Gene Regulatory Dynamics in Growing Systems

Jan T. Kim, jtk@cmp.uea.ac.uk
J. Andrew Bangham, ab@cmp.uea.ac.uk

School of Computing Sciences, University of East Anglia
http://www.cmp.uea.ac.uk/

16th March 2005

1 Regulatory and Structural Information

The information contained in a gene can conceptually be separated into a regulatory and a structural component. Structural information determines the structure of the gene product and typically takes the form of a coding sequence of nucleotides, which is then (according to the "central dogma of molecular biology") translated to an amino acid sequence which, in turn, folds into a three-dimensional, biological structure (possibly in interaction with molecular chaperones etc.).

Regulatory information determines under which condition the encoded structure is expressed. The process at the core of interpreting regulatory information is the binding of transcription factors to their cognate sites (see e.g. [2]). As transcription factors are encoded by genes which are subject to differential regulation themselves, regulatory networks are constituted.

2 Modelling Gene Expression in Growing Systems

`transsys` is language [3] for representing regulatory networks and for simulating gene expression dynamics. It has been combined with the concept of Lindenmayer systems to explore how regulatory networks can organise spatially extended processes of development [4]. By including a diffusion model, `transsys` allows feedback from the spatial structures onto the regulatory dynamics. Fig. 1 shows an example of such a model. Further studies to systematically explore the interplay between spatial growth and dynamics in regulatory gene networks are currently underway, directed towards the long-term objective of understanding further principles underlying the genetics of geometry [1].

3 Regulatory Information and Programming

Regulatory information can be likened to elements that control variable scoping and the extent of flow control in programming languages. Like structural information specifies what biomolecule is to be synthesized and regulatory information determines under which conditions it is expressed, program statements determine what is to be done

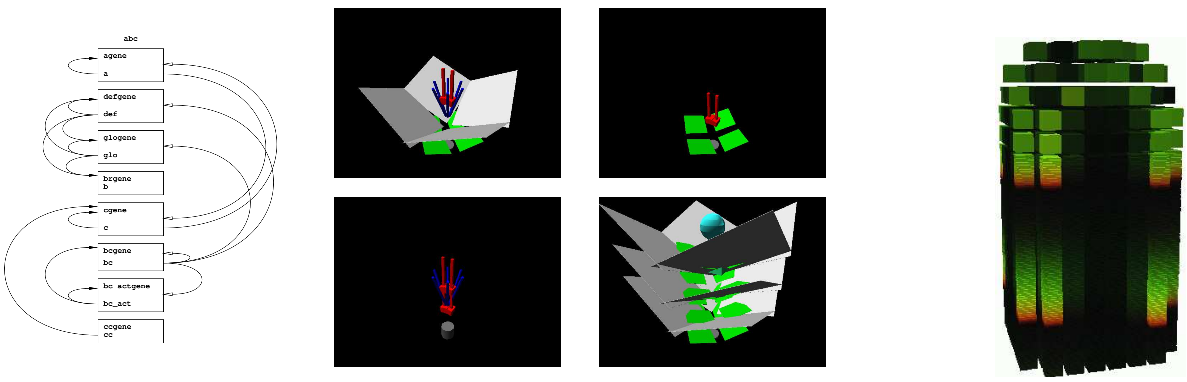

Figure 1: Examples of spatially extended structures organised by, or organising `transsys` regulatory network dynamics.

and program flow specifies under which conditions it is to be executed. Like in genetics, regulatory information in programs is somewhat elusive. In C-like languages, it takes the form of curly braces which are frequently overlooked and misinterpreted by the untrained. Nonetheless, these elements are pivotal to understand a program sufficiently to predict its output, as this example illustrates:

With regulatory information

```
int n = 5, x = 3;
int a[] = {3, 1, 5, 3, 2};
int m = 0;
for (int i = 0; i < n; i++)
{
  if (a[i] == x)
  {
    m = m + 1;
    printf("%d %d\n", i, m);
  }
}
```

Without regulatory information

```
int n = 5, x = 3;
int a[] = {3, 1, 5, 3, 2};
int m = 0;
for (int i = 0; i < n; i++)

if (a[i] == x)

m = m + 1;
printf("%d %d\n", i, m);
```

Predicted output

With regulatory information	Without regulatory information
0 1 3 2	6 2

It is interesting to note that regulatory information in classical programming languages tends to be organised in hierarchies (trees). Departures from this concepts are frequently notoriously difficult to handle, e.g. multiple inheritance in C++. Genetic regulatory networks are not restricted to hierarchies. Thus, these systems appear to transcend the constraint of hierarchy. Understanding the organisational principles underlying these networks may enable progress towards developing novel, non-classical programming paradigms that facilitate exploitation of the computational potential of growing systems.

4 Regulatory Networks and Open-Ended Evolution

Traditional evolutionary algorithms are focused on evolving structural information: The structure of a well-adapted, or optimal solution to some problem at hand is the objective. This traditional approach can be complemented and enhanced by integrating the regulatory aspect. In such an approach, the genome may contain multiple solutions in multiple genes, where each gene represents a solution that is adequate *under certain environmental conditions*. Clearly, this makes it necessary to equip the organism with sensors to perceive the current state of the environment. A possible transfer of this approach to a complex technical system is explored in [5].

Genomic research has revealed that the genomes of complex organisms do not have dramatically larger numbers of genes than the genomes of (supposedly) simple life forms. It is commonly believed that evolution of complexity takes place on the regulatory rather than on the structural information. From the view outlined above, and also based on the analogy of regulatory information to program flow control, the perspective emerges that regulatory information is the "glue" that integrates multiple structural elements into a coherent system, and thus enables open-ended growth of complexity of such systems (in terms of the number of different structural elements being integrated).

References

[1] Enrico Coen, Anne-Gaëlle Rolland-Lagan, Mark Matthews, J. Andrew Bangham, and Przemyslaw Prusinkiewicz. The genetics of geometry. *PNAS*, 101:4728–4735, 2004.

[2] Ritsert C. Jansen. Studying complex biological systems using multifactorial perturbation. *Nature Reviews Genetics*, 4:145–151, 2003.

[3] Jan T. Kim. `transsys` home page. `http://www2.cmp.uea.ac.uk/˜jtk/transsys/`.

[4] Jan T. Kim. `transsys`: A generic formalism for modelling regulatory networks in morphogenesis. In Jozef Kelemen and Petr Sosík, editors, *Advances in Artificial Life (Proceedings of the 6th European Conference on Artificial Life)*, volume 2159 of *Lecture Notes in Artificial Intelligence*, pages 242–251, Berlin Heidelberg, 2001. Springer Verlag.

[5] Paul Lukowicz, Erhardt Barth, and Jan T. Kim. Organic architectures for large-scale environment-aware sensor networks. In *Organic Computing Workshop at Architecture of Computing Systems (ARCS, 2005*, page (to appear), 2005.