

Abstract: LBS – a Language of Biological Systems

PhD project by Michael D. Pedersen*, supervised by Prof. Gordon Plotkin
LFCS, School of Informatics, University of Edinburgh, Scotland

*E-mail: m.d.pedersen@sms.ed.ac.uk

A problem in the systems biology cycle

Computational systems biology research can be viewed as a cyclic process involving cooperation between experimental biologists and theoreticians. Experiments are carried out on a system and observations gathered into informal diagrams by biologists. Theoreticians construct a corresponding model using a mathematical language based on e.g. differential equations or Petri nets. Observations from computer simulations can then be used to design new experiments, and the cycle repeats.

However, there are problems: first, constructing big models is difficult due to the gap in abstraction level between biology and mathematical modelling languages. Second, diagrams are ambiguous, so it may not be clear when the model is a correct abstraction of a system.

Our work aims to address these problems by designing a high-level language with tools for modelling complex biological systems. The language should be intuitive for biologists, yet have unambiguous, mathematically defined meaning. We call the language LBS – a Language of Biological Systems.

Key features of LBS

The basic entities of LBS are species with modification sites; they may be composed to form complexes. Simple rules are used for defining reactions which may involve constraints on the modification sites. Multiple reactions take place simultaneously, within a hierarchy of compartments. General modules can be defined and used repeatedly with different parameter values for compartments, species and rate constants, thus facilitating reuse.

Species modification sites are most often of true/false types (e.g., phosphorylated or not), but data types, and functions on these, can be more general. This allows, for example, DNA sequences to be represented by strings, and translation with possible frame shifts by string functions.

LBS is given meaning in terms of coloured Petri nets, a well established mathematical formalism for which both quantitative and qualitative simulation and analysis methods are known. This will enable reuse of existing software in the future development of simulation tools for LBS.

Example: gene expression

Program 1 shows how a general module for gene expression can be written in LBS. Inside the module three reactions are taking place in parallel (indicated by the vertical bar): transcription inside the nucleus, transport out of the nucleus, and finally translation to protein. The `mrna` species is declared locally, inside the module, reflecting that `mrna` is specific to each

individual gene being expressed. The remaining species are provided as parameters together with the nucleus compartment.

Program 1 A reusable module for gene expression

```
module dogma(comp nucleus, spec gene, spec rnap, spec protein) {
  spec mrna;
  nucleus[gene + rnap -> gene + rnap + mrna] |
  nucleus[mrna] -> mrna |
  mrna -> protein
}
```

Program 2 now shows how the gene expression module can be used to produce two proteins inside the nucleus of a cell. All the compartments and species involved must be declared before use, and they are then passed as parameters to two instances of the expression module. Note that both instances use the same **rnap** and must compete for this if supply runs out.

Details such as complex formation, modification sites and reaction rates have been omitted for clarity.

Program 2 Expression of two genes using previously declared module

```
comp cell vol 1E-15;
comp nuc vol 1E-14 inside cell;
spec gene1, gene2;
spec protein1, protein2;
spec rnap;

cell[
  exp(nuc, gene1, rnap, protein1) |
  exp(nuc, gene2, rnap, protein2)
]
```

Conclusion and future work

LBS is an intuitive, high-level language for describing biological systems and has mathematical meaning in terms of coloured Petri nets. Plans for future work include the development of tools for simulation and visualisation, which will subsequently be used in a non-trivial case study. The possibility to translate LBS to SBML will also be investigated, although the latter does not presently embody notions of modification sites and modularity.

Acknowledgements

This work was supported by Microsoft Research through its European PhD Scholarship Programme.