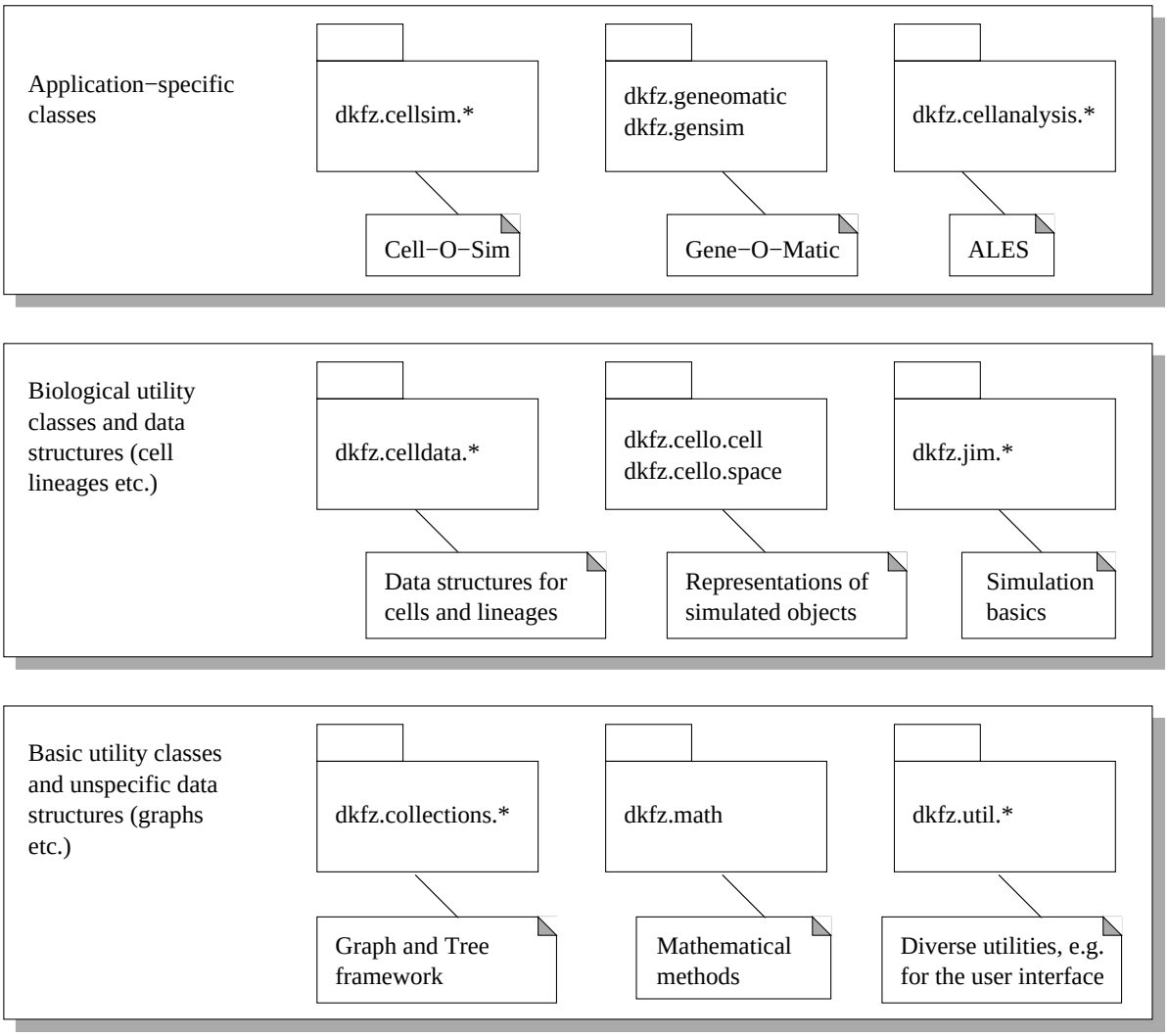

Figure 5.1.: Packages and structural organisation of the CELL-O framework. There are three layers (from bottom to top): basic, biological, and application layer.

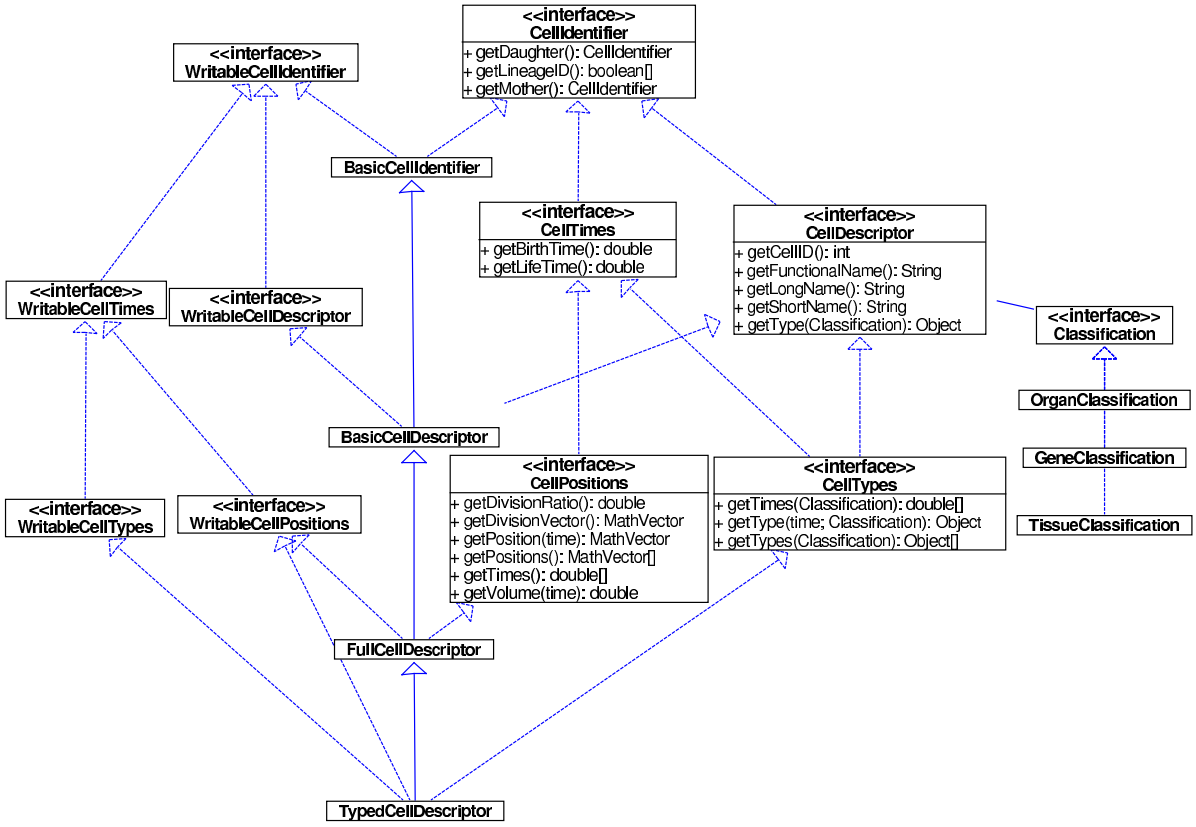

Figure 5.2.: UML class diagram of the classes for cell data management. The interfaces for accessing cell data are on the right, while on the left those for writing data into the objects are shown. The most important classes that implement the interfaces are in the middle (`Basic-` and `FullCellDescriptor` etc.). A short overview of the UML notation can be found in appendix B.

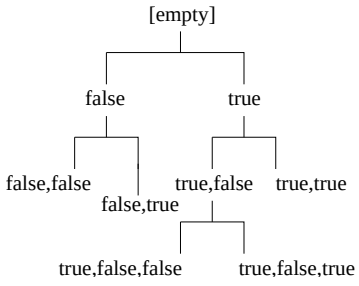

Figure 5.3.: Cell lineage IDs. `false` indicates the anterior, left, or dorsal daughter cell, and `true` indicates the posterior, right, or ventral daughter. The root cell has an empty lineage ID.

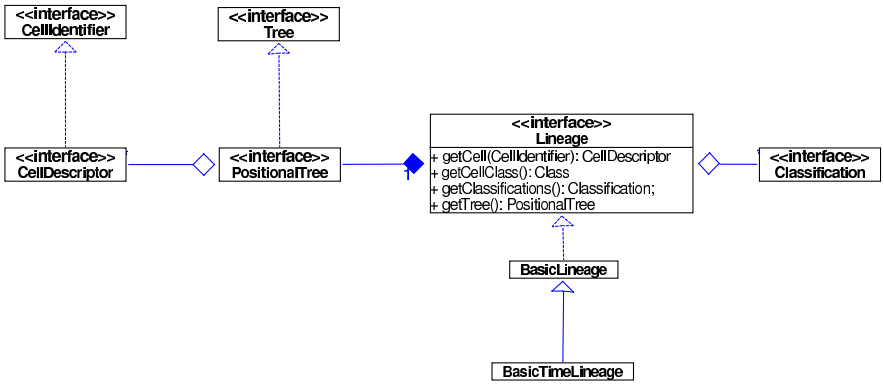

Figure 5.4.: UML class diagram of the data structure for cell lineages. The `Lineage` contains a `PositionalTree` which contains the cells (instances of `CellDescriptor`). Via the `getCell(CellIdentifier)` method one can access the `CellDescriptor` for any cell to get more information. The `BasicTimeLineage` is an extension of the `BasicLineage` that contains cells which implement the `CellTimes` interface. It does not provide additional methods.
An overview of the UML notation can be found in appendix B.

type according to a given `Classification`, for example a cell may have the type "muscle" in the `TissueClassification` and "intestine" in the `OrganClassification`. Not all methods must return useful information: if no long name is available, for example, the `getLongName` method may return the short name, too.

Implementing the `CellTimes`, `CellPositions`, and `CellTypes` interfaces indicates that further specific information about a cell is available (birth- and lifetime, positions, and changes in cell type, respectively). For each of these interfaces there exists a `Writable...` interface which specifies the methods required for storing data in the cell object.

The interfaces `CellTypes` and `WritableCellTypes` are of special importance to GENE-O-MATIC. In contrast to the normal `getType` method defined in the `CellDescriptor` interface, which assumes that each cell has a specific type which does not change during the cell's lifetime, they allow to deal with cell types that change over time, such as genetic states.

The information about all cells in an organism is stored in a `Lineage`. The `Lineage` is a wrapper for a `Tree` that contains the cells (`CellDescriptors`) and stores additional information, for example for which `Classifications` the cells have type information. The relationship between the `Lineage`, the `Tree`, and the cells within are shown in figure 5.4.

5.2. Cell Model

In the CELL-O framework cells have a spherical representation and thus are parameterized by their position and size. The position is a point in 3D specifying the centre of the sphere. The size represents the radius of the sphere.

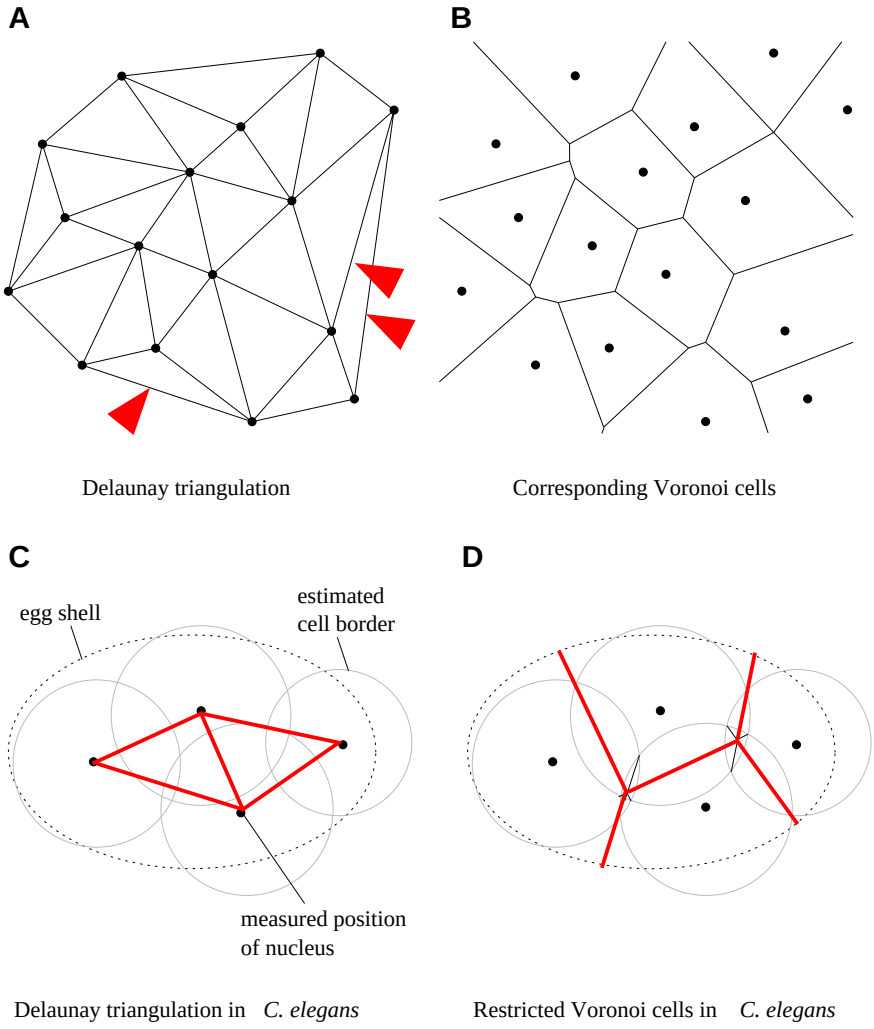

Figure 5.5.: Delaunay triangulation for calculation of cell-cell contacts. **A.** An arbitrary set of points and its Delaunay triangulation. The edges of the Delaunay triangles represent the connectivity between the points. Arrowheads indicate connections which would not be accepted in CELL-O because the cells are too distant. **B.** The corresponding Voronoi cells. The cell borders are perpendicular bisectors of the Delaunay edges. **C.** Delaunay triangulation (red) in the 2D projection of the 4-cell stage *C. elegans* embryo. The dotted line represents the egg shell, and the approximated cell shapes are grey circles. The edges show which cells are connected. To prevent unrealistic connections (see arrowheads in A), connections are only accepted if the distance between the two nuclei is not much larger than the sum of the radii of the spheres. **D.** Voronoi cells in *C. elegans*. Cell size is restricted by the egg shell. One can see how Voronoi cell borders (red) can also be obtained by connecting the intersections of the circles (grey) by which the cells are approximated. A and B are adapted from Sedgewick (1992, p. 466).

To calculate cell-cell contacts, the cell shapes are approximated as Voronoi cells (see figure 5.5 B and D). Voronoi cells are areas around so-called Voronoi points, which are the nuclei positions here. These areas are defined such that each point within a Voronoi cell is closer to its own cell's Voronoi point than to the Voronoi point of any other cell. The border between the two Voronoi cells contains all points that are equally distant from the Voronoi points of both cells. Another application of Voronoi cells stems from social studies: all people who live in one Voronoi cell go to the nearest supermarket, i.e. the cell's Voronoi point. Because for the genetic network simulation the size of the contact area or the shape of the cells is irrelevant, and only the question whether there is any contact at all is of interest, the related method of Delaunay triangulation is used in the calculation (see figure 5.5 A and C). The Delaunay triangulation of a set of points, for example cell centres, is a collection of edges satisfying an "empty circle" property: for each edge a circle containing the edge's endpoints but not containing any other points can be found. The spheres which represent the cells are the three-dimensional equivalent of those circles. Cells not connected by a Delaunay edge do not contact each other. The borders of the Voronoi cells are the perpendicular bisectors of the Delaunay edges, and only those cells that share a Voronoi cell border are connected by a Delaunay edge.

For the display, spherical cells are used, because their rendering is very fast compared to more complex shapes. However, if a more exact representation is desired, Voronoi cell shapes can be displayed.

6. The GENE-O-MATIC Software

This chapter explains the implementation details of the GENE-O-MATIC software. I tried to keep in mind that many readers of this thesis are no computer scientists, or at least no experts in JAVA and object-oriented programming. Therefore the concepts have been simplified by leaving out unnecessary detail, and the chapter contains many examples and figures. For those who want to use the software to investigate new networks there is a user manual in appendix A.

GENE-O-MATIC was developed as part of the CELL-O framework. It is one of the three applications that use the framework at the moment. Some classes and methods used by GENE-O-MATIC are located within the core framework, as in the future other applications may need them. Also, GENE-O-MATIC makes uses of many functions offered by the framework.

6.1. Networks and Cell Lineages

6.1.1. Graphs as Data Structures

From a computer science point of view, the representations of genetic networks and cell lineages are related. Network structures are called *graphs*, and cell lineages are *trees*. A tree is a special type of graph. First some terms will be defined (adapted from Sedgewick, 1992).

Definition 1 A *graph* is a set of *vertices* (or nodes) and *edges* (see figure 6.1). Vertices are simple objects that can have a name and other properties (BICs or cells, for example). An edge is a connection between two vertices.

Definition 2 A *path* is a list of subsequent vertices connected by edges. A *cycle* is a path on which no vertex occurs twice, except that the first vertex equals the last.

Definition 3 A *connected graph* is a graph where a path exists between any two of the vertices.

Definition 4 A *directed graph* is a graph where each edge has a direction. Therefore, two edges may exist between two vertices A and B: one from A to B, and one from B to A. Genetic networks are directed graphs.

Definition 5 A connected graph without cycles is called a *tree*. Usually, trees have directed edges, from top to bottom. Trees can be used to represent cell lineages.

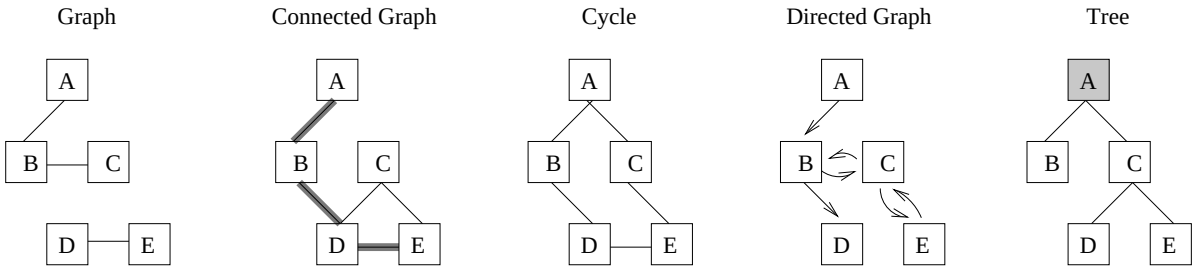

Figure 6.1.: Different types of graphs. The squares with the letters are the vertices, the lines represent edges. The grey line in the connected graph represents a path from vertex A to vertex E. The grey vertex in the tree represents the root vertex.

Graphs and trees are of enormous importance in computer science, as they can be used to store data and to solve different kinds of problems efficiently. Therefore many software libraries offer methods and algorithms to deal with graphs and trees. However, in this project, the graph is to be used as a data structure only, not as a basis for complex algorithms. The complexity of most graph libraries is thus not required. Therefore a simpler structure was developed that fulfils all requirements of our project but does not require the software developer to busy himself with the implementation of numerous complex methods which will never be used.

This graph framework consists of three main interfaces: `Graph`, `Vertex`, and `Edge`. They are located in the `dkfz.collections.graph` package. `Graph` defines 12 methods, `Vertex` two, and `Edge` four (18 in total). In contrast, the `DirectedNet` class of IBM's Graph Foundation Classes (IBM, 2002a) contains over 200 methods. OPENJGRAPH, an open source graph library (Salvo, 2002), still specifies 43 methods in its DirectedGraph interface. These packages offer a lot of convenience methods that make the life of programmers who write algorithms for graphs a lot easier, but which are not necessary for people who simply want to use a graph as a data structure for a network. In addition, the possibility to associate objects with the edges is not available in either of the two frameworks. The reason for this is probably that most computer science problems are not interested in the edges themselves, but only in the fact whether there is an edge between two vertices. At most, they allow to define a weight for the edge, but only as a real number.

For this application, however, it is useful, if not essential, to associate more complex objects with edges and vertices alike, for example to store literature references that describe the experiments that led to the discovery of the gene and its interactions. The components of the graph framework and their relationships are portrayed in figure 6.2.

6.1.2. Graph Methods

The methods defined by the `Graph` interface are listed in the following paragraphs. The syntax is as follows:

`methodName(Parameters): returnValue`

The first word is the name of the method, followed by method parameters in brackets. After the colon, the return type is given, in case the method returns a value.

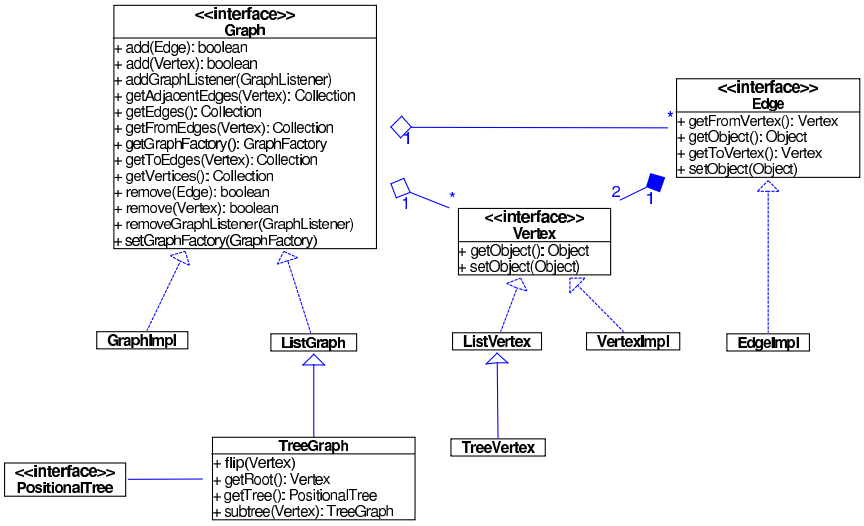

Figure 6.2.: UML class diagram depicting the relationships of the interfaces and classes in the graph framework. An overview of the UML notation can be found in appendix B.

Modifying Methods The following methods are required to modify the structure of the graph, i.e. to add or remove elements. If one tries to add an element which is already part of the graph, or tries to remove one which does not exist, the methods do nothing and return `false` to indicate that the graph structure was not modified. If the modification was successful, the methods return `true`.

`add(Edge): boolean` adds an edge to the graph, and with it the adjacent vertices (depending on the implementation, see below).

`add(Vertex): boolean` adds a vertex to the graph.

`remove(Vertex): boolean` removes a vertex from the graph, and with it all adjacent edges.

`remove(Edge): boolean` removes an edge from the graph.

Information Methods These methods allow the user to get information about the graph and its structure. Each of these methods return a Collection of edges or vertices.

`getFromEdges(Vertex): Collection` get a collection of all edges that start at the given vertex.

`getToEdges(Vertex): Collection` get all edges that point towards a given vertex.

`getAdjacentEdges(Vertex): Collection` get all edges which are adjacent to a

given vertex, i.e. start at that vertex or point towards it. The result equals the union of the results from the `getFromEdges` and `getToEdges` methods.

`getEdges(): Collection` get all edges in the graph.

`getVertices(): Collection` get all vertices in the graph.

Other Methods These methods simplify working with the graph, they are not required for the data structure itself.

`addGraphListener(GraphListener)` and
`removeGraphListener(GraphListener)` can be used to add a listener to the graph. The graph will inform the listener of any changes that take place, e.g. if a new vertex is added to the graph. By means of a listener, the graph may be monitored easily.

`getGraphFactory(): GraphFactory` gives access to a GraphFactory that can be used to create new edges and vertices. Some graphs accept only specific types of edge or vertex objects, and the factory lets the user create these required types conveniently.

Tree-specific Methods These methods are not part of the `Graph` interface, but are implemented in the `TreeGraph` class which is an implementation of `Graph` that behaves like a tree. Before genetic networks were invented in GENE-O-MATIC, CELL-O already offered methods and data structures to deal with cell lineage trees (the "tree framework"). It became apparent that the new graph framework described here was more flexible than the tree framework. Therefore some parts of CELL-O were reimplemented to use the graph framework instead of the tree framework. Nonetheless, the tree framework offers more advanced methods to deal specifically with cell lineage trees, therefore compatibility between the tree and graph systems was essential. To support this, the graph system provides some methods which are similar to methods from the tree framework.

`getRoot(): Vertex` allows access to the root of a tree, i.e. the only vertex in the tree that has no edge pointing towards it and from which you can reach all other vertices by following the directed edges.
`flip(Vertex)` swaps two children in a binary tree. In the tree framework's trees, the order of the adjacent edges of a vertex is important and can be changed deliberately.

`subTree(Vertex): TreeGraph` creates a subtree of this tree-like graph with the given vertex as the root. A similar method is available in the `PositionalTree`.

`getTree(): PositionalTree` converts this tree-like graph into a `PositionalTree`. The `PositionalTree` is a data type defined in the tree framework, it does not implement the `Graph` interface.

6.1.3. Graph Implementations

The two most common graph implementations are adjacency lists and adjacency matrices. Figure 6.3 illustrates the different implementations for an example graph (figure 6.3 A).

Adjacency List An adjacency list is a list that contains all vertices in the graph, and each vertex contains a list of adjacent edges. The edges contain a reference to the next adjacent vertex (figure 6.3 B). Adjacency lists are useful for sparse graphs with few edges. A drawback (or feature, depending on the point of view, see below) of the adjacency list is that, if a vertex is added which is already connected by an edge in another graph (graph 2 in figure 6.4), this edge, and the vertex at the edge's other end, and all vertices connected with them, are now also accessible within the graph if one follows the edges. If the `add` method is not implemented correctly, this may lead to an inconsistent state of the graph: Vertex E is not part of the list of vertices in graph 1, but can be reached via the edges. This can be prevented if the `add` method moves through the whole second graph, finds all connected vertices and edges and adds them, too. However, if the second graph contains cycles, it must be made sure that no vertex is added twice, and no edge is followed twice in moving through the graph, which makes the implementation of an adjacency list very complex. This complexity is the drawback of the adjacency list, the feature is that it is afterwards easy to merge two graphs, but this was not required in GENE-O-MATIC.

Adjacency Matrix An adjacency matrix (figure 6.3 C) is a square matrix where each field in the matrix contains the edge between the vertex in the respective column and the vertex in the respective row. This implementation is useful for graphs with a lot of edges, so-called dense graphs. One problem with the adjacency matrix is that it is rather complicated to add or remove vertices, as this requires a modification of the size of the matrix, which is a computationally expensive operation.

GENE-O-MATIC's Graph Implementation To avoid the problems with the consistency of the adjacency list and with resizing of the adjacency matrix, a special graph similar to an adjacency list was implemented (the `GraphImpl`, figure 6.3 D). It contains a list of all vertices, but the vertices do not know about the edges connected to them. So if a vertex is added, it does not matter if in another graph, an edge is connected to it, because in this graph nobody will know about it. In addition, the graph contains a list of all edges which know about their adjacent vertices. Thus, the information about the connectivity has been separated from the vertices.

Simplified Adjacency List The disadvantage of GENE-O-MATIC's graph implementation is that it is very slow to find out which edges are connected to a given vertex, as all edges in the graph's edge-list have to be examined to find out. For handling the cell lineages, this is a serious problem as the number of vertices (cells) may be quite large (959 living cells in the adult worm, plus the cells which have divided during development) compared to a genetic network (in the worm database produced during this thesis, there are less than 200 genes).

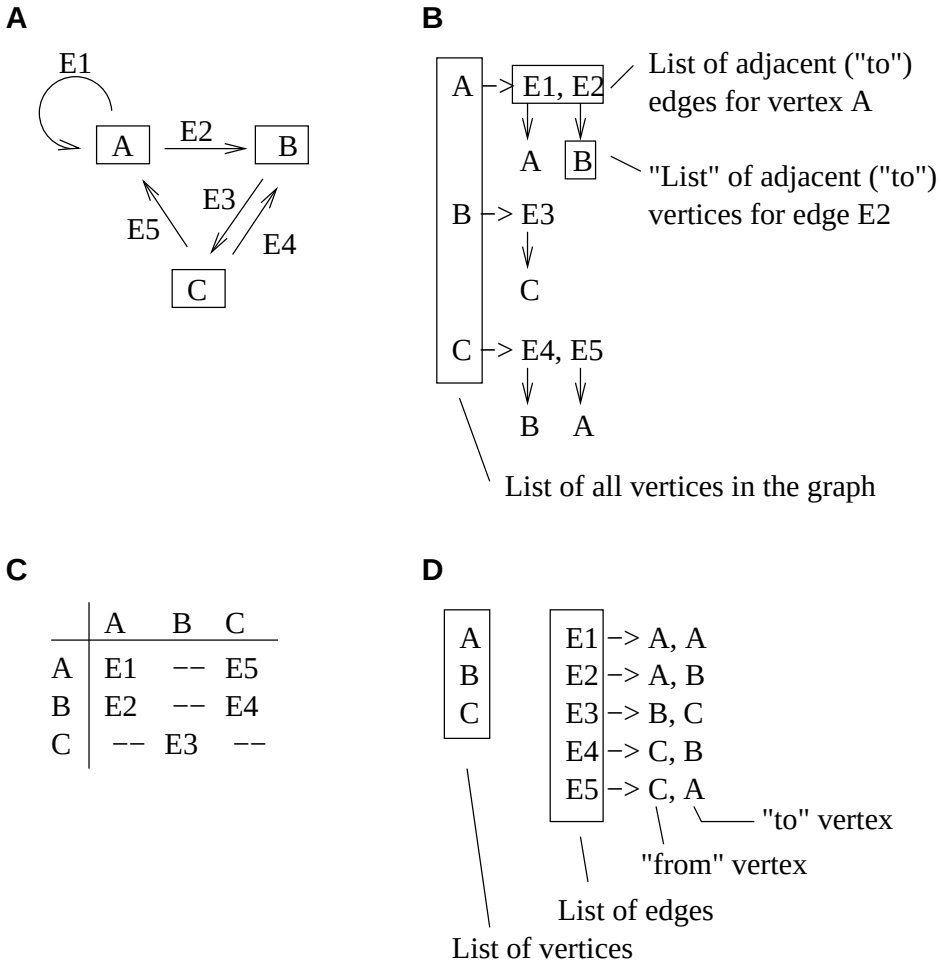

Figure 6.3.: Graph implementations. **A.** An example graph. **B.** The adjacency list representation for this graph. **C.** The adjacency matrix for this graph. **D.** The implementation used in GENE-O-MATIC.

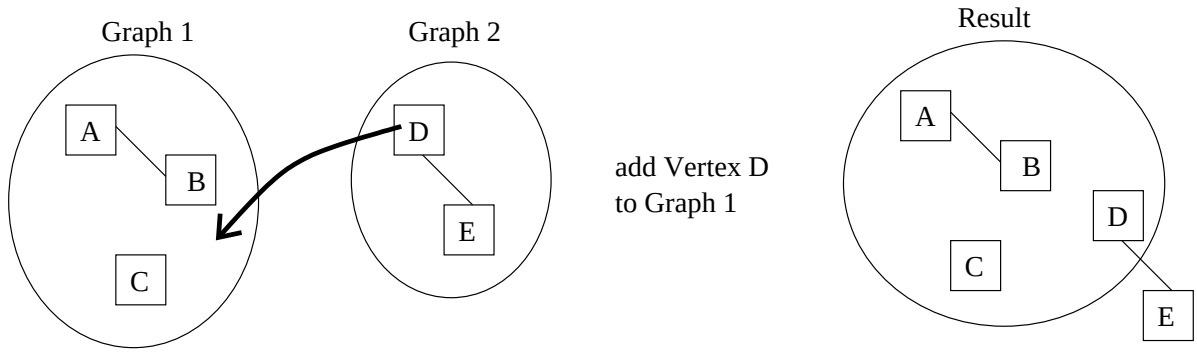

Figure 6.4.: Adjacency list consistency problem. If vertex D is added to graph 1, the edges and vertices connected to it become part of graph 1, too, in the sense that they can be reached by following the edges. But they are not part of the list of all edges if they are not explicitly added to it.

Therefore a faster implementation was required, and thus an adjacency list was implemented (`ListGraph`), with a special extension for trees (`TreeGraph`). To avoid the consistency problem, in this implementation it is not allowed to add vertices that have adjacent edges. The `TreeGraph` extends the `ListGraph` only so far as required to conserve a true tree structure, for example, to avoid circles or multiple root vertices (it overrides the `add` and `remove` methods to get control over how the structure of the graph is modified).

6.2. Graphical User Interface

The graphical user interface (GUI) of GENE-O-MATIC is shown in figure 6.5. It consists of three parts: the database area, which contains a collection of "reference" genes, the network area where the user can enter and modify the genetic network, and the cell panel for entering information about the simulated cells. This section highlights some specific issues of the implementation. A description of the different functions from the user's point of view is given in the user manual in appendix A.

6.2.1. Model – View – Controller

Software developers often make use of design patterns for common programming tasks. Thus, they do not have to develop the software architecture from scratch, but can use mechanisms other people already applied to solve a specific problem. One of these patterns is called Model-View-Controller. It is used to program graphical user interfaces (views) that allow a user to interact with a dataset (model) by means of a controller that detects user actions and relays them to the view or the model (see figure 6.6).

One advantage of this pattern is that it keeps the data independent from the way it is presented. Different views could require additional visual information, for example for a genetic network, the coordinates of the genes on the screen, or the colour of the edges. But if the genes

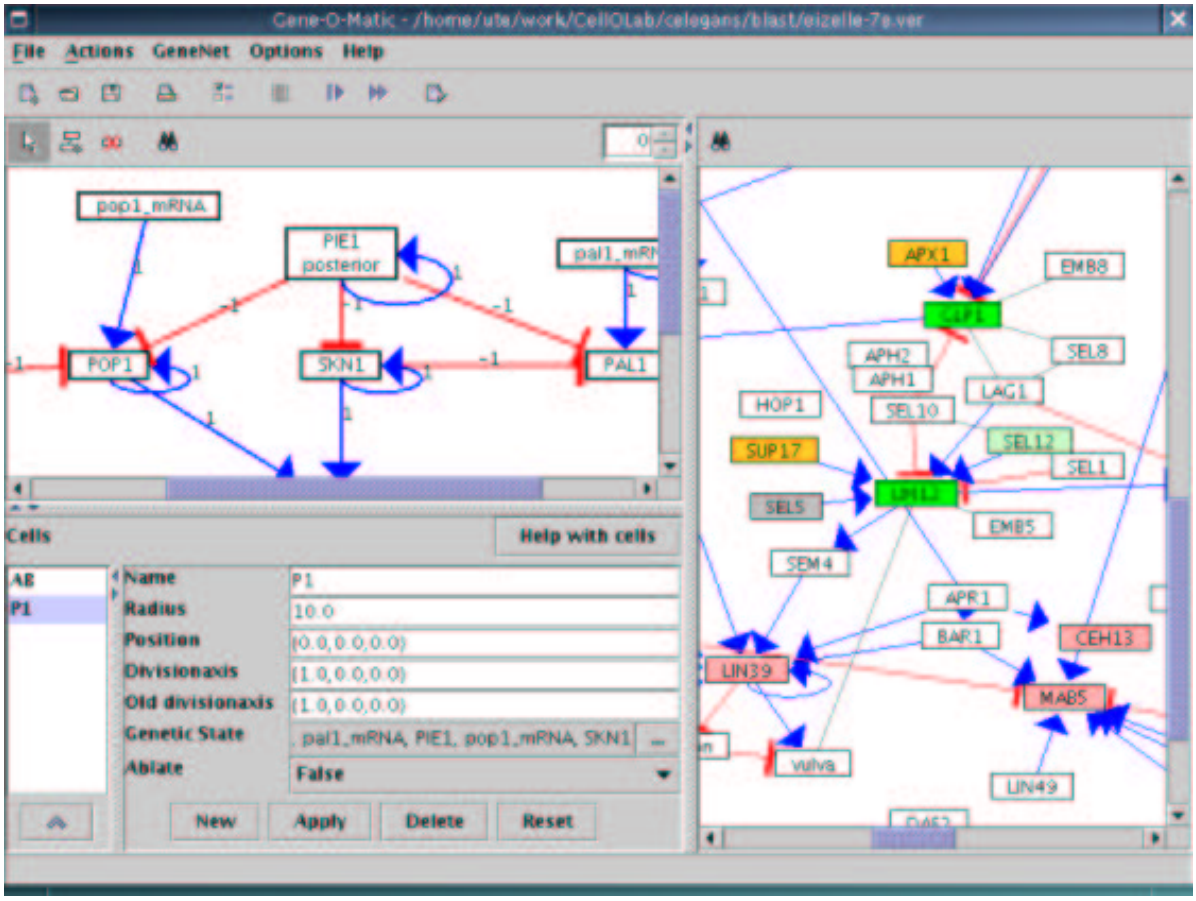

Figure 6.5.: Graphical user interface of GENE-O-MATIC. On the right is the database area, on the left the network area (top) and the panel for the cells (bottom). The toolbars allow quick access to most GENE-O-MATIC functions.

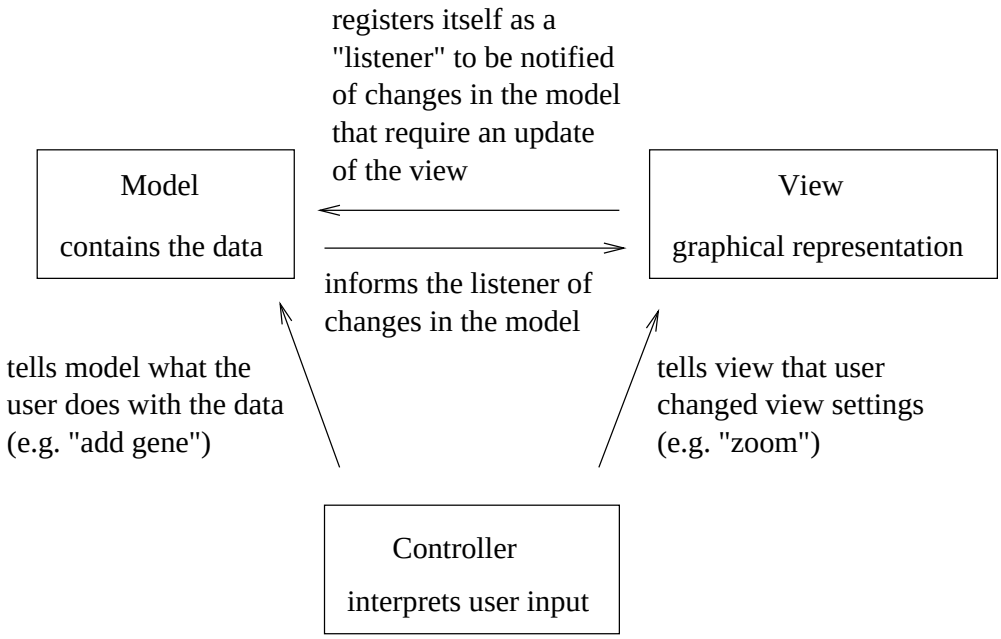

Figure 6.6.: The Model-View-Controller design pattern.

are presented in a list, this data would not be required, and instead, some information about the ordering of the genes in the list may be of interest. Each view must of course be capable of presenting the data adequately if additional visual information is not available, for instance by using default or random values.
Another advantage is that the way the user can interact with the data or the view is flexible and can be changed easily by using a different controller. Neither the data nor the view needs to know anything about the way user interactions are handled.

Using this pattern, the software developer can concentrate on the best way to represent the data internally. Then he can develop one or more graphical representations for the data, and finally the user interaction schemes can be developed.

For GENE-O-MATIC, this pattern is used for the display of the genetic network. The model consists of implementations of `Graph`, `Edge`, `Vertex`, and the objects inside the edges and vertices. View and Controller are described in the next paragraphs.

The View The class `dkfz.collections.graph.view.ViewGraph` is the most important component of the view. It extends `javax.swing.JPanel` and can thus be used within any type of graphical JAVA application. Via the `setModel` method the view is told which model, i.e. which dataset to display. For each vertex and edge in the graph, a view object is created: `ViewVertex` and `ViewEdge`, respectively. A `ViewVertex` contains information about the vertex's position in the panel. Both `ViewVertex` and `ViewEdge` extend the abstract class `DelegateShape` which implements the interface `java.awt.Shape`. The `DelegateShape` — as the name implies — delegates all method calls to its internal `Shape` object, which is set at the time of painting (see below). The relationship between the differ-

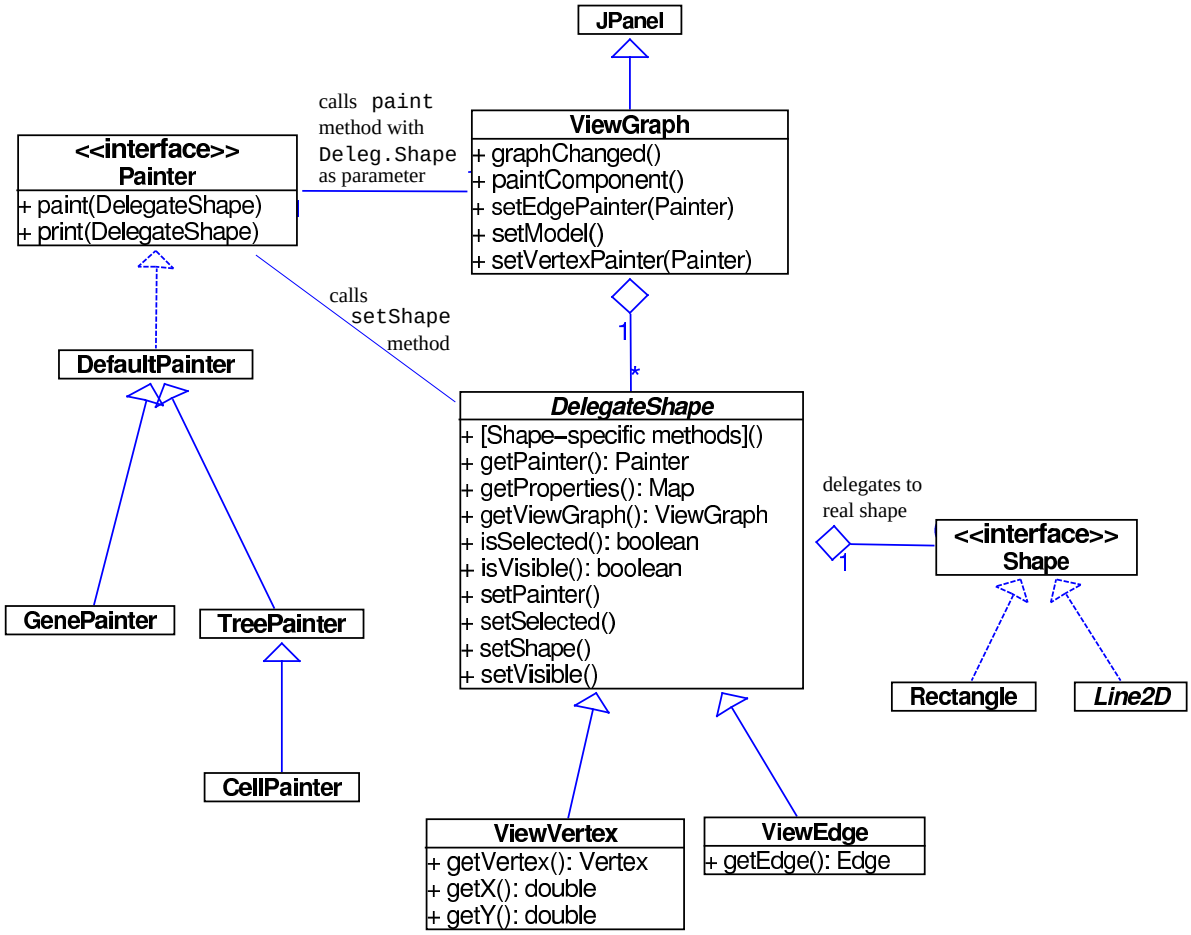

Figure 6.7.: The Graph: UML class diagram for view and painting. An overview of the UML notation can be found in appendix B.

ent view classes is shown in figure 6.7. Details of the painting procedure will be explained in section 6.2.2.b.

The Controller The controller allows the user to modify the model, i.e. to add or remove edges and vertices, and to manipulate the view, for example to change the size of the display, the colours, etc. For this it presents the user with popup menus that contain all possible actions. Some operations affect single objects only, while others influence the whole graph. Therefore two different popup menus were implemented, one that appears if the user right-clicks on an object (the `ObjectPopup`), and one that appears if the user right-clicks on the background of the graph display (the `GraphPopup`). In addition, the controller has two states: one for normal operation, where the view can be modified (e.g. move vertices), and one where the model can be modified (e.g. add new vertices and edges).

In GENE-O-MATIC, the controller is implemented by the class `GraphController` in the package `dkfz.collections.graph.view` and the classes that extend it. There are

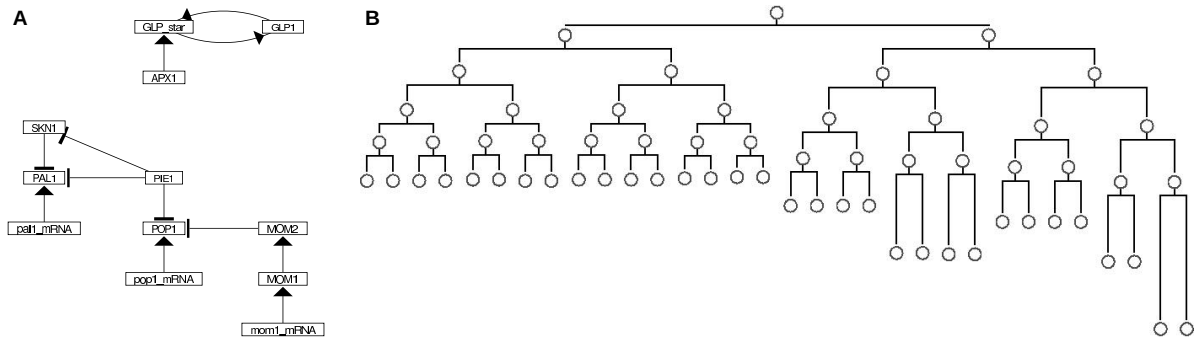

Figure 6.8.: Graph layout and painting. **A.** network layed out with the `RandomNetLayouter` and painted with the `StraightGenePainter`. The two arched lines between GLP1 and GLP_star are painted by the `GenePainter`. **B.** Cell lineage tree layed out with the `CellTimesLayouter` and painted with the `CellPainter`. Both images were converted to black-and-white for simplicity.

several controllers available which differ in the possibilities they offer to interact with the model and the view. For the database, for example, there is a controller that does not offer possibilities to add or remove BICs or interactions.

6.2.2. Layout and Painting

Laying out a graph is the process of determining the coordinates of the vertices on the drawing area. Painting is the process of drawing the vertices and edges. Together, layout and painting determine the visual appearance of a graph. It is useful to distinguish these two functions, as one may want to keep the layout but select a different painting style, or vice versa.

6.2.2.a. Laying out the graph

There are three different approaches to layout a graph:

- manual: the user determines the positions of the vertices.
- triggered: the user can trigger an automatic calculation of the positions and then manually improve the arrangement.
- automatic: the computer calculates the positions and the user cannot influence this process or change positions later. If new vertices are added, they are automatically assigned a position.

For the genetic network, it is best to let the user determine the position of the BICs manually. He can arrange them according to criteria not available to — or not usable by — the software, for example the biological processes they are involved in. And besides, an automatic layout may change the network appearance completely after re-loading the network, or after some

BICs or interactions have been added. This would confuse the user. Therefore no fully automated layout mechanism has been implemented. However, it is possible to trigger a one-time rearrangement of the BICs manually, for example to get a good initial layout from which to start for fine-tuned BIC positions. This so-called `RandomNetLayouter` starts with a random vertex and determines a random position and then arranges all connected vertices on a grid around it. The next unconnected vertex gets a random position again. This leads to a relatively clear layout that gives a good overview over connected parts of the network (cf. figure 6.8 A). If the network is saved to a file, the positions of the BICs are also stored and of course re-stored upon loading.

In contrast, for the cell lineage tree an automatic layout mechanism is adequate, because the tree structure makes it possible to calculate the positions of the cells in a deterministic manner. Besides, manually laying out 959 cells would be nearly impossible. Depending on the requirements, different layouters were implemented: `TreeLayouter` for a quick and simple display as a tree, where all cells are equidistant, and `CellTimesLayouter`, where the y axis represents time, i.e. cells born later appear lower in the diagram (cf. figure 6.8 B).

6.2.2.b. Painting

For painting a vertex or an edge, two types of information are used. First, the information in the object associated with the vertex or edge, for example the BIC or the cell. Second, the graphical information about the vertex or edge, for instance the coordinates of the vertex, line width, or text size.

Painting is done by implementations of the `Painter` interface. There is a base class for painters, called `DefaultPainter`, that can be extended. It provides methods to change the font, the line width, and the stroke (pattern of the line). It is necessary, or at least useful, to have different kinds of painters because different types of graphs require different representations. A genetic network, for example, is quite well represented by simple rectangular genes and straight lines for the interactions. A cell lineage tree — or any other tree — should have only horizontal and vertical lines, and the cells could be represented as circles, for example (see figure 6.8).

While usually one painter is used for all vertices and edges in a graph, it is possible to use different painters for each single vertex and edge. One example for this is the `GenePainter`. The normal `StraightGenePainter` draws straight lines for the interactions between the BICs in the network. If there are two interactions between two BICs (they influence each other), straight lines are not useful as both lines would be superimposed. Therefore, the `GenePainter` draws arched lines (see figure 6.8 A, top). Also, if the user wants to change the colour or font for a single interaction or BIC, this is done by creating a new instance of the painter which can be configured independently of the other painter instances.

Painting Procedure In its `paintComponent` method, the `ViewGraph` calls for each of the `ViewEdges` and `ViewVertices` the `paint` method of the `Painter`. The edges are painted first, and then the vertices, as the boxes of the vertices then hide the ends of the edge lines. The `Painter` sets the `Shape` for the `ViewVertices` and `ViewEdges`,

because obviously the shape of an object depends on the way it is painted. Vertices often get an instance of `java.awt.Rectangle` as their shape, and edges get instances of `java.awt.geom.Line2D`, for example.

6.2.3. Generic Editors for Objects – JAVA Beans Customizer

In GENE-O-MATIC, editors are required for many different types of objects, such as genes, interactions, and cells. It would be a very tedious task to implement editors for each of these object types manually, as the basic principle is very similar: each property of the object has a name and needs some kind of editor component, like a text box. JAVA offers a concept for simple creation of editors for different types of objects. It is based on the assumption that each object has a set of properties that the user may want to modify. These properties must be accessible via `set` and `get` methods. Such objects are then called *beans* (Sun Microsystems, 2003c). For simple properties, such as text or numbers, default editor components are defined (text boxes where the user can enter the value). For more complex properties, such as the genetic state of a gene, the programmer must write the property editors. To tell JAVA which property editor to use for the complex properties, a `BeanInfo` class must be implemented for the bean. The `BeanInfo` contains for each property the name of the editor class. Alternatively, a `Customizer` for the whole bean may be specified. The `BeanInfo` also contains information about which properties shall be shown to the user at all, as there may be some properties which are to be changed by the software only, and not by the user. And it can offer descriptions and human-readable names in case the method names in the program are cryptic.

The ORBITAL library for JAVA (Platzer, 2003) contains the `DefaultCustomizer` that creates a `Customizer` for a bean based on the `BeanInfo`, or, if that is absent, based on default property editor components. This was extended to create `Customizers` that respect the order of the properties as specified in the `BeanInfo` (the default behaviour of JAVA is to ignore the order, as the information about properties may be integrated from different sources and thus the order is ambiguous). The resulting class is the `OrderedCustomizer`. Its disadvantage is that it does not consider information from sources other than the `BeanInfo` for the object to edit. Figure 6.9 shows an example of a generated `Customizer`.

6.3. Data Storage

The problem with storing the data is similar to that with the editors for the objects. To avoid having to write specific code for storing each type of object, a generic mechanism was implemented. Using reflection (Sun Microsystems, 2003b) it is possible to find out the properties and their values for any given object at runtime. With the same mechanism it is also possible to set property values if the name of the property is known.

The classes for writing and reading the objects from a file have two main methods, called `write(Object)` and `read(): Object`. Depending on the specific implementation, the information what type of object is to be read is either read from the data source (file) itself, or has to be told to the reading class beforehand.

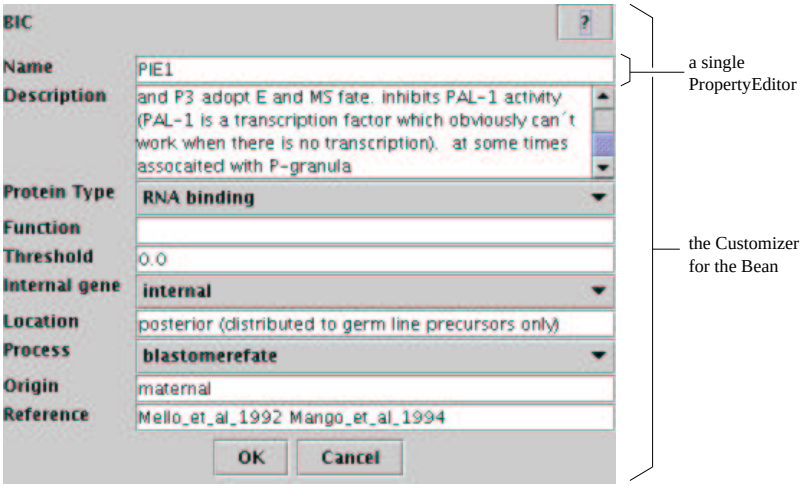

Figure 6.9.: Bean customizer and property editor. The figure shows the customizer for objects of the class `BIC`. It is generated at runtime by the `OrderedCustomizer` class. Each property has its own `PropertyEditor`, the one for the property *name* is indicated. It is a normal text field, which is the default for properties with textual content (`Strings`). The PropertyEditor for the *description* property is a text area, which is non-standard and has to be specified in the BIC's BeanInfo.

To write an object (for example a BIC), the `write(Object)` method is called with the object as a parameter. The `write` method then determines what type of object to write by calling the object's `getClass(): Class` method. The class provides information about the methods (via the `getMethods(): Method[]` method). For each method in the array, it is then determined whether it is useful, i.e. if it is the "get" method for a property:

- The name of the method must start with "get", as defined in the convention for JAVA beans. For properties of type `boolean`, the name must start with "is", as in "isKnocked-Out".
- There must be a corresponding "set" method, because it does not make sense to store information that cannot be restored.
- The method must not expect parameters, and must return a value (must not have the return type `void`).
- The object returned must be of a useful type, this is explained later.

Once the method is analysed as being the "get" method of a property, the value of the property is determined by invoking the method via reflection:

```
Object propertyValue = method.invoke(object, null);
```

This invokes the method for the given object and returns the value the method would return if you called it normally. `null` as the second parameter indicates that the method does not have any parameters, as is usual for "get" methods.

The property value can then be written to the file. Two different file formats are available: one is a simple text file, and the other is an XML file.

6.3.1. Text File

The text file contains one line for each property, and two distinct objects are separated by an empty line. Property values may not contain newline characters. An example text file is shown in listing 6.1. The disadvantage is that one file can only contain objects of a single type, and before loading, the program has to know what type of object it is, because that information is not stored in the file. Also, if objects within objects are stored, the file becomes very unclear. Storing objects with a `Collection` as a property is not possible.

Listing 6.1 Text file example. A file containing two objects of type BIC is shown. As the file does not contain any information about the type of object being stored (`BIC`), the software must know this before trying to load a file again.

```
name = gene1
description = a test BIC

name = secondGene
description = another gene in the network
```

For loading, the file is read line by line. New objects are created by using their classes' `newInstance()` method. Property names are determined (the text before the equals sign), and values are converted from the text in the file to objects of the specified type (most simple types in JAVA have methods to convert text to the type). The values are set for the new object by calling the corresponding "set" method via reflection.

Useful types of properties for objects stored in text files are `int`, `double`, `boolean`, and the other so-called primitive types which are not themselves objects, and the class `String` (text). Also, arrays of these types can be written. If specified by the programmer who wrote the object, other objects may also be stored as properties. This is useful only if these objects contain properties of useful types themselves.

6.3.2. XML File

XML offers the possibility to group information hierarchically. This allows to store complex objects that contain other objects. In JAVA, there are already classes that can store objects in an XML file and re-load them (`java.beans.XMLEncoder` and `XMLDecoder`), but they can handle only one object per file. For storing complex structures such as graphs, this is not sufficient because the graph contains different types of objects, such as the objects inside the edges and vertices and the corresponding visual information. Therefore, a new class was written to handle XML files: the `XMLManager`. The file format was adopted from that generated by the `XMLEncoder`. An example is shown in listing 6.2.

Listing 6.2 XML file example. An XML file that contains a `LinkedList` (a special type of `Collection`) with two `BIC` objects.

```
<object class="java.util.LinkedList">
  <void method="add">
    <object class="dkfz.cello.genet.BIC">
      <void property="name">
         <string>gene1</string>
      </void>
      <void property="description">
         <string>a test BIC</string>
      </void>
    </object>
  </void>
  <void method="add">
    <object class="dkfz.cello.genet.BIC">
      <void property="name">
         <string>secondGene</string>
      </void>
      <void property="description">
         <string>another gene in the network</string>
      </void>
    </object>
  </void>
</object>
```

The difference to the `XMLEncoder` is that more than one object, and additional tags, can be written to a file, and can also be read again. Of course, this requires that at the time of reading, the program knows what to expect, i.e. how to interpret the additional tags read from the file.

The XML files are parsed using a `SAXParser` which is part of the JAVA library. It is really not a parser, but only a scanner, i.e. it scans through the file and announces all tokens, for example start tags, character data, and end tags. The `XMLManager` is the parser which then interprets the tokens semantically and creates the objects in a process analogous to that for the text file.

In addition to the property types that are accepted by the text file writer, in the XML file also properties of type `Collection` and `Map` can be stored. This was achieved by explicitly defining what to do if the object to be stored is one of these types. The default behaviour, to determine or set their properties by “get” and “set” method calls does not work because neither `Collection` nor `Map` are beans, i.e. have properties accessible via “get” and “set” methods. In the future, the `XMLManager` can be extended further to implement handling of other types of objects, such as those which do not have “set” methods but get their property values via the constructor (“read-only” properties).

7. Simulation Algorithms

GENE-O-MATIC simulates genetic networks in a multicellular context. The genetic states are Boolean variables, it is therefore a qualitative simulation method. The time is also discrete, state changes occur in defined time steps. The next states of the BICs in a cell depend on the states of the BICs in this cell and in neighbouring cells. Cell positions are continuous variables, or as good an approximation as the precision of the computer allows.

In this chapter, the simulation algorithms used by GENE-O-MATIC are described. For an overview of other simulation techniques, refer to chapter 2.

7.1. Simulating the Genetic Network

A biological regulatory network consists of different types of elements, namely genes, mRNAs, and proteins. Between these, interactions are used to describe the way they influence or regulate each other. Typically, there are three levels of regulation: transcription (DNA), splicing and translation (mRNA), and protein activity and degradation (protein). In our model, the three levels of regulation are treated equally. Genes, mRNAs, and proteins are jointly called biological information carriers (BICs). The three levels may be modelled explicitly by using three BICs: one BIC for the gene, one for the mRNA, and one for the protein, or implicitly by using one single BIC that represents gene, mRNA, and protein together. Even complex regulatory mechanisms can be modelled. Alternative splicing of mRNA can be modelled by using several BICs for different splice variants, for instance. The same is possible for different states of the same protein (phosphorylation at different residues, complex formation).

7.1.1. Representation of the Genetic State of a Cell

If a gene is transcribed, its mRNA translated, and the protein present and active, the corresponding BIC's state is said to be `1`, else, if the gene is not transcribed, the mRNA not correctly processed, or the protein inactive or not present, the state of the BIC is `0`. The genetic state of a cell is the collection of the states of all BICs in the cell. It can be described by a vector of values in $\{0, 1\}$.

To include cell-cell interactions in the simulation there must be some way that BICs can influence other BICs not only in their own cell but also in neighbouring cells. Jackson *et al.* (1986) describe a technique of dividing a genetic network into an internal and an external part. In our model, this is represented by BICs that can have different ranges of activity. Some BICs are *internal*, their state in a specific cell is relevant only for the calculation of the next state of the same cell. In contrast, *external* BICs do not influence the state of their own cell, but rather that of all neighbouring cells (growth factors or secreted proteins, for instance). BICs

may also have both internal and external activities. They can then affect the state of their own cell as well as that of neighbouring cells.

Prior to the calculation of the next state of a cell the states of the adjacent cells have to be considered to identify potential inter-cellular interactions. This is done in several successive steps (see also the example on page 41). First, the external state of the cell is determined: the neighbouring cells are identified (by Delaunay triangulation, see section 5.2) and their states are collected to form a combined "external" state vector. If in any of the neighbouring cells a BIC has state `1`, its state in the external state vector is set to `1` as well. Else, i.e. if this BIC is inactive in all surrounding cells, its external state is set to `0`. In a second step, the new overall state of the cell is determined by combining external and internal state. A vector is created that contains for all BICs with *internal* range their state in the respective cell and for all *external* BICs their state from the external state vector. The resulting vector is used for calculation of the next state for the respective cell.

Setting the Next State of the Cells Only after the next states have been calculated for all cells, these states will be set. Therefore all cells change their states simultaneously and not successively. This is important to make sure the model retains its deterministic behaviour in the cell-cell interactions. If cells would change their states successively one would have to define which cell changes its state first. In real living cells state changes do not occur as discrete steps but rather gradually, therefore it would not make sense to model them as successive events.

The state changes in our model occur in regular time intervals. The precise interval length as well as the time of the first change can be defined by the user. This makes it possible to co-ordinate state changes with cell data in simulations where cell positions and division times are taken from a database, as is the case for the *C. elegans* simulation described in section 8.2.

In the next sections, different methods for calculating the next state are described.

7.1.2. Matrix multiplication

For the matrix calculation technique, the genetic network is treated as a set of interactions between BICs. Each interaction has a specific strength. Positive strength values indicate activation, negative values indicate inhibition. The absolute strength can be used to specify that one interaction has a stronger effect than another, for example an inhibiting interaction with strength -1 can be outweighed by an activating interaction of strength +2. Though the program allows interaction strengths to be any real number, integer values are more intuitive and are sufficient for all tested models and simulations. All interactions together are formulated as an interaction matrix where each component corresponds to the strength of the influence of the BIC in the respective column on the BIC in the respective row.

The calculation algorithm is implemented in the `MatrixCalculator` class. It is based on a method used by Mendoza and Alvarez-Buylla (1998). They use an asynchronous approach, i.e. the BICs can change their states successively. GENE-O-MATIC applies a synchronous approach, where all BICs change states simultaneously. This turned out to be sufficient for the simulation of *C. elegans*.

The next state of a cell is obtained by multiplying the interaction matrix with the state

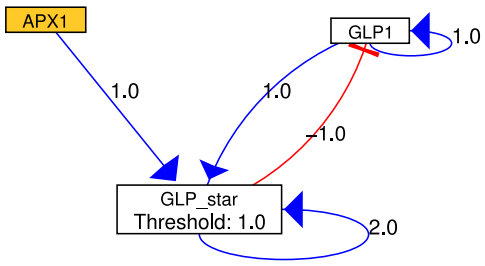

Figure 7.1.: A small example network. APX-1 is a signalling protein on the cell surface. The GLP-1 receptor is expressed on neighbouring cells. Upon contact, GLP-1 is split and a fragment, which we name GLP-1* (GLP_star), moves to the nucleus and activates transcription of target genes.

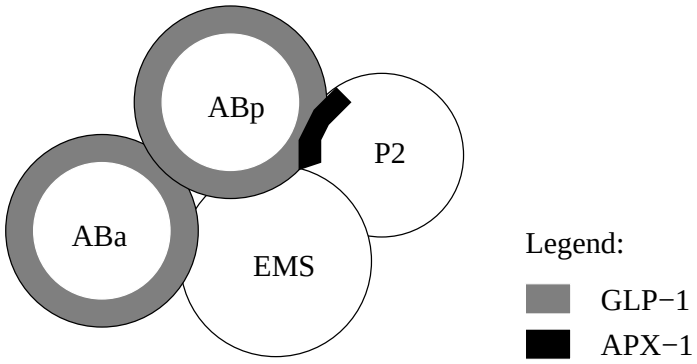

Figure 7.2.: Localization of GLP-1 and APX-1 in 4-cell embryos. GLP-1 (grey) is present at the cortex of ABa and ABp. In P_2, APX-1 (black) is present at the border to ABp.

vector of a cell (after combining it with the states of the neighbouring cells). The resulting vector cannot be used for the new state vector directly as it may contain values other than `0` and `1` (e.g. if an interaction strength was larger than 1 or if two interactions activate the same gene). Therefore it is re-discretized by applying a step function, assigning `1` whenever the result vector entry is larger than the respective BIC's threshold, and `0` if the result vector entry is smaller than or equal to the BIC's threshold. The re-discretized vector represents the next genetic state of the cell. The threshold can be freely defined by the user. An example of a genetic network and the calculation of the next states is given below.

Calculation Example To illustrate the matrix calculation algorithm, we demonstrate an example calculation here. The network for ABa vs. ABp specification in *C. elegans* is used (see figure 7.1 and sections 8.1.4.a and 8.2.5). The interaction matrix that describes the genetic network looks like this:

$$\mathbf{M} = \left(\begin{array}{c|ccc} & \text{APX-1} & \text{GLP-1} & \text{GLP-1*} \\ \hline \text{APX-1} & 0 & 0 & 0 \\ \text{GLP-1} & 0 & 1 & -1 \\ \text{GLP-1*} & 1 & 1 & 2 \end{array}\right)$$

Table 7.1.: BIC States

BIC	**Internal genetic state of**				**State of ABp**	
	ABp	ABa	EMS	P_2	external	calculation-relevant
APX-1	0	0	0	1	1	1 (ext.)
GLP-1	1	1	0	0	1	1 (int.)
GLP-1*	0	0	0	0	0	0 (int.)

The field in row GLP-1* and column APX-1 describes the influence that APX-1 has on GLP-1*, i.e. the strength of the arrow from APX-1 to GLP-1* in figure 7.1.

The cell positions and protein activities at the time of the induction of ABp are shown in figure 7.2. Because a detailed representation of the distribution of molecules within cells is not possible in our model, for calculation purposes it is assumed that APX-1 is active in the whole P_2 cell. The mathematical description of the states of all BICs in this example is shown in table 7.1.

Let us assume we want to know the next state in cell ABp. First, ABp's external state is determined by finding out which BICs are active in any of the neighbouring cells (of course, beforehand it is determined which cells *are* neighbours of ABp, which are all cells in this case). GLP-1 (in ABa) and APX-1 (in P_2) are active (`1`) in the external state. The external state vector is determined by the `ExternalStateCombiner` class. Then the calculation-relevant state of ABp is determined by combining ABp's external and internal states in such a way that for all internal genes (in this case, GLP-1 and GLP-1*) the internal state of ABp and for all external genes (here, APX-1) the external state of ABp is used. For this, the `ExtIntCombiner` is used. The sources of the calculation-relevant states are additionally indicated by "(ext.)" and "(int.)" behind the states in table 7.1. The calculation-relevant state of ABp is then multiplied with the interaction matrix M:

$$\underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 1 & 1 & 2 \end{pmatrix}}_{\text{interaction matrix}} \cdot \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}}_{\text{calc.-rel. state of ABp}} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix} \overset{?}{>} \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}}_{\text{thresholds}} \Rightarrow \underbrace{\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}}_{\text{ABp's new state}}$$

After the multiplication, the re-discretization is done by comparing the result with the thresholds and assigning `1` whenever the result is larger than the threshold. The end result is the new state of cell ABp. Both the multiplication and the re-discretization is done by the `MatrixCalculator`.

The same calculation is done simultaneously for the other cells, after which the new states are set for all cells at once. The whole process is repeated at every state change time.

7.1.3. Boolean Functions

In this representation, interactions are formulated as Boolean functions. Each BIC is assigned a Boolean (or logic) function (or formula) that combines all input BICs for that BIC using

logical operators like `AND` (`&`), `OR` (`|`), and `NOT` (`!`). A BIC that is activated by BIC `B` and inhibited by BIC `C` could have a Boolean function that looks like this: `B&(!C)`. Note that no explicit interactions (`"arrows"`) are required here, because they are implicit in the formulas. The BICs from the network in figure 7.1 would have the following Boolean functions if the same interactions were to be modelled:

BIC	Boolean function
APX1	(empty)
GLP1	`GLP1`
GLP_star	`(APX1 & GLP1) | GLP_star`

To calculate the next state of a cell in this model the Boolean functions for all BICs are evaluated, interpreting BICs with state `1` as `TRUE` and BICs with state `0` as `FALSE`. This is done synchronously, i.e. all functions are evaluated before any new states are actually set. The algorithms for the interpretation of the formulas and for their evaluation are part of the ORBITAL library for JAVA (Platzer, 2003). The class that applies them to the genetic network is the `LogicCalculator`.

7.1.4. Converting Between Matrix and Boolean Network

The matrix representation is a more intuitive description for genetic networks as one can create a network basically by drawing a picture of it. The Boolean model is available because there are some special cases that cannot be modelled accurately by the matrix representation without introducing additional, artificial BICs into the network and without using “Multi-BICs” that combine the functionality of several sequentially connected BICs (e.g. the Boolean operator `XOR` (`^`) that means that exactly one of two BICs is on). On the other hand, different-strength interactions can be modelled only with the matrix but not with the Boolean network. Whether these special cases are really biologically relevant is a question not to be discussed here.

For convenience, conversion mechanisms between the two models are provided. They do a near conversion (because as stated above an exact conversion may not be possible). The user can thus specify part of the network in the Boolean formalism and another part using the matrix formalism. The program can then combine the information to provide a complete model in both of these representations. It might also be interesting to see how these models differ in their behaviour.

7.1.5. Mixed Calculation Method

There is also a calculator that combines both calculation modes. For those BICs to which a Boolean function was assigned, the `LogicCalculator` is used, and if there is no Boolean function, the `MatrixCalculator` is used.

7.2. The Cell Model

Two methods exist in GENE-O-MATIC to model cell behaviour. First, the cell positions can be taken from a database of recorded (or pre-calculated) positions. The recorded cell positions

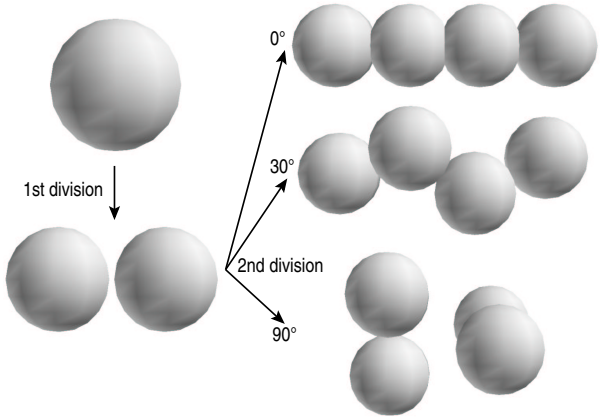

Figure 7.3.: Simulated cell division directions. The mother cell divides along a horizontal line (1st division). Different rotation angles for the daughters' division directions with respect to the mother cell's division direction are shown: no rotation ($0°$), $30°$ rotation, and division orthogonal ($90°$) to the mother's division direction.

for the *C. elegans* embryo (Schnabel *et al.*, 1997) are an example of such a database. The user does not need to specify any cell information in the genetic network editor except which cells are present at the beginning of the simulation and what their genetic states are.

The second possibility is to simulate cell movement and divisions. In this case the user has to specify position and size of the initial cells. Additionally some information about the division direction of the cells has to be provided. The division direction of a cell is the direction of the mitotic spindle, a vector orthogonal to the actual plane along which the daughter cells separate. The simulation environment assumes that cells divide in a direction orthogonal to the division direction of their mother, parallel to the division plane of the mother cell. So two random vectors parallel to the mother cell's division plane and orthogonal to each other are calculated as the division directions of the two daughter cells. The division direction vectors may be rotated if the cell is in a specific genetic state. The rotation angle and the Boolean function describing this genetic state can be specified by the user. Some examples for default and modified division directions are shown in figure 7.3. The initial cell (top left) divides along the horizontal x-axis. To each of the daughter cells a new axis is assigned which is orthogonal to the mother's division direction. If this is not modified, the daughter cells divide in exactly these directions ($90°$).

The time of division can be regulated via the genetic network, too. For this purpose a Boolean formula can be defined that describes the genetic state which triggers a division.

7.2.1. Asymmetric Divisions

A crude mechanism for the simulation of asymmetric cell divisions is included in the cell representations. For BICs which are to be distributed unequally or asymmetrically upon cell division, their preferred location has to be specified. This can be done by setting the location to one of the keywords *anterior*, *posterior*, *dorsal*, *ventral*, *left*, or *right* or to a combination of

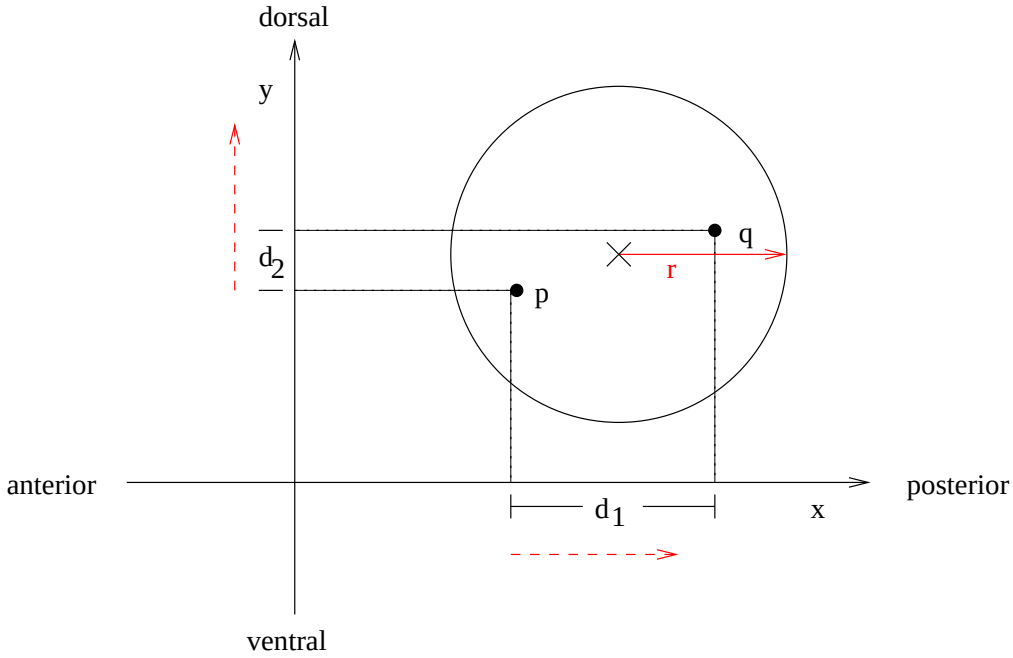

Figure 7.4.: Determination of localised asymmetric distribution of BICs following a cell division. Black dots represent the centres of the daughter cells, the large circle represents the mother cell, the cross its centre. The mother cell's radius r is indicated by the red arrow. The projection of the distance vector onto x-axis (d_1) and y-axis (d_2) are shown, the third dimension (z) was left out for simplicity. For comparison, the length of the mother cell's radius is indicated by dashed red lines next to the projections d_1 and d_2. This division is asymmetrical along the x axis (if a scaling factor of 1 is assumed), because the distance projection in x direction is larger than r. It is symmetric along the y axis because d_2 is smaller than r.

Table 7.2.: Asymmetric distribution of BICs according to their preferred location.

BIC location	Associated coordinate axis	Cell with BIC activity
anterior	x	smaller x-value
posterior	x	larger x-value
dorsal	y	larger y-value
ventral	y	smaller y-value
left	z	larger z-value
right	z	smaller z-value

these (see also table 7.2). The calculation mechanism is illustrated in figure 7.4. The division direction vector is the difference d between the positions p and q of the daughter cells. The i-th component of that vector is its projection onto the i-th coordinate axis: $d_i = q_i - p_i$. It is then evaluated whether the absolute value of the respective component is sufficiently large to justify an asymmetric distribution of BICs in that direction. A small deviation along one of the axes will still result in an uniform distribution of BIC activities. The evaluation is done by comparing it to the radius r of the mother cell, multiplied by a scaling factor s:

$$|q_i - p_i| > s \cdot r \Rightarrow \text{division is asymmetric along axis } i$$

The user can change the value of the factor s to adapt the simulation to his specific needs. If a division has been evaluated to be asymmetric along a specific axis, a lookup in table 7.2 is done to determine for all BICs with a specific preferred location in which of the two daughter cells the BIC remains active, and in which cell the activity has to be suppressed. In the example in figure 7.4 a BIC with the preferred location "anterior" which was active in the mother, will still be active in daughter p (the anterior daughter), and will be set to `OFF` in daughter q, the posterior daughter. A BIC with the preferred location "dorsal" will still be `ON` in both daughters because the y component of the division direction vector is not large enough for the division to be considered asymmetric along the y axis.

7.3. Additional Simulation Parameters

Several additional parameters which have not been mentioned above are available to configure the behaviour of the simulation.

Calculator Specifies the type of calculator to use for calculating the next state of the cells: Matrix, Logic, or Mixed

Start time for state changes The time when the first genetic state change occurs, i.e. when the genetic network starts working. This parameter is available to adjust the genetic network simulation to pre-recorded cell position and division information from a database.

State change interval The time between two state changes, i.e. between two calculations of the next genetic state of all cells. For simulated positions, this can be left unchanged, but if cell data is taken from a database, larger intervals may be required for synchronisation.

Cell cycle time A time interval after which all cells in the simulation divide. This may be easier than explicitly using BICs to simulate a cell cycle. A drawback is that all cells divide synchronously if a cycle time is specified, while with a simulated cell cycle some cells can divide earlier or more often than others, depending on their genetic state.

Use egg shell constraints For the worm simulation, the "world" that contains the simulated cells provides an egg shell which keeps all cells inside an oval space. This can be switched off for other simulations.

Consider gene locations If this is disabled, the BIC locations, as desribed in section 7.2.1, are not considered and asymmetric divisions are impossible.

Division direction calculation factor If "consider gene locations" is enabled, the mother cell radius will be multiplied by this factor to determine the minimum distance two daughter cells must have in anterior-posterior (or dorsal-ventral or left-right) direction to distribute BICs asymmetrically in this direction.

Neighbourhood factor To calculate whether two cells are neighbours, their radii are added and it is checked whether the result is smaller than the distance between the two cell centres (restricted Delaunay triangulation, cf. figure 5.5 A). As this is sometimes too strict a measure, the sum of the radii is multiplied by this factor to artificially enlarge the cells to make sure all intended neighbours are found.

Part III.

Results

This part contains the scientific results achieved in this thesis. Two models have been studied: *Caenorhabditis elegans* and *Arabidopsis thaliana*. For each of them, known biological data will be reviewed, a model (genetic network) constructed, and then, simulation results will be presented.

8. Blastomere Fate Specification in *C. elegans*

Blastomere fate specification in *C. elegans* is a prime example for a multicellular developmental process. Apart from intra-cellular signalling cascades (e.g. MAP cascades), interactions between cells and asymmetric divisions are also part of the regulatory repertoire applied to determine the embryonic founder cells.

The cell lineage of the early embryo is shown in figure 8.1. The first division of the zygote and the generation of two different daughter cells is a result of polarity establishment rather than a first step in blastomere fate specification. Therefore, this chapter will deal only with those regulatory interactions that happen after the initial division, i. e. beginning with the two-cell stage. An overview of the whole process is given in figure 8.2.

The specification of blastomere fates is — as is the polarity establishment and transduction — regulated by maternally provided mRNA and proteins. The embryo can develop without any own transcription up to the 24-cell stage (Newman-Smith and Rothman, 1998). The maternal transcription factors set the stage for embryonic expression of tissue and organ identity genes.

Blastomere fate specification is finished when the fates of the so-called embryonic founder cells AB, C, D, E, MS, and P_4 have been specified. This has happened after the division of P_3, in the 24-cell stage. However, all specification events except for one are finished in the 12-cell stage. The only thing that is missing there is the division of P_3 and therewith birth of P_4 and D. This is not discussed here because the mechanism of D vs. P_4 specification is analogous to that of C vs. P_3.

Section 8.1 reviews biological data about regulatory interactions during blastomere fate specification. As there is currently no publication describing the genetic network in its entirety, data collection was an important part of the thesis. The process of model building — how biological data is interpreted to create a model of the genetic network which can then be simulated — is described in section 8.2. The results of the simulation and their relation to experimental data are listed in section 8.3.

8.1. Experimental Evidence for Regulatory Networks

Blastomere fate specification is one of the best studied processes in embryonic development of *C. elegans*. One of the reasons for this is the fact that mutations that disrupt this process lead to dramatic changes in later development, preventing the analysis of later developmental stages. Therefore only the early steps in development are easy to study.

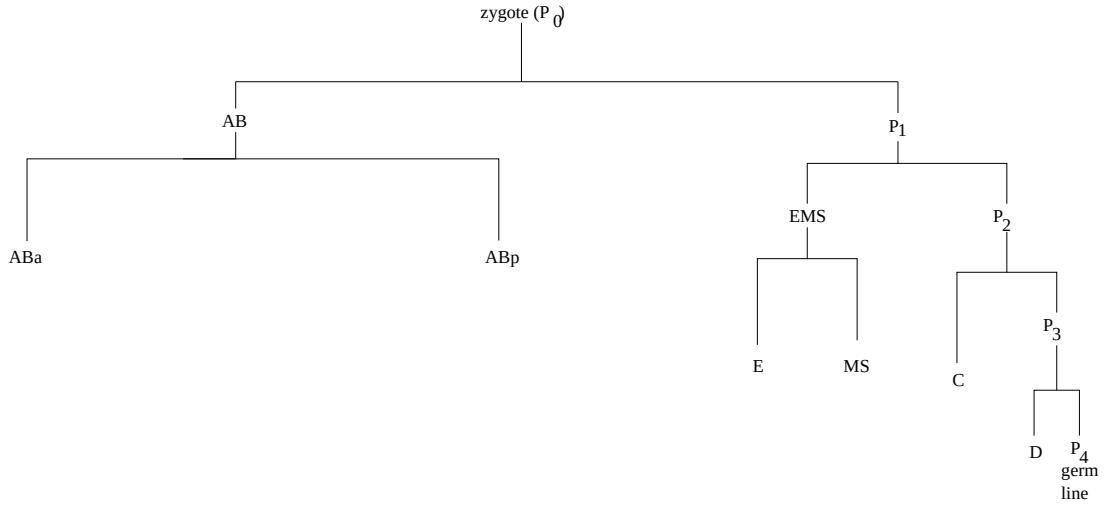

Figure 8.1.: The early cell lineage of *C. elegans*. The formation of the six founder cells AB, E, MS, C, D, and P_4 is shown.

The genetic network that governs blastomere fate specification is shown in figure 8.3. It has not been published in this form before, encompassing so many elements and showing the multitude of connections between them. The genes and interactions and the literature references are listed in tables C.1 and C.2, respectively, in appendix C

8.1.1. Initial Situation

The starting point for blastomere fate specification is the presence of two distinct cells — AB and P_1 (see figure 8.2 A). The most important molecules in them are GLP-1 in AB (Evans *et al.*, 1994) and PIE-1 and P granules in P_1 (Mello *et al.*, 1996; Strome and Wood, 1983; Wolf *et al.*, 1983).
PIE-1 asymmetry is a direct result of zygotic polarity. Therefore it will not be discussed here. The asymmetry in GLP-1 distribution is not the result of an asymmetric distribution of the mRNA or protein during zygotic polarity establishment. The *glp-1* mRNA is distributed homogeneously in all early embryonic blastomeres. Instead, control happens at the translational level. The 3' untranslated region (UTR) of the mRNA is required for the anterior localization of the protein (Evans *et al.*, 1994). This mechanism is often found in the regulation of maternal mRNAs. Of course, many other molecules are present in AB and P_1. Their function in blastomere fate specification will be discussed in the next sections where the single blastomeres are described.

The following paragraphs will shed light onto the molecular regulatory processes that lead to the specification of the founder cells. The events leading to the specification of single blastomeres will be examined separately, though there are of course interdependencies between the different specification pathways.

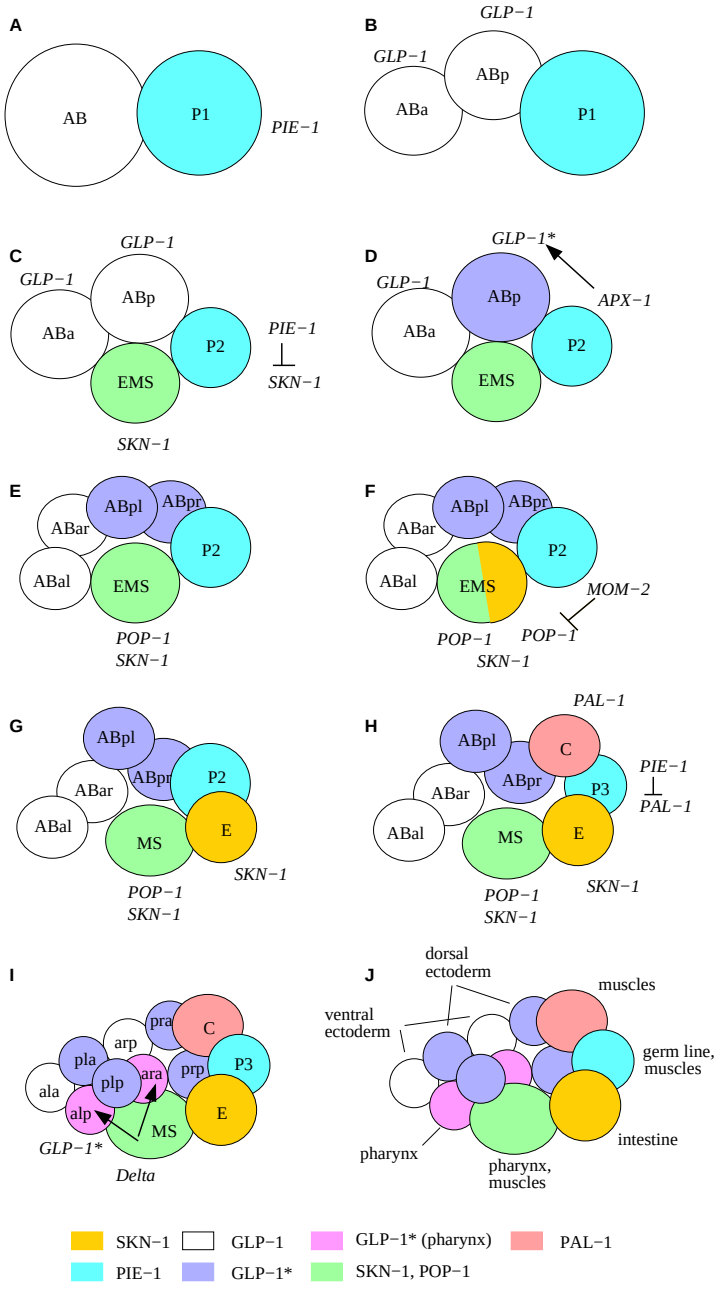

Figure 8.2.: Overview of blastomere specification. Important regulatory mechanisms are shown, the names of the proteins involved are given in italics. Different cell types (in terms of protein activity) are indicated by different colours. **A.** The 2-cell stage. **B.** 3-cell stage. **C, D.** The 4-cell stage. Cell-autonomous specification of P_2 fate by PIE-1, induction of ABp by P_2 via APX-1. **E.** Division of AB granddaughters. **F, G.** Induction of EMS by P_2 via MOM-2, followed by asymmetric division of EMS. **H.** The 8-cell stage. Specification of P_3 vs. C fate by PAL-1 and PIE-1. **I.** Induction of ABalp and ABara by MS. For simplicity, the names of the AB granddaughters are abbreviated. **J.** Seven different cell types have been specified. Cell arrangement as in I.

8.1.2. Germ Line Specification, P_4 Cell

The germ line is the part of the worm whose specification is the simplest. It works by a cell-autonomous process that distributes germ-line specific proteins to the posterior (P_1, P_2, P_3, P_4) during the first divisions (the cyan cells in figure 8.2). After the third division (that of P_2), a polarity reversal takes place, and the anterior daughter inherits the germ line specifying molecules. In addition, all these divisions are asymmetric also in terms of cell size; the germ line precursor inherits only 30 to 40% of the mother cell's volume, while the somatic daughter gets the remaining 60 to 70%. This mechanism is cell-autonomous, i.e. independent of any inter-cellular interactions, and works even if all other cells are ablated. The most important actor in the blastomere fate specification process is the germ line determinant PIE-1. It is a transcriptional repressor that interferes with RNA polymerase II phosphorylation and thereby totally inhibits embryonic transcription in the cells where PIE-1 is expressed. By this mechanism, the effect of all other fate-specifying transcription factors is eliminated. The germ line precursors retain their omnipotent state because they do not express fate-specifying transcription factors that would restrict their developmental potential. After fertilization, PIE-1 is localized to the posterior half of the zygote (Mello *et al.*, 1992). It remains in the posterior part of the cells during the following cell divisions. It is therefore distributed to P_1, P_2, P_3, and P_4, but not to their sister cells (AB, EMS, C, and D, respectively) (Strome and Wood, 1982; Seydoux and Fire, 1994).

Together with PIE-1, RNA-rich particles called P granules are segregated to the germ line precursors, as well (Strome and Wood, 1983; Wolf *et al.*, 1983). Their exact function is unknown, but they contain several components important for blastomere fate specification, e.g. POS-1 and MEX-1 (Seydoux and Schedl, 2001).

8.1.3. Posterior Blastomeres

8.1.3.a. EMS Cell

SKN-1 is a maternally encoded transcription factor present in EMS and P_2, but not in AB and descendants (Bowerman *et al.*, 1993, see figure 8.2 C). It is responsible for specifying the EMS fate (Bowerman *et al.*, 1992a), and it is prevented from acting in P_2 because in P_2 PIE-1 generally inhibits transcription (Mello *et al.*, 1992; Seydoux *et al.*, 1996, see figure 8.2 C). The initial unequal distribution of SKN-1 (its absence from AB) depends on MEX-1 and PAR-1 (Bowerman *et al.*, 1993), but the exact mechanism is unknown.

Activity of SKN-1 results in activation of the transcription of embryonic genes required for correct development of the mesendoderm, e.g. *med-1* and *med-2* (Maduro *et al.*, 2001).

Another protein present in EMS is POP-1. The joint action of SKN-1 and POP-1 determine the fate of EMS, and also of its anterior daughter MS, as mesodermal precursors (see below).

8.1.3.b. E Cell

E is the posterior daughter of EMS. It is different from MS, the anterior daughter, due to an inductive signal sent by P_2 before division (see figure 8.2 F). The signalling mechanism uses MOM-2, a protein homologous to mammalian Wnt and *Drosophila* Wingless (Thorpe

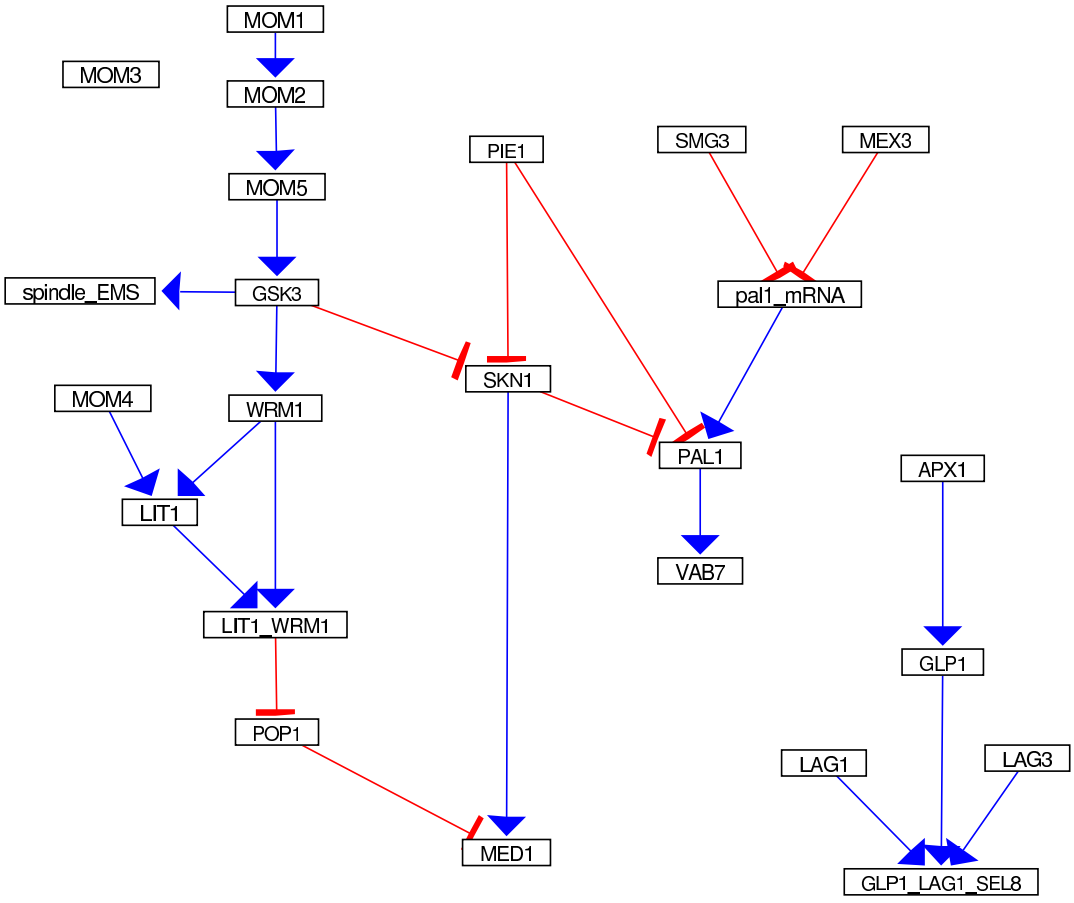

Figure 8.3.: Blastomere specification network. Pointed arrows represent activating interactions, while arrows with cross bars represent inhibition. GLP1_LAG1_SEL8 and LIT1_WRM1 are protein complexes, spindle_EMS means the rotation of the spindle in EMS. The references for all elements of the network are given in table C.1 in the appendix.

et al., 1997; Rocheleau *et al.*, 1997). The receptor, in agreement with this, is the Frizzled homolog MOM-5 (Thorpe *et al.*, 1997; Rocheleau *et al.*, 1997). In P_2, MOM-1, a Porcupine homolog, is required for correct processing and secretion of MOM-2, the signal. This has not been determined experimentally, but is a conclusion based on sequence homology with the *Drosophila* proteins (Thorpe *et al.*, 1997; Rocheleau *et al.*, 1997). MOM-1 and MOM-2 are expressed exclusively in P_2, but in no other cell at the four-cell stage, alas, it is not known how this is achieved. A possible explanation would be the association of their mRNAs with the P granules.

MOM-3 is another protein required in P_2 for successful Wnt signalling (Thorpe *et al.*, 1997; Rocheleau *et al.*, 1997). Its function is unknown, which is why there are no interactions involving MOM-3 in figure 8.3.

The receptor MOM-5 relays the signal in EMS via GSK-3, as it is called because of its homology to human GSK-3 β (Schlesinger *et al.*, 1999). Here, the pathway splits up, because GSK-3 is not only involved in the specification of the fate of the posterior half of EMS, but is also responsible for regulating spindle orientation in EMS (Schlesinger *et al.*, 1999). Components of the Wnt pathway that act downstream of GSK-3 have no influence on spindle orientation in EMS. How GSK-3 influences spindle orientation has not been elucidated yet. From GSK-3, the signal is passed to WRM-1, an armadillo/β-catenin homolog (Rocheleau *et al.*, 1997). WRM-1 forms a complex with the Nemo-like kinase LIT-1, a MAPK[1]. At this point, an integration with a MAPK cascade takes place: the MAPKK[2] or MEK[3] MOM-4 also acts on LIT-1. Functional MOM-4 is required for the Wnt signal (Thorpe *et al.*, 1997), but the reason is unknown.
LIT-1 is activated by interaction with WRM-1 and undergoes auto-phosphorylation to further enhance its interaction with WRM-1. LIT-1, or the LIT-1–WRM-1 complex, deactivate POP-1. POP-1 is a TCF/LEF-like (T cell factor/lymphoid enhancer factor) transcription factor with a HMG (high mobility group) domain (Lin *et al.*, 1995). It is responsible for many, if not all anterior-posterior fate decisions in embryonic development, and an asymmetry in POP-1 distribution can be detected in the daughters of most such divisions (Lin *et al.*, 1998). High concentrations of POP-1 can be detected in the nuclei of anterior daughters, and only low concentrations in posterior nuclei. How it happens that inactive POP-1 in the posterior does not show up in immunofluorescence staining and thus seems to have disappeared in the posterior daughters is unknown.

POP-1 is required maternally as well as zygotically, and the two functions are independent from each other because a mutation that shows maternal-effect lethality and defects in pharynx development leaves the embryonic function of POP-1 unimpaired (Lin *et al.*, 1998).

To summarize: E is different from its sister MS because it does not contain active POP-1 protein (figure 8.2 G). The presence of SKN-1 as the only active transcription factor in E results in E's adoption of endodermal fate; the E cell is the immediate precursor of all intestinal cells of the worm.

[1] mitogen-activated protein kinase
[2] MAPK kinase
[3] MAPK/ERK kinase, ERK is a special MAPK in humans

8.1.3.c. MS Cell

MS is the anterior daughter of EMS. Its fate is not different from EMS in terms of transcription factor activity, i.e. SKN-1 and POP-1 are both present and active. POP-1 specifies mesodermal versus endodermal fate (Lin *et al.*, 1995) and is the basis for the different developmental potential of E and MS. SKN-1 and POP-1 together specify mesodermal cell fate (Thorpe *et al.*, 1997, see figure 8.2 G). Therefore, MS does not adopt an endodermal fate, but becomes a mesodermal precursor instead.

8.1.3.d. C Cell

The C cell is defined by activity of PAL-1 (figure 8.2 H). PAL-1 is a maternal transcription factor that activates muscle-specific structural genes such as *vab-7* (Ahringer, 1997). By several different mechanisms, PAL-1 activity is directed to the C cell and its descendants. *pal-1* mRNA is present in both initial cells in the embryo. MEX-3, which is present in AB only, inhibits *pal-1* mRNA translation. It has been proposed that SMG-3 causes the degradation of untranslated *pal-1* mRNA, because *pal-1* mRNA degradation depends on *smg-3* function (Hunter and Kenyon, 1996). SMGs are proteins involved in the degradation of abnormal or truncated mRNAs (Pulak and Anderson, 1993). Though SMG-3 is also present in the posterior, *pal-1* mRNA is degraded only in the anterior, because its translation is inhibited only there. Because of this mechanism, PAL-1 protein exists only in the posterior half, i.e. in P_1, EMS and P_2, etc.
In EMS and descendants, PAL-1 is inhibited in a SKN-1-dependent manner (Hunter and Kenyon, 1996). Maduro and colleagues report that not SKN-1 itself, but END-1 and/or END-3, two GATA transcription factors activated by SKN-1, may be responsible for PAL-1 inhibition (Maduro *et al.*, 2001).
In P_2, the effect of PAL-1 activity (transcription of muscle-specific genes) is prevented because the presence of PIE-1 blocks all transcription. The same holds true for P_3. In C, however, a lack of any inhibiting factors allows PAL-1 to exercise its function and to activate transcription of muscle-specific genes such as *vab-7* (Hunter and Kenyon, 1996).

8.1.4. Anterior Blastomeres

8.1.4.a. ABa versus ABp

ABa and ABp initially have the same developmental potential, since the division of AB is absolutely symmetric (Sulston *et al.*, 1983, see figure 8.2 B). In the four-cell stage, however, an inductive signal emanating from P_2 distinguishes ABp from ABa, driving ABp to adopt a dorsal fate (figure 8.2 D). The signal uses a pathway homologous to the Notch/Delta/Serrate pathway in *Drosophila*.

The ligand involved here is the Delta-homolog APX-1, a protein expressed in P_1 at the border to ABa and ABp, and later in P_2 at the border to ABp (Mickey *et al.*, 1996), see also figure 7.2. The mRNA is present in all blastomeres (Mickey *et al.*, 1996). How the localized expression is achieved is unknown.

The membrane-bound receptor on ABp is the Notch-homologue GLP-1 (Mello *et al.*, 1994). Its mRNA is present in all blastomeres, but it is translated only in AB and its descendants. This is due to a regulatory mechanism that involves the 3' UTR in the *glp-1* mRNA (Evans *et al.*, 1994). Upon contact with APX-1, GLP-1 is split and the cytosolic part migrates to the nucleus. At least, this is how the pathway works in other organisms (Schroeter *et al.*, 1998; Struhl and Adachi, 1998). On its way, it binds to SEL-8. SEL-8 (also called LAG-3) is a transcriptional activator that has no DNA-binding capabilities (Petcherski and Kimble, 2000). Therefore, to activate transcription, interaction with DNA-binding proteins is required. LAG-1 is a transcription factor (Christensen *et al.*, 1996) that is bound to the promotor of SEL-8 target genes all the time (Petcherski and Kimble, 2000). It recruits the fragment of GLP-1 and with it SEL-8 which is bound to that fragment. The ternary complex binds to the DNA and enables SEL-8 to activate the transcription of target genes (Petcherski and Kimble, 2000). ABa is not induced although it expresses GLP-1, since it is not in direct contact with P_2 and therefore cannot receive the signal.

Because the promotor of *glp-1* contains LAG-1 binding sites, it has been proposed that a positive feedback loop exists for this signalling pathway (Christensen *et al.*, 1996). But as blastomere fate specification does not require embryonic transcription, the relevance of the feedback loop for this process is questionable. It may have more impact later in development with *lin-12*, another Notch-homolog that also has LAG-1 binding sites in its promotor.

8.1.4.b. Pharynx Induction in AB Granddaughters

Without interference, four of the AB granddaughters would produce mainly hypodermis and four of them would produce neuronal tissue. However, two of them, ABalp and ABara, are induced by a signal from MS to produce pharyngeal cells instead (Priess and Thomson, 1987; Hutter and Schnabel, 1994, see figure 8.2 I). Therefore, neurons and hypodermis are each produced by only three AB granddaughters (Sulston *et al.*, 1983). Strictly speaking, this induction is not part of the specification of the embryonic founder cells. However, as this event takes place even before all founder cells are born — namely, at the 12-cell stage, with D and P_4 still missing — it was included it in this description. Also, the mechanism is well studied.

The induction works similar to that of ABp by P_2: it also involves a Notch/Delta pathway, and the receptor is the same as in the specification of ABa vs. ABp: GLP-1 (Mello *et al.*, 1994). The Delta-homolog involved has not been identified yet. For obvious reasons it is clear that APX-1 is not involved: beginning with the 8-cell stage, its mRNA is degraded in all somatic blastomeres, therefore it is simply not present in MS (Mickey *et al.*, 1996).

Experiments by Hutter and Schnabel (1994) have established the fact that only MS is required for this induction. Therefore, the signalling protein must be expressed in MS, but not in any other blastomere that is in contact with AB descendants at the time of signalling. In spite of this rather exact description, it has not been possible to identify the Delta homolog unambiguously yet. Possible candidates are LAG-2 or ARG-1, which are the only other Delta homologs described in *C. elegans* (Henderson *et al.*, 1994; Fitzgerald and Greenwald, 1995). Other homologs can be predicted from the genome sequence, but they have not been verified. However, the presence of either LAG-2 or ARG-1 in MS and their requirement for the induction still remain to be verified, too.

Two interesting questions emerge: first, how is expression of the signalling Delta homolog restricted to the MS cell, and second, why do only two of the AB descendants react to the signal?

The answer to the second question is relatively simple. Of the ABa descendants, only ABalp and ABara are in contact with MS at the time of signalling. And the ABp descendants have probably lost the ability to react because they have previously been induced to become dorsal cells (see section 8.1.4.a).

The answer to the first question is more complex. The obvious solution is to assume that the Delta homolog is activated by a mechanism specific to MS. As far as the genes and proteins described so far are concerned, this could only be done by POP-1 or by a joint action of SKN-1 and POP-1. But Lin and colleagues (1995) observed that the induction is normal in POP-1 mutants. In contrast, SKN-1 activity in MS is required for the induction (Mello *et al.*, 1994). Because SKN-1 is a transcription factor, and because destruction of DNA in MS prevents the induction (Mello *et al.*, 1994), the regulation of the induction probably involves embryonic transcription.

8.2. Translating the Regulatory Network into a Computer Model

In this section, the translation of the genetic network described in the previous section is explained. Generally, this is a rather straightforward process. However, some adaptations and modifications were necessary to compensate for the limited capabilities of the simulation software. Also, interaction strengths and BIC activity threshold values needed to be defined. Not all BICs described in the previous sections have been included in the model. For long signalling cascades such as the Wnt pathway, and for complex mechanisms such as the transcription activation by a ternary complex including GLP-1, simplifications were made and BICs left out.

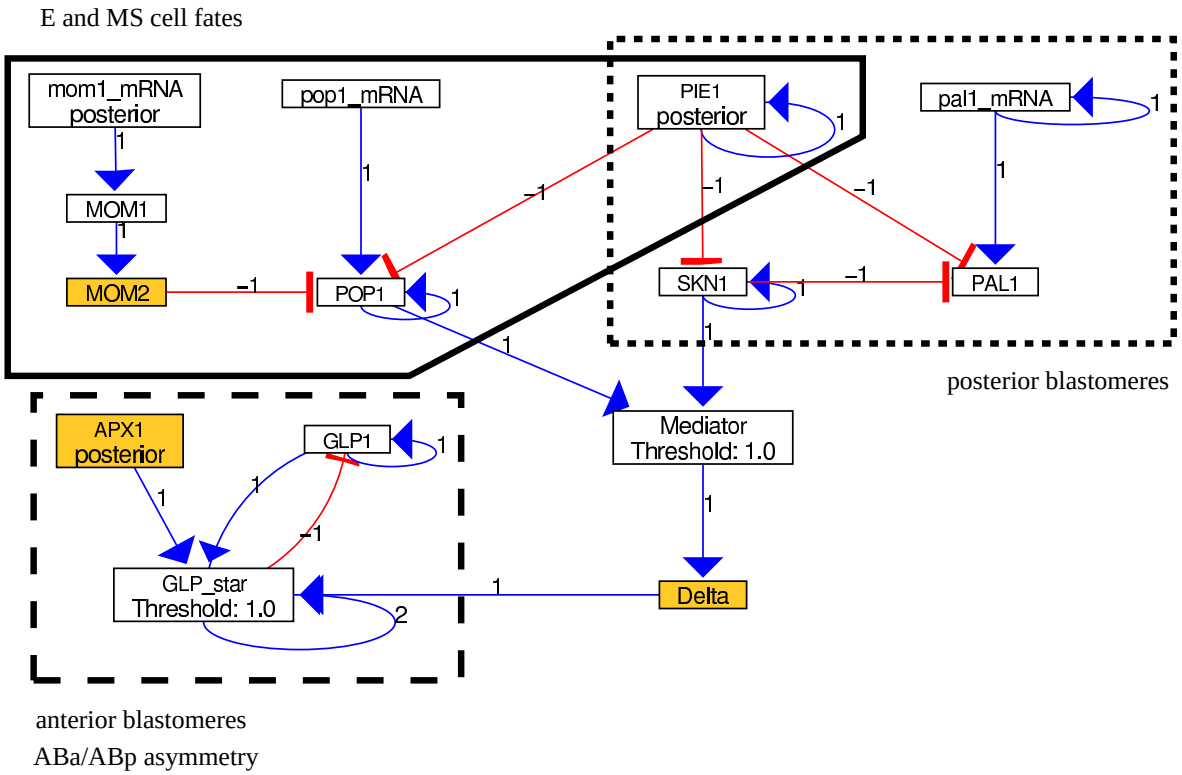

Figure 8.4.: Model of the genetic network regulating blastomere fate specification. For those BICs with a special location, the location is typed below the BIC name. Also, if a BIC's threshold is not zero, this is indicated. External BICs are highlighted in orange. The frames indicate the modules of the network. The dashed rectangle contains the module for the specification of anterior cell fates, the dotted line that for posterior blastomere fates. The solid line indicates the module for E vs. MS specification. The BICs which are not part of any module can fulfil their function correctly only if all other modules are present.

Table 8.1.: The parts of the network.

BIC	location	range of activity	threshold	initial state in AB	initial state in P_1
PIE-1	posterior	internal	0	0	1
SKN-1		internal	0	0	1
pal-1 mRNA		internal	0	0	1
PAL-1		internal	0	0	0
mom-1 mRNA	posterior	internal	0	0	1
MOM-1		internal	0	0	0
MOM-2		external	0	0	0
pop-1 mRNA		internal	0	0	1
POP-1		internal	0	0	0
APX-1	posterior	external	0	0	1
GLP-1		internal	0	1	0
GLP-1*		internal	1	0	0
Mediator		internal	1	0	0
Delta		external	0	0	0

As indicated in figure 8.4, the modelled genetic network consists of several modules. Each module can be simulated separately and is necessary and sufficient to cause a specific subset of the observed developmental effects. Two main modules can be distinguished: BICs for anterior cell fates (APX-1, GLP-1, GLP-1*), and BICs for posterior cell fates (PIE-1, SKN-1, *pal-1* mRNA, PAL-1, *mom-1* mRNA, MOM-1, MOM-2, *pop-1* mRNA, POP-1). The module for the posterior cell fates can be further divided into sub-modules. PIE-1 is a central element for the fate determination of the posterior cells. Two BICs (Mediator and Delta) form a bridge via which the two main modules interact. The modular architecture of the network became evident during the modelling process. It turned out to simplify the modelling considerably, as each module of the whole network could be optimized for correct simulation results separately. The following paragraphs name the BICs and the interactions found to be important and explain how they were incorporated into the model. Details about the BICs and their states in the two initial cells are given in tables 8.1 and 8.2.

8.2.1. PIE-1

PIE-1 is the protein responsible for germ line specification (see section 8.1.2). To make sure it becomes distributed to the germ line precursor cells in our model, it was assigned the location "posterior". Its initial state was set to `1` in the P_1 cell and to `0` in the AB cell to represent its initially asymmetric distribution. PIE-1 is responsible for suppressing embryonic transcription in the germ line precursors, thus specifying their identity. By totally suppressing transcription, it impedes the activity of the transcriptional activators POP-1, SKN-1, and PAL-1. Therefore we introduced inhibiting interactions from PIE-1 to these proteins into our model. To keep PIE-1

Table 8.2.: Biological meaning of BICs. This table explains the meaning of the BICs and which parts from the real worm they represent.

BIC	**If state is 1, this means that ...**
PIE-1	PIE-1 protein is present and active and inhibits transcription by promoting RNA Pol-II phosphorylation.
SKN-1	SKN-1 protein is present and active and generally, transcription takes place, so that it can really effect something (i.e. its function is not made useless by PIE-1).
pal-1 mRNA	*pal-1* mRNA is present and being translated.
PAL-1	analogous to SKN-1.
mom-1 mRNA	*mom-1* mRNA is present and being translated.
MOM-1	MOM-1 protein is present and active, helping MOM-2 secretion.
MOM-2	MOM-2 protein is present and being secreted.
pop-1 mRNA	*pop-1* mRNA is present and being translated.
POP-1	analogous to SKN-1.
APX-1	APX-1 protein is present and active on the cell surface, ready to signal.
GLP-1	GLP-1 protein is present in the cell membrane, ready to receive signals.
GLP-1*	GLP-1 protein is active, i.e. it has been split and the fragment moved to the nucleus to activate transcription of target genes.
Mediator	a protein responsible for the activation of Delta is present and active.
Delta	analogous to APX-1.

active, the model requires a self-activating interaction from PIE-1 to itself; this is not based on experimental data but is a requirement for the modelling mechanism. Otherwise, PIE-1 would switch off again after one step. the self-activating interaction ensures PIE-1 remains active unless it is inhibited by some other BIC. For a detailed explanation of the calculation mechanism, see section 7.1.2.

8.2.2. SKN-1

SKN-1 is represented in our model with an initial state of `1` in P_1, and `0` in AB. The inactivity in the presence of PIE-1 is modelled by an inhibiting interaction from PIE-1 to SKN-1. The target genes of SKN-1 have not been modelled explicitly because the identity of the blastomere is specified already by SKN-1 activity. SKN-1 is an antagonist to PAL-1, which is also a transcriptional activator, and thus an inhibiting interaction from SKN-1 to PAL-1 is included (see also section 8.2.3). SKN-1 remains active (unless explicitly inhibited), represented by a self-activating interaction.

8.2.3. PAL-1

The complex mechanism by which PAL-1 protein is directed to the P_1 cell and removed from AB was not explicitly modelled. Instead, *pal-1* mRNA activity was simply set to `0` in AB and to `1` in P_1. SKN-1 and PIE-1 influence PAL-1 activity negatively — even if SKN-1 does not do so directly, see section 8.1.3.d (Maduro *et al.*, 2001) — therefore inhibiting interactions from SKN-1 and PIE-1 to PAL-1 are included in our model.

8.2.4. Wnt Pathway

Of the Wnt signalling cascade, only the BICs *mom-1* mRNA, MOM-1, MOM-2, *pop-1* mRNA, and POP-1 are incorporated. MOM-2 acts directly on POP-1 here. The other BICs (MOM-3, MOM-5, GSK-3, WRM-1, and LIT-1) were left out because their presence would not have altered the outcome, i.e. the inhibition of POP-1, as they do only pass on the signal.

MOM-1 and MOM-2 are expressed in P_2, but not in any other cell. This is modelled by assigning a posterior location to *mom-1* mRNA (the other BICs become active only in cells with *mom-1* mRNA activity and therefore need no specific location). As it is not necessary (and would even lead to undesired results, see section 8.3.3) that MOM-1 and MOM-2 remain active for a long time, no self-activating interactions are provided for any of these BICs.

Biological observations show that POP-1 is present in the nuclei of all blastomeres in the 4-cell-stage (Lin *et al.*, 1995). As no data exist about POP-1 influence on blastomere specification in ABa and ABp, I decided to ignore its presence there and to constrain it to P_1 and its descendants. This was done by introducing *pop-1* mRNA, with the initial state `0` in AB and `1` in P_1. The mRNA "activates" the protein POP-1. As a transcription factor, POP-1 cannot act in cells where PIE-1 inhibits all transcription, therefore an inhibiting interaction from PIE-1 to POP-1 was introduced into the network model. Once active, POP-1 will remain active until it is inhibited, so a self-activating interaction was provided for POP-1.

8.2.5. Notch Pathway for ABa vs. ABp Distinction

To make sure APX-1 is active only in germ-line precursors (and not in EMS, where it would erroneously induce ABa), it was assigned a posterior location (similar to PIE-1), and its initial state is set to `0` in AB, and to `1` in P_1. It was made an external gene, as its biological function is to transmit signals from one cell to another. As APX-1 cannot be detected after the 8-cell stage, there was no need to make sure the protein remains active, so no self-activating mechanism was modelled.

GLP-1 is translated only in AB and descendants. Thus it was assigned the state `1` in AB, but `0` in P_1. In contrast to APX-1, GLP-1 activity is required later in the embryo, too, for the signal given to the ABa granddaughters, (Mello *et al.*, 1994). Therefore we included a self-activating interaction for GLP-1. GLP-1* represents the intra-cellular fragment that is created by proteolytic cleavage of GLP-1 upon contact with APX-1. This fragment migrates to the nucleus to activate gene transcription. Creation of this fragment requires both the presence of GLP-1 and the activity of APX-1, and to model this, the threshold for GLP-1* was set to 1 (which ensures it becomes active only if the combined activating influence is larger than 1, i.e. if both APX-1 and GLP-1 are active). Once activated, we wanted GLP-1* to remain active (as a proxy for the genes to be activated by GLP-1* which are not known yet), thus we introduced a self-activating interaction. This needs a strength of 2 to overcome the higher threshold. Upon splitting of GLP-1, the membrane-bound form becomes deactivated (it cannot be split twice), thus GLP-1* deactivates GLP-1. GLP-1* is initially off in both cells (state `0`).

8.2.6. Notch Pathway for Pharynx Induction in AB Granddaughters

The not-yet-identified Delta homolog responsible for the induction of ABara and ABalp is represented by the BIC Delta. It is activated — via a mediator BIC — by joint action of SKN-1 and POP-1. A threshold of 1 and activating interactions from both of the proteins to the mediator guarantee that its activity is restrained to MS since a threshold of 1 means it can be activated only by joint action of both proteins, which occurs only in MS. This mechanism is in contrast to biological observations stating that pharynx induction in AB descendants is normal in *pop-1* mutants and therefore is independent of POP-1 (Lin *et al.*, 1995). However, no POP-1-independent possibility to constrain the activity of the Delta-homologue to MS was found for this model. This is an example how experiments in the computer can help identify critical points in the data gathered by laboratory experiments.
The mediator BIC is required for correct timing; because the state changes occur in regular time intervals, without the mediator the signal would reach the AB descendants one time step too early, when the four granddaughters have not yet divided. The mediator therefore is not really part of the signalling mechanism, it is merely an emulation for temporal effects in the simulation.

Delta is, like APX-1, an external protein. Delta activates GLP-1* in neighbouring cells with GLP-1 activity, applying the same mechanism as that used with APX-1.

8.2.7. Parameters for the Simulation

Above, the genetic network used in the simulation was described. Now the other simulation parameters will be listed shortly to allow other users to reproduce the results.

Cell positions and division times were taken from a database of recorded positions (Schnabel *et al.*, 1997). The calculator for the next state was the matrix calculator. Start time for state changes was 19.5 minutes, and the interval between two state changes was 10.5 minutes. Thus, state changes occur at 19.5, 30, 40.5, and 51 minutes. It is crucial to have a state change between 39 and 42.5 minutes, after EMS divides and before P2 does, to simulate the induction of asymmetry in EMS. The simulation was observed closely up to 52 minutes, and the cell lineages were recorded for up to 70 minutes. Egg shell constraints were applied. The BIC locations are used, so assymetric divisions are possible. All other parameters were left unchanged from the default values.

8.3. Simulation Results

As mentioned already, the modules of the network can be simulated independent of each other. While creating the network each module was tested separately. After that, the network was simulated as a whole. The results are described in the following paragraphs, and an overview of the results is given in fig. 8.5.

8.3.1. Germ Line: PIE-1

A network containing only PIE-1 produces a worm where all cells are genetically identical except for P_1, then later P_2 and P_3. In these germ line precursors the simulation correctly generates PIE-1 activity while at the same time PIE-1 is inactive in all other cells. Due to constraints of the model, it is not yet possible to simulate the correct distribution of PIE-1 into the P_4 cell: the P_4 cell is created somewhat anterior of the D cell according to the published position dataset (Schnabel *et al.*, 1997). This results in PIE-1 — as a posteriorly located BIC — being distributed to D instead of P_4 upon division of P_3. A graphical representation of the simulated activity of PIE-1 during embryonic development is shown in figure 8.6 A.

8.3.2. Posterior Blastomeres

SKN-1 is active from the beginning. Upon division of P_1 it is equally distributed to both daughter cells. When the first state change occurs, it is switched off in P_2 (by PIE-1) while it remains active in EMS, thus determining EMS identity (see figure 8.6 A).

pal-1 mRNA is present in all P_1 descendants, but the protein is active only in C and its descendants, because this is the only place with neither SKN-1 nor PIE-1 activity. This experimental observation (Hunter and Kenyon, 1996) is accurately reproduced in our model (see figure 8.6 B).

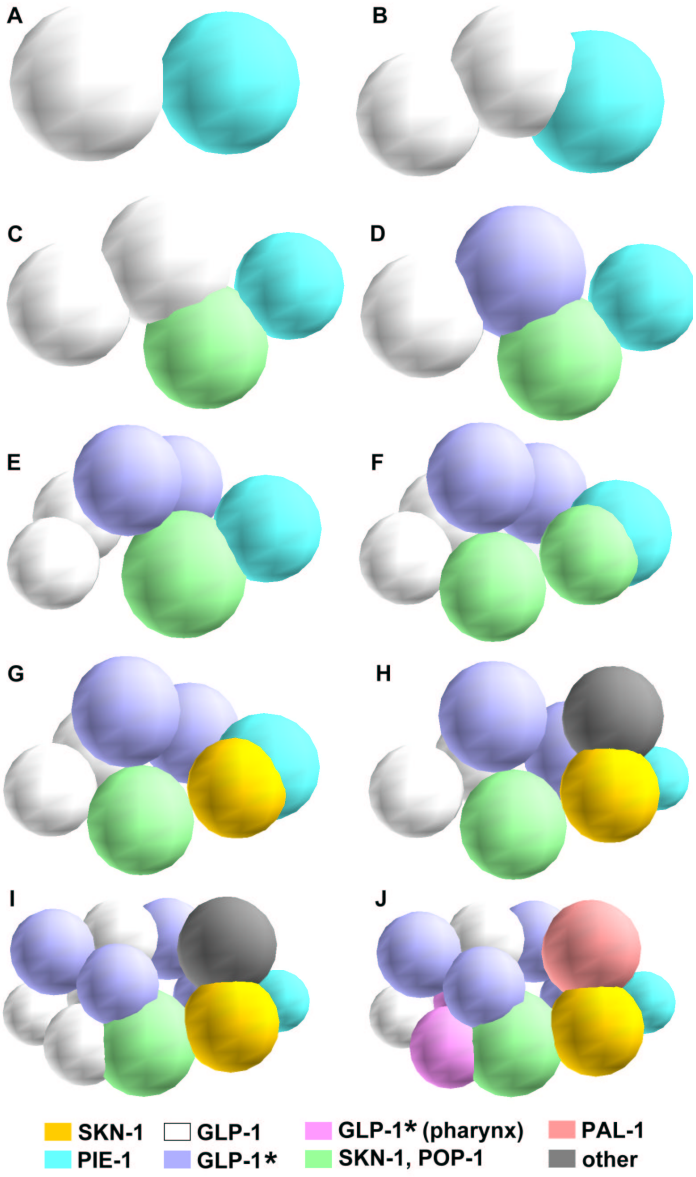

Figure 8.5.: Snapshots of simulation results. The screenshots from the simulation at different stages are shown. Colours indicate different protein activities. A comparison with figure 8.2 reveals a good agreement between the simulation results and real protein expression patterns, with the exception of the division of EMS (F and G). **A.** The beginning of the simulation, 0 minutes. **B.** After the division of AB, 17 minutes. **C.** Division of P_1, 18 minutes. **D.** After the first state change, 19.5 minutes. **E.** After the second state change (nothing happened), and after division of AB descendants, 35 minutes. **F.** After division of EMS, 39 minutes. Note that in the simulation, MS divides symmetrically, in contrast to reality (cf.figure 8.2 F,G) **G.** After the next state change, 40.5 minutes, the asymmetry of the two MS daughters is restored. **H.** After division of P2, 42.5 minutes. **I.** After division of AB descendants, 50 minutes. **J.** After the next state change, seven different cell types have been generated. End of the simulation at 51.5 minutes.

8.3.3. EMS Asymmetry

P_2 sends a signal to EMS to induce asymmetry in EMS. This results in an asymmetric division of EMS. Because our model does not support inhomogeneous cells, the signal cannot be sent before division of EMS. If it were sent beforehand, POP-1 would be switched off in the whole EMS cell, resulting in no POP-1 activity in either E or MS, which would contradict biological evidence (Thorpe *et al.*, 1997).

mom-1 mRNA is distributed to P_2, but not to EMS, because of its posterior location in the model. In P_2 it leads to activity of MOM-1 and thereby activity of MOM-2. Immediately after the division of EMS, P_2 is located in such a way that it contacts only E but not MS. This constellation results in an E-specific deactivation of POP-1 through the externally acting protein MOM-2, as observed in experiments (Thorpe *et al.*, 1997).

Since *pop-1* mRNA is not continuously active (no self-activating interaction) POP-1 does not get re-activated in E descendants when contact with a MOM-2 expressing cell is lost. Biologically, this effect may be attributed to the fact that in reality inactive POP-1 is degraded in E (Lin *et al.*, 1995). The simulated pattern of POP-1 activity is shown in figure 8.6 B.

8.3.4. ABp/ABa Asymmetry

Due to the spatial arrangement of the cells in the simulation and the time constraints on APX-1 activity there is only one contact between a GLP-1 expressing and an APX-1 expressing cell: between ABp and P_2. Therefore only ABp is induced and not ABa (see figure 8.6 A). GLP-1* remains active in ABp and prevents further GLP-1 activity. This ensures that ABp descendants do not react to the later signal from MS (see below). The induction of asymmetry between ABa and ABp is therefore reproduced correctly.

8.3.5. Pharynx Induction in AB Granddaughters

As the BICs responsible for sending a signal from MS to ABa granddaughters have not yet been detected experimentally, I have “invented” a Delta homologue. It is activated in MS by joint action of POP-1 and SKN-1. Delta successfully activates GLP-1* in ABalp and ABara (see figure 8.6 C). After this induction, ABara and ABalp are genetically identical with the ABp descendants that have been induced by the first Notch signal. A hitherto unknown mechanism is used in the embryo to interpret GLP-1* activity differently in these two groups of cells.

If the simulation is allowed to continue further (beyond the induction of ABara and ABalp), other ABa descendants are also induced because they come into contact with MS or MS descendants. This is in contrast to biological observations. Perhaps in the worm, later activity of the Delta homolog is prevented by timely degradation of the maternal mRNA coding for that protein, or by degradation of GLP-1, the receptor, in the target cells. This is another example where the simulation shows results that suggest further experimental studies to explain the effect observed *in silico*.

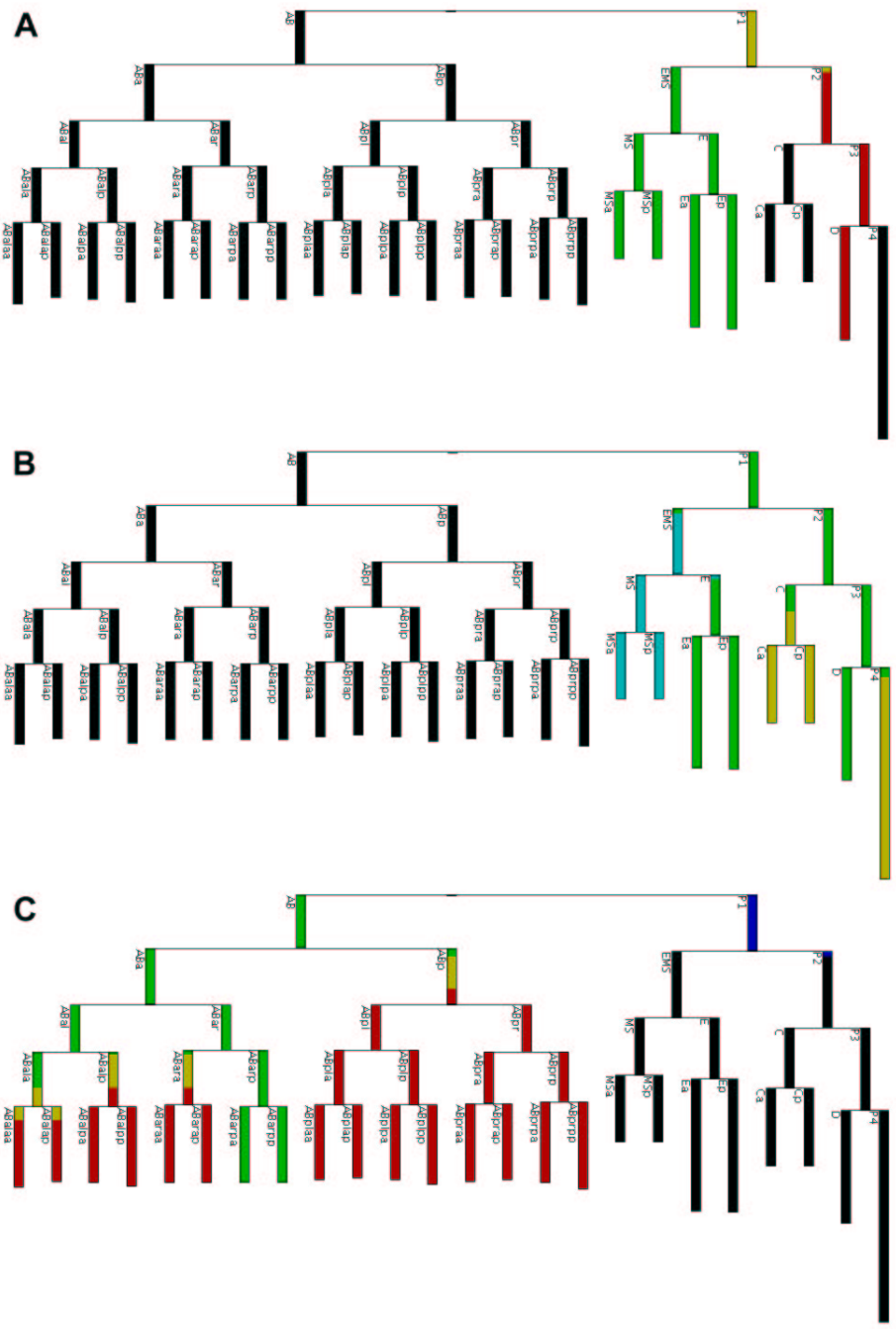

Figure 8.6.: Simulated expression patterns. Basic colours are red, green, and blue. Combination colours (yellow, turquoise) indicate combined expression of several proteins, as if the cells were stained with fluorescent dyes. Black cells do not contain any active BICs. The length of the bars represents the lifetime of the cells. The images are screenshots from the `Cell lineage` view of GENE-O-MATIC. **A.** Simulated expression of SKN-1 (green) and PIE-1 (red). **B.** Simulated activity resp. presence of POP-1 (blue), *pal-1* mRNA (green), and PAL-1 (red). **C.** APX-1 (blue), GLP-1 (green), and GLP-1* (red).

8.4. Summary

In general, the simulation results of the simplified genetic network for blastomere specification match experimental results quite well. The simulation runs completely analogous to real development up to the 22-cell stage. It thereby generates six different cell types (or seven, if the ABp descendants and the ABa descendants induced to produce pharynx are considered different). Via BICs with external range of activity (for example ligands) all inductive interactions occurring in the embryo up to that stage have been reconstructed. This leads to two statements: first, GENE-O-MATIC is obviously suitable for simulation of cell differentiation in multicellular organisms, and second, the genetic network used is correct in the sense that it is good enough to reproduce reality.

Some aspects of the modelling and simulation process are worthy of a more detailed discussion, which will be done in section 11.1.2.

9. Flower development in *Arabidopsis thaliana*

Some of the simulations of flower development in *Arabidopsis thaliana* were performed in cooperation with Aitor González, a postgraduate biologist. The purpose was to prove the applicability of GENE-O-MATIC under the following conditions:

- simulation of models where no cell positions are known
- for other organisms than *C. elegans*
- if the user is a biologist who is not a computer expert and who has no knowledge of the internals of GENE-O-MATIC.

Also, the application of the program by another person than the developer often leads to the exposure of bugs and of inconveniences in the graphical user interface and thus is an important means to test and improve software.

9.1. Biological Background

Flower development in *Arabidopsis thaliana* was chosen because it is a very well known process. Extensive experimental and some computational work has led to a well characterized genetic network controlling this process.

The aerial parts of higher plants develop from the apical meristem. This is a group of stem cells at the tip of the shoot. The apical meristem identity defines two main phases of plant life, a vegetative and a reproductive phase (Poethig, 1990). The apical meristem during vegetative phase is called shoot apical meristem (SAM) and is characterized by slow growth of the shoot and production of leaf primordia. The leafs are arranged in a rosette (see figure 9.1 A, B). At a certain point the SAM transforms into an inflorescence meristem (IM) and the reproductive phase starts. The IM is characterized by fast shoot growth and the production of inflorescence primordia (see figure 9.1 C, D). The inflorescence primordia give rise to twigs branching from the main shoot. At the top of each twig there is again an IM. Later arising inflorescence primordia develop into flower primordia with a floral meristem (FM) that gives rise to the floral organs (see figure 9.1 E). The transformation of the SAM into an IM is promoted by different environmental and endogenous signals (Howell, 1998b). Other endogenous signals later trigger the transformation of an IM into an FM. In contrast to the SAM and IM, the FM is determined and cannot grow indefinitely. Each FM gives rise to a single flower.

The model described here focuses on stem cell maintenance in the shoot apical meristem. Further simulation results are described in González (2002).

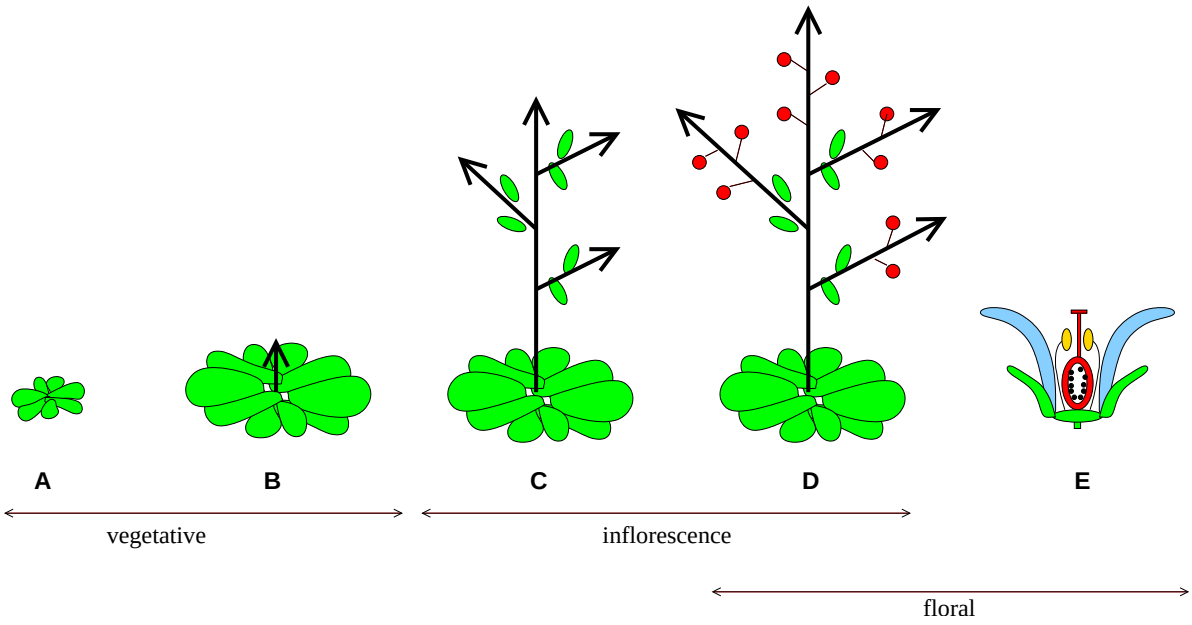

Figure 9.1.: Phase changes in *Arabidopsis thaliana* (adapted from Kieffer and Davies, 2001). **A, B.** During the vegetative stage leaf primordia are formed by the SAM, which then grow to form leafs. **C.** When the SAM transforms into an IM, shoot growth increases and inflorescence primordia are formed in addition to leaf primordia. **D, E.** Finally the IMs transform into FMs that are determinate and form flowers.

Cell	Description	Active Genes
L1	represents the *CLV3*-expressing region in the SAM.	*intCLV3*
L2, L3	dividing stem cells	
OC	the organising centre, region of *WUS* expression	*intWUS*, *WUS*
P1, P2	the region around the OC where *CLV1* is expressed	*intCLV1*, *CLV1*

Table 9.1.: Cells in the SAM maintenance simulation.

9.2. Stem Cell Maintenance in the SAM

The shoot apical meristem contains pluripotent stem cells that divide and are responsible for the growth of a plant. To maintain the meristem, the loss of stem cells due to organ differentiation at the sides and cell replenishment by stem cell division have to be tightly regulated. The structure of the shoot apical meristem is shown in figure 9.2. *WUS* (*WUSCHEL*) is expressed below the tip in a group of cells called the organising centre (OC). It activates *CLV3* (*CLAVATA 3*) expression in the overlying cells. This signal specifies them as stem cells. CLV3 is secreted into the extracellular space (Fletcher *et al.*, 1999) and diffuses downward. It has a negative influence on *WUS* activity (Schoof *et al.*, 2000). Stem cells divide continuously so that there is a flow of cells and CLV3 through the organizing centre. Therefore there has to be a mechanism that inactivates CLV3 before the cells enter the OC, because otherwise CLV3 would inhibit *WUS* there and thus destroy the OC. Cells in and around the OC express *CLV1* (*CLAVATA 1*) on their surface. CLV1 has been proposed to form a receptor complex together with CLV2 (CLAVATA 3) which can bind and inactivate CLV3 (Trotochaud *et al.*, 1999). On its way towards the OC, CLV3 is bound by excess CLV1 and thus prevented from inactivating *WUS*. The negative feedback loop between *WUS* and *CLV3* is one of the main mechanisms for maintaining the correct cell number in the SAM (Haecker and Laux, 2001; Sharma and Fletcher, 2002).

The genetic network that was used to simulate this process is shown in figure 9.3. *WUS*'s and *CLV3*'s ability to move between cells is represented by making them external BICs which can influence BICs in neighbouring cells. A drawback of the modelling environment is that it does not support long range signals as would be required here to model WUS and CLV3 diffusion appropriately. The "*int*" BICs (with internal range of activity) intCLV1, intCLV3, and intWUS, and their activating influence on CLV1, CLV3, and WUS, respectively, make sure the (externally acting) BICs CLV1, CLV3, and WUS remain constantly active (the normal self-activating interaction mechanism cannot be used for external BICs). The activating interaction from WUS to CLV3 was omitted from the model and instead intCLV3 was set to "active" in the L1 cell which represents the whole area of *CLV3* expression. CLV2 does not have a function in the model, it is present only to indicate the receptor complex. Table 9.1 shows the initial cells and their genetic states.

Cell positions and division information could not be taken from a database because there is no such database for *Arabidopsis*. Therefore initial cell positions were defined such that the cell arrangement represents the position of different tissues in the SAM. Cell division is regulated by the genetic network, the Boolean formula defining the state is `DIV`, so cells divide whenever DIV becomes active. L2 and L3 are the only cells that will divide; in all other cells,

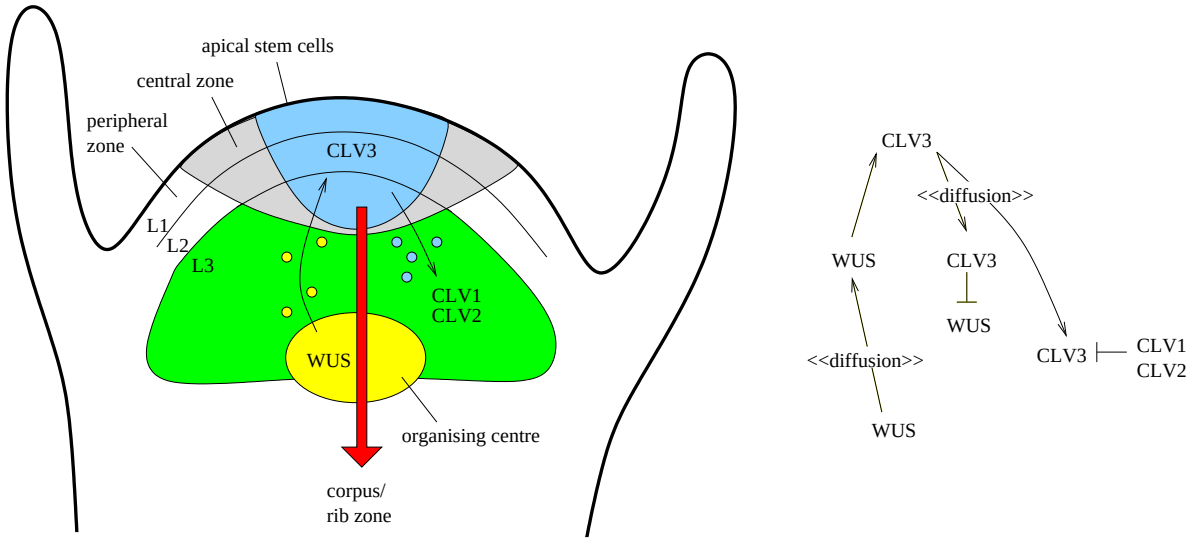

Figure 9.2.: Regulation of the maintenance of the SAM (after Haecker and Laux, 2001; Sharma and Fletcher, 2002). The apical stem cells in the central zone of the SAM express CLV3 (blue). This is induced by WUS which moves from the organising centre (OC, yellow) upwards. CLV3 itself diffuses downwards and represses WUS expression in L3. Because WUS is required in the OC, CLV3 is caught by binding to the CLV1–CLV2 receptor complex which is expressed in the L3 area (green) around the OC. Bound CLV3 cannot inactivate WUS any more. The fast-dividing cells in the shoot apex move towards the rib zone (red arrow) through the OC. OC identity and position is probably maintained by signals from surrounding cells at the periphery (not shown).

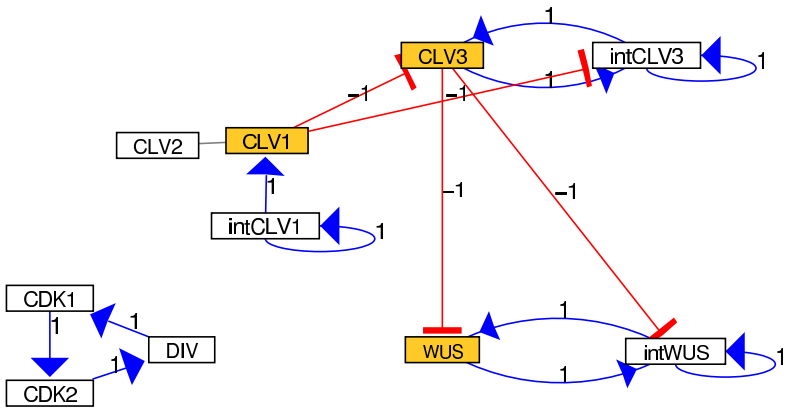

Figure 9.3.: Genetic network for the simulation of SAM maintenance.

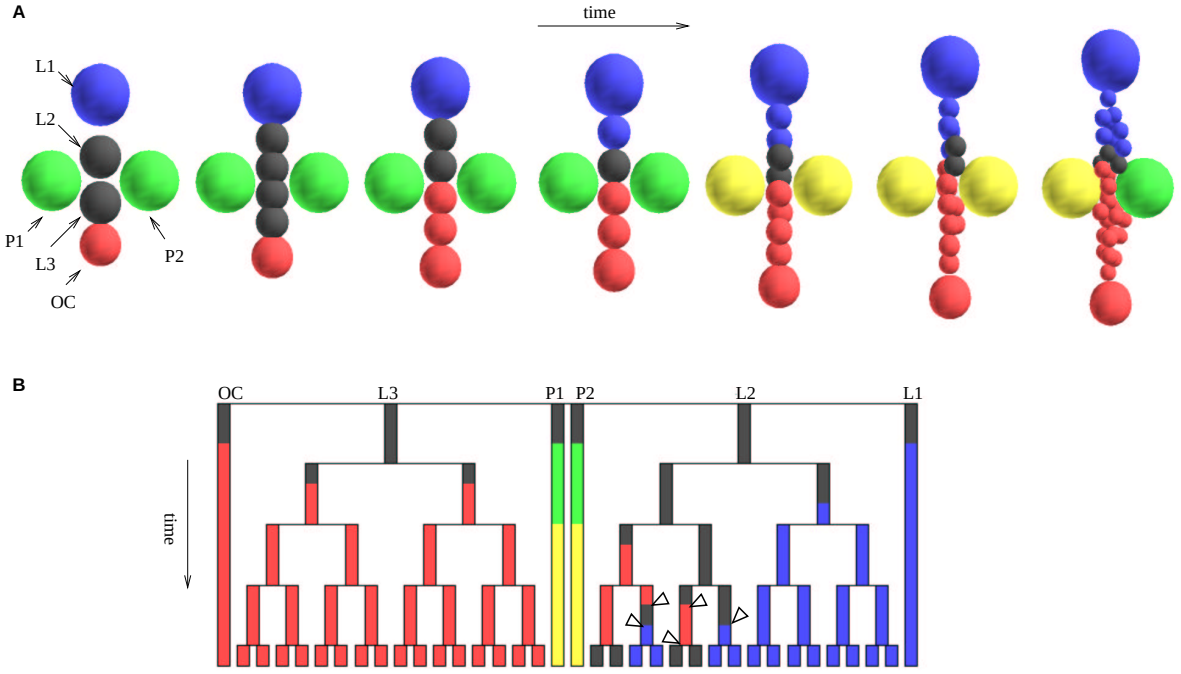

Figure 9.4.: Simulation results for SAM maintenance. **A.** 3D view. Stem cells (L2 and L3) divide rapidly, and the new cells move toward the organizing centre where *WUS* is expressed (red). CLV3 (blue) is inactivated in these cells through the influence of CLV1 which is expressed in the neighbouring cells (green). Yellow indicates joint expression of *WUS* and *CLV1*. In the grey cells, none of the BICs is active. **B.** Cell lineage. Arrowheads in the cell lineage indicate spontaneous (i.e. non-clonal) expression or non-expression of WUS or CLV3.

no element of the "cell cycle" (CDK1, CDK2, DIV) is active. The initial division direction for L2 and L3 was set to be vertical, along the y axis to mimic vertical growth of the meristem.

The results of the simulation are shown in figure 9.4. In the cell lineage drawing it can be seen that cell fate is not dependent on clonal origin, but on the surroundings: In some sublineages, *WUS* or *CLV3* expression starts or ends abruptly (arrowheads in figure 9.4 B). In figure 9.4 A one can see that CLV1 prevents CLV3 activity in the cells adjacent to P1 and P2. Also, the negative effect of *CLV3* on *WUS* is apparent as the central cells adjacent to *CLV3*-expressing cells do not display *WUS* activity.

9.3. Conclusions

The concordance of experimental and simulated results demonstrates that the genetic network and the cell arrangement used in this simulation is an adequate representation of reality.

The simulations performed in *Arabidopsis thaliana*, the one described here as well as the others described by González (2002) have shown that GENE-O-MATIC can be used to model

reality also under difficult conditions:

- where no cell positions are known, but only a rough approximation is possible
- if the user has no prior experience in the usage of such programs

Some problems do exist, though. As there is no way to model long-range signals or the diffusion of molecules between cells, it is difficult to create a working model. And some such interactions, in the example described here the activation of *CLV3* by WUS, had to be left out of the simulation so that they not disturb the results or lead to experimentally unobserved phenomena. Therefore, an extension of GENE-O-MATIC that offers the possibility of free and/or regulated diffusion between cells would be highly desirable. A model for diffusion within single cells has already been implemented by a diploma student under my guidance (Billewitz, 2003), and it is planned to extend it to inter-cellular diffusion in the near future.

During the setup of the *Arabidopsis thaliana* simulations it became obvious that though the program can in principle be used, there are several ways to further enhance the convenience. One of these, which was duly implemented, is the genetic state table that allows to monitor the states of all BICs in all cells simultaneously, not as a (sometimes difficult to interpret) colour pattern, but in a table. Another example is the feature that the user can trigger cell divisions during the simulation by clicking on the cells in the 3D display. This allows for a quick determination of the correct division time, which can then later be programmed into the simulation by means of cell cycle genes.

Part IV.

Discussion

In this part of the thesis, the strengths and weaknesses of the approach taken in GENE-O-MATIC are described. It is compared to other modelling and simulation software, and the results achieved with GENE-O-MATIC are be analysed regarding their value for the scientific community. Finally, a short chapter summarises the contents.

10. GENE-O-MATIC as a Network Database

The network editor of GENE-O-MATIC is in itself a useful tool. It allows to collect data about genes, proteins, and interactions and represents them graphically. Though there are many gene and protein databases available on the internet (DIP, 2003; Apweiler, 2003; WormBase, 2000), very few of them offer graphical representations of the interactions the molecules are involved in. And even then, this is often a static picture which cannot be modified by the user (KEGG, 2003). The databases often contain only a very small part of all available information, because data is gathered by high-throughput techniques faster than it can be entered into curated databases. Therefore many experimentalists to whom GENE-O-MATIC was shown on conferences were happy to see a tool that will allow them to create and maintain their own database.

The advantages of having a graphical, visual representation rather than a text-based database, are manifold:

- It is much easier to get a good overview over a process if it is represented as a picture: *a picture is worth a thousand words*.
- Missing elements, be it interactions or BICs, become apparent simply because there is a gap in the drawing.
- It is much easier to compare two regulatory networks visually than comparing their textual descriptions, especially if similar genes in different organisms have totally different names. Therefore, a graphical display is an aid in the identification of analogous pathways in different or related organisms, or in different processes.
- It can help with the identification of developmental or regulatory modules: genes that belong to one module will probably be closer together in the drawing than those which are involved in distinct processes.

To strengthen this aspect of GENE-O-MATIC, a direct link to the internet database WORMBASE (WormBase, 2000) has been implemented, allowing quick access to more information on worm genes. In the future it is planned to further enhance this functionality and make possible access to databases for other organisms, too. In addition, it is planned to implement automated network search algorithms that can collect gene and protein interaction data for a given organism directly from the internet. This will greatly facilitate and accelerate the process of network creation.

11. Simulation Methodology

This chapter discusses pros and cons of the simulation methods in GENE-O-MATIC.

11.1. Tissue Simulation

11.1.1. Interweaving of Network and Cell Simulation

One of the main features that distinguishes GENE-O-MATIC from other genetic network simulation and analysis software is the integration with three-dimensional cell simulation. Each cell maintains its own genetic state independently of the other cells, although they all use the same genetic network. This represents the fact that all cells in an organism contain the same DNA, but may exhibit different gene expression patterns. The presence of multiple cells alone is not really new: computer scientist have long known the concept of cellular automata. The following definition was taken from an online dictionary (Webnox Corp., 2000-2003):

> [A cellular automaton] is a regular spatial lattice of "cells", each of which can have any one of a finite number of states. The state of all cells in the lattice are updated simultaneously and the state of the entire lattice advances in discrete time steps. The state of each cell in the lattice is updated according to a local rule which may depend on the state of the cell and its neighbours at the previous time step.
>
> Each cell in a cellular automaton could be considered to be a finite state machine which takes its neighbours' states as input and outputs its own state.

In GENE-O-MATIC, the rules according to which the states of the cells are updated are given by the genetic network. The difference between GENE-O-MATIC and a cellular automaton is that in GENE-O-MATIC cells are not arranged on a regular spatial lattice. And GENE-O-MATIC's cells can move and multiply.

This is also the main difference to those other genetic network simulation tools that offer multicellular simulation: their cells do neither move nor divide. As the aim of GENE-O-MATIC was the simulation of developmental processes, the ability to simulate divisions, which are an inherent part of every developmental process, was crucial to the success of the project.

Inter-cellular interactions are very important for cell differentiation because cell-autonomous mechanisms alone are not powerful enough to direct the formation of many distinct cell types. Environmental conditions are most important for the specification of cell fates, and many cell determination events in *C. elegans* as well as in *Arabidopsis* could not take place if there was no inter-cellular signalling. GENE-O-MATIC is not the first nor the only program that can simulate cell differentiation, but it is definitely the first that combines a flexible cell

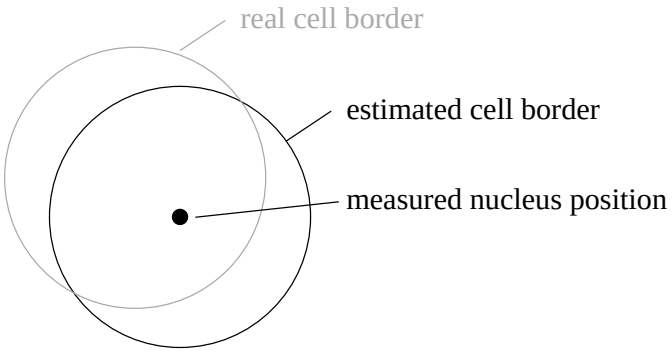

Figure 11.1.: Difference between estimated (black) and real (grey) cell position in *C. elegans* if the nucleus is not in the cell centre.

model including cell movement and division with the simulation of genetic networks to create a tool for the simulation of network-directed cell-cell interactions in developmental processes such as cell differentiation.

11.1.2. Cell Shape and Position

Compared to other cell and tissue models, GENE-O-MATIC's cell model lacks one important feature: a detailed representation of the cell shape or surface is not possible. Instead, cells are approximated as spheres around a central point. Voronoi cells, which at first sight seem to be a better approximation of the cell shape, do not solve this problem, as the Voronoi cell border between two cells is identical with the plane defined by the intersection of the two spheres by which the cells are approximated (cf. figure 5.5 on page 21). As for the determination of the neighbours of a cell only the connectivity between the cells is important — *if* they have a common border — and not their shape, the lack of a detailed cell shape representation does not hinder a successful simulation.

For the simulation of the early embryo of *C. elegans*, this problem is further aggravated by the fact that the assumption that the measured positions of the cell nuclei correspond to the centre of the cell is not always true. Often the nucleus can be dislocated to one side of the cell. In that case, the spherical, and also the Voronoi approximation of the cell will appear shifted compared to the real cell position (see figure 11.1). The reason for this inaccurate description of cell positions in measured data is that the actual location of the cell membranes, which would be needed for precise determination of cell neighbourhood relations, cannot be recorded easily because the exact membrane position in the images would have to be marked manually — automatic detection by the computer is not even possible for the position of the nuclei which are characterised by prominent grey circles in the images. Already the manual marking of the nuclei, where only a single spot has to be marked for each nucleus in the developing worm, takes ages.

It is probably due to these drawbacks that the simulation of blastomere fate specification in *C. elegans* shows a severe sensitivity to changes in state change times because some cell-cell contacts are established only for very short periods of time. For the simulation to produce the

correct results it is therefore very important to tune the times of state changes to the recorded cell positions, i.e. a state change has to occur exactly when the cells do make the correct contacts. An example for such a problematic induction is the Notch signal sent from MS to exactly two ABa descendants. The induction takes place at approximately 50 minutes of simulated embryonic development. Around that time, not only ABalp and ABara, but also ABarp are very close to MS, and there is only a short time span where MS is not in contact with ABarp. That contact would result in inappropriate activation of GLP-1* in ABarp. Since it is highly unlikely that worm development relies on such a narrow time frame, this problematic effect is likely an artefact resulting from imprecise cell position measurements or, more probably, incorrectly calculated cell shapes: the assumption that the nucleus is in the cell centre is wrong.

The problem is restricted to simulations that use recorded cell positions with very specific inductive interactions between the cells. For more general simulations of whole tissues (collections of cells) this becomes irrelevant. This is demonstrated satisfactorily in the simulation of *Arabidopsis thaliana* flower development, where whole tissues contact each other and inductions do not depend on specific cell-cell interactions, and also in simulations of vulval development in *C. elegans*, which were performed by Rolf Lohaus, a Diploma student, under my guidance (Lohaus, 2003).

11.1.3. Localisation of Intracellular Compounds

GENE-O-MATIC is not yet capable of a detailed representation of inhomogeneous molecule distributions within single cells, such as the *C. elegans* zygote after polarity establishment. The asymmetric distribution of a compound can only be specified generally via the BIC's preferred location, but cannot be modelled on a per-cell basis. And BICs cannot influence other BICs' locations. There are two biological processes where this problem occurs:

- Molecules move by themselves into one half of the cell, or they are fixed or fix themselves to specific locations in the cell (cell-autonomous).
- Induction by a signal (external or internal) affects only a spatially distinct population of molecules in the target cell.

Some workarounds employed to compensate for the inadequacy of the modelling environment are listed here. To model the initial asymmetry in the worm zygote, the simulation had to be started with two different cells (AB and P_1) instead of one single cell with an inhomogeneous distribution of cell contents (P_0). Another example is the induction of an asymmetry in EMS and the following asymmetric division; in the simulation these two events occur in reverse order compared to the real embryo to achieve the desired result.

An extension of GENE-O-MATIC with a more detailed representation of the cell interior and a direction for cell polarity has been implemented (Billewitz *et al.*, 2002; Billewitz, 2003), however it does not yet provide cell-cell interactions and thus was not used in this thesis.

Another aspect also due to inadequate representation of intra-cellular asymmetry is the mislocalisation of PIE-1 at the division of P_3 in the worm simulation. In the embryo, the divisions of P_2 and P_3 are controlled via a different mechanism than those of P_0 and P_1 (Guo and

Kemphues, 1996a). This leads to a polarity reversal and changed division directions, and thus to the fact that the germ line precursor P_4 is generated as the anterior daughter of P_3. Because the model used for describing the BICs in the computer does not allow polarity reversals, i.e. changes in the preferred location of BICs, PIE-1 — having the location "posterior" — is mislocated to the D cell. As a result, PAL-1 activity does not occur in D, but in P_4 instead, i.e. the two cells switched their fate.

The concept of a mitotic spindle controlling the cell division direction, and that the orientation of the spindle can be influenced by the genetic network, is already in the queue, but could not yet be implemented due to time constraints.

11.1.4. Long-Range Signals and Morphogen Gradients

GENE-O-MATIC does not support long-range signals. It assumes that signals are propagated from one cell to the neighbours. Therefore, neither diffusion of molecules through the extracellular space (where they can also reach non-neighbouring cells) nor the establishment of a gradient of a substance throughout the whole organism can be modelled. This is a disadvantage, as in many multicellular organisms, long-range signals play an important role in the regulation of cell division and other cellular processes. Whole tissues are regulated that way, for example by hormones and growth factors. Because of this drawback, classical developmental model organisms such as *Drosophila* (Struhl, 1989) and hydra (Meinhardt and Gierer, 1974) cannot be modelled with GENE-O-MATIC. The *Drosophila* embryo develops as a syncytium, and only after more than ten cell divisions cell walls are made. The cell types are specified by morphogen gradients, which are initialized by asymmetries in the oocyte and stabilized and expanded in the syncytium. In hydra, morphogen gradients form over the whole organism in a self-emerging pattern starting from small asymmetries.

Even for cellular models, long-range signals cannot be modelled. Aitor González, who created most of the *Arabidopsis* simulations, complained that the diffusion of molecules from one cell into the surrounding tissue or extra-cellular matrix (ECM) cannot be modelled. The concept of plasmodesmata, channels that link plant cells and allow the movement of larger molecules such as proteins between the cells, is not available in GENE-O-MATIC, either. It is clear that this is a major downside of the cell-based modelling concept in GENE-O-MATIC. As flexibility and independence from specific model organisms are among the main goals of GENE-O-MATIC, the implementation of an ECM-like structure surrounding the cells is one of the priorities in future development of GENE-O-MATIC. However, because different processes like embryonic development of *C. elegans*, vulval development of *C. elegans*, and flower development in *Arabidopsis* could indeed by simulated realistically in spite of missing long-range signals, GENE-O-MATIC is already much more flexible than most other multicellular modelling tools.

11.2. Network Simulation

There are two main approaches to the simulation of biological networks: qualitative and quantitative. GENE-O-MATIC makes use of the qualitative approach. The question "Which is the

better method?" is an ongoing reason for discussion and has not yet been — and in my opinion cannot be — answered satisfactorily. It always depends on the requirements and conditions under which a simulation is to be performed. The computational model has to be tuned to the biological circumstances. If nature allows to measure lots of parameters, concentrations, rate constants, etc., there is no reason why one should not use a quantitative model, if every parameter in the model is backed by some measured experimental value. In contrast, it does not make sense to apply a quantitative model if there is no data on which it could be based. In that case, the fewer parameters a model requires to generate realistic results, the better, because then fewer values have to be guessed or even chosen arbitrarily.

For processes like transcriptional regulation, quantitative, especially continuous models do not provide a big advantage compared to qualitative models. The reason is that transcription is really a Boolean process: either a gene is transcribed, or it is not. The "most quantitative" case, the diplont, has two copies of gene, so there are three possible states: zero, one, or two transcribed genes. But this does not really sanctify the use of a quantitative model.

The same is true for cell differentiation. No matter what the gene expression patterns within a cell are, the cell either adopts a specific type or it does not (i.e. it adopts another type). There are no cells which are half muscle and half neuron. Of course, there are so-called "weak" mutations that interfere partially with the formation of a specific type of cells, so that the organism has fewer or more cells of that type than would be normal. But this does not change the fact that each of the missing resp. superfluous cells definitely belongs to a specific cell type.

As mentioned already, it may be difficult to acquire the amount of data necessary for a plausible quantitative model. In the *C. elegans* embryo, this is indeed the case. The techniques most often used, and on whose results the simulation described here is based, are purely qualitative: immunofluorescence or GFP-tagging, and microscopic observations. Cell differentiation in development is never a process well suited for mass acquisition of data: microarrays are useless because differences in the expression patterns from single cells need to be assessed.

Therefore GENE-O-MATIC was initially designed for qualitative simulations only. Since is was obvious from the beginning, however, that sooner or later one would like to do quantitative simulations as well, the software was designed for a high degree of flexibility and extensibility that will allow the later addition of quantitative calculators.

11.3. *In Silico* Experiments

Computer simulation becomes a simple and cheap alternative to "real" experiments. Especially in the area of metabolic networks, computational models of metabolic pathways help biologists to design organisms which will be better suited to solve a specific task, for example yeasts for beer brewing or bacteria for the production of yogurt (see Wiechert (2002) for a review). The effect of mutations or overexpression of genes can be tested much quicker and cheaper than would be possible in an *in vitro* or worse, *in vivo* experiment. The reason why *in silico* experiments are much more popular for metabolic than for developmental regulation is that it is easy to measure reaction rates and substrate concentrations *in vitro*, therefore much more and detailed data is available than for developmental processes. Thus the models are much

more accurate and the predictions made with their help are more reliable. Also, biochemists are used to kinetic, quantitative models, while geneticists and developmental biologists are not (yet).

This picture is changing, however. Recently, more developmental models and simulations are being published. GENE-O-MATIC is one of the tools available for that. It offers several experimental techniques which are also available for *in vivo* and *in vitro* experiments:

Knockouts. Without modifying the original network structure ("wildtype", the reference network) knockout mutants can be analyzed quickly. This has been done for several genes in the blastomere fate specification network, and the simulation results show a significant concordance with the corresponding *in vivo* knockouts, especially if considering the fact that cell positions and divisions in the simulation cannot change in the *in silico* mutants as they would in the *in vivo* mutant because they are defined in the database.

Overexpression or ectopic expression of genes can be simulated by setting genes to be active from the beginning, and by providing self-activating interactions which will prevent them from being switched off.

Cell ablation is possible for those simulations which are not based on a cell position database. Cells can be ablated from the beginning, or later during the simulation. Thus specific inductive interactions can be prevented and the resulting alterations in development be observed. This has been tested successfully in simulations of vulval development (Lohaus, 2003).

All these methods are independent of the specific organism or system which is modelled and simulated. GENE-O-MATIC — in contrast to many simulation programs which are written only for a specific application — offers high flexibility in the modelling environment and is applicable to all organisms with cellular development. It is capable of simulating well-defined developmental processes even in large organisms, if only specific parts, such as single organs, of the organism are considered.

GENE-O-MATIC therefore provides a choice of experimental techniques that are also used by biologists in *in vitro* experiments. They do not have to accustom themselves to new methods. GENE-O-MATIC fulfils the requirements to become a powerful tool for *in silico* experiments, aiding experimentalists in the design of new *in vitro* and *in vivo* experiments.

Mutant analysis can also help to support or disprove hypotheses on how regulatory mechanisms function: if the *in silico* mutant behaves like the *in vivo* mutant, this adds evidence to the validity of the network, while a mismatch in experimental and simulated result of a mutation indicates inconsistencies in the model.

12. Benefits of Computational Analysis and Simulation

Computer simulation in biology suffers from the same phenomenon as any other predictive tool, or any clairvoyant: if something is foretold which is already known, nobody will be impressed because everybody says “but we knew that already”, and if something is predicted which is yet unknown, nobody will be impressed, either, because “we do not know if it really is true”. Thankfully, in computational biology some people are impressed even if nothing new was achieved, but only existing, known results reproduced.

After a short overview of the model-building process, I will therefore discuss the value of GENE-O-MATIC as a tool for the analysis of existing data. Then I will show what additional information can be gained by computer simulation which would not be available from experimental data alone, and the advantages this may bring for researchers. This will include a discussion of the results obtained in the worm simulation.

12.1. Building the Model

The modelling process itself often is the first step to a better understanding of a biological phenomenon. It promotes a detailed understanding of each single element of the model. In creating the model for *C. elegans* blastomere fate specification, I started with a single protein, PIE-1, and gradually added the other proteins and mRNAs, going back a step whenever the results did not match experimental observations. Which molecules can be left out, because they would unnecessarily complicate the model without bringing real new information? What is the best way to model protein complex formation or proteolytic cleavage that transforms one protein into another with a different function?

Though the possibility to get answers to these questions looks promising, it must be said that building a model is a complicated procedure that already requires a lot of knowledge from the start. One cannot simply copy a drawing somebody else has made of the process in question. Often, genes need to be grouped into one, and other genes need to be left out, to get correct simulation results. Some improvement concerning these issues are planned for the future (see chapter 13), but until then, the user will have to make up for the inadequacies of the modelling environment by means of creative thinking.

The disadvantage of having to put a lot of knowledge into model-building is that there is probably not much left to be discovered if enough data is available to build a working model. This will be discussed in detail in section 12.3.

12.2. Analysis of Existing Data

What can be gained from a simulation that merely reproduces reality? If a model has been set up based on experimental data, and if the simulation of that model produces results which are consistent with the data, this is an affirmation of the correctness of the model. Unfortunately, simulation can never produce proof — there may be different possibilities to achieve the given results which are all consistent with the data, and there is no way to tell which one is the one that Nature took to solve the problem. Nonetheless, a working, functional model gives a strong hint that the current model really is the correct one, because otherwise most probably inconsistencies or errors would have occurred.

To be able to make predictions, first, a realistic[1] model has to be created, thus a simulation that reproduces reality is the first step towards a simulation that makes predictions.

If it is not possible to set up a realistic simulation, this is valuable information, too: it tells us that some crucial element of the model must be missing or that the model is otherwise erroneous. The specific difference between simulation and experimental results may also give hints as to where the error may be, and what experiments need to be performed to collect missing data. Instead of being in the dark, experimentalists can direct their research efforts to promising targets in the first place.

12.3. Predictive Power

The predictive power of simulation results may be smaller than expected because a lot of *a priori* knowledge is required to build a model. Nonetheless, many interesting new insights can be gained by a simulation.

Simulation enables scientists to quickly try out different hypotheses about a process yet unknown and enables them to make predictions about the underlying biological mechanism. Construction of a model, simulation of its behaviour, and comparison of the simulation results with experimental observations allows confirmation of the existing data. If the simulation results equal the experimental results, one can conclude that everything — or at least enough to grasp the basic mechanism — about that process is known. If the simulation gives other results, however, one may assume that some elements crucial to the investigated process are misplaced in the model or even still missing.

In the simulations described here, I succeeded in reproducing observed results even though the molecules involved are still unknown: for the induction of two ABa descendants to become pharyngeal precursors, the signalling protein and the way it is activated in MS is still unknown, but nonetheless this can be simulated with the model. This clearly demonstrates that GENE-O-MATIC can indeed be used not only to recapitulate long-known facts but also to make predictions about unknown or only partly known processes.

Even for mechanisms that seem to be understood completely, this approach can give new insights. One question arising from the simulation is whether SKN-1 is active in C, when after division of P_2 PIE-1 is not present any more. There is no biological data supporting the

[1] Of course all models should be based on experimental data, and not be collections of purely fictitious BICs and interactions.

simulation results (where no SKN-1 activity occurs in C), but there is no data disproving it, either. Bowerman and colleagues reported the presence of SKN-1 in all P_1-descendants in the 8-cell-stage (Bowerman *et al.*, 1992a), however nothing is said about the activity of SKN-1 there. All the same, the effect of recurring activity in C can easily be simulated by introducing another BIC representing continuously active *skn-1* mRNA. In this case, PAL-1 is deactivated in C after a short period of time, which would *in vivo* probably interfere with C's specification as a muscle precursor. So the prediction derived from this simulation is that SKN-1 will not become active in C. This is a good example of how a computer model may be used to suggest experiments biologists could make. If experimental data were available here, more precise statements about the underlying molecular mechanism could be made.

12.4. Robustness

Robustness is often named as one of the prominent features of biological regulatory networks. The regulation works even if the concentrations of involved molecules are modified 10-, 100-, or even 1000-fold. A robust model is more probably a correct description of the biological process than one that is not robust, because most biological processes work reliably in different and difficult conditions.

Because GENE-O-MATIC models do not contain molecule concentrations, the application of the term is not fitting in this case. However, the question remains whether good simulation results are an inherent property of a model (which could then be termed "robust"), or whether they are a result of chance.

Using the Genetic Network Analyzer (de Jong *et al.*, 2003) it has been shown that the regulatory network used in the blastomere fate specification model has a specific set of stable states. These can be divided into two groups: those which have biological equivalents and correspond to determined blastomeres, and those which do not have equivalents in worm embryology (Ute Platzer and Hidde de Jong, unpublished data). It turned out that those stable states with no biological counterpart result from initial states which do not occur in the worm embryo under normal conditions, for example GLP-1 and PIE-1 activity together in the same cell. Therefore it can be said that the ability for the generation of the embryonic founder cells is an inherent property of the network, i.e. that the network proposed in this thesis for blastomere fate specification in *C. elegans* is robust to changes in the calculation or simulation technique.

13. Future Prospects

While GENE-O-MATIC already is a valuable and powerful tool, it can still be improved in many ways. Cooperation with experimental biologists, e.g. Ricardo B. Azevedo of the University of Houston, Texas, and Aitor González, who worked in our group at the DKFZ, has resulted in many suggestions. In addition, comments from the audience of presentations on various conferences (9th and 11th International Conference on Intelligent Systems in Molecular Biology, 2001 and 2003; 12th and 13th International *C. elegans* Meetings, 2001 and 2003; 5th German Workshop on Artificial Life, 2002; International Conference on Systems Biology, 2002; 2nd and rd Workshop on Computation of Biochemical Pathways and Genetic Networks, 2001 and 2003) have confirmed the fact that further development of GENE-O-MATIC will lead to valuable improvements and aid in reaching our ultimate goal: to make GENE-O-MATIC a universal tool for biological simulation.

The following sections will describe enhancements and extensions to GENE-O-MATIC which are planned for the future.

13.1. Enhancement of the Network Editor

As mentioned earlier, creating a genetic network from scratch is a tedious, complicated process. To support the user in this, it is planned to add several mechanisms that will provide initial cues as to how the network for a specific process and organism will look like. The first step towards this has already been done by providing the database area which contains all genes important for the process. Unfortunately, at the moment the user has to create the database himself. The following sections describe possible approaches to support the user in this.

13.1.1. Data Collection

There are several bioinformatics techniques available to support the automated construction of a genetic network.

Text Mining Text mining is a method to extract relevant information directly from the literature. By means of keyword search and by applying rudimentary grammatical rules to interpret the text, it is possible to gather information about interactions between molecules from the literature (see for example Friedman *et al.*, 2001). Currently, text mining is not advanced enough to provide a guarantee that the detected information is correct. Humans must verify the information first. Nonetheless text mining would simplify the process of data acquisition, as it is much simpler to verify a specific information than to

find it in the first place. Therefore it is planned to implement a function in GENE-O-MATIC to initialize a genetic network based on text mining of relevant literature. The relevant literature could also be found automatically by querying the PubMed literature database (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed). In addition to simplifying model building, this GENE-O-MATIC extension will also increase the acceptance of text mining as an information-gathering method in biology, because results can be analysed visually.

Internet Databases To create a new genetic network, to verify an existing one, or to extend it, automated search and query methods for internet databases could be a valuable aid. A first step in this direction was the implementation of a link to WormBase (WormBase, 2000) which allows to display all information available in WormBase for a specific gene or a specific cell in a browser directly from within GENE-O-MATIC. This functionality will be extended to enable access to other databases for other organisms, and also to access interaction databases such as KEGG (KEGG, 2003) or DIP (DIP, 2003). This will allow quick addition of new interactions which the user had not yet included into his model, and will help to verify the existing interactions.

A long-term goal is to offer a web-based version of GENE-O-MATIC to enable biologists worldwide to build models. Also, the genetic network editor could be integrated with protein and gene databases directly to display regulatory interactions graphically.

13.1.2. Integration of Experimental Data

A major challenge is the quick and easy generation of genetic networks from experimental data. This is often called “reverse engineering”. It would increase the attractiveness of GENE-O-MATIC for experimentalists for whom it would be made more clear that it can help them with the analysis of their own experimental data. GENE-O-MATIC would then be accepted as a regular laboratory tool and not some exotic software that is only used by theoreticians. Currently, one extension towards this goal is in concrete planning (integrate data from genetic experiments), while the other (usage of microarray data) lies in the distant future.

Genetic Experiments GENEPATH (Zupan *et al.*, 2003b) is a program that assists in the construction of genetic pathways from experimental genetic data. It mimics the geneticist by applying a set of patterns that search for specific gene-to-gene or gene-to-biological process relations. It would be very useful if it was possible to import a genetic network created with GENEPATH into GENE-O-MATIC and to simulate it immediately. Our group has already established contact with Blaz Zupan from the GENEPATH project and negotiated an XML-based data exchange format. In the near future, GENE-O-MATIC will be extended to be able to load GENEPATH data.

Microarrays In the introduction to this thesis, I stated that

> [...] the fully automated elucidation of regulatory interactions from experimental data is not possible yet, and the significance of the interpretation of large datasets by computational methods such as clustering is arguable.

It is all the more important to support these automated methods with additional tools, because the steadily increasing flood of experimental data needs to be converted to useful information. A network resulting from a cluster analysis, for example, could be simulated with GENE-O-MATIC to confirm its correctness, or to demonstrate its invalidity. The network initialized by automatic mechanisms can be further refined by the scientist to create a working model of the process in question. Therefore, it will surely be a worthwhile undertaking to couple GENE-O-MATIC to existing microarray analysis tools to initialise the networks.

13.2. Evolution of GENE-O-MATIC

Recently, the analysis of genetic networks for the discovery of evolutionary relationships between organisms has been an upcoming topic. The comparison of network structures can lead to new insights into evolutionary processes. This could be done manually by looking at the graphical representations, or computationally by graph-theoretic approaches. Also, by means of comparing simulation results of networks from different organisms it may be possible to detect elements in the networks which are under strong evolutionary pressure, and those which are more flexible. GENE-O-MATIC could be used to retrace evolution: given the genetic networks of a specific process in two related organisms, it could be studied which modifications in one network may lead to the other one. One could reproduce the mutations one organism went through during evolution. Only the future will reveal the direction in which GENE-O-MATIC, and the whole CELL-O framework, will evolve....

14. Summary

The accumulation of data on regulatory processes in and between cells by means of high-throughput experimental methods calls for computer-assisted evaluation and analysis techniques. The first step towards a global picture, which scientists hope to assemble, is to sort and present data comprehensively. One way to achieve this is to create genetic networks, graphical representations of regulatory interactions between genes, mRNAs, and proteins. They are connected by arrows that depict interactions and influences between them.

Computational analysis and simulation of genetic networks can give new insights into the process being studied. Most methods available today are restricted to the analysis of single cells or fixed arrangements of cells. In this thesis, GENE-O-MATIC, a software tool for the simulation of genetic networks in multicellular context is presented. It features a flexible, three-dimensional cell model with cells that can move, divide, and interact with each other via the genetic network. The latter is simulated using a qualitative model. Since the network as well as the cell positions can be defined freely by the user, GENE-O-MATIC is not restricted to the analysis of a specific organism. It has a broad range of applicability.

With the help of GENE-O-MATIC, a model was created of the process of blastomere fate specification in the early embryo of the nematode *Caenorhabditis elegans*. The genetic network model contains 14 mRNAs and proteins and is based on experimental data collected from published experimental results. Cell positions are taken from microscope recordings. The results of this simulation show striking agreement with experimental observations, and manipulation of the *in silico* model results in phenotypes very similar to those of the respective mutants *in vivo*. It was even possible to make predictions and discover gaps in the data.

To demonstrate the applicability of GENE-O-MATIC to different organisms and conditions, several simulations were set up for flower development in *Arabidopsis thaliana*. The networks are also based on experimental results. No *a priori* information about cell positions was available, instead, cell behaviour was simulated. Here, too, the simulations could reproduce experimental results quite well.

During model building for *C. elegans* and *Arabidopsis*, the network editor alone turned out to be a valuable tool for gathering and sorting interaction data and genetic networks. Though simulation and experimental results match, the cell and tissue simulation functionality still has some drawbacks that restrict the choice of organisms which can be simulated. Asymmetric cells and long-range signals cannot be modelled. For developmental processes such as cell differentiation, the simple, qualitative model for the genetic network simulation turned out to be completely sufficient, however. The possibility to make experiments *in silico* further supports GENE-O-MATIC's predictive capabilities.

It is planned to further enhance and extend the modelling environment and the simulation methods. GENE-O-MATIC is a powerful aid to model building and hypothesis testing, and it has the potential to become a valuable extension of the experimental repertoire of biologists.

References

Ahringer, J. (1997). Maternal control of a zygotic patterning gene in *Caenorhabditis elegans*. *Development* 124:3865–3869.

Apache (2003). Apache Ant. http://ant.apache.org/.

Apweiler, R. (2003). Swiss-Prot. http://www.ebi.ac.uk/swissprot/index.html. Maintained by EBI and SIB.

Billewitz, H. (2003). Simulation genetischer Netzwerke unter Berücksichtigung der inhomogenen Verteilung der Zellbestandteile. Master's thesis, Medizinische Fakultät der Ruprecht-Karls-Universität, Heidelberg.

Billewitz, H., Platzer, U., and Meinzer, H.-P. (2002). Simulation of genetic networks with an inhomogeneous distribution of cell contents. *In* Fifth German workshop on Artificial Life: Abstracting and Synthesizing the Principles of Living Systems (D. Polani, J. Kim, and T. Martinez, editors). Akademische Verlags-Gesellschaft Aka GmbH, pp. 77–79.

Bodnar, J. (1997). Programming the *Drosophila* embryo. *J Theor Biol* 188:391–445.

Bodnar, J. W., and Bradley, M. K. (2001). Programming the *Drosophila* embryo 2: from genotype to phenotype. *Cell Biochem Biophys* 34:153–190. Review.

Boveri, T. (1888). Zellenstudien II. Die Befruchtung und Teilung des Eies von *Ascaris megalocephala*. *Jena Zeit Naturw* 22:685–882.

Bowerman, B., Draper, B., Mello, C., and Priess, J. (1993). The maternal gene skn-1 encodes a protein that is distributed unequally in early *C. elegans* embryos. *Cell* 74:443–452.

Bowerman, B., Eaton, B., and Priess, J. (1992a). skn-1, a maternally expressed gene required to specify the fate of ventral blastomeres in the early *C. elegans* embryo. *Cell* 68:1061–1075.

Braun, V. (2003). Analyse, Vergleich, und Simulation der embryonalen Zellstammlinien verschiedener Nematoden. Ph.D. thesis, Medizinische Fakultät der Ruprecht-Karls-Universität, Heidelberg.

Braun, V., Azevedo, R. B., Gumbel, M., Agapow, P. M., Leroi, A. M., and Meinzer, H. P. (2003). ALES: cell lineage analysis and mapping of developmental events. *Bioinformatics* 19:851–858.

Brenner, S. (1973). The genetics of behaviour. *Br Med Bull* 29:269–271.

Brenner, S. (1974). The genetics of *Caenorhabditis elegans*. *Genetics* 77:71–94.

Calvo, D., Victor, M., Gay, F., Sui, G., Luke, M. P.-S., Dufurcq, P., Wen, G., Maduro, M., Rothman, J., and Shi, Y. (2001). A POP-1 repressor complex restricts inappropriate cell type-specific gene transcription during *C. elegans* embryogenesis. *EMBO J* 20:7197–7208.

Cederqvist, P. (1993). Concurrent Versions System. http://www.cvshome.org.

Christensen, S., Kodoyianni, V., Bosenberg, M., Friedman, L., and Kimble, J. (1996). lag-1, a gene required for lin-12 and glp-1 signaling in *Caenorhabditis elegans*, is homologous to human CBF1 and *Drosophila* Su(H). *Development* 122:1373–1383.

Crittenden, S., Rudel, D., Binder, J., Evans, T., and Kimble, J. (1997). Genes required for GLP-1 asymmetry in the early *Caenorhabditis elegans* embryo. *Dev Biol* 181:36–46.

de Jong, H. (2002). Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol* 9:67–103. Review.

de Jong, H., Geiselmann, J., Hernandez, C., and Page, M. (2003). Genetic Network Analyzer: qualitative simulation of genetic regulatory networks. *Bioinformatics* 19:336–344.

DIP (2003). Database of Interacting Proteins. http://dip.doe-mbi.ucla.edu/.

Draper, B., Mello, C., Bowerman, B., Hardin, J., and Priess, J. (1996). MEX-3 is a KH domain protein that regulates blastomere identity in early *C. elegans* embryos. *Cell* 87:205–216.

Evans, T., Crittenden, S., Kodoyianni, V., and Kimble, J. (1994). Translational control of maternal glp-1 mRNA establishes an asymmetry in the *C. elegans* embryo. *Cell* 77:183–194.

Fire, A., Xu, S., Montgomery, M. K., Kostas, S. A., Driver, S. E., and Mello, C. C. (1998). Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. *Nature* 391:806–811.

Fitzgerald, K., and Greenwald, I. (1995). Interchangeability of *Caenorhabditis elegans* DSL proteins and intrinsic signalling activity of their extracellular domains in vivo. *Development* 121:4275–4282.

Fletcher, J. C., Brand, U., Running, M. P., Simon, R., and Meyerowitz, E. M. (1999). Signaling of cell fate decisions by CLAVATA3 in *Arabidopsis* shoot meristems. *Science* 283:1911–1914.

Fowler, M., and Scott, K. (2000). UML konzentriert. Addison-Wesley, München, 2nd edition.

Friedman, C., Kra, P., Yu, H., Krauthammer, M., and Rzhetsky, A. (2001). GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics* 17 Suppl 1:S74–S82.

Goldstein, B., and Hird, S. (1996). Specification of the anterior-posterior axis in *C. elegans*. *Development* 122:1467–1474.

González, A. (2002). Use of Gene-O-Matic to simulate a genetic network in Thale Cress. Technical report, Deutsches Krebsforschungszentrum.

Guedes, S., and Priess, J. (1996). The *C. elegans* MEX-1 protein is present in germline blastomeres and is a P granule component. *Development* 124:731–739.

Gumbel, M. (2001). Eine dreidimensionale Computersimulation zur Analyse der frühen Embryogenese des Nematoden *Caenorhabditis elegans*. Ph.D. thesis, Medizinische Fakultät der Ruprecht-Karls-Universität, Heidelebrg.

Gumbel, M., Schnabel, R., and Meinzer, H.-P. (2000). Analysis of Cell Migrations in *C. elegans* using Computer Simulations. *In* Proceedings of the 14th European Simulation Multiconference. SCS Publications, pp. 605–611.

Guo, S., and Kemphues, K. (1996a). Molecular genetics of asymmetric cleavage in the early *Caenorhabditis elegans* embryo. *Curr Opin Genet Dev* 6:408–415.

Haecker, A., and Laux, T. (2001). Cell-cell signaling in the shoot meristem. *Curr Opin Plant Biol* 4:441–446.

Henderson, S. T., Gao, D., Lambie, E. J., and Kimble, J. (1994). lag-2 may encode a signaling ligand for the GLP-1 and LIN-12 receptors of *C. elegans*. *Development* 120:2913–2924.

Herman, M. (2001). *C. elegans* POP-1/TCF functions in a canonical Wnt pathway that controls cell migration and in a noncanonical Wnt pathway that controls cell polarity. *Development* 128:581–590.

Howell, S. (1998b). Molecular Genetics of Plant Development, chapter 7. Transition to Flowering. Cambridge University Press.

Huang, N., Mootz, D., Walhout, A., Vidal, M., and Hunter, C. (2002). MEX-3 interacting proteins link cell polarity to asymmetric gene expression in *Caenorhabditis elegans*. *Development* 129:747–759.

Hung, T., and Kemphues, K. (1999). PAR-6 is a conserved PDZ domain-containing protein that colocalizes with PAR-3 in *Caenorhabditis elegans* embryos. *Development* 126:127–135.

Hunter, C., and Kenyon, C. (1996). Spatial and temporal controls target pal-1 blastomere-specification activity to a single blastomere lineage in *C. elegans* embryos. *Cell* 87:217–226.

Hutter, H., and Schnabel, R. (1994). glp-1 and inductions establishing embryonic axes in *C. elegans*. *Development* 120:2051–2064.

IBM (2002a). Graph Foundation Classes. http://www.alphaworks.ibm.com/tech/gfc.

IBM (2002b). Jikes Version 1.18. http://www-124.ibm.com/developerworks/opensource/jikes/.

Jackson, E., Johnson, D., and Nash, W. (1986). Gene networks in development. *J Theor Biol* 119:379–396.

JetBrains, I. (2003). IntelliJ Idea. http://www.intellij.com/.

Joberty, G., Petersen, C., Gao, L., and Macara, I. (2000). The cell-polarity protein Par6 links Par3 and atypical protein kinase C to Cdc42. *Nat Cell Biol* 2:531–539.

KEGG (2003). Kyoto Encyclopedia of Genes and Genomes. http://www.genome.ad.jp/kegg/.

Kieffer, M., and Davies, B. (2001). Developmental programmes in floral organ formation. *Semin Cell Dev Biol* 12:373–380.

Lin, R., Hill, R., and Priess, J. (1998). POP-1 and anterior-posterior fate decisions in *C. elegans* embryos. *Cell* 92:229–239.

Lin, R., Thompson, S., and Priess, J. (1995). pop-1 encodes an HMG box protein required for the specification of a mesoderm precursor in early *C. elegans* embryos. *Cell* 83:599–609.

Lohaus, R. (2003). Simulation of vulval development in the nematode *C. elegans*. Master's thesis, Medizinische Fakultät der Ruprecht-Karls-Universität, Heidelberg. In process.

Maduro, M., Meneghini, M., Bowerman, B., Broitman-Maduro, G., and Rothman, J. (2001). Restriction of mesendoderm to a single blastomere by the combined action of SKN-1 and a GSK-3beta homolog is mediated by MED-1 and -2 in *C. elegans*. *Mol Cell* 7:475–485.

Maduro, M. F., Lin, R., and Rothman, J. H. (2002). Dynamics of a developmental switch: recursive intracellular and intranuclear redistribution of *Caenorhabditis elegans* POP-1 parallels Wnt-inhibited transcriptional repression. *Dev Biol* 248:128–142.

Mango, S., Thorpe, C., Martin, P., Chamberlain, S., and Bowerman, B. (1994b). Two maternal genes, apx-1 and pie-1, are required to distinguish the fates of equivalent blastomeres in the early *Caenorhabditis elegans* embryo. *Development* 120:2305–2315.

Meinhardt, H., and Gierer, A. (1974). Applications of a theory of biological pattern formation based on lateral inhibition. *J Cell Sci* 15:321–346.

Mello, C., Draper, B., Krause, M., Weintraub, H., and Priess, J. (1992). The pie-1 and mex-1 genes and maternal control of blastomere identity in early *C. elegans* embryos. *Cell* 70:163–176.

Mello, C., Draper, B., and Priess, J. (1994). The maternal genes apx-1 and glp-1 and establishment of dorsal-ventral polarity in the early *C. elegans* embryo. *Cell* 77:95–106.

Mello, C., Schubert, C., Draper, B., Zhang, W., Lobel, R., and Priess, J. (1996). The PIE-1 protein and germline specification in *C. elegans* embryos. *Nature* 382:710–712.

Mendoza, L., and Alvarez-Buylla, E. (1998). Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis. *J Theor Biol* 193:307–319.

Mickey, K., Mello, C., Montgomery, M., Fire, A., and Priess, J. (1996). An inductive interaction in 4-cell stage *C. elegans* embryos involves APX-1 expression in the signalling cell. *Development* 122:1791–1798.

Molin, L., Schnabel, H., Kaletta, T., Feichtinger, R., Hope, I., and Schnabel, R. (1999). Complexity of developmental control: analysis of embryonic cell lineage specification in *Caenorhabditis elegans* using pes-1 as an early marker. *Genetics* 151:131–141.

Newman-Smith, E., and Rothman, J. (1998). The maternal-to-zygotic transition in embryonic patterning of *Caenorhabditis elegans*. *Curr Opin Genet Dev* 8:472–480.

Object Management Group (2003). The UML Resource Page. http://www.omg.org/uml/.

Petcherski, A., and Kimble, J. (2000). LAG-3 is a putative transcriptional activator in the *C. elegans* Notch pathway. *Nature* 405:364–368.

Platzer, A. (2003). The Orbital Library for Java. http://functologic.com.

Poethig, R. (1990). Phase change and the regulation of shoot morphogenesis in plants. *Science* 250:923–930.

Priess, J., and Thomson, J. (1987). Cellular interactions in early *C. elegans* embryos. *Cell* 48:241–250.

Pulak, R., and Anderson, P. (1993). mRNA surveillance by the *Caenorhabditis elegans* smg genes. *Genes Dev* 7:1885–1897.

Rocheleau, C., Downs, W., Lin, R., Wittmann, C., Bei, Y., Cha, Y., Ali, M., Priess, J., and Mello, C. (1997). Wnt signaling and an APC-related gene specify endoderm in early *C. elegans* embryos. *Cell* 90:707–716.

Salvo, J. (2002). OpenJGraph. http://openjgraph.sourceforge.net.

Schlesinger, A., Shelton, C., Maloof, J., Meneghini, M., and Bowerman, B. (1999). Wnt pathway components orient a mitotic spindle in the early *Caenorhabditis elegans* embryo without requiring gene transcription in the responding cell. *Genes Dev* 13:2028–2038.

Schnabel, R., Hutter, H., Moerman, D., and Schnabel, H. (1997). Assessing normal embryogenesis in *Caenorhabditis elegans* using a 4D microscope: variability of development and regional specification. *Dev Biol* 184:234–265.

Schoof, H., Lenhard, M., Haecker, A., Mayer, K., Jurgens, G., and Laux, T. (2000). The stem cell population of *Arabidopsis* shoot meristems in maintained by a regulatory loop between the CLAVATA and WUSCHEL genes. *Cell* 100:635–644.

Schroeter, E., Kisslinger, J., and Kopan, R. (1998). Notch-1 signalling requires ligand-induced proteolytic release of intracellular domain. *Nature* 393:382–386.

Schubert, C., Lin, R., de Vries, C., Plasterk, R., and Priess, J. (2000). MEX-5 and MEX-6 function to establish soma/germline asymmetry in early *C. elegans* embryos. *Mol Cell* 5:671–682.

Sedgewick, R. (1992). Algorithmen in C++. Addison-Wesley, Bonn, München, Paris, 1st edition.

Seydoux, G., and Fire, A. (1994). Soma-germline asymmetry in the distribution of embryonic RNAs in *Caenorhabditis elegans*. *Development* 120:2823–2834.

Seydoux, G., Mello, C., Pettitt, J., Wood, W., Priess, J., and Fire, A. (1996). Repression of gene expression in the embryonic germ lineage of *C. elegans*. *Nature* 382:713–716.

Seydoux, G., and Schedl, T. (2001). The germline in *C. elegans*: origins, proliferation, and silencing. *Int Rev Cytol* 203:139–185. Review.

Sharma, V. K., and Fletcher, J. C. (2002). Maintenance of shoot and floral meristem cell proliferation and fate. *Plant Physiol* 129:31–39. Review.

Shin, T., Yasuda, J., Rocheleau, C., Lin, R., Soto, M., Bei, Y., Davis, R., and Mello, C. (1999). MOM-4, a MAP kinase kinase kinase-related protein, activates WRM-1/LIT-1 kinase to transduce anterior/posterior polarity signals in *C. elegans*. *Mol Cell* 4:275–280.

Strome, S., and Wood, W. (1982). Immunofluorescence visualization of germ-line-specific cytoplasmic granules in embryos, larvae, and adults of *Caenorhabditis elegans*. *Proc Natl Acad Sci USA* 79:1558–1562.

Strome, S., and Wood, W. B. (1983). Generation of asymmetry and segregation of germ-line granules in early *C. elegans* embryos. *Cell* 35:15–25.

Struhl, G. (1989). Morphogen gradients and the control of body pattern in insect embryos. *Ciba Found Symp* 144:65–86. Review.

Struhl, G., and Adachi, A. (1998). Nuclear access and action of notch in vivo. *Cell* 93:649–660.

Sulston, J., and Horvitz, H. (1977). Post-embryonic cell lineages of the nematode, *Caenorhabditis elegans*. *Dev Biol* 56:110–156.

Sulston, J., Schierenberg, E., White, J., and Thomson, J. (1983). The embryonic cell lineage of the nematode *Caenorhabditis elegans*. *Dev Biol* 100:64–119.

Sulston, J. E., and Brenner, S. (1974). The DNA of *Caenorhabditis elegans*. *Genetics* 77:95–104.

Sun Microsystems (2003a). Java Development Kit Version 1.4.1. http://java.sun.com/j2se/1.4.2/.

Sun Microsystems (2003b). Java Reflection. http://java.sun.com/j2se/1.4.2/docs/guide/reflection/index.html.

Sun Microsystems (2003c). JavaBeans. http://java.sun.com/products/javabeans/.

Tabuse, Y., Izumi, Y., Piano, F., Kemphues, K., Miwa, J., and Ohno, S. (1998). Atypical protein kinase C cooperates with PAR-3 to establish embryonic polarity in *Caenorhabditis elegans*. *Development* 125:3607–3614.

Thorpe, C., Schlesinger, A., Carter, J., and Bowerman, B. (1997). Wnt signaling polarizes an early *C. elegans* blastomere to distinguish endoderm from mesoderm. *Cell* 90:695–705.

Trotochaud, A. E., Hao, T., Wu, G., Yang, Z., and Clark, S. E. (1999). The CLAVATA1 receptor-like kinase requires CLAVATA3 for its assembly into a signaling complex that includes KAPP and a Rho-related protein. *Plant Cell* 11:393–406.

Webnox Corp. (2000-2003). Hyperdictionary. http://www.hyperdictionary.com/.

Wiechert, W. (2002). Modeling and simulation: tools for metabolic engineering. *J Biotechnol* 94:37–63. Review.

Wolf, N., Priess, J., and Hirsh, D. (1983). Segregation of germline granules in early embryos of Caenorhabditis elegans: an electron microscopic analysis. *J Embryol Exp Morphol* 73:297–306.

Wood, W., editor (1988). The nematode *C. elegans*. Cold Spring Harbor Laboratory.

WormBase (2000). http://www.wormbase.org/.

Zupan, B., Demsar, J., Bratko, I., Juvan, P., Halter, J. A., Kuspa, A., and Shaulsky, G. (2003b). GenePath: a system for automated construction of genetic networks from mutant data. *Bioinformatics* 19:383–389.

Own Publications

The following publications describe different aspects of the research and results achieved in this thesis.

Azevedo, R. B., Gumbel, M., Braun, V., Agapow, P. M., Houthoofd, W., Jacobsen, K., Platzer, U., Meinzer, H.-P., Borgonie, G., and Leroi, A. (2002). Nematode embryonic cell lineages are computationally efficient. *In* Proceedings of the 1st Computational Embryology Workshop (V. Braun, M. Gumbel, A. M. Leroi, and H.-P. Meinzer, editors), Technical Report 125. Abteilung Medizinische und Biologische Informatik, Deutsches Krebsforschungszentrum Heidelberg, INF 280, 69210 Heidelberg, pp. 13–24.

Billewitz, H., Platzer, U., and Meinzer, H.-P. (2002). Simulation of genetic networks with an inhomogeneous distribution of cell contents. *In* Fifth German workshop on Artificial Life: Abstracting and Synthesizing the Principles of Living Systems (D. Polani, J. Kim, and T. Martinez, editors). Akad. Verl.-Ges. Aka, pp. 77–79. Peer-reviewed.

Lohaus, R., Platzer, U., Meinzer, H.-P., and Azevedo, R. (2003). Simulation of the genetic network controlling *C. elegans* vulval development. *In* Proceedings of the fourth International Conference on Systems Biology. St. Louis, MI. In press.

Platzer, U., Braun, V., and Meinzer, H.-P. (2002). Simulation of genetic networks in multiple cells controlling cell differentiation in the *C. elegans* embryo. *In* Proceedings of the 1st Computational Embryology Workshop (V. Braun, M. Gumbel, A. M. Leroi, and H.-P. Meinzer, editors), Technical Report 125. Abteilung Medizinische und Biologische Informatik, Deutsches Krebsforschungszentrum Heidelberg, INF 280, 69210 Heidelberg, pp. 43–62.

Platzer, U., and Meinzer, H.-P. (2002a). Genetic networks in multicellular organisms. *In* Proceedings of the third International Conference on Systems Biology: The Logic of Life (E. Aurell, J. Elf, and J. Jeppsson, editors). Karolinska Institutet, Stockholm, Sweden, pp. 181–182.

Platzer, U., and Meinzer, H.-P. (2002b). Simulation of genetic networks in multicellular context. *In* Fifth German workshop on Artificial Life: Abstracting and Synthesizing the Principles of Living Systems (D. Polani, J. Kim, and T. Martinez, editors). Akad. Verl.-Ges. Aka, Lübeck, Germany, pp. 43–51. Peer-reviewed.

Platzer, U., and Meinzer, H.-P. (in press). Genetic networks in the early development of *Caenorhabditis elegans*. *Int Rev Cytology* Invited review.

Appendix

A. GENE-O-MATIC User Manual

The aims of GENE-O-MATIC are:

- to create and to edit a genetic regulatory network
- to simulate genetic interactions within and between cells
- to simulate organism development

The set of genes, mRNAs, and proteins that can be found in each cell, together with their interactions, is represented by the genetic network. The editor provides you with the means to add or remove genes, mRNAs, proteins, and interactions as well as to assign certain attributes. To avoid having to say “genes, mRNAs, and proteins” all the time, the term “BIC” (biological information carrier) is used here to address all three of them at once.

The user manual is supposed to help people in applying GENE-O-MATIC, i.e. it describes how to use GENE-O-MATIC to create networks, simulate their behaviour, and use the results. It gives a quick overview on how to set up and run a simulation. You should read it to get a general impression of the capabilities of the software. More information about the specific functions and options can be found in the online help delivered with the program.

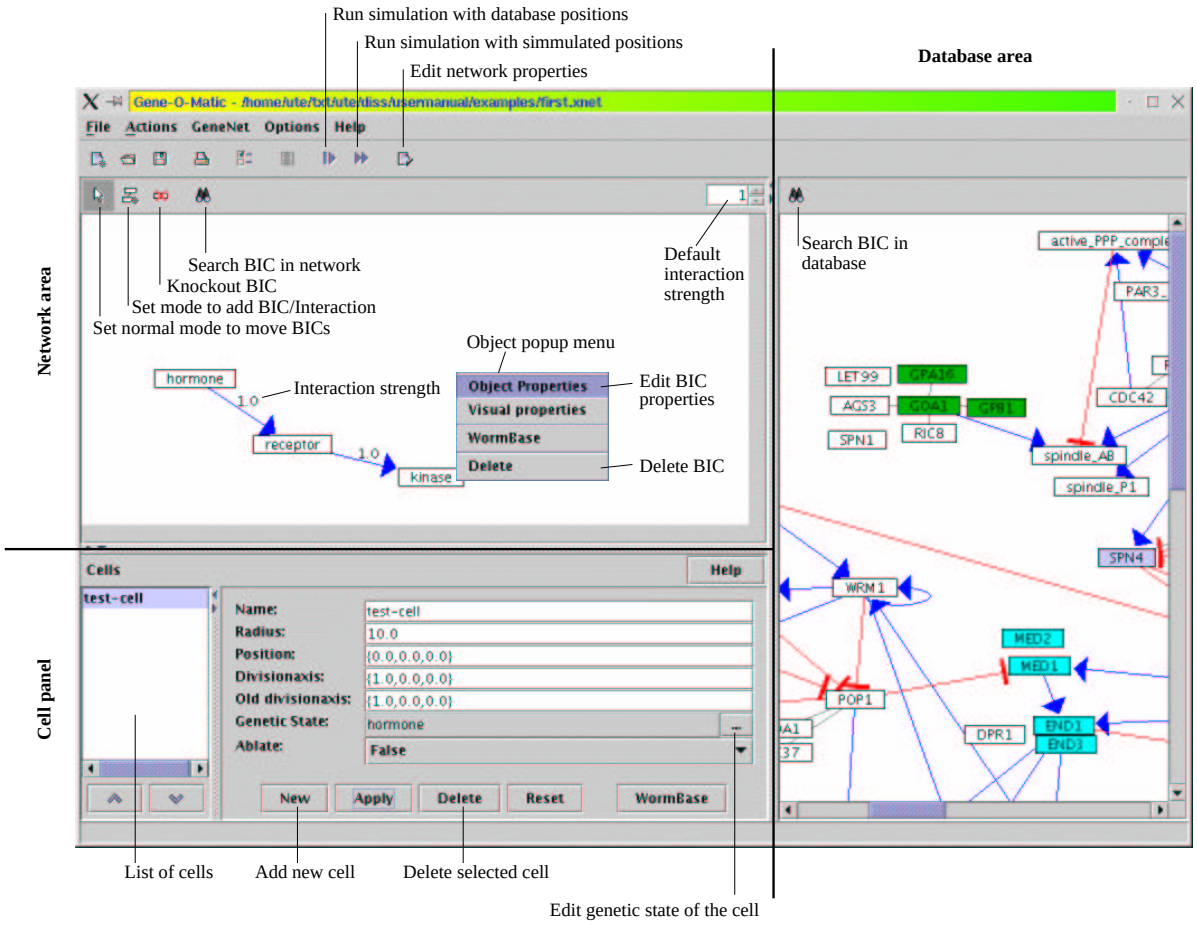

Figure A.1.: User interface of the network editor.

A.1. The Editor

The editor consists of three major parts (see figure A.1):

1. The network area where the user can create a genetic network by adding BICs and interactions.

2. The database area that presents often-used or other interesting BICs and interactions; these can be added to the user's network by drag-and-drop. The database itself cannot be modified.

3. The cell panel where the user can specify the cells for the simulation start.

If you start GENE-O-MATIC, it displays the editor window with a new, empty genetic network.

A.2. Adding Genes to the Network

The network area has two modes, one for adding BICs and interactions, and one for moving BICs. To switch between the modes, use the buttons in the toolbar ( to add,  for normal mode, see also figure A.1) or right-click on the background of the network area and select the mode you want from the popup menu.

Now add some BICs to the network. Switch to the `add BIC/Interaction` mode. This puts GENE-O-MATIC in the correct mode for adding new BICs and interactions. By clicking with the left mouse button on the graph, you add new BICs. For each new BIC, a dialog with the properties will be displayed. You should at least enter a name for each BIC. You can always change the BIC properties later by right-clicking on the BIC and selecting `Object Properties` from the popup menu.

In the `add BIC/Interaction` mode, you can add interactions between BICs by clicking on one BIC and dragging the mouse (while you still hold down the left mouse button) to another BIC (or back to the same, if it influences itself in a feedback mechanism). Similar to the BICs, upon creation of the interaction a dialog will show up where you can edit the properties of the interaction (e.g. the strength). You can do this later by right-clicking on the line and selecting `Object Properties` from the popup menu (don't worry if you don't hit the line on the first try; the normal `graph popup menu` will be displayed). The default strength for interactions is 0, meaning `"no influence"`. To set a different default strength for the interactions, use the control in the upper right of the network area (see figure A.1). Most often you will need strength values of +1 or -1.

Try to create the network shown in the screenshot (figure A.1): three BICs (*hormone*, *receptor*, and *kinase*), connected by two interactions with strength +1.

A.3. Adding Cells

To add a cell to your virtual organism, click `New` on the cell panel (lower left). A new cell entry will be displayed.

To specify what BICs are active in the cell, click on the button "`...`" to the right of the `Genetic State` property. A new window will be displayed. Ticking a checkbox will set the BIC's initial state to `ON` (active). Set the *hormone* BIC to active.

For simplicity, we will not alter the default values for radius, position, and division axes here. You can adjust them later to see the effect of different division settings.

A.4. Running the Simulation

To avoid unnecessary trouble, you should save the network before you continue. Though it does not happen very often, some circumstances may lead to an abnormal termination of program execution, in which case you would have to start from the beginning if you did not save the network. To save it, select `Save as...` from the `File` menu and enter a filename.

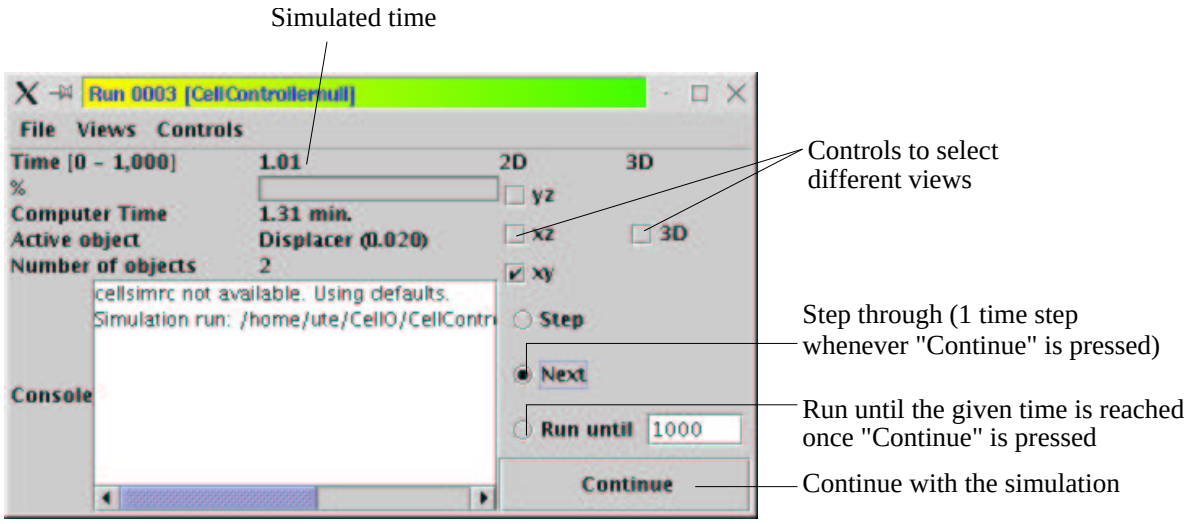

Figure A.2.: `Run` window of the simulation

To load it again if the program really crashed, select `Load` from the `File` menu and enter the same filename or select it in the file browser.

To start the simulation, press the ≫ (`Run simulation with simulated positions`) button in the toolbar, or select `Run simulation...` → `Simulated positions` from the `GeneNet` menu. A dialog will appear that lets you enter a note – just press `OK` for now. You'll see two windows, the `Run` window (figure A.2) and a 2D view of the cell. The cell is black, and that is because we have not yet specified any colouring. To do so, select `Cell Colours` from the `Options` menu of the GENE-O-MATIC editor window. You can choose one colour for each of the BICs; if you have more than four BICs, you must leave out some. Unfortunately the display is not updated automatically, but if you resize the 2D view window, or if you minimize it and maximize it again, the colour will show. To step through the simulation, select the `Next` option in the `Run` window and press `Continue`. You will see that the simulated time changes in steps of 0.5 minutes. This is because after 0.5 minutes, the next states are calculated, and after another 0.5 minutes, they are set. Therefore, at 2.0 minutes, the cell in the viewer should switch to another colour, indicating that its genetic state (BIC activities) has changed.

A.5. View the Results

There are three possibilities to look at the results, or to look at what is going on during a simulation:

- Use the 3D view to obtain information about the cells during the simulation. Select `3D` in the `Run` window. Then hold down the `Control` key and click on the cell with the left mouse button. An `Info` window will be displayed that shows information about the current state of the cell.

- Display the current genetic states for all cells in a table. Select `View Result` → `Genetic states...` in the `GeneNet` menu. The table is continuously updated during the simulation. Use the filters to select which BICs and cells shall *not* be displayed in the table.

- Display the lineage of the simulated organism. Select `View Result` → `Cell lineage...` in the `GeneNet` menu. The cell lineage will not look very impressive for a singe cell, but you can imagine how useful it is for lots of dividing cells. The colours in the lineage are the same as those in the 2D and 3D views. The lineage is also updated during the simulation, as you can see if you press the `Continue` button in the `Run` window several more times.

Well, that is it! Congratulations to your first simulation with GENE-O-MATIC. Further information concerning all options and properties can be found in the online help which is available via the `Help` menu and the `Help` buttons in the dialogs and panels.

B. UML Notation

UML stands for *Unified Modelling Language*. It is a graphical language for the formal description of software. Here a short overview of the notation of UML class diagrams is given. Class diagrams are used to depict the relationships between different classes (types of objects). This chapter is by no means a complete description of UML. For further information on UML, consult Fowler and Scott (2000) and Object Management Group (2003), for instance.

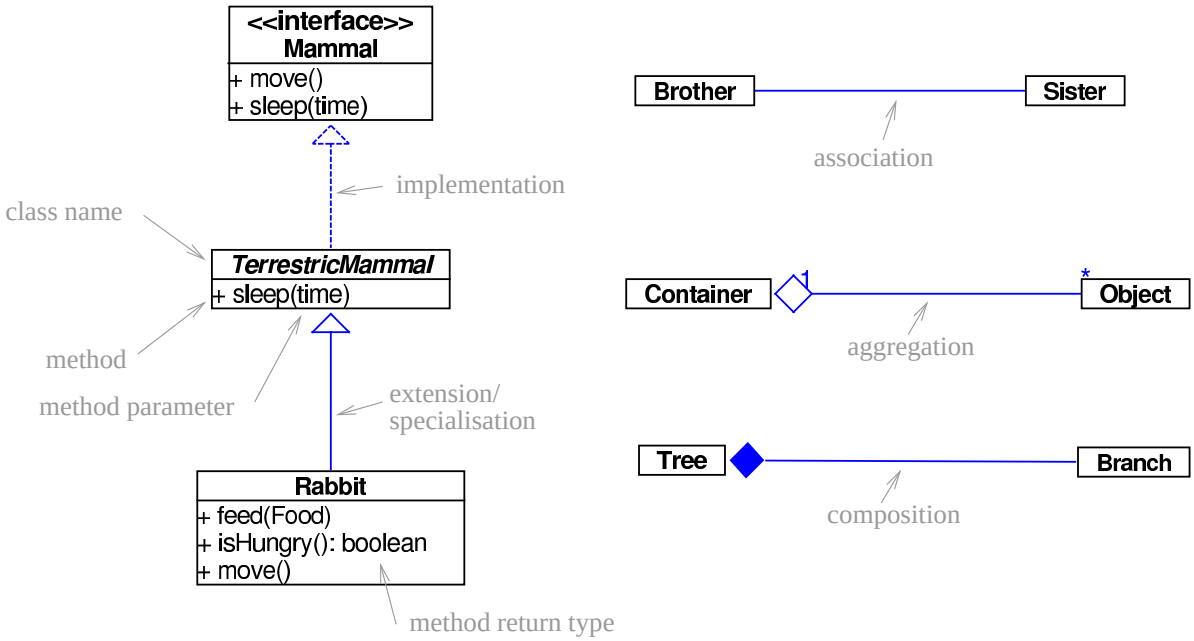

Figure B.1.: Common symbols in UML class diagrams. Grey text and arrows are not part of the diagram; they explain its elements.

Figure B.1 shows the most commonly used symbols in UML class diagrams. These diagrams may contain different types of model elements:

≪**Interfaces**≫ define the methods a type of object needs to interact with its surroundings ("behaviour"). They do not specify how the behaviour is implemented in the code.
Example: the interface `Mammal` defines the methods `move` and `sleep`.

Abstract classes contain some code for common behavioural patterns of related types of objects, but do not provide a functional object by themselves.
Example: the abstract class `TerrestricMammal` implements the method `sleep` as

something like "do nothing", but does not implement `move`, as different animals may have different forms of movement (run, hop, fly, ...).

Classes can be seen as programmatic representations of types of objects in the real world. Example: the class `Rabbit` implements the method `move` as "hop". It does not need to implement `sleep` because that is already done in the superclass (`TerrestricMammal`), the one it extends (see below). In addition, two food-related methods are also available (specialisation of the `TerrestricMammal`).

Classes and interfaces can be connected by different types of relationships, drawn as "arcs", as computer scientists call the arrows:

Implementation A class implements an interface if it provides code for all (or some, for abstract classes) of the methods defined in the interface. The class is then the concrete description of one possible type of object defined by the interface.

Extension or Specialisation One class (the *subclass*) extends another class (the *superclass*) if it offers more or special methods, more ways to interact with the surroundings, than the other class. A subclass may also redefine some behaviour of the superclass: all terrestric mammals do nothing while they sleep, but bears snore.

Association Two classes are associated if they have some kind of relationship, for example if one knows about the other and sometimes sends messages. Aggregation and composition are special kinds of associations.

Aggregation One class aggregates another if it contains objects of the other type. Aggregations can be annotated with a cardinality, for example a one-to-many aggregation says that one object of the first type contains any number (between zero and infinity) of objects of the other type.

Composition This is a special type of aggregation where one class cannot exist without the other, i.e. always contains at least one object of the other type.

C. BICs and Interactions in Blastomere Fate Specification in *Caenorhabditis elegans*

Table C.1.: BICs involved in the regulation of blastomere fate specification in the *Caenorhabditis elegans* embryo.

Name	**Location**	**Description**	**Reference**
APX-1	P_1, P_2, at the border to AB descendants	Delta/Serrate-homolog (ligand)	Mickey *et al.*, 1996
GLP-1	translated only in AB and descendants	Notch homolog (receptor)	Evans *et al.*, 1994; Hutter and Schnabel, 1994; Mello *et al.*, 1994
GSK-3	EMS	also called SGG-1; involved in Wnt signalling and spindle rotation in EMS	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997; Schlesinger *et al.*, 1999
LAG-1		transcription factor, no activating function but binds to promotor of GLP-1 target genes	Petcherski and Kimble, 2000
LAG-3		see SEL-8	
LIT-1	EMS	kinase	Molin *et al.*, 1999
MED-1, MED-2	EMS cell	mesendoderm-specific GATA transcription factors	Maduro *et al.*, 2001
MEX-3	AB	maternal RNA-binding protein, inhibits PAL-1 translation	Draper *et al.*, 1996
MOM-1	P_2	porcupine-homolog, required for processing and secretion of MOM-2	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-2	P_2	Wnt-homolog (ligand)	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-3	P_2	unknown function	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-4	EMS	MAPKKK	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997; Shin *et al.*, 1999
MOM-5	EMS	frizzled-homolog (receptor)	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
PAL-1	C, D	maternal muscle-specific transcription factor	Hunter and Kenyon, 1996
PIE-1	P_0,P_1,P_2, P_3, P_4	germ line specification, repressor of embryonic transcription	Mello *et al.*, 1992; Mango *et al.*, 1994b
PKC-3	anterior cortical	atypical protein kinase	Tabuse *et al.*, 1998

Table C.1.: (continued)

Name	**Location**	**Description**	**Reference**
POP-1	EMS, MS (and other anterior daughters)	maternal transcription factor of the tcf/lef HMG box family, required for specification of MS and distinction of daughters of a/p divisions	Lin *et al.*, 1995, 1998; Herman, 2001; Calvo *et al.*, 2001
SEL-8		also called LAG-3. Transcriptional activator without DNA binding capability	Petcherski and Kimble, 2000
SGG-1		see GSK-3	
SKN-1	P_1 and descendants	maternal transcription factor, specifies EMS fate	Bowerman *et al.*, 1992a, 1993
SMG-3	AB, P_1	degrades nonsense (and untranslated?) mRNA, e.g. of *pal-1*	Pulak and Anderson, 1993
VAB-7	C cell	embryonic muscle-specific patterning gene (homeodomain)	Hunter and Kenyon, 1996
WRM-1	EMS	armadillo/β-catenin-homolog	Shin *et al.*, 1999

Table C.2.: Regulatory interactions in blastomere fate specification in the early embryonic development of *Caenorhabditis elegans*.

Interaction	**Description**	**Reference**
APX-1 → GLP-1	ligand binds to receptor	Mello *et al.*, 1994; Mickey *et al.*, 1996
GLP-1 ⇌ LAG-1	LAG-1 recruits GLP-1, and with it SEL-8, to the DNA	Petcherski and Kimble, 2000
GLP-1 ⇌ SEL-8	interact with each other	Petcherski and Kimble, 2000
GSK-3 ⊣ SKN-1	GSK-3 inhibits SKN-1 in the C lineage, preventing mesendoderm development there	Maduro *et al.*, 2001
GSK-3 → WRM-1	Wnt signal is passed on	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
LIT-1 → LIT-1	autophosphorylation	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MEX-3 ⊣ *pal-1* mRNA	inhibit translation	Hunter and Kenyon, 1996
MOM-1 → MOM-2	MOM-1 helps secretion of MOM-2	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-2 → MOM-5	ligand binds to and activates receptor	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-4 → LIT-1	MOM-4 is required for activity of LIT-1	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
MOM-5 → GSK-3	Wnt signal is passed on	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
PAL-1 → *vab-7*	PAL-1 activates muscle-specific genes	Hunter and Kenyon, 1996
PIE-1 ⊣ PAL-1	PIE-1 inhibits all transcription, so the transcriptional regulator PAL-1 is useless.	Seydoux *et al.*, 1996
PIE-1 ⊣ SKN-1	PIE-1 inhibits all transcription, so the transcriptional regulator SKN-1 is useless.	Seydoux *et al.*, 1996
POP-1 ⊣ MED-1,2	POP-1 inhibits activity of MED-1,2 transcriptional activators	Maduro *et al.*, 2002
SKN-1 → *med-1, -2*	SKN-1 activates transcription of embryonic transcription factors	Maduro *et al.*, 2001
SKN-1 ⊣ PAL-1	in EMS and descendants	Hunter and Kenyon, 1996; Maduro *et al.*, 2001
SMG-3 ⊣ *pal-1* mRNA	degradation of untranslated mRNA	Pulak and Anderson, 1993

Table C.2.: (continued)

Interaction	**Description**	**Reference**
WRM-1 → LIT-1	Wnt signal is passed on	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997
WRM-1–LIT-1 complex ⊣ POP-1	complex inhibits POP-1 activity	Rocheleau *et al.*, 1997; Thorpe *et al.*, 1997