

# LBS: a language of biological systems

PhD project by Michael D. Pedersen  
Supervised by Gordon Plotkin

LFCS, School of Informatics  
University of Edinburgh, Scotland

The Twelfth Workshop on Software Platforms for Systems  
Biology  
Long Beach, California  
5-6 October, 2007

*This work was supported in part by Microsoft Research through the European PhD Scholarship Programme*

# Overview of LBS

- ▶ LBS features not presently found in SBML:
  - ▶ Species modification sites.
  - ▶ Complexes.
  - ▶ Pattern-based reactions.
  - ▶ Parameterised modules.

# Overview of LBS

- ▶ LBS features not presently found in SBML:
  - ▶ Species modification sites.
  - ▶ Complexes.
  - ▶ Pattern-based reactions.
  - ▶ Parameterised modules.
- ▶ Formal foundations:
  - ▶ A type system.
  - ▶ Semantics in terms of coloured Petri nets.
  - ▶ Translation to SBML possible in some cases.

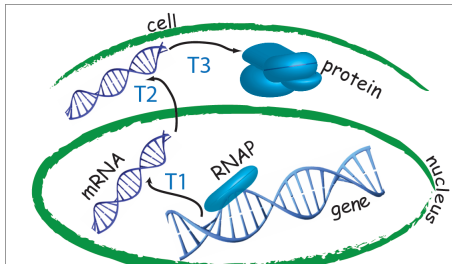




# Example: a module for transcription and translation

```
module exp(comp nuc, spec gene{act:bool}, )  
{
```

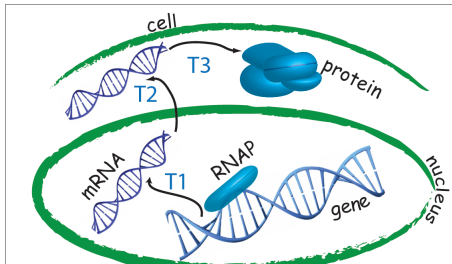
```
}
```



## Example: a module for transcription and translation

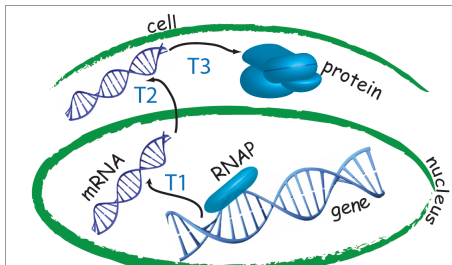
```
module exp(comp nuc, spec gene{act:bool}, spec prot)  
{
```

```
}
```



## Example: a module for transcription and translation

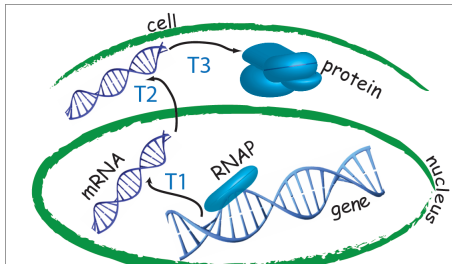
```
module exp(comp nuc, spec gene{act:bool}, spec prot)
{
  spec mrna;
  pattern geneAct = gene{act=true};
}
```



# Example: a module for transcription and translation

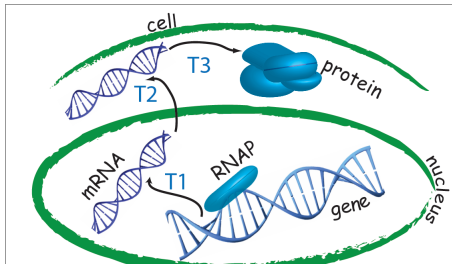
```
module exp(comp nuc, spec gene{act:bool}, spec prot)
{
  spec mrna;
  pattern geneAct = gene{act=true};

  nuc[
    geneAct + rnap <-> geneAct-rnap
    | geneAct-rnap -> geneAct + rnap + mrna
  ]
}
```



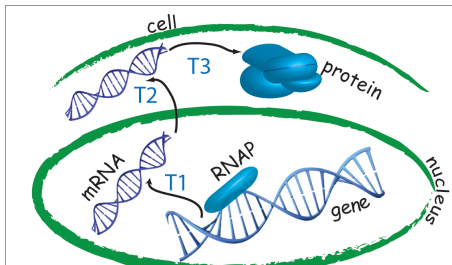
# Example: a module for transcription and translation

```
spec rnap;  
module exp(comp nuc, spec gene{act:bool}, spec prot)  
{  
  spec mrna;  
  pattern geneAct = gene{act=true};  
  
  nuc[  
    geneAct + rnap <-> geneAct-rnap  
    | geneAct-rnap -> geneAct + rnap + mrna  
  ]  
  
}
```



## Example: a module for transcription and translation

```
spec rnap;  
module exp(comp nuc, spec gene{act:bool}, spec prot)  
{  
  spec mrna;  
  pattern geneAct = gene{act=true};  
  
  nuc[  
    geneAct + rnap <-> geneAct-rnap  
    | geneAct-rnap -> geneAct + rnap + mrna  
  ]  
  
  | nuc[mrna] -> mrna  
  | mrna -> prot  
}
```



## Example continued: using the module

```
(* compartment declarations: *)  
comp cell      vol 1E-14;  
comp nucleus vol 1E-15 inside cell;
```

## Example continued: using the module

*(\* compartment declarations: \*)*

**comp** cell **vol** 1E-14;

**comp** nucleus **vol** 1E-15 **inside** cell;

*(\* species declarations: \*)*

**spec** gene1{act:**bool**};

**spec** gene2{act:**bool**, dna:**string**};

**spec** protein1;

**spec** protein2;

## Example continued: using the module

```
(* compartment declarations: *)
```

```
comp cell      vol 1E-14;
```

```
comp nucleus vol 1E-15 inside cell;
```

```
(* species declarations: *)
```

```
spec gene1{act:bool};
```

```
spec gene2{act:bool, dna:string};
```

```
spec protein1;
```

```
spec protein2;
```

```
(* use module to produce two proteins inside cell: *)
```

```
cell[
```

```
    exp(nuc, gene1, protein1)
```

```
    |exp(nuc, gene2, protein2)
```

```
]
```