

# Statistical Model-Checking for Biological Systems

Thomas Anung Basuki

UNU-IIST and University of Pisa

Macau, January 20, 2009

# Outline

- Introduction: Background, Motivation and related works
- Progress report: What I have done
- Future and On Going Work: What I am doing and plan to do

# INTRODUCTION

# Background

- Modelling biological systems
  - concurrency theory to model many interacting processes
  - system structure (e.g. compartment) cannot be modelled
- P Systems, Brane Calculi, Calculi of Looping Sequences
- Calculi of Looping Sequences models computations inside, outside and on the membrane
  - More flexible to model biological systems

# Motivation

- A simulator for Stochastic CLS has been developed in Pisa (Guido Scatena's Master Thesis)
- Could we do another kind of analysis?
- *Model-checking*
- Model-Checking for Stochastic CLS models?

# Calculi of Looping Sequences

A class of formalisms developed in Pisa for modelling biological systems.

- Calculus of Looping Sequences (CLS), for high-level modelling
- CLS with links (LCLS), for protein modelling in domain level
- Stochastic CLS, for stochastic modelling
- Spatial CLS, for modelling spatial information

# CLS Terms and Sequences

- Terms

$$T ::= S \quad | \quad (T)^L \quad | \quad T \mid T$$

# CLS Terms and Sequences

- Terms

$$T ::= S \mid (T)^L \mid T \mid T$$

- Sequences

$$S ::= \epsilon \mid a \mid S \cdot S$$

where  $\mathcal{E}$  is the alphabet,  $a \in \mathcal{E}$  and  $\epsilon$  is the empty sequence

# CLS Terms and Sequences

- Terms

$$T ::= S \quad | \quad (T)^L \rfloor T \quad | \quad T | T$$

- Sequences

$$S ::= \epsilon \quad | \quad a \quad | \quad S \cdot S$$

where  $\mathcal{E}$  is the alphabet,  $a \in \mathcal{E}$  and  $\epsilon$  is the empty sequence

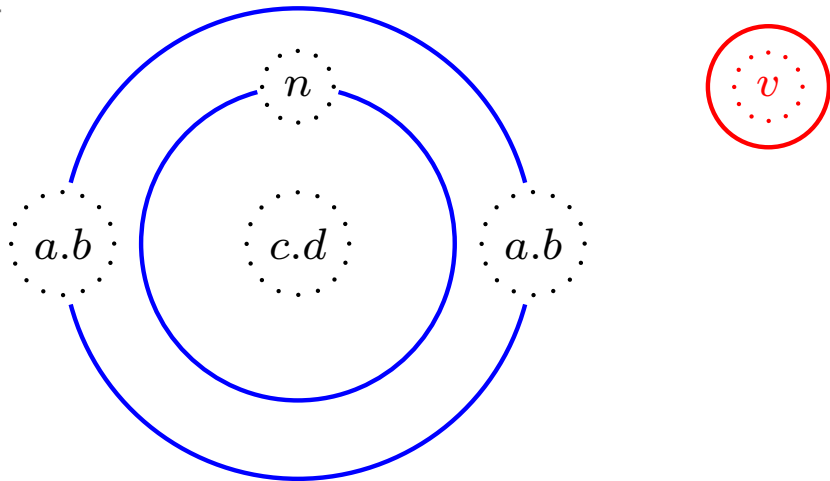
- Operators

$T | T$  : parallel composition

$(T)^L \rfloor T$  : looping and containment

$S \cdot S$  : sequence

# Term example



CLS Term:

$$((a.b \mid a.b)^L \mid ((n)^L \mid c.d)) \mid (\epsilon)^L \mid v$$

# CLS Patterns

- enrich terms with variables
- $P ::= S \mid (P)^L \mid P \mid P \mid X$   
where  $\mathcal{V}$  is a set of variables and  $X \in \mathcal{V}$ .

# CLS Patterns

- enrich terms with variables
- $P ::= S \mid (P)^L \mid P \mid P \mid X$   
where  $\mathcal{V}$  is a set of variables and  $X \in \mathcal{V}$ .

$P\sigma$  denotes the term obtained by instantiating all variables in  $P$  with corresponding terms.

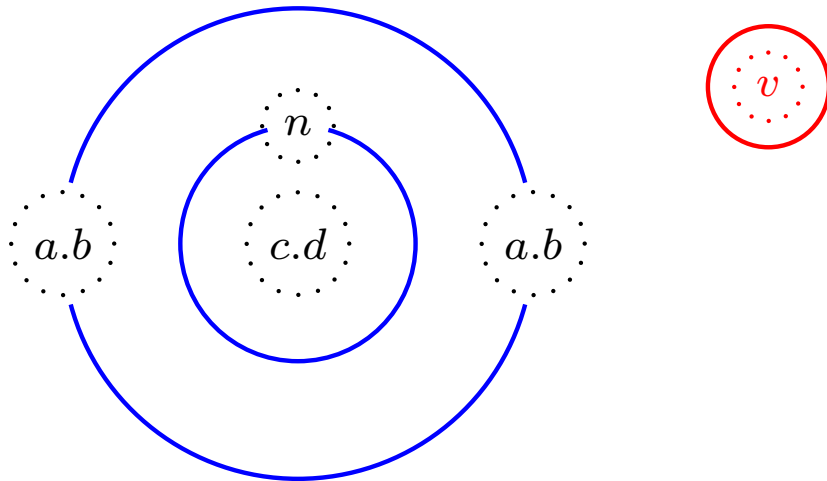
# CLS Rewrite Rules

A *rewrite rule* is a pair  $(P_1, P_2)$ , denoted with  $P_1 \mapsto P_2$ , such that

- $P_1$  and  $P_2$  are patterns
- $Var(P_2) \subseteq Var(P_1)$

A rule  $P_1 \mapsto P_2$  can be applied to all terms  $P_1\sigma$ .

# Rewrite Rule Example

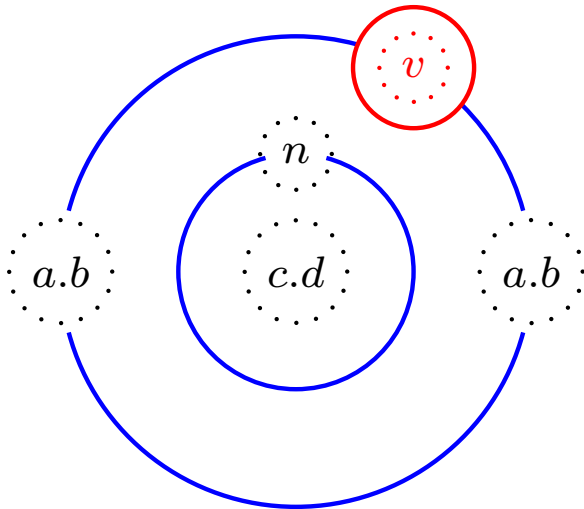


$$R_1: (X)^L \rfloor Y \mid (\epsilon)^L \rfloor v \mapsto (X \mid (\epsilon)^L \rfloor v)^L \rfloor Y$$

$$R_2: (X \mid (\epsilon)^L \rfloor v)^L \rfloor Y \mapsto (X)^L \rfloor (Y \mid v)$$

$$((a.b \mid a.b)^L \rfloor ((n)^L \rfloor c.d)) \mid (\epsilon)^L \rfloor v \xrightarrow{R_1}$$

# Rewrite Rule Example



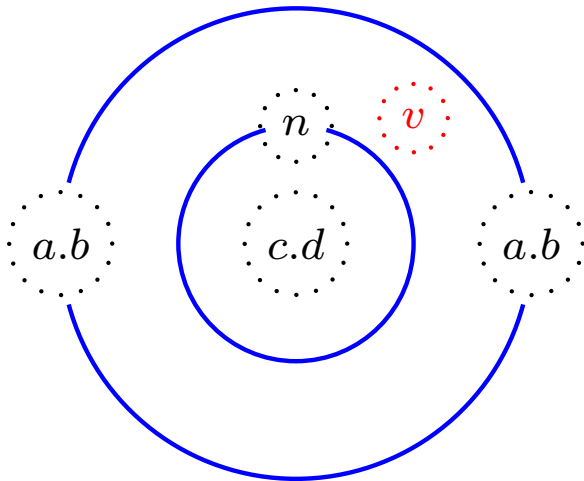
$$R_1: (X)^L \mid Y \mid (\epsilon)^L \mid v \mapsto (X \mid (\epsilon)^L \mid v)^L \mid Y$$

$$R_2: (X \mid (\epsilon)^L \mid v)^L \mid Y \mapsto (X)^L \mid (Y \mid v)$$

$$((a.b \mid a.b)^L \mid ((n)^L \mid c.d)) \mid (\epsilon)^L \mid v \xrightarrow{R_1}$$

$$(a.b \mid a.b \mid (\epsilon)^L \mid v)^L \mid ((n)^L \mid c.d) \xrightarrow{R_2}$$

# Rewrite Rule Example



$$R_1: (X)^L \rfloor Y \mid (\epsilon)^L \rfloor v \mapsto (X \mid (\epsilon)^L \rfloor v)^L \rfloor Y$$

$$R_2: (X \mid (\epsilon)^L \rfloor v)^L \rfloor Y \mapsto (X)^L \rfloor (Y \mid v)$$

$$((a.b \mid a.b)^L \rfloor ((n)^L \rfloor c.d)) \mid (\epsilon)^L \rfloor v \xrightarrow{R_1}$$

$$(a.b \mid a.b \mid (\epsilon)^L \rfloor v)^L \rfloor ((n)^L \rfloor c.d) \xrightarrow{R_2}$$

$$(a.b \mid a.b)^L \rfloor (((n)^L \rfloor c.d) \mid v)$$

# Stochastic CLS

Stochastic CLS patterns are given by the following grammar:

$$P_L ::= S \mid (P_X)^L \mid P_X \mid P_L \mid P_L$$

$$P_X ::= P_L \mid P_L \mid X$$

$$P_R ::= S \mid (P_R)^L \mid P_R \mid P_R \mid P_R \mid X$$

where  $X \in \mathcal{V}$ .

# Stochastic CLS

Stochastic CLS patterns are given by the following grammar:

$$P_L ::= S \mid (P_X)^L \mid P_X \mid P_L \mid P_L$$

$$P_X ::= P_L \mid P_L \mid X$$

$$P_R ::= S \mid (P_R)^L \mid P_R \mid P_R \mid P_R \mid X$$

where  $X \in \mathcal{V}$ .

A rewrite rule is a triplet  $(P_1, k, P_2)$  where

- $P_1$  and  $P_2$  are patterns
- $k \in \mathbb{R}$  is a kinetic constant
- $Var(P_2) \subseteq Var(P_1)$

# Gillespie's SSA

- is incorporated into Stochastic CLS semantics

# Gillespie's SSA

- is incorporated into Stochastic CLS semantics
- chemical solution with a set of reactions  $\{R_1, \dots, R_M\}$

# Gillespie's SSA

- is incorporated into Stochastic CLS semantics
- chemical solution with a set of reactions  $\{R_1, \dots, R_M\}$
- $X_1, \dots, X_N$  represent molecular population

# Gillespie's SSA

- is incorporated into Stochastic CLS semantics
- chemical solution with a set of reactions  $\{R_1, \dots, R_M\}$
- $X_1, \dots, X_N$  represent molecular population
- for each  $R_\mu$  there is a kinetic constant  $k_\mu$

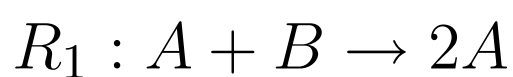
# Gillespie's SSA

- is incorporated into Stochastic CLS semantics
- chemical solution with a set of reactions  $\{R_1, \dots, R_M\}$
- $X_1, \dots, X_N$  represent molecular population
- for each  $R_\mu$  there is a kinetic constant  $k_\mu$
- reaction rate  $a_\mu = k_\mu \times h_\mu$  (in *moles.s*<sup>-1</sup>)  
where  $h_\mu$  = number of possible combinations of reactants

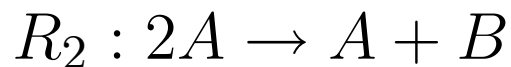
# Gillespie's SSA

- is incorporated into Stochastic CLS semantics
- chemical solution with a set of reactions  $\{R_1, \dots, R_M\}$
- $X_1, \dots, X_N$  represent molecular population
- for each  $R_\mu$  there is a kinetic constant  $k_\mu$
- reaction rate  $a_\mu = k_\mu \times h_\mu$  (in *moles.s*<sup>-1</sup>)  
where  $h_\mu$  = number of possible combinations of reactants

Ex:  $X_1$  molecules of  $A$  and  $X_2$  molecules of  $B$



$$a_1 = \binom{X_1}{1} \binom{X_2}{1} k_1 = X_1 X_2 k_1$$



$$a_2 = \binom{X_1}{2} k_2 = \frac{X_1(X_1-1)k_2}{2}$$

# Next Reaction in SSA

if  $t$  is the current time

- $t + \tau$  is the time next reaction occurs with  $\tau$  is a random variable exponentially distributed with parameter  $\sum_{v=1}^M a_v$
- every  $R_\mu$  has a probability  $\frac{a_\mu}{\sum_{v=1}^M a_v}$  to occur

# Next Reaction in SSA

if  $t$  is the current time

- $t + \tau$  is the time next reaction occurs with  $\tau$  is a random variable exponentially distributed with parameter  $\sum_{v=1}^M a_v$
- every  $R_\mu$  has a probability  $\frac{a_\mu}{\sum_{v=1}^M a_v}$  to occur

Direct Method to choose  $\tau$  and  $\mu$

- Generate two random numbers  $r_1$  and  $r_2 \in [0, 1]$
- $\tau = \frac{1}{\sum_{v=1}^M a_v} \ln\left(\frac{1}{r_1}\right)$
- Choose  $\mu$  such that  $\sum_{v=1}^{\mu-1} a_v < r_2 \sum_{v=1}^M a_v \leq \sum_{v=1}^{\mu} a_v$

# PROGRESS REPORT

# Model-Checking Stochastic CLS

- Real-Time Maude is a suitable tool
  - rewriting logic
  - support the notion of time
  - support model checking
- Statistical model checking
  - applying SSA to simulate biological systems
  - sampling the state space by executing simulation for a number of times
  - model check the result

# Modified SSA

Given  $\{R_1, \dots, R_M\}$  and  $X_1, \dots, X_N$

**Step 0** Initialise  $t$  and  $tstep$  to 0.

**Step 1** Compute  $a_1, \dots, a_M$  and  $a_0 = \sum_{v=1}^M a_v$ .

**Step 2** Compute  $\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$

**Step 3** While  $t + \tau \geq tstep$  do increase  $tstep$  by  $\Delta t$

    Compute  $\mu$  such that  $\sum_{v=1}^{\mu-1} a_v < r_2 a_0 \leq \sum_{v=1}^{\mu} a_v$  and  
    increase  $t$  by  $\tau$ .

**Step 4** Execute  $R_\mu$  and update  $X$  and  $a$  accordingly .

**Step 5** Compute  $\sum_{v=1}^M a_v$ . Return to **Step 2**.

# Computing Reactant Combinations

- define a function *occ* to count combinations of reactants

# Computing Reactant Combinations

- define a function *occ* to count combinations of reactants
- minimise recursive calls to optimise execution time

# Computing Reactant Combinations

- define a function *occ* to count combinations of reactants
- minimise recursive calls to optimise execution time
  - define a normal form for terms, that group similar terms in parallel composition

# Computing Reactant Combinations

- define a function *occ* to count combinations of reactants
- minimise recursive calls to optimise execution time
  - define a normal form for terms, that group similar terms in parallel composition
  - represented as the term and the number of occurrences

# Computing Reactant Combinations

- define a function *occ* to count combinations of reactants
- minimise recursive calls to optimise execution time
  - define a normal form for terms, that group similar terms in parallel composition
  - represented as the term and the number of occurrences
  - define *occ* in the normal form

# The Normal Form

- Grouped Term

$$GT ::= BT^N \quad | \quad GT \mid GT$$

$$BT ::= S \quad | \quad (GT)^L \rfloor GT$$

where  $N$  is a natural number

- every CLS term can be represented as a grouped term

# The Normal Form

- Grouped Term

$$GT ::= BT^N \quad | \quad GT \mid GT$$

$$BT ::= S \quad | \quad (GT)^L \quad \rfloor \quad GT$$

where  $N$  is a natural number

- every CLS term can be represented as a grouped term

Grouped Term subset ( $\subseteq$ )

(i)  $BT^N \subseteq BT^M \mid GT$  if  $N \leq M$

(ii)  $BT^N \mid GT \subseteq BT^M \mid GT_1$  if  $N \leq M$  and  $GT \subseteq GT_1$

(iii)  $GT \not\subseteq GT_1$  if both (i) and (ii) are not satisfied

# Layered Terms

Set of layers of term T:

- T itself
- subterms of T that are operands of  $(\_)^L \mid \_$

Example:

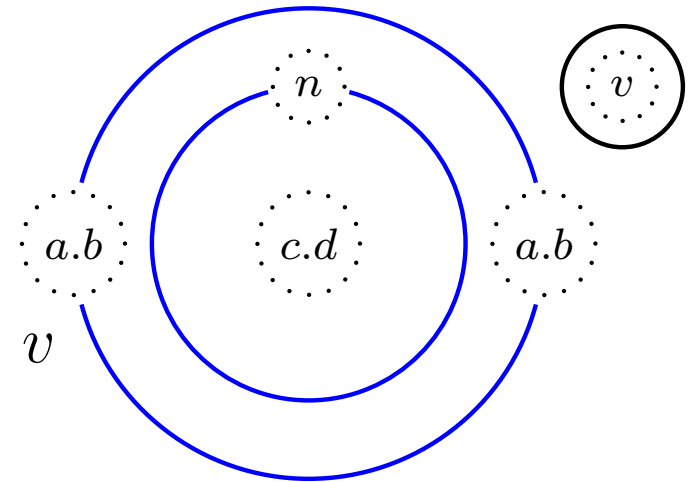
$$((a.b \mid a.b)^L \mid ((n)^L \mid c.d)) \mid (\epsilon)^L \mid v$$

Layers:

1)  $(a.b \mid a.b)^L \mid ((n)^L \mid c.d)$  and  $(\epsilon)^L \mid v$

2)  $a.b \mid a.b$  and  $(n)^L \mid c.d$

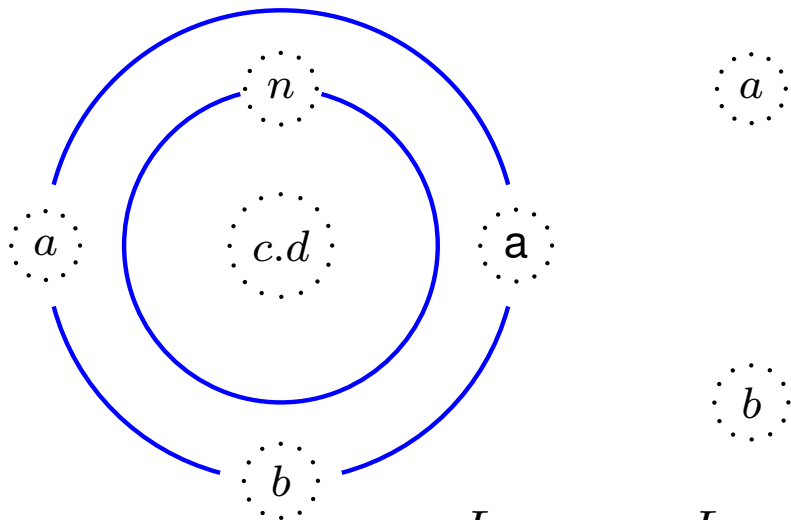
3)  $n$  and  $c.d$



# Computing *occ* in layers

Computing reactant combinations consists of two parts:

- computing in the same layer
- computing in the inner layer
- example:



$$occ(a \mid b, (a \mid a \mid b)^L \mid ((n)^L \mid c.d) \mid a \mid b)$$

# Defining *occ*

Function *occ* is defined as follows:

$$occ(GT, \epsilon) = 0 \text{ if } GT \not\equiv \epsilon \qquad occ(\epsilon, GT) = 1$$

$$occ(GT_1, GT_2) = occ(GT_1, GT_3) \text{ if } GT_2 \equiv GT_3$$

$$occ(BT^M | GT_1, BT^N | GT_2) = \\ \binom{N}{M} \times occ'(GT_1, GT_2) + occ(BT^M | GT_1, GT_2)$$

$$occ(BT^M | GT_1, ((GT_2)^L \rfloor GT_3)^N | GT_4) = \\ N \times (occ(BT^M | GT_1, GT_2) + occ(BT^M | GT_1, GT_3)) + \\ occ(BT^M | GT_1, GT_4) \text{ if } BT^M \not\subseteq ((GT_2)^L \rfloor GT_3)^N | GT_4$$

$$occ(BT^M | GT_1, GT_2) = 0$$

$$\text{if } BT^M \not\subseteq GT_2 \text{ and } \forall GT_3, GT_4, N. ((GT_3)^L \rfloor GT_4)^N \not\subseteq GT_2$$

# Definition of $occ'$

- function  $occ'$  is used to compute in the same layer:

$$occ'(GT, \epsilon) = 0 \text{ if } GT \neq \epsilon$$

$$occ'(\epsilon, GT) = 1$$

$$occ'(GT_1, GT_2) = occ'(GT_1, GT_3) \text{ if } GT_2 \equiv GT_3$$

$$occ'(BT^M | GT_1, BT^N | GT_2) = \binom{N}{M} \times occ'(GT_1, GT_2)$$

$$occ'(BT^M | GT_1, GT_2) = 0 \text{ if } BT^M \not\subseteq GT_2$$

# Translating to Maude

1. Define a Maude module for CLS syntax
  - Define `occ` and `occ'`
2. Define a Maude module containing data structure for modified SSA
3. Define a Maude module for the biological system
  - Define each step of modified SSA as a Maude rule
  - Define each rewrite rule as a Maude rule
4. Define a Maude module for initialising the biological system and sampling its state space

# Analysis

1. Statistical model checking
  - Define a Maude module for logical properties
  - Check them on the samples
2. Search the earliest time an event occurs

# Application and Result

- We apply the approach to *lactose operon* case study

# Application and Result

- We apply the approach to *lactose operon* case study
  - *lactose operon* is the sequence of genes in *E. Coli* that controls transportation and metabolism of lactose

# Application and Result

- We apply the approach to *lactose operon* case study
  - *lactose operon* is the sequence of genes in *E. Coli* that controls transportation and metabolism of lactose
  - is responsible for producing three enzymes
    - *lactose permease*, transports lactose from the environment into the cell
    - $\beta$ -*galactosidase*, hydrolises lactose into glucose and galactose
    - *transacetylase*

# Application and Result

- We apply the approach to *lactose operon* case study
- We check how the presence of *LACTOSE* influence the number of resultant enzymes

# Application and Result

- We apply the approach to *lactose operon* case study
  - We check how the presence of *LACTOSE* influence the number of resultant enzymes
  - Define a predicate  $IsGreaterN(X,N)$  which states that  $X$  is greater than  $N$  molecules
  - Find a number  $N$ , such that for a certain initial state:
    - the system always satisfies  $\diamond IsGreaterN(X,N)$  if *LACTOSE* is present
    - the system never satisfies  $IsGreaterN(X,N)$  if *LACTOSE* is absent
- with  $X$  is the enzymes number of molecules

# Publications

- T.A. Basuki, A. Cerone, A. Griesmayer and R. Schlatte. *Model-checking user behaviour using interacting components*. to appear in Formal Aspects of Computing, 2009.
- T.A. Basuki, A. Cerone and P. Milazzo. *Translating Stochastic CLS into Maude*. Proceedings of MECBIC 2008, ENTCS 227 pp 37 - 58, 2009.

# FUTURE WORK

# On Going Work

Applying the same approach to model-check Spatial CLS.  
Challenges:

- Separating the spatial information from the terms
- Modify the Gillespie's SSA to deal with autonomous movement of elements in the system
- Implementing rearrangement algorithm efficiently

# Improvement in The Query

- Currently only querying:
  - yes or no answer
  - time related information
- Extend the query to include probabilistic notions (e.g. what is the probability of an event to occur)
- Define a case study to show the applicability of statistical model checking in biological systems

THANK YOU

# CLS Structural Congruences

Least congruence relations

- $\equiv_S$  on sequences
- $\equiv$  on terms

such that

# CLS Structural Congruences

Least congruence relations

- $\equiv_S$  on sequences
- $\equiv$  on terms

such that

- $S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3$
- $S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S$

# CLS Structural Congruences

Least congruence relations

- $\equiv_S$  on sequences
- $\equiv$  on terms

such that

- $S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3$
- $S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S$
- $T_1 \mid T_2 \equiv T_2 \mid T_1$
- $T_1 \mid (T_2 \mid T_3) \equiv (T_1 \mid T_2) \mid T_3$
- $T \mid \epsilon \equiv T$

# Gillespie's SSA

Given  $\{R_1, \dots, R_M\}$  and  $X_1, \dots, X_N$

**Step 0** Initialise  $t$  to 0.

**Step 1** Compute  $a_1, \dots, a_M$  and  $a_0 = \sum_{v=1}^M a_v$ .

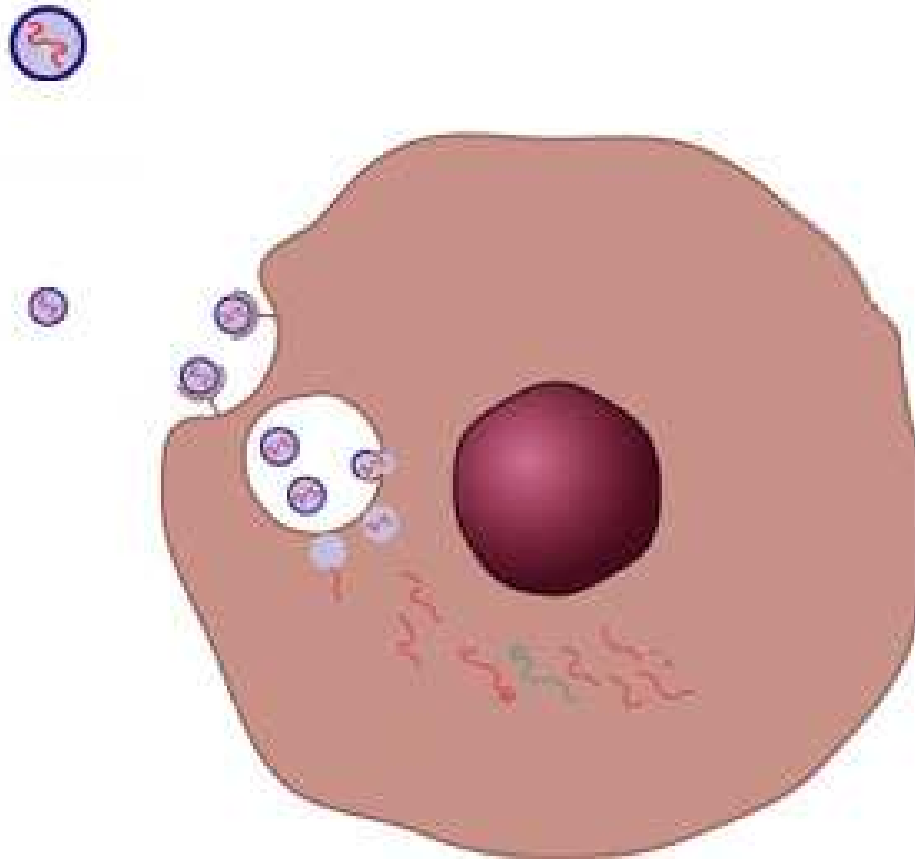
**Step 2** Compute  $\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$  and  $\mu$  such that

$$\sum_{v=1}^{\mu-1} a_v < r_2 a_0 \leq \sum_{v=1}^{\mu} a_v.$$

**Step 4** Execute  $R_\mu$  and update  $X$  accordingly. Increase  $t$  by  $\tau$

**Step 5** Return to **Step 1**.

# Virus attack



# Computing occ example

$$\text{Ex: } \text{occ}(a | b, (a | a | b)^L \rfloor ((n)^L \rfloor c.d) | a | b) =$$

$$\binom{1}{1} \times \text{occ}'(b, (a|a|b)^L \rfloor ((n)^L \rfloor c.d)|b) + \text{occ}(a|b, (a|a|b)^L \rfloor ((n)^L \rfloor c.d)|b) =$$

$$\binom{1}{1} \times \text{occ}'(\epsilon, (a|a|b)^L \rfloor ((n)^L \rfloor c.d)) + \binom{1}{1} \times \text{occ}'(a, (a|a|b)^L \rfloor ((n)^L \rfloor c.d))$$

$$+ \text{occ}(a|b, (a|a|b)^L \rfloor ((n)^L \rfloor c.d)) =$$

$$1 + 0 + \text{occ}(a|b, a|a|b) + \text{occ}(a|b, (n)^L \rfloor c.d) =$$

$$1 + \binom{2}{1} \times \text{occ}'(b, b) + \text{occ}(a|b, b) + \text{occ}(a|b, n) + \text{occ}(a|b, c.d) =$$

$$1 + 2 = 3$$