BMC Systems Biology

Research article

Simulation-based model checking approach to cell fate specification during *Caenorhabditis elegans* vulval development by hybrid functional Petri net with extension

Chen Li†, Masao Nagasaki*†, Kazuko Ueno and Satoru Miyano

Address: Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan

Email: Chen Li - chenli@ims.u-tokyo.ac.jp; Masao Nagasaki* - masao@ims.u-tokyo.ac.jp; Kazuko Ueno - uepi@ims.u-tokyo.ac.jp; Satoru Miyano - miyano@ims.u-tokyo.ac.jp

* Corresponding author †Equal contributors

Published: 27 April 2009

BMC Systems Biology 2009, **3**:42 doi:10.1186/1752-0509-3-42

Received: 18 August 2008
Accepted: 27 April 2009

This article is available from: http://www.biomedcentral.com/1752-0509/3/42

Abstract

Background: Model checking approaches were applied to biological pathway validations around 2003. Recently, Fisher *et al.* have proved the importance of model checking approach by inferring new regulation of signaling crosstalk in *C. elegans* and confirming the regulation with biological experiments. They took a discrete and state-based approach to explore all possible states of the system underlying vulval precursor cell (VPC) fate specification for desired properties. However, since both discrete and continuous features appear to be an indispensable part of biological processes, it is more appropriate to use quantitative models to capture the dynamics of biological systems. Our key motivation of this paper is to establish a quantitative methodology to model and analyze *in silico* models incorporating the use of model checking approach.

Results: A novel method of modeling and simulating biological systems with the use of model checking approach is proposed based on hybrid functional Petri net with extension (HFPNe) as the framework dealing with both discrete and continuous events. Firstly, we construct a quantitative VPC fate model with 1761 components by using HFPNe. Secondly, we employ two major biological fate determination rules – Rule I and Rule II – to VPC fate model. We then conduct 10,000 simulations for each of 48 sets of different genotypes, investigate variations of cell fate patterns under each genotype, and validate the two rules by comparing three simulation targets consisting of fate patterns obtained from *in silico* and *in vivo* experiments. In particular, an evaluation was successfully done by using our VPC fate model to investigate one target derived from biological experiments involving hybrid lineage observations. However, the understandings of hybrid lineages are hard to make on a discrete model because the hybrid lineage occurs when the system comes close to certain thresholds as discussed by Sternberg and Horvitz in 1986. Our simulation results suggest that: Rule I that cannot be applied with qualitative based model checking, is more reasonable than Rule II owing to the high coverage of predicted fate patterns (except for the genotype of *lin-15ko; lin-12ko* double mutants). More insights are also suggested.

Conclusion: The quantitative simulation-based model checking approach is a useful means to provide us valuable biological insights and better understandings of biological systems and observation data that may be hard to capture with the qualitative one.

Background

Model checking is a successful method for automatic verification of software and reactive systems [1], which is usually applied to ensure consistency and correctness of designed models. Practiced verification methods in most cases are still simple simulation and testing. While simple simulation and testing provide a part of the possible results of a model, model checking can conduct an exhaustive exploration of all possible behaviors [1-4].

Among the features, one merit of using model checking is that it is possible to verify a set of rules defined by users. Recently, the size of targeting network becomes enlarged and difficult to check all rules and their combinations by each user, especially, biologist. To solve this, around 2003, several attempts were launched to apply model checking approaches to biological pathway validations [5-12]. With the aid of model checking approach, one can obtain answers to questions such as "what is the probability that the gene finally expressed?" and "does this reaction always lead to DNA fragmentation?" In 2007, Fisher *et al*. proved the importance of model checking approach by inferring the new regulation of inductive and lateral signaling crosstalk of *C. elegans* and confirming the regulation with biological experiments [9]. The approach was applied on a discrete model by using one of the model checking languages named *reactive modules* [13]. However, quantitative properties (e.g. continuous feature) are also important in biological processes, such as the concentration of proteins and the reaction rates. Thus, it is desirable to deal with both discrete and continuous features in the model (called *hybrid model*). From this fact, the next challenge is to apply the model checking approach to such hybrid models. Several hybrid models have been applied to biological pathway modeling, e.g. hybrid automata [6], π-calculus [14], ordinary differential equations (ODEs) [15], and hybrid functional Petri nets (HFPN) [16] and its extension (HFPNe) [17,18]. Among them, HFPNe and its related concepts have been accepted as a formal modeling method due to potential advantages of HFPNe possessing intuitive graphical representation and capabilities for mathematical analysis [19-23].

We use HFPNe to quantitatively model *C. elegans* vulval development mechanism, which best meets the features of biological processes. We have been developing an HFPNe based software "Cell Illustrator" [24,25] for modeling and simulating biological pathways [17,18,26]. It has been successfully employed to develop and analyze some pathway models for gene regulatory networks, metabolic pathways, and signaling pathways [16,27-31]. In this paper, Cell Illustrator is used as a software tool to model and simulate this complicated biological system in *C. elegans*.

The paper is organized as follows: In **Methods**, we first present an introduction of HFPNe and biological background of VPC fate determination mechanism. We demonstrate how to construct VPC fate model by using HFPNe afterwards. When determining cell fate from biological points of view, several biological fate determination rules can be considered. We thus employ two major biological fate determination rules – **Rule I** and **Rule II** – to the VPC fate model. Two rules are used to determine the cell fate from two viewpoints – temporal interval and temporal order – of time course expression of cell fate candidates. Next, we conduct 10,000 simulations for each of 48 sets of different genotypes which are the combination of four mutants and AC (anchor cell). The simulation procedures such as noise parameters, high-speed simulation engine, and the emulation of the temporal stimulations are also described. Finally, we examine the consistency and correctness of the VPC fate model, and evaluate proposed two rules by comparing with three simulation targets consisting of predicted fate patterns obtained from *in silico* and *in vivo* experiments. In **Results and Discussion**, our simulation results suggest that **Rule I** on the temporal interval is more reasonable than **Rule II** owing to the high coverage of predicted fate patterns (except for *lin-15ko; lin-12ko* double mutants), (ii) for the *lin-15ko; lin-12ko* double mutants, the coverage will be considerably augmented, if the number of animal population is increased in the *in vivo* experiments, and (iii) unmatched fate patterns of *lin-15ko* and *ac-; lin-15ko*, still have the possibility to be examined in *in vivo* experiments by enlarging animal numbers. More insights concerning the hybrid lineage are also suggested and discussed. The final section concludes the paper and addresses the contributions of the work.

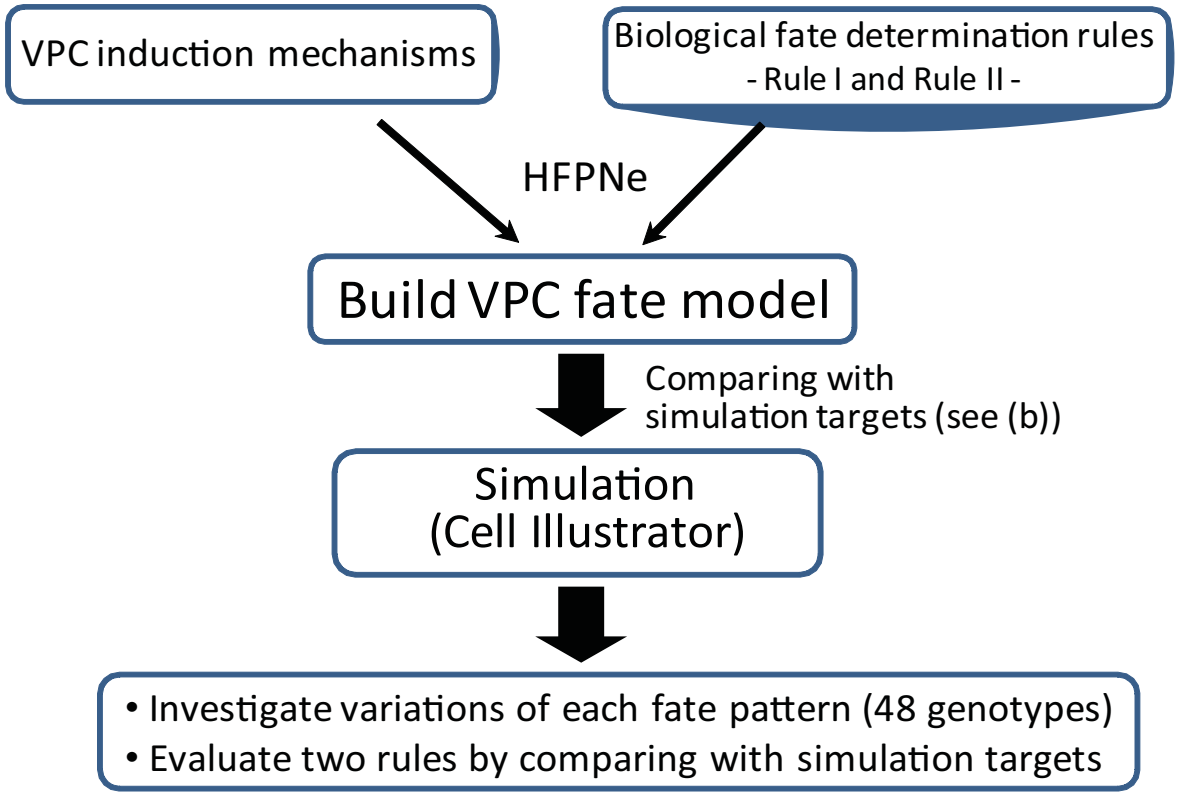
Methods

Figure 1(a) illustrates the procedure overview of **Methods**. Briefly, starting from an introduction of HFPNe and VPC induction mechanisms, we first show the processes of constructing HFPNe-based VPC fate model (VPC fate model for short). We then employ two major biological fate determination rules to the VPC fate model from different viewpoints: temporal interval and temporal order. Finally, we execute 480,000 simulations in total for different genotypes by comparing with three simulation targets (i.e., **JA**, **ST**, and **STA**) (see Figure 1(b)). **JA** consists of fate patterns obtained by using model checking approach (with MOCHA); **ST** is the fate patterns summarized by Sternberg and Horvitz [32]; and **STA** is the fate patterns derived from [32] including hybrid lineage data. The aims of the simulation are: (i) to investigate predicted fate patterns variations of each genotype, and (ii) to evaluate two rules employed to VPC fate model.

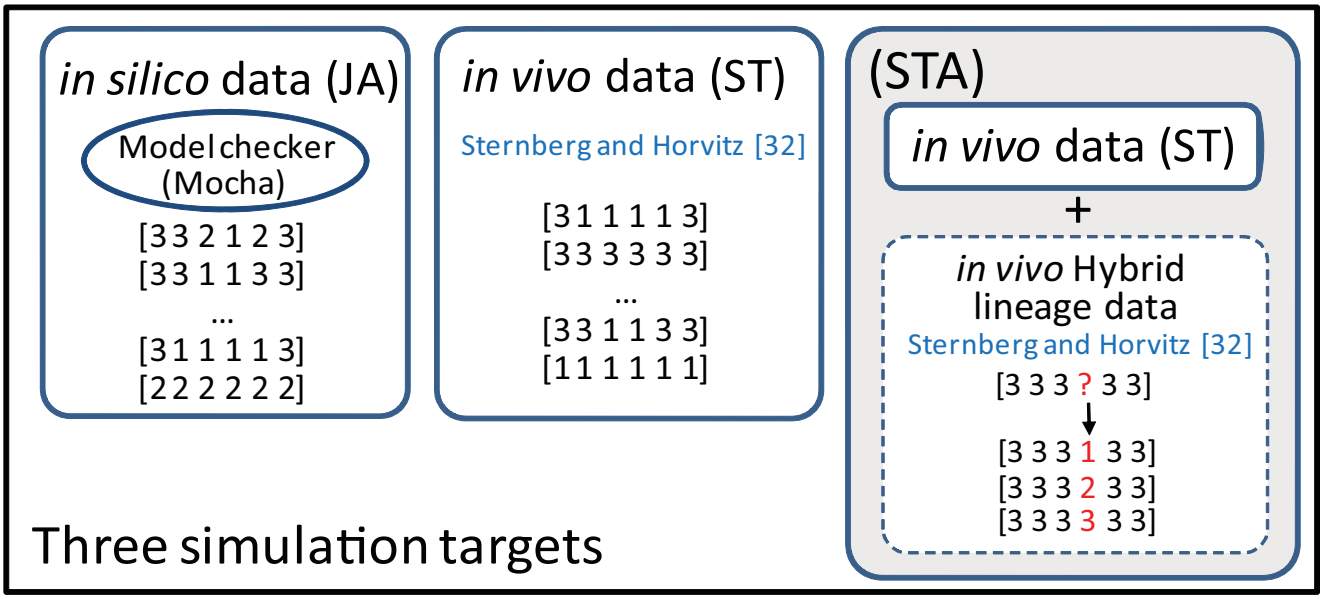
Modeling biological pathways with hybrid functional Petri net with extension (HFPNe)

Brief introduction of HFPNe: an enhanced Petri net for modeling biological interactions

Petri net is a network which consists of *place*, *transition*, *arc*, and *token*. A place can hold tokens as its content. A



(a)



(b)

Figure 1
(a) Procedure overview of the Methods section. (b) Schematic view of three simulation targets.

transition has arcs coming from places and arcs going out from the transition to some places. A transition with these arcs defines a firing rule in terms of the contents of the places where the arcs are attached [33].

Due to the limitation of conventional Petri net and more requirements in modeling, Matsuno *et al*. have defined *hybrid functional Petri net* (HFPN for short) in 2003 [16]. However, when modeling biological pathways, it has been noticed that several useful extensions should be applied for modeling and simulating more complicated biopathway processes (e.g., activities of enzymes for a multi-modification protein) and other biological processes that are not normally treated in biological pathways (e.g., alternative splicing and frameshifting) [18]. Therefore, Nagasaki *et al*. have proposed a new enhanced Petri net architecture *hybrid functional Petri net with extension* (HFPNe) in 2004. They have firstly used the new terminology in HFPNe to bridge the gap between the researchers of computer science and biology. In other words, the terms of place, transition, arc, and token are named as *entity*, *process*, *connector*, and *content* respectively.

HFPNe can deal with three types of data – discrete, continuous, and generic – and comprises three types of elements – entities, processes, and connectors – whose symbols are illustrated in Figure 2. A discrete entity holds a positive integer number of content. A discrete process is the same notion as used in the traditional discrete Petri net [33]. A continuous entity holds a nonnegative real number as concentration of a substance such as mRNA and protein. A continuous process is used to represent a biological reaction such as transcription and translation, at which the reaction speed is assigned as a parameter. A generic entity can hold various kinds of types including object, e.g., the string of nucleotide base sequence. A generic process can deal with any kind of operations (e.g., alternative splicing and frameshifting) to all types of entities. Connectors are classified into three types: process connector, associate connector, and inhibitory connector. Process connector connects an entity to a process or vice versa. Associate or inhibitory connector represents a condition and is only directed from an entity to a process. Each of process connector from an entity, associate connector, and inhibitory connector has a threshold by which the parameter assigned to the process at its head is controlled. A process connector from an entity or an associate connector (an inhibitory connector) can participate in activating (repressing) a process at its head, as far as the content of an entity at its tail is over the threshold. For either of asso-

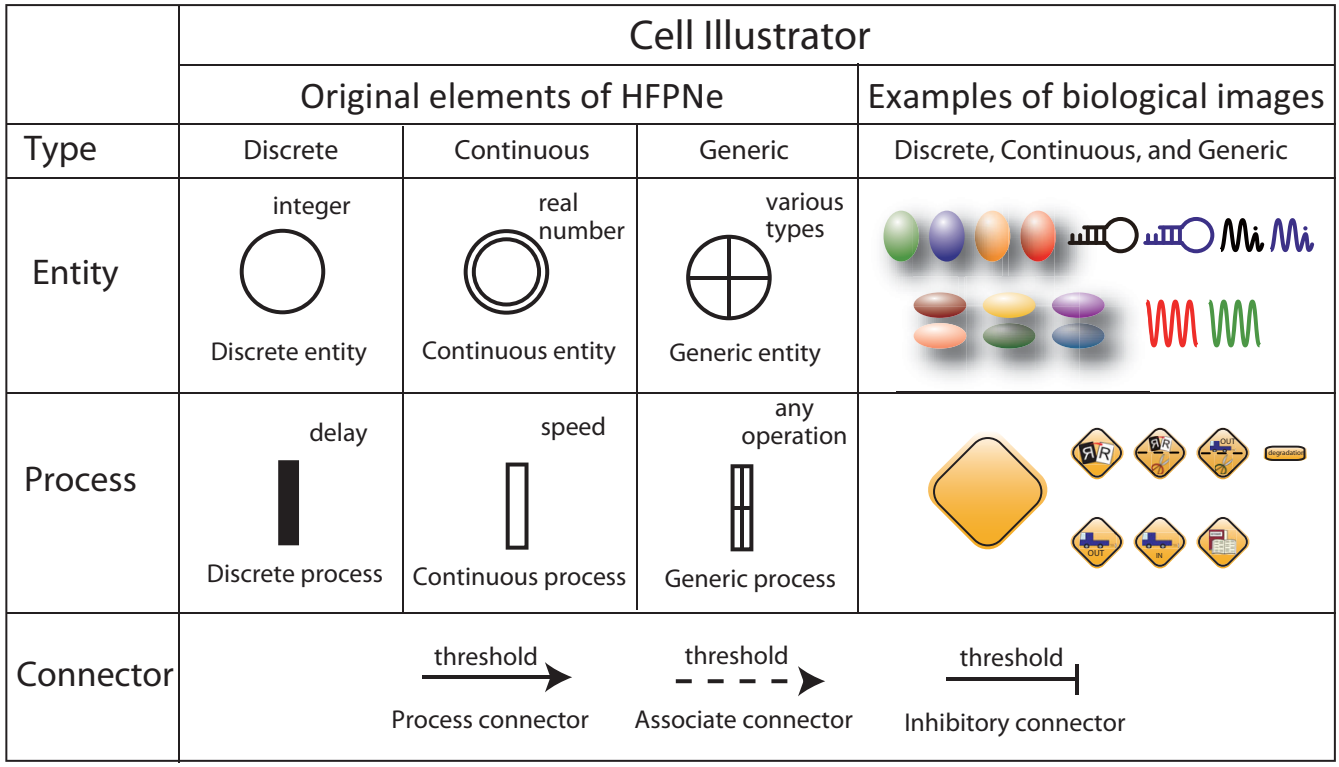


Figure 2
Basic elements of HFPNe and examples of biological icons in Cell Illustrator. In Cell Illustrator [24-26], HFPNe elements are replaced with the biological icons defined in the Cell System Ontology [47]. This replacement makes the HFPNe model of a biological pathway more comprehensible (see [17] for details).

ciate or inhibitory connectors, no amount is consumed from an entity at its tail. The followed section only gives the mathematic definitions of HFPNe. For the detailed formal definition and properties of HFPNe, the readers are suggested to refer to [18].

Basic definitions of HFPNe

For modeling complex biological processes intuitively, we are required to deal with various kinds of biological information, e.g. the density of molecules, the number of molecules, sequences, molecular modifications, binding location, localization of molecules, etc. To cope with this feature in biological system modeling, we introduce *types* for biological entities and processes.

The set T of *types* is defined by the following abstract syntax:

$$\langle type\rangle ::= \text{boolean} \,||\, \text{integer} \,||\, \text{integer+} \,||\, \text{real} \,||\, \text{real+} \,||\, \text{string} \,||\, \text{pair}(\langle type\rangle, \langle type\rangle) \,||\, \text{list}\ \langle type\rangle \,||\, \text{object}(\langle type\rangle, \cdots, \langle type\rangle).$$

Then, for $\theta \in T$, we define the *domain* $D(\theta)$ of θ as follows:

1. $D(\text{boolean}) = \{\text{true, false}\}$, $D(\text{integer}) = \mathbf{Z}$ (the set of integers), $D(\text{integer+}) = \mathbf{N}$ (the set of nonnegative integers), $D(\text{real}) = \mathbf{R}$ (the set of real numbers), $D(\text{real+}) = \mathbf{R}^{\geq 0}$ (the set of nonnegative real numbers), $D(\text{string}) = \mathbf{S}$ (the set of strings over some alphabet).

2. $D(\text{pair}(\theta_1, \theta_2)) = D(\theta_1) \times D(\theta_2)$.

3. $D(\text{list}\theta) = \cup_{k \geq 0} D(\theta)^k$.

4. $D(\text{object}(\theta_1, \bullet, \theta_n)) = D(\theta_1) \times \bullet \times D(\theta_n)$.

For convenience, we denote $D^* = \cup_{\theta \in T} D(\theta)$.

Let E be a finite set. A *type function* for E is a mapping $\tau: E \to T$. For $e \in E$, $\tau(e)$ is called the *type* of e. A *marking* of E is a mapping $M : E \to D^*$ satisfying $M(e) \in D(\tau(e))$ for $e \in E$. For $e \in E$, $M(e)$ called the *mark* of e. We denote by the set of all markings of E. We can regard as the set $\prod_{e \in E} D(\tau(e))$.

Consider a function $f : \to \mathbf{R}$. For a subset $F \subseteq E$ and an element $v \in \prod_{e \in F} D(\tau(e))$, let $f[F = v]: \prod_{e \in E\text{-}F} D(\tau(e)) \to \mathbf{R}$ be the function obtained from f by restricting the value for F to v, i.e. $f[F = v](z) = f(z, v)$ for $z \in \prod_{e \in E\text{-}F} D(\tau(e))$. Let F be a subset of E such that $e \in F$ satisfies $D(\tau(e)) = \mathbf{R}$ or $\mathbf{R}^{\geq 0}$. We say that the function f is *continuous for* F if $f[E - F = v]: \prod_{e \in F} D(\tau(e)) \to \mathbf{R}$ is continuous on $\prod_{e \in F} D(\tau(e))$ for any $v \in \prod_{e \in E\text{-}F} D(\tau(e))$.

Based on the above terminology, we define the notion of hybrid functional Petri net with extension (HFPNe). The basic idea of HFPNe is two-fold. The first is to introduce types with which we can deal with various data types. The second is to employ functions of marking $f(M)$ to determine the weight, delay, and speed, etc. which control the system behavior. In the following definition, we use different names instead of place, transition, arc, etc. which are conventionally used in Petri net theory since biological system modeling requires more intuitive names for representing biological entities and processes.

[Definition 1] We define a *hybrid functional Petri net with extension* (HFPNe) $H = (E, P, h, \tau, C, d, \alpha)$ as follows:

1. $E = \{e_1, \bullet, e_n\}$ is a non-empty finite set of *entities* and $P = \{p_1, \bullet, p_m\}$ is a non-empty finite set of *processes*, where we assume $E \cap P = \varnothing$.

2. $h : E \cup P \to \{\text{discrete, continuous, generic}\}$ is a mapping called the *hybrid function*. Terms "discrete" and "continuous" correspond to those in hybrid Petri net [16] and "generic" is a newly introduced name which can be of any type in T. A process $p \in P$ with $h(p) =$ discrete (resp., continuous, generic) is called a *discrete process* (resp., *continuous process*, *generic process*). An entity $e \in E$ with $h(e) =$ discrete (resp., continuous, generic) is called a *discrete entity* (resp., *continuous entity*, *generic entity*).

3. $\tau: E \to T$ is a type function for E such that $\tau(e) =$ integer+ if e is a discrete entity, and $\tau(e) =$ real+ if e is a continuous entity.

4. $C = (EP, PE, a, w, u)$ consists of subsets $EP \subseteq E \times P$ and $PE \subseteq P \times E$. An element in $EP \cup PE$ is called a *connector*. Each connector has a *connector type* which is given by a mapping $a : EP \cup PE \to \{\text{process, associate, inhibitor}\}$ called the *connector type function* which satisfies the conditions: (i) $a(c) =$ process for $c \in PE$. (ii) All connectors $c = (e, p) \in EP$ satisfy the conditions in Table 1(a) and all connectors $c = (p, e) \in PE$ satisfy the conditions in Table 1(b). A connector $c = (e, p) \in EP$ is called a *process connector* (resp., an *associate connector*, an *inhibitory connector*) if $a(c) =$ process (resp., associate, inhibitor). "Process connector", "associate connector" and "inhibitory connector" correspond to *normal arc*, *test arc* and *inhibitory arc*, respectively. For a connector $c = (p, e) \in PE$, $a(c) =$ process by definition and we also call it a *process connector*. We say that a connector $c = (e, p) \in EP$ is *discrete* (resp., *continuous*, *generic*) if p is a discrete process (resp., continuous process, generic process). In the same way, we also say that $c = (p, e) \in PE$ is *discrete* (resp., *continuous*, *generic*)

Table 1: The conditions of the connector type

	connector type		process connector			associate or inhibitory connector		
		process type	discrete	continuous	generic	discrete	continuous	generic
(a)	entity	discrete	X	-	X	X	X	X
	type	continuous	X	X	X	X	X	X
		generic	-	-	X	X	X	X
	connector type		process connector			associate or inhibitory connector		
		process type	discrete	continuous	generic	discrete	continuous	generic
(b)	entity	discrete	X	-	X	-	-	-
	type	continuous	X	X	X	-	-	-
		generic	X	X	X	-	-	-

Table 1: (a) For a connector $c = (e, p) \in EP$, the entity type $h(e)$, the process type $h(p)$ and the connector type $a(c)$ must satisfy the following conditions, where X means that the connection is allowed and – means that the connection is not allowed. (b) For a connector $c = (p, e) \in PE$, the connector type $a(c)$ is process by definition. The entity type $h(e)$ and the process type $h(p)$ must satisfy the following conditions, where X means that the connection is allowed and – means that the connection is not allowed.

if p is a discrete process (resp., continuous process, generic process). Let be the set of all markings of E and let F be the set of continuous entities in E. Then we denote $\mathcal{D}_{\text{discrete}} = \{f \mid f: \to \mathbf{N}\}$, $\mathcal{D}_{\text{continuous}} = \{f \mid f: \to \mathbf{R}^{\geq 0}$ is continuous for $F\}$, $\mathcal{D}_{\text{generic}} = \{f \mid f: \to D^*\}$, and $\mathcal{D}_{\text{boolean}} = \{f \mid f: \to \{\text{true, false}\}\}$.

Then w and u are given as follows:

(a) $w : EP \to \mathcal{D}_{\text{discrete}} \cup \mathcal{D}_{\text{continuous}} \cup \mathcal{D}_{\text{boolean}}$ is a function called the *activity function* such that for a connector $c \in EP$ (i) $w(c) \in \mathcal{D}_{\text{discrete}}$ if c is discrete, (ii) $w(c) \in \mathcal{D}_{\text{continuous}}$ if c is continuous, (iii) $w(c) \in \mathcal{D}_{\text{boolean}}$ if c is generic. For a connector (e, p), $w(e, p)$ be used as a function giving the threshold in discrete and continuous cases and the condition in generic case which is required for enabling the process p.

(b) $u : EP \cup PE \to \mathcal{D}_{\text{discrete}} \cup \mathcal{D}_{\text{continuous}} \cup \mathcal{D}_{\text{generic}}$ is a function called the *update function* which satisfies the following conditions: For a connector $c \in EP \cup PE$, let $c = (e, p) \in EP$ or $c = (p, e) \in PE$. (i) $u(c) \in \mathcal{D}_{\text{discrete}}$ if c is discrete. (ii) $u(c) \in \mathcal{D}_{\text{continuous}}$ if c is continuous. (iii) If c is generic, then $u(c)$ is a function in $\mathcal{D}_{\text{generic}}$ such that $u(c)(M)$ is in $D(\tau(e))$ for any marking $M \in$. For a connector $c = (e, p)$ or $c = (p, e)$, $u(c)$ is used as a function which will update the mark of e.

5. $d : P_{\text{discrete}} \to \mathcal{D}_{\text{continuous}}$ is a mapping called the *delay*, where P_{discrete} is the set of discrete processes in P. For a discrete process p, $d(p)$: $\to \mathbf{R}^{\geq 0}$ is called the *delay function* of p.

6. $\alpha > 0$ is a real number called the *generic time*. The generic time is used as the clock for generic processes.

We introduce a parameter $t \in \mathbf{R}^{\geq 0}$ called the *time* to a hybrid functional Petri net with extension $H = (E, P, h, \tau, C, d, \alpha)$. Given a marking I called the *initial marking*, we define a marking $M(t)$ called the *marking at time t* and a marking $M_r(t)$ called the *reserved marking at time t* for $t \geq 0$ in the following way.

By convention, we denote $M(e, t) = M(t)(e)$ and $M_r(e, t) = M_r(t)(e)$ for $e \in E$. We define $\tilde{M}(t)$ by $\tilde{M}(e, t) = M(e, t) - M_r(e, t)$ for discrete and continuous entities and $\tilde{M}(e, t) = M(e, t)$ for generic entities e.

First, we define $M(0) = I$, $M_r(e, 0) = 0$ for all discrete and continuous entities e. For all generic entities e, $M_r(e, t) =$ null (the empty list) for any $t \geq 0$. For $t > 0$, we define $M(t)$ and $M_r(t)$ in the following way.

For a process $p \in P$ at time t, if the following conditions are satisfied, then the process p is said to be *enabled* at time t. Otherwise the process is said to be *disabled* at time t.

1. If p is a discrete process, then for all connectors $c = (e, p) \in EP$ the following conditions hold:

(a) $\tilde{M}(e, t) \geq w(e, p)(M(t))$ if $a(c) \neq$ inhibitor.

(b) $\tilde{M}(e, t) < w(e, p)(M(t))$ if $a(c) =$ inhibitor.

2. If p is a continuous process, then for all connectors $c = (e, p) \in EP$ the following conditions hold:

(a) $\tilde{M}(e, t) \geq w(e, p)(M(t))$ if $a(c) \neq$ inhibitor.

(b) $\tilde{M}(e, t) \leq w(e, p)(M(t))$ if $a(c) =$ inhibitor.

3. If p is a generic process, then for all connectors $c = (e, p) \in EP$ the following conditions hold:

(a) $w(e, p)(\tilde{M}(t)) =$ true if $a(c) \neq$ inhibitor.

(b) $w(e, p)(\tilde{M}\, t)) =$ false if $a(c) =$ inhibitor.

If a disabled process turns to be enabled at time t, the process is said to be *triggered* at time t. If an enabled process turns to be disabled or a disabled process turns to be enabled at time t, the process is said to be *switched* at time t. If a discrete process p is triggered at time t, we say that the discrete process can be *fired* at time $t + d(p)(M(t))$. If a generic process p is triggered at time t, we say that the generic process can be *fired* at time $t + \alpha$.

For an entity $e \in E$ and time t, let $S_d(t)$ be the set of discrete processes which can be fired at time t, and let $U_d(t)$ be the set of discrete processes which are triggered at time t. For a discrete process p that can be fired at time t, we denote by $q(p, t)$ the time when p is triggered. Let $S_c(t)$ be the set of continuous processes which are enabled at time t. Let $S_g(t)$ be the set of generic processes which can be fired at time t.

Note that we can choose a sufficiently small $\varepsilon_t > 0$ such that in the interval $[t - \varepsilon_t, t)$, neither discrete nor generic process is triggered or can be fired and no continuous process is switched.

Also note the following facts:

1. $S_c(t - \varepsilon_t) = S_c(t')$ for any $t' \in [t - \varepsilon_t, t)$ since no continuous process is switched in the interval $[t - \varepsilon_t, t)$.

2. $\tilde{M}(t')$ is constant on $E - E_{\text{continuous}}$ in the interval $[t - \varepsilon_t, t)$ since neither discrete nor generic process is triggered or can be fired in the interval $[t - \varepsilon_t, t)$, where $E_{\text{continuous}} = \{e \in E \mid e \text{ is continuous}\}$.

3. For any continuous connector c, $u(c)(\tilde{M}(t'))$ is continuous on $[t - \varepsilon_t, t)$ since by definition $u(c)$ is continuous for $E_{\text{continuous}}$ and $\tilde{M}(t')$ is constant on $E - E_{\text{continuous}}$ in the interval $[t - \varepsilon_t, t)$.

Then $M(t)$ is defined by the following procedure:

1. $Tmp \leftarrow M(t - \varepsilon_t)$, $Tmp_r \leftarrow M_r(t - \varepsilon_t)$

2. **if** $t = \alpha k$ for some integer $k \geq 1$ **then**

for each generic process $p \in S_g(t)$

$Tmp' \leftarrow Tmp$

for each $(e, p) \in EP$ with $a(e, p) =$ process

$Tmp'(e) \leftarrow u(e, p)(Tmp)$

for each $(p, e) \in PE$

$Tmp'(e) \leftarrow u(p, e)(Tmp)$

$Tmp \leftarrow Tmp'$

3. **for each** continuous process $p \in S_c(t - \varepsilon_t)$

$Tmp' \leftarrow Tmp$

for each $(e, p) \in EP$ with $a(e, p) =$ process

$$Tmp'(e) \leftarrow Tmp'(e) - \int_{t-\varepsilon_t}^{t} u(e, p)(\tilde{M}(x))dx$$

for each $(p, e) \in PE$

$$Tmp'(e) \leftarrow Tmp'(e) + \int_{t-\varepsilon_t}^{t} u(p, e)(\tilde{M}(x))dx$$

$Tmp \leftarrow Tmp'$

4. **for each** discrete process $p \in S_d(t)$

$Tmp' \leftarrow Tmp$

for each $(e, p) \in EP$ with $a(e, p) =$ process

$Tmp'(e) \leftarrow Tmp'(e) - u(e, p)(\tilde{M}(q(p, t)))$

for each $(p, e) \in PE$

$Tmp'(e) \leftarrow Tmp'(e) + u(p, e)(\tilde{M}(q(p, t)))$

$Tmp \leftarrow Tmp'$

5. $M(t) \leftarrow Tmp$

Then $M_r(t)$ is defined as follows:

6. **for each** entity e with $h(e)$ = discrete or continuous

$$Tmp_r(e) \leftarrow Tmp_r(e) - \sum_{p \text{ with } p \in S_d(t) \text{ and } (e,p) \in EP} u(e,p)(\tilde{M}(q(p,t))) + \sum_{p \text{ with } p \in U_d(t) \text{ and } (e,p) \in EP} u(e,p)(\tilde{M}(t - \varepsilon_t)).$$

7. $M_r(t) \leftarrow Tmp_r$.

We call $M(t)$ $(t \geq 0)$ the *behavior* of H starting at the initial marking $M(0) = I$. □

In the next section, we will introduce the biological background and explain the modeling method of *C. elegans* vulval development mechanisms.

Biological background and modeling of C. elegans *vulval development*

Biological background of C. elegans *vulval development*

The *C. elegans* vulva is an egg-laying organ which is constituted by the descendants of three VPCs. The three VPCs are the members of six initially equivalent VPCs that are consecutively numbered P3.p – P8.p (termed Pn.p cells). In response to extracellular signaling pathways, each VPC has a potential to adopt one of three alternative cell fates (1°, 2°, 3°) (see Figure 3(a)). Six cell fates of Pn.p cells comprise a cell fate pattern in the form of [P3.p P4.p P5.p P6.p P7.p P8.p]. In wild-type worms, P3.p – P8.p always adopt a same pattern of fates (i.e., [332123]). The sublineage is then generated according to each specified VPC fate. The sublineage is a determined pattern of cell divisions which produces a characteristic set of progeny cell types. Figure 3(b) shows the sublineage of respective VPC fate according to the criteria defined by Sternberg and Horvitz [32].

In wild-type *C. elegans*, an inductive signal LIN-3, an EGF-like signal produced by gonad, activates the EGFR homolog LET-23 and a canonical Ras/MAPK cascade in P6.p adopting the 1° fate. In response to the inductive signal, P6.p produces LIN-12-mediated lateral signals (LS for short) that counteract the inductive signal from AC in two neighboring VPCs, which causes two VPCs to adopt the 2° fate. In other words, LS induces the expression of negative regulators (collectively termed *lateral signal target* (*lst*) genes [34]) against the EGF/Ras/MAPK pathway. LS is encoded by three functionally redundant members of the Delta/Serrate protein family (*dsl-1*, *apx-1*, and *lag-2*) and transduced by the LIN-12/Notch receptor. Each *lst* gene contains a cluster of binding site LBSs for LAG-1 that is a DNA binding protein forming a complex with the LIN-12 intracellular domain to activate the transcription of the target genes [34-36]. Furthermore, two functionally redundant synthetic Multivulva (*synMuv*) genes transcriptionally repress the target gene of *lin-3* in the hypodermis, i.e., *synMuv* genes prevent the surrounding hypodermal syncytium hyp7 from generating inductive LIN-3 signals [37]. Without the activation of either inductive or lateral signaling pathways, P3.p, P4.p, and P8.p adopt the 3° fate [35]. Thus, the fates of 1°, 2°, and 3° are the productions of the coordination regulated by three signaling pathways, i.e., AC induced signaling pathways, the lateral signaling pathways, and the signaling pathways induced from hyp7 [32,35,38-41]. Figure 4 illustrates the biological diagram for the multiple signaling events underlying the VPC fate specification.

Modeling VPC fate specification mechanisms with HFPNe based on literature

Figure 5 exhibits a whole HFPNe based VPC fate model that is constructed by compiling and interpreting the information appeared in the literature concerning the VPC fate specification mechanisms [9,32,34-38,42-46]. The whole VPC fate model totally includes 427 entities, 554 processes and 780 connectors. The elements of HFPNe are changed to biological icons in Cell Illustrator (on the right side of Figure 2). The icons have been defined with one of the biological ontology information, called Cell System Ontology [47]. Although these changes have no effect on mathematical meaning, it is helpful for biologists to understand the pathways.

Since six VPCs are initially equivalent, we hereafter explain modeling operations by using a single VPC fate model as shown in Figure 6 whose six copies are embedded in the whole VPC fate model in Figure 5. In Figure 6, 24 events directly attending VPC fate specification are assigned to the processes $p_i \in \{p1, \bullet, p_{24}\}$ (see Table 2). A set of sink processes $\{p_{d_1}, p_{d_2}, \cdots, p_{d_{28}}\}$ denotes the natural degradation of the substances, whereas a set of source processes $\{p_{s_1}, p_{s_2}, \cdots, p_{s_{10}}\}$ denotes the translation reactions of initial entities. The left 13 processes $\{p_{25}, p_{26}, \bullet,$

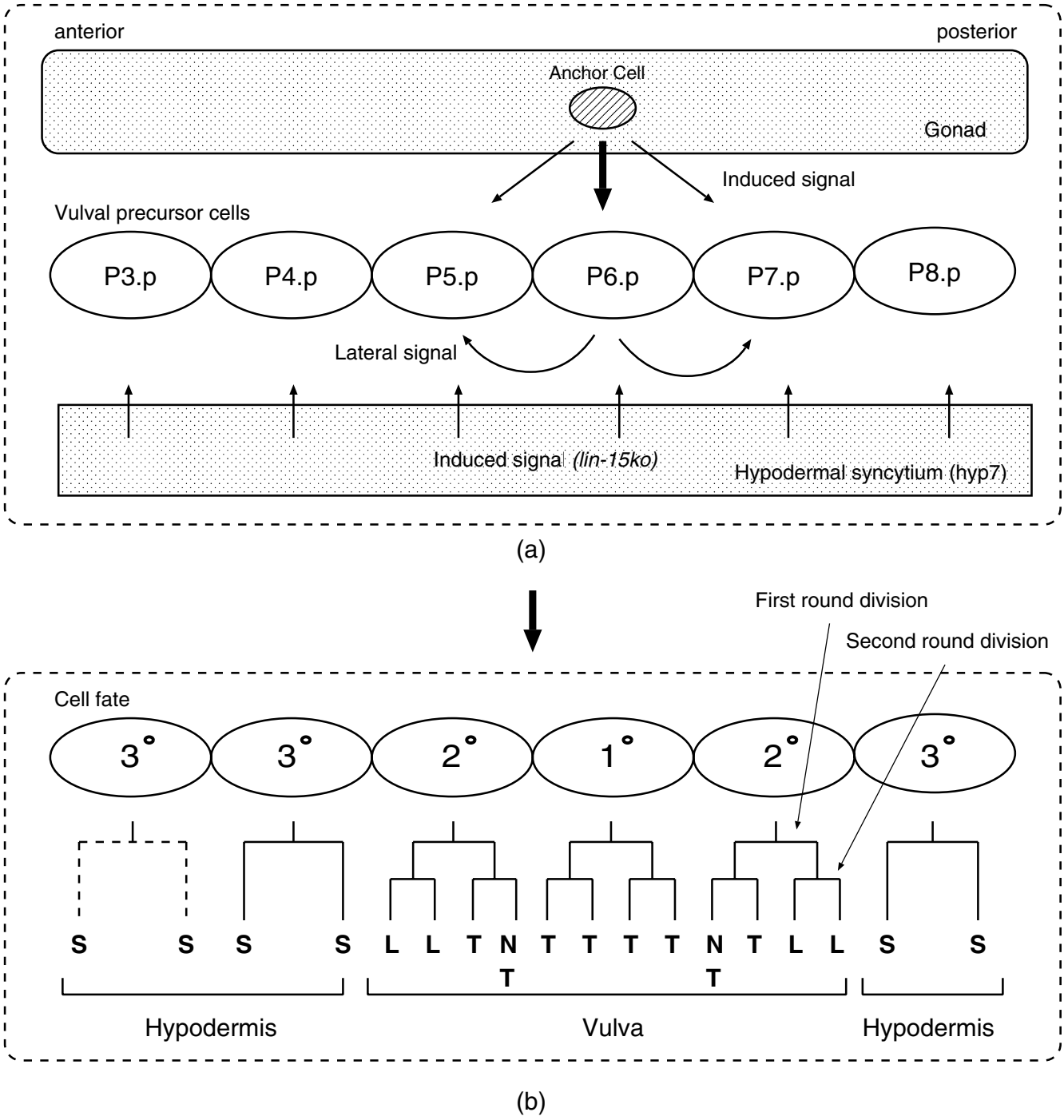


Figure 3

Schematic representations of VPC fate specification in wild-type hermaphrodites. (a) The anchor cell produces a graded inductive signal and causes six equivalent cells to adopt fates in a precise pattern. (b) The sublineage is generated according to each specified cell fate. The sublineage is a determined pattern of cell divisions which produces a characteristic set of progeny cell types. Sternberg and Horvitz have defined vulval cell types (after two rounds of VPC divisions) by two criteria as follows: the axis of the third round nuclear divisions (L, longitudinal axis; T, transverse axis; N, no division; S is to join the large hypodermal syncytium (hyp7)), and adherence to the ventral cuticle [32].

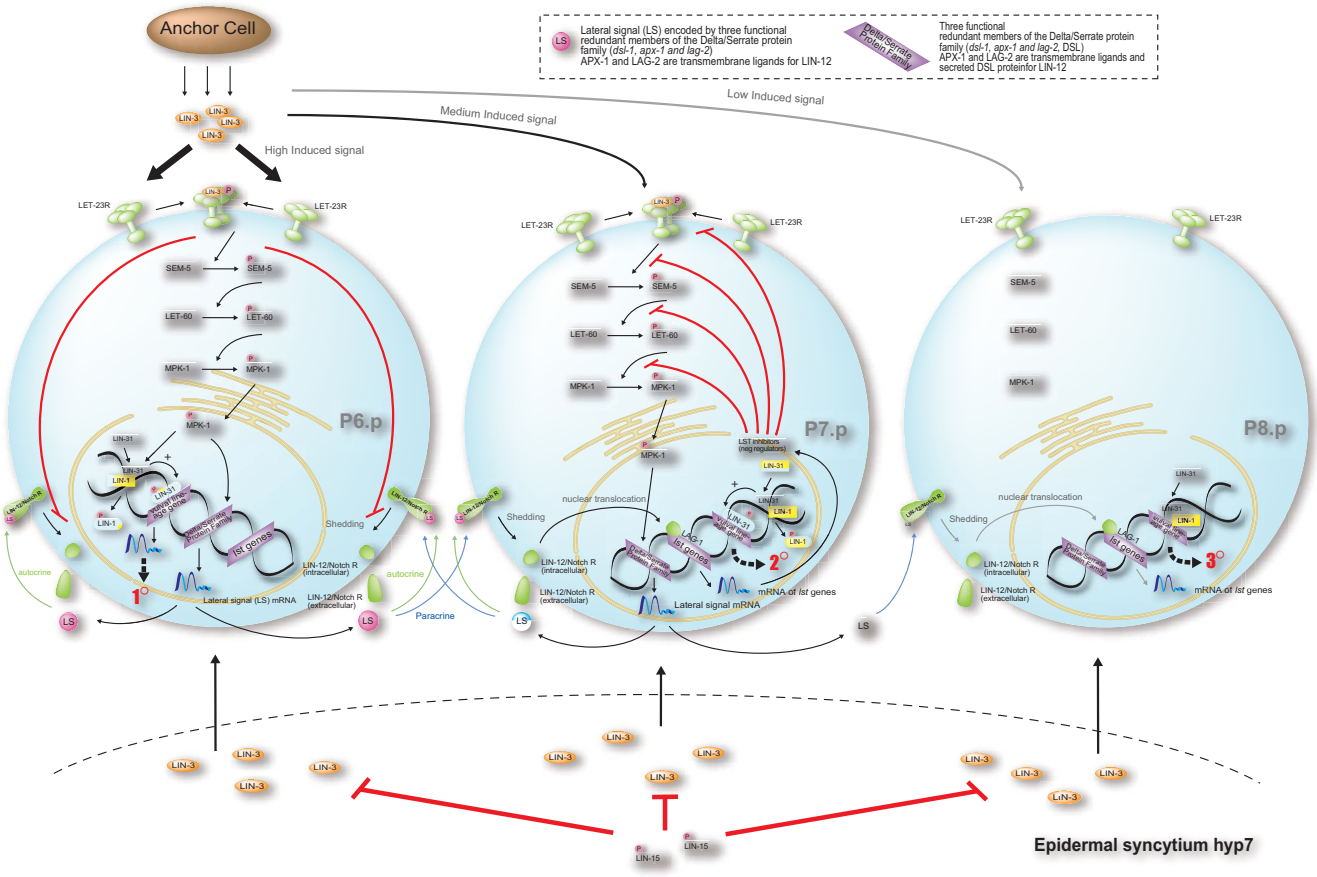


Figure 4

Biological diagram for the multiple signaling events underlying VPC specification. Biological diagram for the multiple regulatory signaling pathways underlying VPC fate specification. Three VPCs (they can be P6.p, P7.p and P8.p, or P6.p, P5.p and P4.p) according to the relative distance to the AC are selected to depict the signaling crosstalk. In the rightmost cell, the EGF/MAPK pathway cannot be activated because the induced signal (indicated by a grey line) received by LET-23, is lower than the threshold for induction, and the VPC hence adopts the 3° fate. The induction signal with high concentration (indicated by a heavy black line) activates the EGF/MAPK pathway and causes the 1° fate. It also has been known that two transcription factors, LIN-31 and LIN-1 are likely to be the downregulation targets of the MAPK pathway [42]. Both LIN-31 and LIN-1 can be phosphorylated by MAPK kinase MPK-1. LIN-31 and LIN-1 usually form a complex that is disrupted by the phosphorylation of LIN-31, and LIN-1 dissociates from LIN-31 to let LIN-31 play a role in the proper specification of VPCs. On the other hands, ligands for LIN-12 are members of the "DSL" family, an acronym derived from canonical ligands from *Drosophila* (*D*elta, *S*errate) and *C. elegans* (*L*AG-2). Binding of DSL ligands to LIN-12/Notch leads to the shedding of the LIN-12/Notch ectodomain (extracellular domain) via cleavage. The remaining transmembrane protein is cleaved constitutively, and the intracellular domain translocates to the nucleus, binding to the LAG-1 that usually exists as a transcription factor to repress *lst* genes. With the binding to LAG-1, *lst* genes expresses LST proteins to counteract the operations of EGF/MAPK pathways by inhibiting VPCs from becoming the 1° fate. For the details of LIN-12/Notch signaling in *C. elegans*, the readers are suggested to refer to [34].

$p_{37}\}$ are generic processes, which contribute to play the roles of extracellular stimuli ($\{p25\}$) and the cell fate specification ($\{p_{26}, p_{27}, \bullet, p_{37}\}$). Variable $m_x \in \{m_1, \bullet, m_{32}\}$ denotes the concentration of corresponding substance (see Table 3).

The modeling operation starts from the source process p_{s_1} denoting an activity that the substance takes part in the reaction. Here we only explain the case of Ras/MAPK inductive signaling pathway: LIN-3 ligands and corresponding receptors LET-23 are assigned to the entities (m1 and m2) connected from the source processes (p_{s_1} and p_{s_2}), respectively. Next, the ligand-receptor binding reaction where two entities (m1 and m2) merge into an entity m3 denoting ligand-receptor complex is represented via the process p3. Since two ligand-receptor com-

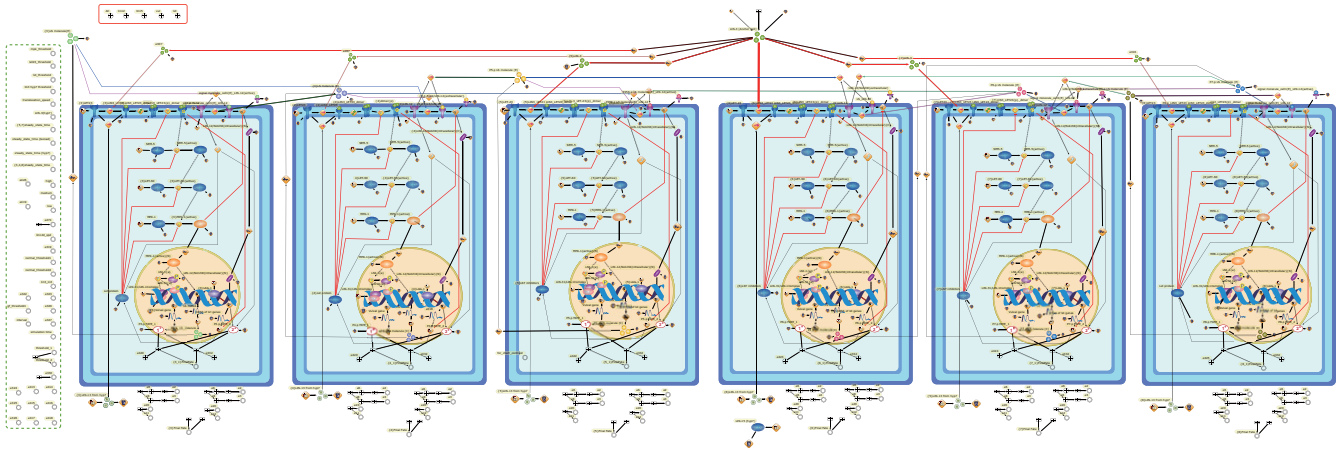


Figure 5
The whole HFPNe based VPC fate model underlying the fate specification mechanisms involving six equivalent VPCs.

plexes shape a dimer, the process p4 is used to represent the homodimerization connecting from the input entity m3 as well as connecting to the output entity m4 of LIN-3/LET-23 dimer. Note that the stoichiometry of the input connector of p4 is set to 2 due to the dimerization. The dimer is autophosphorylated subsequently which is modeled by using a phosphorylation process p5 connecting from and to the entities of m4 and m5, respectively. Succedent reactions for the products of the canonical cascades: active SEM-5 (m7), LET-60 (m9), and MPK-1 (m11) are modeled in the same way using the processes of activation (p6, p7 and p8). Activated MPK-1 then moves from the cytoplasm to the nucleus. This movement is modeled as a translocation reaction by p9.

Based on the understanding of the literature and the hypothesis reported in [9], we first model downstream regulation of Ras/MAPK cascades and succedent intracellular reactions induced by LIN-12/Notch signaling according to newly published literature. The details of the new biological facts are described in the caption of Figure 4. The new facts concerning the mechanisms of the downstream regulation of Ras/MAPK cascades are modeled as follows: The phosphorylation by m_{12} further disrupts the formation of LIN-31/LIN-1 complex (m_{13}) which applies the process of phosphorylation (p_{13}). The processes p_{14} and p_{15} are used to model the transcription of target genes and the translation of mRNA (m_{16}) regulated by LIN-31 (m_{31}) that acts as a transcriptional activator promoting the 1° fate as a fate candidate (m_{18}).

Two biological fate determination rules applied to VPC fate model

When determining cell fates from biological points of view, several biological fate determination rules can be taken into account. One major biological fate determination rule (**Rule II**) based on temporal order has been applied to a qualitative model by Fisher *et al*. [9]. The approach is based on a discrete model. However, the other rules such as temporal interval based one (**Rule I**) cannot be handled in the same discrete model. This is because the model cannot deal with the quantitative properties (e.g., continuous feature) that are also important to biological processes, such as the concentration of proteins and reaction rates. With inspiring by this limitation of qualitative models, we employ these two major biological fate determination rules – **Rule I** and **Rule II** – to VPC fate model. The model has both discrete and continuous features.

Two fate determination rules are as follows: (i) For **Rule I**, the fate will be determined if it satisfies the conditions that (1) the fate can sustain the behaviors at a certain over-threshold state within a given length of time, and (2) the time epoch of the certain over-threshold in (1) is earlier than the other fate candidate. Figure 7 shows a schematized diagram of a simulation result. In Figure 7, the cell fate is determined to the 2° fate by using **Rule I**. The time course expression of 1° is maintained at the over-threshold state during *interval*_1, and that of 2° is maintained at the over-threshold state during *interval*_2. Although the time course expression of 1° has two over-threshold states, only the second interval *interval*_1 is longer than the given *interval*. Thus, the time epoch labeled with "τ_i of Rule I" is regarded as the initial time epoch inducing the second over-threshold state. The VPC adopts the 2° fate due to $\tau_j <$ "τ_i of Rule I".

(ii) For **Rule II**, the cell fate will be priorly adopted according to the temporal sequence of the first time epoch inducing over-threshold state. In Figure 7, we can observe that the first time epoch (i.e., "τ_i of Rule II") inducing

Table 2: Biological interpretation based on literature and assignment of each process in Figure 6

Wet experiments results published in literature	#1	#2	Reaction type	Refs
Translocation of LIN-3 emanated from AC to P5.p and P7.p	p_1	*LSMass*(m_1*0.1**low*, 0.1)	Translocation	[9,35]
Translocation of LIN-3 emanated from AC to P3.p, P4.p and P8.p	p_2	*LSMass*(m_1*0.1**mid*, 0.1)	Translocation	[9,35]
ligands LIN-3 binding to LET-23 to form a ligand-receptor complex	p_3	*LSMass*(m_1*0.1**high*, 0.1)	Binding	[43,44]
Two identical LIN-3 LET-23 complex combining to form a dimer	p_4	*LSMass*(m_3*0.1, 0.5)	Dimerization	[43,44]
Autophosphorylation following the dimerization of ligand-receptor complex	p_5	*LSMass*(m_4*0.1, 0.5)	Autophosphorylation	[43,44]
SEM-5 is activated by LIN-3 LET-23 dimer	p_6	*LSMass*(m_5*m_6*0.1, 0.5)	Enzymic reaction	[9,35]
LET-60 is activated by upstream SEM-5	p_7	*LSMass*(m_7*m_8*0.1, 0.5)	Enzymic reaction	[9,35]
Inactive MPK-1 is activated by upstream LET-60	p_8	*LSMass*(m_9*m_{10}*0.1, 0.5)	Enzymic reaction	[9,35]
The movement of MPK-1 from cytoplasm to nucleus	p_9	*LSMass*(m_{11}*0.1, 0.5)	Translocation	-
Active MPK-1(N) downregulates the target genes of *lst* and transcribes the mRNA of lateral signal (LS)	p_{10}	*LSMass*(m_{12}*0.1, 0.5)	Transcription	[9]
LS mRNA is translated to LS molecules	p_{11}	*LSMass*(m_{17}*0.1, 0.5)	Translation	-
Translated LS molecules are released to combine with LIN-12 receptors	p_{12}	*LSMass*(m_{19}*0.1, 0.5)	Translocation	[34]
LIN-31/LIN-1 complex is dissociated to individual active LIN-31 and LIN-1 by the phosphorylation of active MPK-1	p_{13}	*LSMass*(m_{12}*m_{13}*0.1, 0.5)	Phosphorylation	[38,45]
Active a MPK-1(N) acts as transcription factor to transcribe vulval genes to mRNA	p_{14}	*LSMass*(m_{14}*0.1, 0.5)	Transcription	[42]
mRNA of vulval genes is translated and cause the 1 cell fate	p_{15}	*LSMass*(m_{16}*0.1, 0.5)	Translation	[42]
LIN-12 receptor received the LS molecules from the adjacent Pn.p and shape a ligand-receptor complex	p_{16}	*LSMass*(m_{20}*m_k*1.0, 0.1)	Binding	[32,34]
LIN-12 receptor received the LS molecules from its own Pn.p and shape a ligand-receptor complex	p_{17}	*LSMass*(m_{20}*m_k*1.0, 0.1)	Binding	[32,34]
LIN-12 receptor received the LS molecules from the adjacent Pn.p and shape a ligand-receptor complex	p_{18}	*LSMass*(m_{20}*m_k*0.1, 0.1)	Binding	[32,34]
Binding of LS ligands to LIN-12/Notch receptor leads to shedding of the LIN-12/Notch extracellular domain via cleavage	p_{19}	*LSMass*(m_{21}*0.1, 0.5)	Shedding/Cleavage	[34]
Cleaved intracellular domain of LIN-12/Notch receptor move from cytoplasm to nucleus	p_{20}	*LSMass*(m_{23}*0.1, 0.5)	Translocation	[34,46]
Cleaved LIN-12/Notch receptor promote the target *lst* genes transcribed into mRNA of *lst* genes	p_{21}	*LSMass*(m_{24}*0.1, 0.5)	Transcription	[34,36]
LST mRNA is translated to LST in cytoplasm	p_{22}	*LSMass*(m_{26}*0.1, 0.5)	Translation	-
LIN-12 immediately induces *lst* expression thus prevents cells from engaging the mechanisms reducing LIN-12 activity	p_{23}	*LSMass*(m_{20}**lin*12_*init*, 0.1)	Production	[9]
LIN-3 emanating from hyp7 binds to LET-23 to form a complex	p_{24}	*LSMass*(m_2*m_{28}*0.1, 0.5)	Expression	[9,37]

Table 2: The column of #1 represents corresponding processes in the HFPNe model. Twenty-four events are assigned to the processes $p_i \in \{p_1, \bullet, p_{24}\}$. Each reaction speed of the processes is assigned as shown in the column of #2, in which several reaction speeds have been tuned manually. Reaction types of the processes are described in the fourth column with the literature facts given in the fifth column. Variable $m_x \in \{m_1, \bullet, m_{32}\}$ denotes the concentration of corresponding substance (see Table 3). The variable *mk* is used to collectively denote the concentration of the LS molecules generated from the adjacent Pn.p. The values of *high, mid*, and *low* are assigned to 100, 1, and 0.01. For example, the process p_9 has a reaction speed with a noise denoted by *LSMass*(m_{11}*0.1, 0.5), i.e., the reaction speed depends on the concentration of MPK-1{active} (C) in the cytoplasm (m_{11}). *LSMass*() is a function of log-normal distribution (see text).

Table 3: Entities in the HFPNe model of Figure 6

Entity Name	Variable (m_x)
LIN-3(AC)	m_1
LET-23	m_2
LIN-3/LET-23 complex	m_3
LIN-3/LET-23 dimer	m_4
LIN-3/LET-23 dimer{p}	m_5
SEM-5	m_6
SEM-5{active}	m_7
LET-60	m_8
LET-60{active}	m_9
MPK-1	m_{10}
MPK-1{active}(C)	m_{11}
MPK-1{active}(N)	m_{12}
LIN-1/LIN-31 complex	m_{13}
LIN-31{active}	m_{14}
LIN-1{active}	m_{15}
Vulval gene mRNA	m_{16}
LS mRNA	m_{17}
1° cell fate	m_{18}
LS molecules(C)	m_{19}
LIN-12/Notch receptor	m_{20}
LIN-12R/LS complex	m_{21}
Cleaved fraction of LIN-12R/LS (extracellular domain)	m_{22}
Intracellular domain of LIN-12/LS complex (C)	m_{23}
Intracellular domain of LIN-12/LS complex (N)	m_{24}
LAG-1	m_{25}
LST mRNA	m_{26}
LST inhibitors	m_{27}
LIN-3 emanating from hyp7	m_{28}
LIN-15 in hyp7	m_{29}
2° cell fate	m_{30}
Final fate determined by Rule II	m_{31}
Final fate determined by Rule I	m_{32}

Table 3: Variable $m_x \in \{m_1, \bullet, m_{32}\}$ indicates the concentration of each substance that actually takes part in the biological reactions. Initial value represents the initial content of an entity.

over-threshold state of 1° is earlier than 2°, and therefore the 1° fate will be adopted by **Rule II** no matter how long the over-threshold state of 1° will continue. Preliminary notations and mathematical definitions of two rules are given in the Additional file 1.

In Figure 6, two dashed-line ellipses illustrate the topological connections representing **Rule I** (blue ellipse) and **Rule II** (red ellipse) which jointly use the values of m_{18} and m_{30} as the inputs for the fate specification. Variables m_{18} (denoting the concentration of 1°) and m_{30} (denoting the concentration of 2°) are regarded as two cell fate candidates for determining final VPC fate. Two rules are implemented with a script-based language Pnuts in Cell Illustrator [48]. The detailed properties and descriptions of the entities concerning the fate specification are summarized in Table 4. The activity functions of related processes and update functions of the connectors are summarized in Tables 5 and 6.

Further, several variables are designed to reserve alterable values according to different genotypes (see Table 7). Figure 8 shows a Genotype Configuration panel and a structure of auto-switching mechanism. The Genotype Configuration panel includes an AC and four mutant variables. AC, lin15, vul and lst can toggle between true and false. lin12 has three string-type values, i.e., "wt", "ko", "gf", indicating three genetic conditions of wild, knockout and overexpression of lin-12.

It is obvious that the combination number is 48 as listed in Table 8. Forty-eight genotypes have the features as follows:

(i) For 15 genotypes whose RowID are indicated by boldface and labeled with square brackets, the fate patterns of each genotype have the *in vivo* experimental data reported in [32] (refer to the column of "**Fate Patterns (ST)**" in Tables 8 and 9).

(ii) For eight genotypes whose RowID are indicated with the superscript (†), the fate patterns of each genotype have been confirmed by model checking approach in the discrete model in [9], which are also consistent with the biological facts [35,46,49-53].

(iii) For the left 25 genotypes, the patterns of each genotype have only been verified in [9], because these genotypes are usually experimentally intractable. The reason is that, the population of double, triple or quadruple mutants might be technically difficult to be generated [9].

(iv) For four genotypes (i.e., RowID 5, 21, 29, and 45) are called *unstable patterns*, because genetic condition will lead to an unstable fate pattern as shown in Table 9.

In the next subsection, we will execute simulation to investigate the characteristics of predicted fate patterns of 48 genotypes, in particular, four genotypes leading to the unstable fate pattern, and to evaluate two rules employed to the VPC fate model by comparing with three simulation targets (i.e., **JA**, **ST**, and **STA**) as shown in Figure 1(b).

Simulation-based model checking approach of determining VPC specification by using Cell Illustrator

We first demonstrate how to assign parameters to the VPC fate model. Then, we consider three simulation targets to investigate the properties of fate patterns, and to evaluate two proposed rules. Finally, a large number of simulations are performed with the use of "High-Speed Simula-

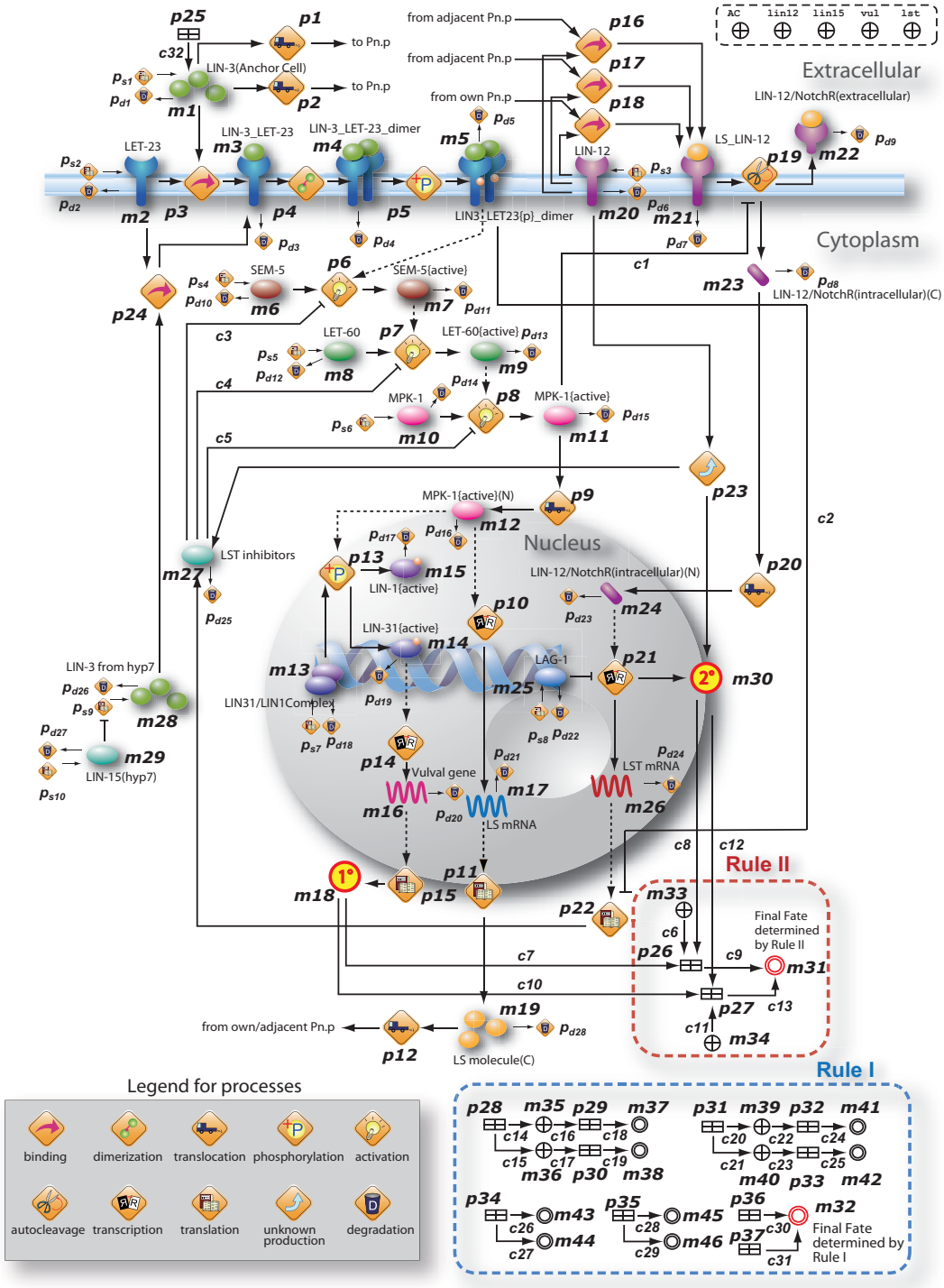


Figure 6

HFPNe model of a single VPC in Figure 5. The naming rules of the VPC fate model are defined as follows: (i) each "entity" is labeled with the name of a substance (e.g. LIN-3, LET-60, and *lst* gene), (ii) the name of a complex consisting of two or more protein components $A_1, A_2, \bullet, A_M$ ($M \in Z^+$) is represented as $A_1\_A_2\_\bullet\_A_M$, and (iii) an additional label (C) or (N) is attached at the end of a substance name, when it happens to distinguish the location of the substance in the cytoplasm or the nucleus. The label of {active} denotes the active state of the enzyme, and {p} denotes that the substance is phosphorylated.

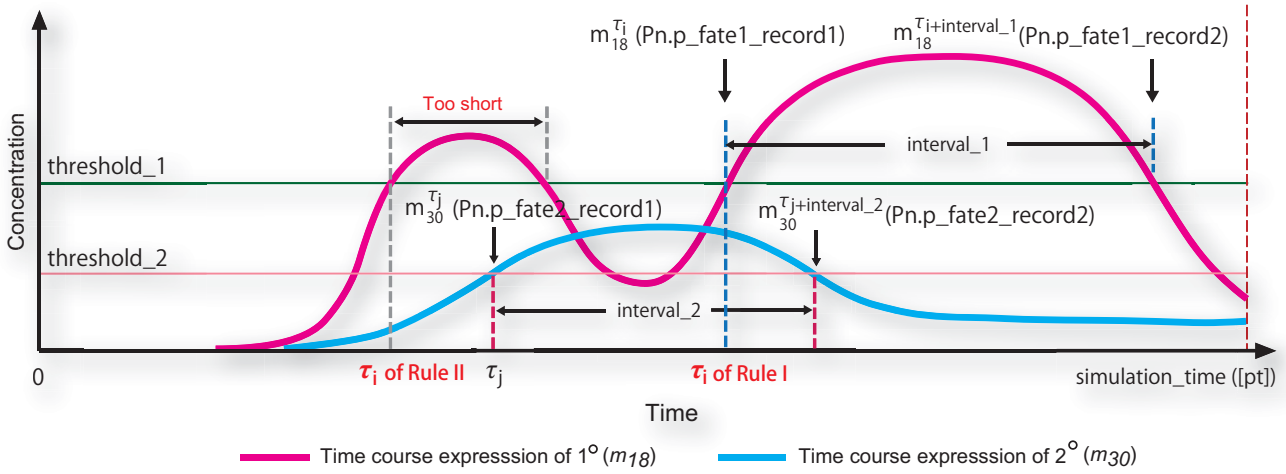


Figure 7
A schematized diagram depicting the fate specification methods by using Rule I and Rule II. "τ_i of Rule I" represents the first time epoch that the concentration of 1° (m_{18}) exceeds *threshold*_1, and the time length *interval*_starting from "τ_i of Rule I" is longer than or equal to the given *interval* when using **Rule I**. "τ_j of Rule II" represents the first time epoch that the concentration of 1° exceeds *threshold*_1, no matter how long this interval will continue when using **Rule II**. τ_i represents the first time epoch that the concentration of 2° (m_{30}) exceeds *threshold*_2. Variables m_{18} (denoting the concentration of 1°) and m_{30} (denoting the concentration of 1°) are regarded as two cell fate candidates for determining final VPC fate. *Pn.p_fate*1/2_*record*1/2 indicated in the brackets are the entity names used in Cell Illustrator.

tion Module" of Cell Illustrator. The simulation results are given in the form of *fitting score* and *variation frequency* of each pattern of 48 genotypes. The fitting score is a percentage of the total numbers of predicted patterns appeared in each genotype); the variation frequency is the number of each appearing fate pattern in the respective genotype during the simulation experiments.

Parameter assignment

The VPC fate model in Figure 5 reflects the interrelation of each substance. All parameters for the reaction rates of processes and the steady state of the substance are tuned manually with repeating simulation until concentration behaviors of proteins correspond to the biological facts. Note that it is hard to decide optimal values of these parameters, since data from biological experiments are very insufficient to determine them.

We thus simplify the kinetic parameters to the same speeds m_x*0.1 with noise for the process $p_i \in \{p_1, p_2, \bullet, p_{24}\}$ (see Table 2); the speed of the source process $p_{s_i} \in \{p_{s_1}, p_{s_2}, \cdots, p_{s_{10}}\}$ is assigned to 1.0 denoting the production rate of substance. When constructing the quantitative VPC fate model, threshold values (partly shown in Figure 8 and Table 10) of each reaction are assigned as real number, and the parameters for the steady states of the substances induced by the stimulations from anchor cell and hyp7 are carefully tuned by hand. Lots of trial and error operations have been performed repeatedly until appropriate parameters for simulation are determined. The VPC fate model in Figure 5 is available from [54], which can be executed on Cell Illustrator [24] or Java web start software Cell Illustrator Online 4.0 (CIO) [25]. All the parameters have been saved in the csv format, which are also available online at the same website [54].

Three simulation targets for validation

[Obtaining fate patterns (JA) by using model checker: MOCHA]

Model checking is powerful technique for automatically verifying the system requirements. The essential idea of model checking is that, with an exhaustive exploring of all reachable states and transitions of a modeled system, system properties (expressed as a formal specification) are examined whether the properties are satisfied or not. A model checker (e.g., MOCHA[55]) accepts the modeled system and the specification as its inputs. The checker then outputs yes if the model satisfies the given specification and generates a *counterexample* otherwise. The counterexample indicates why the model does not satisfy the given specification. By repeating operations of revising the errors examined by the counterexamples, the modeled system can be refined to satisfy enough system specifications [1,2].

Fisher *et al*. have firstly applied the model checking approach for validating biological systems of *C. elegans*

Table 4: Properties of entities for the HFPNe model of Figure 6

Variable	Entity name	Entity type	Value type	Initial value	Variable description
AC	AC	Generic	Boolean	true/false	Entity can be switched according to the genotype
lin12	lin-12	Generic	String	"wt"/"ko"/"gf"	Entity can be switched according to the genotype
lin15	lin-15	Generic	Boolean	true/false	Entity can be switched according to the genotype
vul	vul	Generic	Boolean	true/false	Entity can be switched according to the genotype
lst	lst	Generic	Boolean	true/false	Entity can be switched according to the genotype
m_{31}	Pn.p_FinalFateII	Continuous	Double	0	Final Fate determined by Rule II
m_{32}	Pn.p_FinalFateI	Continuous	Double	0	Final Fate determined by Rule I
m_{33}	m_{33}	Generic	Boolean	true	An entity designed to judge if m_{18}/m_{30} exceeds respective threshold when lin-12 is "wt"/"ko"
m_{34}	m_{34}	Generic	Boolean	true	An entity for judging if m_{18}/m_{30} exceeds its threshold when lin-12 is "gf"
m_{35}	m_{35}	Generic	Boolean	false	An entity used as a flag to judge if m_{37} can get the time epoch when m_{18} is over threshold_1
m_{36}	m_{36}	Generic	Boolean	false	An entity used as a flag to judge if m_{38} can get the time epoch when m_{18} is below threshold_1
m_{37}	Pn.p_fate1_record1	Continuous	Double	0	An entity designed to reserve a time epoch that m_{18} exceeds threshold_1
M_{38}	Pn.p_fate1_record2	Continuous	Double	0	An entity designed to reserve a time epoch that m_{18} decreases below threshold_1
m_{39}	m_{39}	Generic	Boolean	false	An entity used as a flag to judge if m_{41} can get the time epoch when m_{30} is over threshold_2
m_{40}	m_{40}	Generic	Boolean	false	An entity used as a flag to judge if m_{42} can get the time epoch when m_{30} is below threshold_2
m_{41}	Pn.p_fate2_record1	Continuous	Double	0	An entity designed to reserve a time epoch that m_{30} exceeds threshold_2
m_{42}	Pn.p fate2_record2	Continuous	Double	0	An entity designed to reserve a time epoch that m_{30} decreases below threshold_2
m_{43}	Pn.p_m1_interval	Continuous	Double	0	Time difference between m_{38} and m_{37} that is the longest time interval at the present time epoch
m_{44}	Pn.p_fate1_init	Continuous	Double	0	Time epoch that m_{18} exceeds threshold_1 to the initial one in the time span of m_{43} corresponding
m_{45}	Pn.p_m2_interval	Continuous	Double	0	Time difference between m_{42} and m_{41} that is the longest time interval at the present time epoch
m_{46}	Pn.p_fate2_init	Continuous	Double	0	Time epoch that m_{30} exceeds threshold_2 corresponding to the initial one in the time span of m_{45}
-	*lin3_init*	Continuous	Double	100	Amount of LIN-3 impulse emanating from AC
-	*Steady_state_time(Gonad)*	Continuous	Double	400	Time epoch of substance that achieve a steady state level
simultime	*Simulation_time*	Continuous	Double	2000-getSamplingInterval(simulator)	The length of simulation time

Table 5: Properties of processes for the HFPNe model of Figure 6

Process	Process type	Process activity
p_{25}	Generic	if (IfTime(simulator, steady_state_time(Gonad)) && AC == true) {return true;} else {return false;}
p_{26}	Generic	if ("gf".equals(lin12)) {return false;} else {return true;}
p_{27}	Generic	if ("gf".equals(lin12) && (getElapsedTime(simulator) > simultime+getSamplingInterval(simulator))) {return true;} else {return false;}
p_{28}	Generic	if ((m35 == true && m36 == true) && m18 > = threshold_1) {return true;} else {return false;}
p_{29}	Generic	if (m18 > = threshold_1 && m35 == false) {return true;} else {return false;}
p_{30}	Generic	if ((m35 == true && m36 == false && m18 < threshold_1) || (m35 == true && m36 == false && m18 > = threshold_1 && IfTime(simulator, simultime))) {return true;} else {return false;}
p_{31}	Generic	if (((m39 == true) && (m40 == true)) && (m30 > = threshold_2)) {return true;} else {return false;}
p_{32}	Generic	if ((m30 > = threshold_2) && (m39 == false)) {return true;} else {return false;}
p_{33}	Generic	if ((m39 == true && m40 == false && m30 < threshold_2) || (m39 == true && m40 == false && m30 > = threshold_2 && IfTime(simulator, simultime))) {return true;} else {return false;}
p_{34}	Generic	if ((m35 == true && m36 == true) || (m35 == true && getElapsedTime(simulator)>simultime)) {return true;} else {return false;}
p_{35}	Generic	if ((m39 == true && m40 == true) || (m39 == true && getElapsedTime(simulator)>simultime)) {return true;} else {return false;}
p_{36}	Generic	if (getElapsedTime(simulator) > simultime+getSamplingInterval(simulator) && ("gf".equals(lin12) == false)) {return true;} else {return false;}
p_{37}	Generic	if ((getElapsedTime(simulator) > simultime + getSamplingInterval(simulator)) && "gf".equals(lin12)) {return true;} else {return false;}

Table 5: External Java functions getElapsedTime(simulator), getSamplingInterval(simulator), and IfTime(simulator, time) return the value of elapsed time during simulation, the sampling interval of simulator, and true/false by judging if elapsed time equals to the value of time, respectively. All the functions are written in Pnuts language [48].

with biological experiments [9]. They have constructed a discrete and state-based mechanistic model underlying the inductive and lateral signaling crosstalk by using the language of *reactive modules* [13] and MOCHA. Further, they have confirmed that the model can reproduce reported biological behavior observed in 22 genotypes with MOCHA. However, their method has the following weaknesses that are desired to be solved in this paper: (1) The details of predicted fate patterns of four unstable patterns (RowID 5, 21, 29, and 45) are not explicitly given. That is, only summarized patterns are given. However, it is important to reveal if all the expansions of predicted (summarized) patterns will appear even though the predicted patterns satisfy the given specification. For example, in the case of *ac-; lin-12gf; lin-15ko* double mutants (RowID 45 in Table 8), the pre-

Table 6: Update functions of connectors for HFPNe model of Figure 6

Connector Name	Connector type	Update function
Rule I: if ("gf" .equals(lin12)) == false holds, the connectors are updated as follows:		
$c_{14}(p_{28}, m_{35})$	process	return false;
$c_{15}(p_{28}, m_{36})$	process	return false;
$c_{16}(m_{35}, p_{29})$	process	return true;
$c_{17}(m_{36}, p_{30})$	process	return true;
$c_{18}(p_{29}, m_{37})$	process	return getElapsedTime(simulator);
$c_{19}(p_{30}, m_{38})$	process	return getElapsedTime(simulator);
$c_{20}(p_{31}, m_{39})$	process	return false;
$c_{21}(p_{31}, m_{40})$	process	return false;
$c_{22}(m_{39}, p_{32})$	process	return true;
$c_{23}(m_{40}, p_{33})$	process	return true;
$c_{24}(p_{32}, m_{41})$	process	return getElapsedTime(simulator);
$c_{25}(p_{33}, m_{42})$	process	return getElapsedTime(simulator);
$c_{26}(p_{34}, m_{43})$	process	if (m43! = 0 && m43 < m38-m37) {return m38-m37;} else if (m43! = 0 && P3p_m1_interval > = m38-m37) {return m43;} else {return m38-m37;}
$c_{27}(p_{34}, m_{44})$	process	if (m44! = 0 && m43 < m38-m37) {return m37;} else if (m44! = 0 && m43 > = m38-m37) {return m44;} else {return m37;}
$c_{28}(p_{35}, m_{45})$	process	if (m45! = 0 && m45 < m42-m41) {return m42-m41;} else if (m45! = 0 && m45 > = m42-m41) {return m45;} else {return m42-m41;}
$c_{29}(p_{35}, m_{46})$	process	if (m46! = 0 && m45 < m42-m41) {return m41;} else if (m46! = 0 && m45 > = m42-m41) {return m46;} else {return m41;}
$c_{30}(p_{36}, m_{32})$	process	if (m43 > = interval && m45 > = interval) {if (m44 < = m46) {return 1;} else {return 2;}} else if (m43 > = interval && m45 < interval) {return 1;} else if (m43 < interval && m45 > = interval) {return 2;} else{return 3;}
$c_{31}(p_{37}, m_{32})$	process	if (m43 > = interval && m45 > = interval) {return 1;} else if (m43 > = interval && m45 < interval) {return 1;} else if (m43 < interval && m45 > = interval) {return 2;} else {return 3;}
$c_{32}(p_{25}, m1)$	process	lin3_init
Rule II: if ("gf" .equals(lin12)) == false holds, the connectors are updated as follows:		

Table 6: Update functions of connectors for HFPNe model of Figure 6 *(Continued)*

$c6(m_{33}, p_{26})$	process	if (m18 > = threshold_1 || m30 > = threshold_2) {return false;} else {return true;}
$c7(m_{18}, p_{26})$	process	return m18;
$c8(m_{30}, p_{26})$	process	return m30;
$c9(p_{26}, m_{31})$	process	if(m33 == true) {if(m18 > = threshold_1) {return 1;} else if (m30 > = threshold_2) {return 2;} else {return 3;}} else {return m31;}
if ("gf" .equals(lin12)) == true holds, the connectors are updated as follows:		
$c_{10}(m_{18}, p_{27})$	process	return m18;
$c_{11}(m_{34}, p_{27})$	process	if (m18 > = threshold_1 && m30 > = threshold_2) {return false;} else {return true;}
$c_{12}(m_{30}, p_{27})$	process	return m30;
$c_{13}(p_{27}, m_{31})$	process	if(m18 < threshold_1 && m30 < threshold_2) {return 3;} else if(m285 > = threshold_1) {return 1;} else {return 2;}

dicted pattern is [1/2, 1/2, 1/2, 1/2, 1/2, 1/2] in [9]. It is obvious that the pattern includes totally $2^6 = 64$ possible fate patterns, but it is still ambiguous if all these 64 patterns will be adopted in *in vivo* experiments.

(2) The distribution and variation of individual behavior of predicted fate patterns are not sufficient to understand the features of the fate patterns. In other words, it is necessary to clarify the occurrence probability (called *variation frequency*) of each predicted pattern, which is considered to facilitate obtaining an overall distribution of the predicted fate patterns, especially, the four unstable patterns.

To improve the first weakness, we introduce a new procedure GENERATENECESSARYFATEPATTERNS by repeating the use of "counterexample" to obtain necessary fate patterns on MOCHA.

«PROCEDURE GENERATENECESSARYFATEPATTERNS»

[******]: A state pattern of six VPC fates; * is one of three alternative fates (1°, 2°, and 3°); i.e., [111111] denotes a

Table 7: Detailed parameters and functions of Figure 8.

Variable name	Update function
$c_i(P_1, threshold\_1)$	if ("gf".equals(lin12)) {return lin12gf_threshold_1;} else {return normal_threshold_1;}
$c_j(P_1, threshold\_2)$	if ("gf".equals(lin12)) {return lin12gf_threshold_2;} else {return normal_threshold_2;}

$c_k(P_2, interval)$	if ("gf".equals(lin12)) {return lin12gf_interval;}
	else
	{return normal_interval;}

$c_p(P_3, lin12\_init)$	if ("gf".equals(lin12)) {return LSMass(2.0,1.0);} else if ("ko".equals(lin12)) {return 0;} else {return LSMass(1.0,1.0);}

Table 7: The auto-switching mechanism of determining the value of *threshold*_1, *threshold*_2, *interval* and expression speed of lin12 under different mutant combinations as shown in the block of "Genotype Configuration" of Figure 8. If lin12 equals to "wt", a generic process P_3 will fire to deposit LSMass(1.0,1.0) as an expression speed of *lin-12*; if lin12 equals to "ko", P_3 will deposit 0; and if lin12 equals to "gf", P_3 will fire to deposit LSMass(2.0,1.0) as the speed denoting an overexpression of *lin-12*. The variables of *threshold*_1, *threshold*_2 and *interval* are designed likewise. All update functions are written in Pnuts language [48]

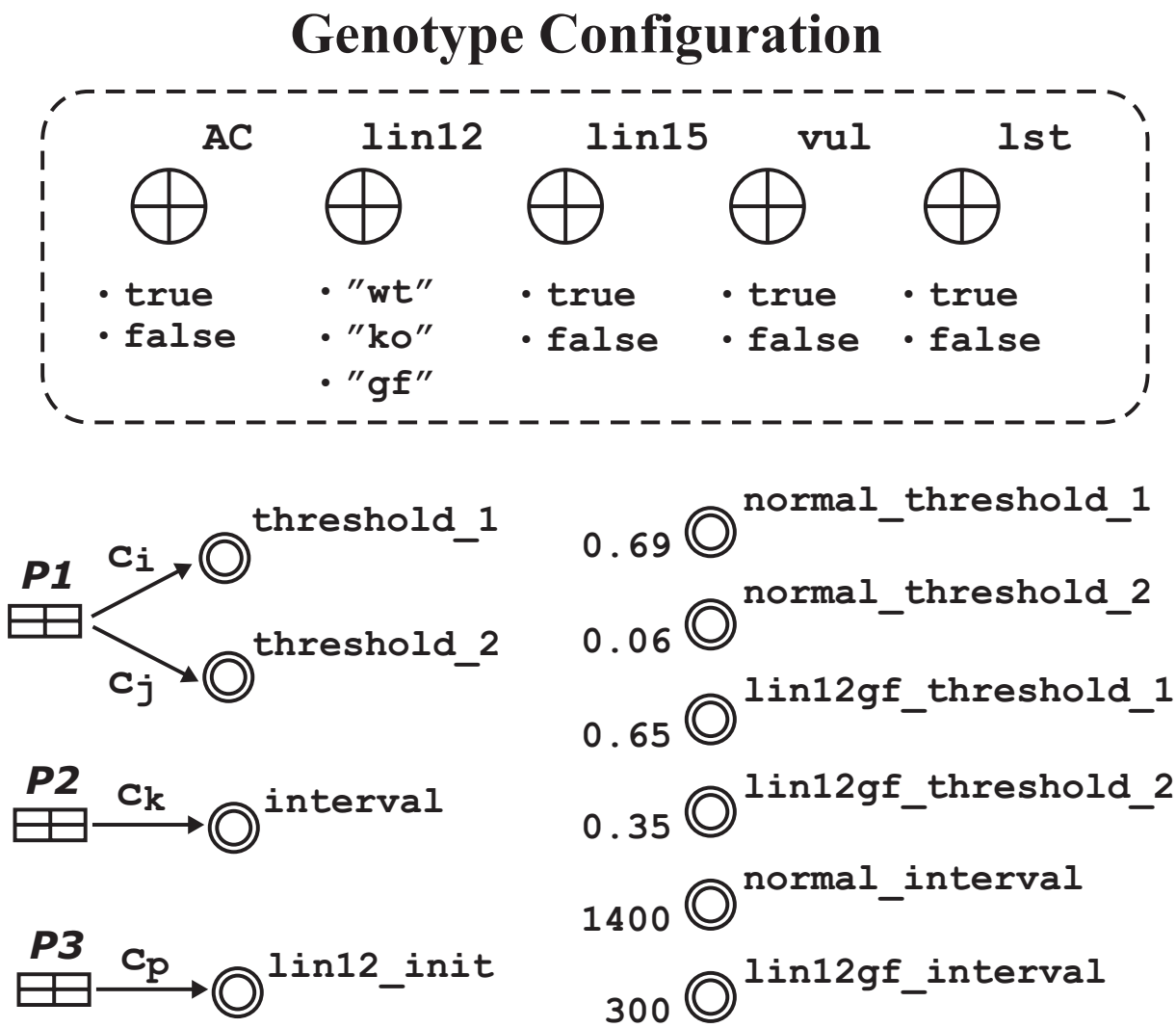


Figure 8
The panel of Genotype Configuration and the automatic-switching mechanisms to determine the values of *threshold_1*/*threshold_2*, *interval*, and *lin12_init* according to the different genetic conditions.

fate pattern with 1° for all VPCs. In total, there are 729(= 3^6) fate patterns.

RS: A set to store intermediate fate patterns required for a genotype.

Verify: A MOCHA procedure to verify if the predication holds for the model. The input is the *RS*. If there is no counterexample for the *RS*, φ will be returned. If the counterexample is generated for the *RS*, one of the counterexamples will be returned at random.

The new procedure to derive whole fate patterns for a genotype is as follows: 1. $RS \leftarrow \{[111111]\}$

2. **do** $c \leftarrow Verify(RS)$

3. $RS \leftarrow RS \cup \{c\}$

4. **while** $c \neq \varphi$

5. $c \leftarrow Verify(RS \backslash \{[111111]\})$

6. **if** $c = \varphi$

7. return $RS \backslash \{[111111]\}$

8. **else**

Table 8: Summary of the VPC fate patterns of 48 combinations concerning the AC and four mutants.

RowID	AC	Genotype	Fate Patterns (JA)	Fate Patterns (ST)	Fate Patterns (STA)	Ref.
			lin-12 lin-15 vul lst P3.pP4.pP5.pP6.pP7.pP8.p	P3.pP4.pP5.pP6.pP7.pP8.p	P3.pP4.pP5.pP6.pP7.pP8.p	
1†	+	wt wt wt wt	[332123]	-	-	[49]
2†	+	wt wt wt ko	[331113]	-	-	[35,50]
[3]	+	wt wt ko wt	[333333]	[333333]	[331113] [331123] [331133] [332313] [332323] [332333] [333323] [333333]	[32]
4†	+	wt wt ko ko	[333333]	-	-	u.d.
[5]	+	wt ko wt wt		Unstable pattern (refer to Table 9)		[32]
6†	+	wt ko wt ko	[1111111]	-	-	[46]
[7]	+	wt ko ko wt	[333333]	[333333]	[333333]	[32,37,51]
8	+	wt ko ko ko	[333333]	-	-	n.d.
[9]	+	ko wt wt wt	[331113]	[331113]	[311111] [311112] [311113] [321111] [321112] [321113] [331111] [331112] [331113] -	[32]
10†	+	ko wt wt ko	[331113]	-	-	u.d.
[11]	+	ko wt ko wt	[333333]	[333333]	[311113] [311123] [311133] [331133] [331213] [331223] [332133] [332213] [332223] [333133] [333213] [333223] -	[32]
12	+	ko wt ko ko	[333333]	-	-	n.d.
[13]	+	ko ko wt wt	[1111111]	[1111111]	[1111111]	[32]
14	+	ko ko wt ko	[1111111]	-	-	n.d.
15	+	ko ko ko wt	[333333]	-	-	n.d.
16	+	ko ko ko ko	[333333]	-	-	n.d.
[17]	+	gf wt wt wt	[222122]	[222122]	[122122] [222122] [322122]	[32]
18	+	gf wt wt ko	[221112]	-	-	n.d.
[19]	+	gf wt ko wt	[222222]	[222222]	[122222] [222222] [322222]	[32]
20	+	gf wt ko ko	[222222]	-	-	n.d.
[21]	+	gf ko wt wt		Unstable pattern (refer to Table 9)		[32]

Table 8: Summary of the VPC fate patterns of 48 combinations concerning the AC and four mutants. *(Continued)*

22	+	gf ko wt ko	[1 1 1 1 1 1]	-	-	n.d.
23	+	gf ko ko wt	[222222]	-	-	n.d.
24	+	gf ko ko ko	[222222]	-	-	n.d.
25†	-	wt wt wt wt	[333333]	-	-	[52]
26†	-	wt wt wt ko	[333333]	-	-	[50]
27	-	wt wt ko wt	[333333]	-	-	n.d.
28	-	wt wt ko ko	[333333]	-	-	n.d.
[29]	-	wt ko wt wt		Unstable pattern (refer to Table 9)		[32]
30	-	wt ko wt ko	[1 1 1 1 1 1]	-	-	n.d.
31	-	wt ko ko wt	[333333]	-	-	n.d.
32	-	wt ko ko ko	[333333]	-	-	n.d.
[33]	-	ko wt wt wt	[333333]	[333333]	[333333]	[32]
34	-	ko wt wt ko	[333333]	-	-	n.d.
35	-	ko wt ko wt	[333333]	-	-	n.d.
36	-	ko wt ko ko	[333333]	-	-	n.d.
[37]	-	ko ko wt wt	[1 1 1 1 1 1]	[1 1 1 1 1 1]	[1 1 1 1 1 1] [1 1 1 1 2 1] [1 1 1 1 3 1]	[32]
38	-	ko ko wt ko	[1 1 1 1 1 1]	-	-	n.d.
39	-	ko ko ko wt	[333333]	-	-	n.d.
40	-	ko ko ko ko	[333333]	-	-	n.d.
[41]	-	gf wt wt wt	[222222]	[222222]	[122222] [222222] [322222]	[32]
42†	-	gf wt wt ko	[222222]	-	-	[50]
[43]	-	gf wt ko wt	[222222]	[222222]	[222222]	[32,53]
44	-	gf wt ko ko	[222222]	-	-	n.d.
[45]	-	gf ko wt wt		Unstable pattern (refer to Table 9)		[32]
46	-	gf ko wt ko	[1 1 1 1 1 1]	-	-	n.d.
47	-	gf ko ko wt	[222222]	-	-	n.d.
48	-	gf ko ko ko	[222222]	-	-	n.d.

Table 8: n.d.: not determined, i.e., the fate pattern is only predicted by the *in silico* model without *in vivo* evidences to support. In *lstko* mutant, all *lst* genes are null: *ark-1*, *lip-1*, *dpy-23*, *lst-1*, *lst-2*, *lst-3*, and *lst-4*. In *vulko* mutant, the genes of *let-23*, *sem-5*, *let-60*, *mpk-1* are null. u.d.: unpublished data; AC(+/-): anchor cell formed/ablated; 1: 1°; 2: 2°; 3: 3°.

Table 9: Expanded results of four unstable VPC patterns with respect to the results of [9] and [32]

RowID [5]			**RowID [21]**		
	Predicted patterns			**Predicted patterns**	
Fate Patterns (JA)	**Fate Patterns (ST)**	**Fate Patterns (STA)**	**Fate Patterns (JA)**	**Fate Patterns (ST)**	**Fate Patterns (STA)**
[1/2,1/2,2,1,2,1/2] (Total fate number: 8)	[1/2,1/2,2,1,2,1/2]	[1/2,1/2,2,1,2,1/2] +*a*	[1/2,1/2,2,1,2,1/2] (Total fate number: 8)	[2,1/2,2,1,2,1/2]	[2,1/2,2,1,2,1/2] +*a*
[112121] [122121] [212121]	[112121] [112122] [122121] [122122] [212121] [212122] [222121] [222122]	[112121] [112122] [112123] [122121] [122122] [212121] [212122] [212123] [222121] [222122] [312121] [312122] [312123]	[112121] [122121] [212121]	[212121] [212122] [222121] [222122]	[212121] [212122] [222121] [222122]
Total fate number: 3			**Total fate number: 3**		
RowID [29]			**RowID [45]**		
	Predicted patterns			**Predicted patterns**	
Fate Patterns (JA)	**Fate Patterns (ST)**	**Fate Patterns (STA)**	**Fate Patterns (JA)**	**Fate Patterns (ST)**	**Fate Patterns (STA)**
[1/2,1/2,1/2,1/2,1/2,1/2,1/2] (Total fate number: 64)	[1/2,1/2,1/2,1/2,1/2,1/2,1/2]	[1/2,1/2,1/2,1/2,1/2,1/2,1/2] +*a*	[1/2,1/2,1/2,1/2,1/2,1/2,1/2] (Total fate number: 64)	[1/2,1/2,1/2,1/2,1/2,1/2,1/2]	[1/2,1/2,1/2,1/2,1/2,1/2,1/2] +*a*
[111111] [111112] [111121] [111211] [111212] [111221] [112111] [112112] [112121] [112211] [112212] [121111] [121112] [121121] [121211] [121212] [121221] [122111] [122112] [122121] [122211] [212111] [212112] [212121] [212211] [212212]	[111111] [111112] [111121] [111122] [111211] [111212] [111221] [111222] [112111] [112112] [112121] [112122] [112211] [112212] [112221] [112222] [121111] [121112] [121121] [121122] [121211] [121212] [121221] [121222] [122111] [122112] [122121] [122122] [122211] [122212] [122221] [122222] [211111] [211112] [211121] [211122] [211211] [211212] [211221] [211222] [212111] [212112] [212121] [212122] [212211] [212212] [212221] [212222] [221111] [221112] [221121] [221122] [221211] [221212]	[111111] [111112] [111121] [111122] [111211] [111212] [111221] [111222] [112111] [112112] [112121] [112122] [112211] [112212] [112221] [112222] [121111] [121112] [121121] [121122] [121131] [121211] [121212] [121221] [121222] [122111] [122112] [122121] [122122] [122211] [122212] [122221] [122222] [211111] [211112] [211121] [211122] [211211] [211212] [211221] [211222] [212111] [212112] [212121] [212122] [212211] [212212] [212221] [212222] [221111] [221112] [221121] [221122] [221211] [221212]	[111111] [111112] [111121] [111122] [111211] [111212] [111221] [111222] [112111] [112112] [112121] [112122] [112211] [112212] [112221] [112222] [121111] [121112] [121121] [121122] [121211] [121212] [121221] [121222] [122111] [122112] [122121] [122122] [122211] [122212] [122221] [122222] [211111] [211112] [211121] [211122] [211211] [211212] [211221] [211222] [212111] [212112] [212121] [212122] [212211] [212212] [212221] [212222] [221111] [221112] [221121] [221122] [221211] [221212]	[111111] [111112] [111121] [111122] [111211] [111212] [111221] [111222] [112111] [112112] [112121] [112122] [112211] [112212] [112221] [112222] [121111] [121112] [121121] [121122] [121211] [121212] [121221] [121222] [122111] [122112] [122121] [122122] [122211] [122212] [122221] [122222] [211111] [211112] [211121] [211122] [211211] [211212] [211221] [211222] [212111] [212112] [212121] [212122] [212211] [212212] [212221] [212222] [221111] [221112] [221121] [221122] [221123] [221211]

Table 9: Expanded results of four unstable VPC patterns with respect to the results of [9] and [32] *(Continued)*

[221221] [221222] [222111] [222112] [222121] [222122] [222211] [222212] [222221] [222222]	[221212] [221221] [221222] [222111] [222112] [222121] [222122] [222211] [222212] [222221] [222222]	[221221] [221222] [222111] [222112] [222121] [222122] [222211] [222212] [222221] [222222]	[221212] [221221] [221222] [221223] [221321] [221322] [221323] [222111] [222112] [222121] [222122] [222211] [222212] [222221] [222222] [222312]
Total fate number: 31		**Total fate number: 31**	

9. return *RS*

In the above procedure, step 1 is an initialization of *RS*. Steps 2 – 4 are the main part to obtain the fate patterns by repeating the use of "counterexample". Steps 5 – 9 are designed to check if the initial fate pattern is necessary or not. By executing the procedure, the results of required fate patterns of 48 genotypes are summarized in the fourth column of Table 8, and the first and fourth columns of Table 9. The predicted fate patterns of each genotype derived by using our procedure are collectively called **Fate Patterns (JA)** (**JA** for short). It is clear that the number of predicted fate patterns investigated by our method is far smaller than the one summarized by [9] (see Table 9). The source code used to obtain **JA** with MOCHA can be found at the CSML website [54].

[Fate patterns (ST) obtained from in vivo *data]*

Sternberg and Horvitz have summarized the VPC fate patterns of 15 genotypes by the observation of anatomy and the cell lineages in living *C. elegans* using Nomarski differential interference contrast optics (see Table 4 in the original paper [32]). We list the fate patterns of these 15 genotypes (refer to the fifth column in Table 8, and the second and fifth columns in Table 9).

[Fate patterns (STA) obtained from in vivo *data including hybrid lineage observations]*

In [32], we have noticed that some biological observations of cell lineages have not received enough attentions. These observations are likely difficult to be determined accurately according to the two criteria defined by Sternberg and Horvitz [32,39]. The reason is that some lineages were not interpretable as 1° or 2° by two criteria, i.e., observed lineages may result in possible inaccuracy in the observations which includes mis-scoring adherence to ventral cuticle owing to deformations in cuticle as animals bend, as well as variations in division axes [32]. In some cases, lineages are hybrid, e.g., [S TT] in which [S] is the production of the characteristics of the 3° fate, and [TT] can be the lineage of either the 1° or 2° fate. Sternberg and Horvitz have suggested that hybrid sublineage is a result of imprecise decision, and the specification is allowed to have a number of outcomes. When a system comes close to a certain threshold, it can be considered that partial functions are involved to express the results in the formation of hybrid lineage [39].

Inspired by the observation of hybrid lineage, it seems reasonable to put those uninterpretable experimental data together with the fate patterns of [32] (i.e., **ST**) as the simulation targets for validation. By investigating the experimental results of the vulval cell lineage given in [32], we explain the cell fate of such an uninterpretable lineage as three plausible fate candidates: 1°, 2°, and 3° fates. Thus, the VPC fate pattern including the uninterpretable lineage is extended to three predicted fate patterns (see **STA** block in Figure 1(b)). For example, in the case of *vulko* (RowID 3 in Table 8), we observed such a cell lineage: [[S S] [S S] [S S] [S TT] [S S] [S S]], in which [S S] is a cell lineage of 3° and [S TT] is a hybrid lineage and is interpretable by neither two criteria defined in [32] nor the method in [9]. This cell lineage is determined by the fate pattern [333?33]. We thus suppose to interpret this VPC fate pattern to three fate pattern extensions of [333133], [333233], and [333333], i.e., extend "?" (uncertain fate) to the 1°, 2°, and 3° fate. All the possible cell fate patterns including extended fate patterns are shown in the column

Table 10: Thresholds used in the HFPNe model of Figure 6.

Connector Name	Connector description	Firing threshold
$c_1(m_{11}, p_{19})$	Active MAPK-1 represses the shedding process of ligand-receptor complex	4.3+*rand*()/10
$C_2(m_5, p_{22})$	Active LIN-3/LET-23 dimer represses the expression of *lst*	0.5+*rand*()/10
$c_3(m_{27}, p_6)$	Expressed LST prevents SEM-5 from becoming an active form	0.09+*rand*()/10
$c_4(m_{27}, p_7)$	Expressed LST prevents LET-60 from becoming an active form	0.09+*rand*()/10
$c_5(m_{27}, p_8)$	Expressed LST prevents MPK-1 from becoming an active form	0.09+*rand*()/10

Fate Patterns (STA) of Tables 8 and 9, i.e., **STA** is the combination of **ST** and the pattern extensions.

Simulation procedures

Here, we discuss several considerations of simulation environment adjustment in order to validate the VPC fate model and evaluate two rules.

When emulating cell stimulations in *in silico* experiments, it is important to adjust the cell to a constant condition before the stimulation. This condition is usually called a *steady state*. For example, simulation time periods of 400 [pt] and 200 [pt] are reserved representing LIN-3 stimulations from AC and hyp7 respectively in the VPC fate model ([pt] is the virtual time unit of the HFPNe model). This is because the concentration of the receptor LET-23 will be kept at a steady state after 200 [pt]. The time interval of the stimulations between AC and hyp7 is designed to investigate respective variation of fate patterns.

Meanwhile, the following is taken into account to emulate real environment as well as assessing the robustness of the model:

(i) The notion of log-normal distribution is introduced. For each process, log-normal distribution is applied as a system noise. Each standard deviation is given in the third column in Table 2, where *LSMass*(*arg*1, *arg*2) denotes the function of log-normal distribution, *arg*1 denotes normal reaction speed without noise, and *arg*2 is the standard deviation denoting the strength of the noise.

(ii) In the case of *lin-15ko* mutant, a little temporal difference between the stimulations of LIN-3 emanating from hyp7 to each Pn.p is designed to investigate the influence of ectopic stimuli.

(iii) The key thresholds involved in Ras/MAPK and LIN-12/Notch signaling pathways are assigned to the variables, which depend on the function of *rand*() that fluctuates randomly between 0.0 and 1.0 (refer to Table 10). Here, uniform random number generated by Mersenne Twister random number generator is used.

We have tried 100, 1,000 and 10,000-run simulation experiments. For all cases, the results clearly show that **Rule I** can generate high fitting score than **Rule II**. Among them, since 10,000-run simulation can present the most precise behaviors of each fate pattern, in this paper we have concentrated on the 10,000-run simulation results to investigate the fitting score and variation frequency of 48 genotypes on the VPC fate model. The simulation experiments are carried out on the workstation of Intel Xeon X5450 (3.0 GHz) processors with 16 Gbytes of memory. The machine has two CPUs which has four cores per CPU (number of total processor cores is eight). The theoretical computational performance is 96 GFLOPS. All simulations were run on Java 1.6 environment. The VPC fate model will take nearly 100 [sec] for one simulation on Cell Illustrator with a script-based simulation engine. It provides detailed visualization of system behaviors. On the other hand, Cell Illustrator is equipped with a "High-Speed Simulation Module" specialized in producing simulation results only. It increases the simulation speed from 10 to 100 times than the script-based one. Hence, the 10,000 simulations can be conducted on a day on average, and the 48 sets can be executed on 48 days on a single core of a processor. We carried out all simulations within 6 days on our computing environment. We use the same model, the same procedures and also the same parameters to evaluate the fate patterns of all 48 genotypes with the three simulation targets (refer to Tables 8 and 9).

Results and Discussion

We now exhibit and discuss our simulation results on the fitting score of predicted fate patterns and the variation frequency of each appearing fate pattern, especially focusing on the four unstable patterns.

Using the method addressed in the previous sections, we performed the simulation of the VPC fate model. Tables 10, 11, 12, 13, 14 and Additional file 2 exhibit the simulation results of 48 genotypes. Figures 9, 10, 11 and 12 show the graphs of four unstable patterns (RowID 5, 21, 29, and 45), which depict the variation frequency of each predicted pattern and the variation distribution based on simulation results. For example, in Figure 9, the fate pattern [122121] predominantly appeared in the matched results of JA by using **Rule I** and **Rule II**, while the pattern is regarded as a false pattern because it is not observed in ST and STA.

From the variation distribution as shown in Figure 9, [122121] is the fate pattern that can be most easily observed in in vivo experiments because of the high appearance. In contrast, [112121] is likely difficult to be monitored due to the relative low variation frequency. Each graph comprises three parts of comparison results obtained from JA, ST and STA (see (a), (b), and (c) in Figure 9) by using both Rule I and Rule II. In each part, the variation frequency results of matched patterns are given on the left side while that of unmatched patterns are exhibited on the right side. Figures 10, 11 and 12 are built up in the same way.

From the simulation results, we draw the following conclusions:

Table 11: Details of the simulation results for RowID 5 in Table 9

RuleI_JA				RuleI_ST				RuleI_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[112121]^X	0			[112121]^X	0	[212121]^X	0	[112121]^X	0	[212123]^X	0
[122121]^X	9260			[112122]^X	0	[212122]^X	0	[112122]^X	0	[222121]^X	0
[212121]^X	0			[122121]^X	9260	[222121]^X	0	[112123]^X	0	[222122]^X	0
				[122122]^X	0	[222122]^X	0	[122121]^X	9260	[312121]^X	0
								[122122]^X	0	[312122]^X	0
								[212121]^X	0	[312123]^X	0
								[212122]^X	0		
Total: 9260 (92.6%)				**Total: 9260 (92.6%)**				**Total: 9260 (92.6%)**			
[121121]	19			[121121]	19			[121121]	19		
[121321]	3			[121321]	3			[121321]	3		
[121323]	1			[121323]	1			[121323]	1		
[123121]	11			[123121]	11			[123121]	11		
[123321]	651			[123321]	651			[123321]	651		
[123323]	55			[123323]	55			[123323]	55		
Total: 740 (7.4%)				**Total: 740 (7.4%)**				**Total: 740 (7.4%)**			
RuleII_JA				RuleII_ST				RuleII _STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[112121]^X	11			[112121]^X	11	[212121]^X	1140	[112121]^X	11	[212123]^X	0
[122121]^X	1289			[112122]^X	1	[212122]^X	410	[112122]^X	1	[222121]^X	253
[212121]^X	1140			[122121]^X	1289	[222121]^X	253	[112123]^X	0	[222122]^X	96
				[122122]^X	496	[222122]^X	96	[122121]^X	1289	[312121]^X	0
								[122122]^X	496	[312122]^X	0
								[212121]^X	1140	[312123]^X	0
								[212122]^X	410		
Total: 2440 (24.4%)				**Total: 3696 (36.96%)**				**Total: 3696 (36.96%)**			
[111122]	1	[211212]	25	[111122]	1	[211222]	2	[111122]	1	[211222]	2
[111221]	1	[211221]	45	[111221]	1	[212111]	77	[111221]	1	[212111]	77
[112111]	1	[211222]	2	[112111]	1	[212112]	318	[112111]	1	[212112]	318
[112112]	2	[212111]	77	[112112]	2	[212211]	95	[112112]	2	[212211]	95
[112122]	1	[212112]	318	[112212]	3	[212212]	400	[112212]	3	[212212]	400
[112212]	3	[212122]	410	[112221]	3	[212221]	449	[112221]	3	[212221]	449
[112221]	3	[212211]	95	[121111]	142	[212222]	64	[121111]	142	[212222]	64
[121111]	142	[212212]	400	[121112]	754	[221111]	54	[121112]	754	[221111]	54
[121112]	754	[212221]	449	[121121]	318	[221112]	265	[121121]	318	[221112]	265
[121121]	318	[212222]	64	[121122]	232	[221121]	110	[121122]	232	[221121]	110
[121122]	232	[221111]	54	[121211]	61	[221122]	70	[121211]	61	[221122]	70
[121211]	61	[221112]	265	[121212]	116	[221211]	18	[121212]	116	[221211]	18
[121212]	116	[221121]	110	[121221]	1084	[221212]	32	[121221]	1084	[221212]	32
[121221]	1084	[221122]	70	[121222]	73	[221221]	426	[121222]	73	[221221]	426
[121222]	73	[221211]	18	[122111]	57	[221222]	45	[122111]	57	[221222]	45
[122111]	57	[221212]	32	[122112]	226	[222111]	11	[122112]	226	[222111]	11
[122112]	226	[221221]	426	[122211]	81	[222112]	33	[122211]	81	[222112]	33
[122122]	496	[221222]	45	[122212]	237	[222211]	8	[122212]	237	[222211]	8
[122211]	81	[222111]	11	[122221]	132	[222212]	23	[122221]	132	[222212]	23
[122212]	237	[222112]	33	[122222]	33	[222221]	28	[122222]	33	[222221]	28
[122221]	132	[222121]	253	[211111]	29	[222222]	11	[211111]	29	[222222]	11
[122222]	33	[222122]	96	[211112]	77			[211112]	77		
[211111]	29	[222211]	8	[211121]	17			[211121]	17		

Table 11: Details of the simulation results for RowID 5 in Table 9 *(Continued)*

[211112]	77	[222212]	23	[211122]	5	[211122]	5
[211121]	17	[222221]	28	[211211]	10	[211211]	10
[211122]	5	[222222]	11	[211212]	25	[211212]	25
[211211]	10			[211221]	45	[211221]	45
Total: 7560 (75.6%)				**Total: 6304 (63.4%)**		**Total: 6304 (63.4%)**	

Table 11: X denotes predicted fate pattern, which is also used in Table 12–14.

(1) **Rule I** gives more genotypes (RowID) with high fitting scores nearly 100% than **Rule II** to cover predicted patterns of three simulation targets. In more details,

(i) For 44 genotypes (i.e., the genotypes without unstable fate patterns), each genotype can generate the fitting score nearly 95%, in which the fitting scores of 41 genotypes are all 100% (refer to Additional File 1) by using both **Rule I** and **Rule II**. In other words, our model shows good robustness, and can produce stable cell fate patterns even we intentionally add the noise to capture the behaviors of the model fluctuations.

(ii) For the left three genotypes of four unstable fate patterns, *lin-15ko* mutant (RowID 5) produces all 92.6% for JA, ST and STA by using **Rule I**, as opposed to 24.4% (for JA) and 36.96% (for ST and STA) by using **Rule II**; *ac-; lin-15ko* mutant (RowID 29) produces high coverage of 94.11% (for JA) and 99.86% (for both ST and STA); and *ac-; lin-12gf; lin-15ko* double mutants (RowID 45) generates all 100% with both rules for three simulation targets.

(iii) For *lin-15; lin-12ko* double mutants (RowID 21), the fitting scores in the results of ST and STA drop to a quite low value below 15% by using either **Rule I** or **Rule II**. This is because P3.p is allowed to adopt the 1° fate in JA (*in silico* model), but is not allowed in ST and STA (*in vivo* model). This fact has also been confirmed in the preceding study [9] that *in silico* model can faithfully produce the 1° fate for P3.p. From these observations, it can

Table 12: Details of the simulation results for RowID 21 in Table 9

RuleI_JA				RuleI_ST				RuleI_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[112121]^X	145	[212121]^X	1440	[212121]^X	1440	[222121]^X	0	[212121]^X	1440	[222121]^X	0
[122121]^X	6544			[212122]^X	0	[222122]^X	0	[212122]^X	0	[222122]^X	0
Total: 8129 (81.29%)				**Total: 1440 (14.4%)**				**Total: 1440 (14.4%)**			
[112111]	5	[122111]	8	[112111]	5	[121212]	1339	[112111]	5	[121212]	1339
[121111]	38			[112121]	145	[122111]	8	[112121]	145	[122111]	8
[121121]	481			[121111]	38	[122121]	6544	[121111]	38	[122121]	6544
[121212]	1339			[121121]	481			[121121]	481		
Total: 1871 (18.71%)				**Total: 8560 (85.6%)**				**Total: 8560 (85.6%)**			
RuleII_JA				RuleII_ST				RuleII_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[112121]^X	0	[212121]^X	1440	[212121]^X	1440	[222121]^X	0	[212121]^X	1440	[222121]^X	0
[122121]^X	7221			[212122]^X	0	[222122]^X	0	[212122]^X	0	[222122]^X	0
Total: 8661 (86.61%)				**Total: 1440 (14.4%)**				**Total: 1440 (14.4%)**			
[121212]	1339			[121212]	1339	[122121]	7221	[121212]	1339	[122121]	7221
Total: 1339 (13.39%)				**Total: 8560 (85.6%)**				**Total: 8560 (85.6%)**			

Table 13: Details of the simulation results for RowID 29 in Table 9

RuleI_JA				RuleI_ST				RuleI_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[111111]^X	0			[111111]^X	0	[211112]^X	0	[111111]^X	0	[211111]^X	0
[111112]^X	0			[111112]^X	0	[211121]^X	0	[111112]^X	0	[211112]^X	0
[111121]^X	0			[111121]^X	0	[211122]^X	0	[111121]^X	0	[211121]^X	0
[111211]^X	0			[111122]^X	0	[212111]^X	0	[111122]^X	0	[211122]^X	0
[111212]^X	0			[111211]^X	0	[212112]^X	0	[111211]^X	0	[212111]^X	0
[111221]^X	0			[111212]^X	0	[212121]^X	0	[111212]^X	0	[212112]^X	0
[112111]^X	0			[111221]^X	0	[212122]^X	0	[111221]^X	0	[212121]^X	0
[112112]^X	0			[111222]^X	0	[212211]^X	0	[111222]^X	0	[212122]^X	0
[112121]^X	0			[112111]^X	0	[212112]^X	0	[112111]^X	0	[212111]^X	0
[112211]^X	0			[112112]^X	0	[212121]^X	0	[112112]^X	0	[212112]^X	0
[112212]^X	0			[112121]^X	0	[212122]^X	0	[112121]^X	0	[212121]^X	0
[121111]^X	0			[112122]^X	0	[212211]^X	0	[112122]^X	0	[212122]^X	0
[121112]^X	0			[112211]^X	0	[212212]^X	0	[112211]^X	0	[212211]^X	0
[121121]^X	120			[112212]^X	0	[212221]^X	0	[112212]^X	0	[212212]^X	0
[121211]^X	0			[112221]^X	0	[212222]^X	0	[112221]^X	0	[212221]^X	0
[121212]^X	0			[112222]^X	0	[221111]^X	0	[112222]^X	0	[212222]^X	0
[121221]^X	4393			[121111]^X	0	[221112]^X	0	[121111]^X	0	[221111]^X	0
[122111]^X	0			[121112]^X	0	[221121]^X	0	[121112]^X	0	[221112]^X	0
[122112]^X	0			[121121]^X	120	[221122]^X	0	[121121]^X	120	[221121]^X	0
[122121]^X	4898			[121122]^X	0	[221211]^X	0	[121122]^X	0	[221122]^X	0
[211111]^X	0			[121211]^X	0	[221212]^X	0	[121131]^X	0	[221211]^X	0
[211112]^X	0			[121212]^X	0	[221221]^X	0	[121211]^X	0	[221212]^X	0
[211121]^X	0			[121221]^X	4393	[221222]^X	0	[121212]^X	0	[221221]^X	0
[211211]^X	0			[121222]^X	0	[222111]^X	0	[121221]^X	4393	[221222]^X	0
[211212]^X	0			[122111]^X	0	[222112]^X	0	[121222]^X	0	[222111]^X	0
[211221]^X	0			[122112]^X	0	[222121]^X	0	[122111]^X	0	[222112]^X	0
[212111]^X	0			[122121]^X	4898	[222122]^X	0	[122112]^X	0	[222121]^X	0
[212112]^X	0			[122122]^X	0	[222211]^X	0	[122121]^X	4898	[222122]^X	0
[212121]^X	0			[122211]^X	0	[222212]^X	0	[122122]^X	0	[222211]^X	0
[212211]^X	0			[122212]^X	0	[222221]^X	0	[122211]^X	0	[222212]^X	0
[212212]^X	0			[122221]^X	575	[222222]^X	0	[122212]^X	0	[222221]^X	0
				[122222]^X	0			[122221]^X	575	[222222]^X	0
				[211111]^X	0			[122222]^X	0		
Total: 9411 (94.11%)				**Total: 9986 (99.86%)**				**Total: 9986 (99.86%)**			
[121223]	14	[122221]	575	[121223]	14			[121223]	14		
Total: 589 (5.89%)				**Total: 14 (0.14%)**				**Total: 14 (0.14%)**			
RuleII_JA				RuleII_ST				RuleII_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[111111]^X	0			[111111]^X	0	[211112]^X	7	[111111]^X	0	[211111]^X	5
[111112]^X	0			[111112]^X	0	[211121]^X	12	[111112]^X	0	[211112]^X	7
[111121]^X	1			[111121]^X	1	[211122]^X	5	[111121]^X	1	[211121]^X	12
[111211]^X	0			[111122]^X	0	[212111]^X	38	[111122]^X	0	[211122]^X	5
[111212]^X	0			[111211]^X	0	[212112]^X	110	[111211]^X	0	[212111]^X	38
[111221]^X	3			[111212]^X	0	[212121]^X	62	[111212]^X	0	[212112]^X	110
[112111]^X	0			[111221]^X	3	[212122]^X	15	[111221]^X	3	[212121]^X	62
[112112]^X	1			[111222]^X	0	[212211]^X	47	[111222]^X	0	[212122]^X	15
[112121]^X	14			[112111]^X	0	[212112]^X	124	[112111]^X	0	[212111]^X	47
[112211]^X	0			[112112]^X	1	[212121]^X	1214	[112112]^X	1	[212112]^X	124
[112212]^X	4			[112121]^X	14	[212122]^X	405	[112121]^X	14	[212121]^X	1214
[121111]^X	5			[112122]^X	3	[212211]^X	125	[112122]^X	3	[212122]^X	405

Table 13: Details of the simulation results for RowID 29 in Table 9 *(Continued)*

[121112]^X	12		[112211]^X	0	[212212]^X	533	[112211]^X	0	[212211]^X	125
[121121]^X	98		[112212]^X	4	[212221]^X	448	[112212]^X	4	[212212]^X	533
[121211]^X	238		[112221]^X	2	[212222]^X	82	[112221]^X	2	[212221]^X	448
[121212]^X	850		[112222]^X	1	[221111]^X	3	[112222]^X	1	[212222]^X	82
[121221]^X	1271		[121111]^X	5	[221112]^X	9	[121111]^X	5	[221111]^X	3
[122111]^X	12		[121112]^X	12	[221121]^X	46	[121112]^X	12	[221112]^X	9
[122112]^X	70		[121121]^X	98	[221122]^X	36	[121121]^X	98	[221121]^X	46
[122121]^X	1263		[121122]^X	54	[221211]^X	79	[121122]^X	54	[221122]^X	36
[211111]^X	5		[121211]^X	238	[221212]^X	296	[121131]^X	0	[221211]^X	79
[211112]^X	7		[121212]^X	850	[221221]^X	441	[121211]^X	238	[221212]^X	296
[211121]^X	12		[121221]^X	1271	[221222]^X	88	[121212]^X	850	[221221]^X	441
[211211]^X	38		[121222]^X	246	[222111]^X	3	[121221]^X	1271	[221222]^X	88
[211212]^X	110		[122111]^X	12	[222112]^X	16	[121222]^X	246	[222111]^X	3
[211221]^X	62		[122112]^X	70	[222121]^X	227	[122111]^X	12	[222112]^X	16
[212111]^X	47		[122121]^X	1263	[222122]^X	103	[122112]^X	70	[222121]^X	227
[212112]^X	124		[122122]^X	511	[222211]^X	6	[122121]^X	1263	[222122]^X	103
[212121]^X	1214		[122211]^X	100	[222212]^X	45	[122122]^X	511	[222211]^X	6
[212211]^X	125		[122212]^X	403	[222221]^X	28	[122211]^X	100	[222212]^X	45
[212212]^X	533		[122221]^X	138	[222222]^X	10	[122212]^X	403	[222221]^X	28
			[122222]^X	32			[122221]^X	138	[222222]^X	10
			[211111]^X	5			[122222]^X	32		
Total: 6119 (61.19%)			**Total: 10000 (100%)**				**Total: 10000 (100%)**			
[112122]	3	[122211]	100							
[112221]	2	[122212]	403	[212122]	405	[221121]	46	[221222]	88	[222211]	6
[112222]	1	[122221]	138	[212221]	448	[221122]	36	[222111]	3	[222212]	45
[121122]	54	[122222]	32	[212222]	82	[221211]	79	[222112]	16	[222221]	28
[121222]	246	[211122]	5	[221111]	3	[221212]	296	[222121]	227	[222222]	10
[122122]	511	[211222]	15	[221112]	9	[221221]	441	[222122]	103		
									Total: 3881 (38.81%)		

Table 14: Details of the simulation results for RowID 45 in Table 9

RuleI_JA				RuleI_ST				RuleI_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[111111]^X	0			[1111111]^X	0	[211122]^X	0	[1111111]^X	0	[211122]^X	0
[111112]^X	0			[111112]^X	0	[211211]^X	0	[111112]^X	0	[211211]^X	0
[111121]^X	0			[111121]^X	0	[211212]^X	0	[111121]^X	0	[211212]^X	0
[111211]^X	2			[111122]^X	0	[211221]^X	0	[111122]^X	0	[211221]^X	0
[111212]^X	32			[111211]^X	2	[211222]^X	0	[111211]^X	2	[211222]^X	0
[111221]^X	0			[111212]^X	32	[212111]^X	108	[111212]^X	32	[212111]^X	108
[112111]^X	6			[111221]^X	0	[212112]^X	0	[111221]^X	0	[212112]^X	0
[112112]^X	0			[111222]^X	0	[212121]^X	1336	[111222]^X	0	[212121]^X	1336
[112121]^X	115			[112111]^X	6	[212122]^X	0	[112111]^X	6	[212122]^X	0
[112211]^X	0			[112112]^X	0	[212211]^X	0	[112112]^X	0	[212211]^X	0
[112212]^X	0			[112121]^X	115	[212212]^X	0	[112121]^X	115	[212212]^X	0
[121111]^X	0			[112122]^X	0	[212221]^X	0	[112122]^X	0	[212221]^X	0
[121112]^X	0			[112211]^X	0	[212222]^X	0	[112211]^X	0	[212222]^X	0
[121121]^X	94			[112212]^X	0	[221111]^X	0	[112212]^X	0	[221111]^X	0
[121211]^X	62			[112221]^X	0	[221112]^X	0	[112221]^X	0	[221112]^X	0
[121212]^X	1430			[112222]^X	0	[221121]^X	0	[112222]^X	0	[221121]^X	0
[121221]^X	2795			[121111]^X	0	[221122]^X	0	[121111]^X	0	[221122]^X	0
[122111]^X	1			[121112]^X	0	[221211]^X	0	[121112]^X	0	[221123]^X	0
[122112]^X	0			[121121]^X	94	[221212]^X	0	[121121]^X	94	[221211]^X	0
[122121]^X	4019			[121122]^X	0	[221221]^X	0	[121122]^X	0	[221212]^X	0
[211111]^X	0			[121211]^X	62	[221222]^X	0	[121211]^X	62	[221221]^X	0

Table 14: Details of the simulation results for RowID 45 in Table 9 *(Continued)*

[211112]^X	0			[121212]^X	1430	[222111]^X	0	[121212]^X	1430	[221222]^X	0
[211121]^X	0			[121221]^X	2795	[222112]^X	0	[121221]^X	2795	[221223]^X	0
[211211]^X	0			[121222]^X	0	[222121]^X	0	[121222]^X	0	[221321]^X	0
[211212]^X	0			[122111]^X	1	[222122]^X	0	[122111]^X	1	[221322]^X	0
[211221]^X	0			[122112]^X	0	[222211]^X	0	[122112]^X	0	[221323]^X	0
[212111]^X	108			[122121]^X	4019	[222212]^X	0	[122121]^X	4019	[222111]^X	0
[212112]^X	0			[122122]^X	0	[222221]^X	0	[122122]^X	0	[222112]^X	0
[212121]^X	1336			[122211]^X	0	[222222]^X	0	[122211]^X	0	[222121]^X	0
[212211]^X	0			[122212]^X	0			[122212]^X	0	[222122]^X	0
[212212]^X	0			[122221]^X	0			[122221]^X	0	[222211]^X	0
				[122222]^X	0			[122222]^X	0	[222212]^X	0
				[211111]^X	0			[211111]^X	0	[222221]^X	0
				[211112]^X	0			[211112]^X	0	[222222]^X	0
				[211121]^X	0			[211121]^X	0	[222312]^X	0
Total: 10000 (100%)				**Total: 10000 (100%)**				**Total: 10000 (100%)**			

RuleII_JA				RuleII_ST				RuleII_STA			
Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.	Patterns	Num.
[111111]^X	0			[111111]^X	0	[211122]^X	0	[111111]^X	0	[211122]^X	0
[111112]^X	0			[111112]^X	0	[211211]^X	0	[111112]^X	0	[211211]^X	0
[111121]^X	0			[111121]^X	0	[211212]^X	0	[111121]^X	0	[211212]^X	0
[111211]^X	0			[111122]^X	0	[211221]^X	0	[111122]^X	0	[211221]^X	0
[111212]^X	0			[111211]^X	0	[211222]^X	0	[111211]^X	0	[211222]^X	0
[111221]^X	0			[111212]^X	0	[212111]^X	0	[111212]^X	0	[212111]^X	0
[112111]^X	0			[111221]^X	0	[212112]^X	0	[111221]^X	0	[212112]^X	0
[112112]^X	0			[111222]^X	0	[212121]^X	1444	[111222]^X	0	[212121]^X	1444
[112121]^X	0			[112111]^X	0	[212122]^X	0	[112111]^X	0	[212122]^X	0
[112211]^X	0			[112112]^X	0	[212211]^X	0	[112112]^X	0	[212211]^X	0
[112212]^X	0			[112121]^X	0	[212212]^X	0	[112121]^X	0	[212212]^X	0
[121111]^X	0			[112122]^X	0	[212221]^X	0	[112122]^X	0	[212221]^X	0
[121112]^X	0			[112211]^X	0	[212222]^X	0	[112211]^X	0	[212222]^X	0
[121121]^X	0			[112212]^X	0	[221111]^X	0	[112212]^X	0	[221111]^X	0
[121211]^X	0			[112221]^X	0	[221112]^X	0	[112221]^X	0	[221112]^X	0
[121212]^X	1462			[112222]^X	0	[221121]^X	0	[112222]^X	0	[221121]^X	0
[121221]^X	2859			[121111]^X	0	[221122]^X	0	[121111]^X	0	[221122]^X	0
[122111]^X	0			[121112]^X	0	[221211]^X	0	[121112]^X	0	[221123]^X	0
[122112]^X	0			[121121]^X	0	[221212]^X	0	[121121]^X	0	[221211]^X	0
[122121]^X	4235			[121122]^X	0	[221221]^X	0	[121122]^X	0	[221212]^X	0
[211111]^X	0			[121211]^X	0	[221222]^X	0	[121211]^X	0	[221221]^X	0
[211112]^X	0			[121212]^X	1462	[222111]^X	0	[121212]^X	1462	[221222]^X	0
[211121]^X	0			[121221]^X	2859	[222112]^X	0	[121221]^X	2859	[221223]^X	0
[211211]^X	0			[121222]^X	0	[222121]^X	0	[121222]^X	0	[221321]^X	0
[211212]^X	0			[122111]^X	0	[222122]^X	0	[122111]^X	0	[221322]^X	0
[211221]^X	0			[122112]^X	0	[222211]^X	0	[122112]^X	0	[221323]^X	0
[212111]^X	0			[122121]^X	4235	[222212]^X	0	[122121]^X	4235	[222111]^X	0
[212112]^X	0			[122122]^X	0	[222221]^X	0	[122122]^X	0	[222112]^X	0
[212121]^X	1444			[122211]^X	0	[222222]^X	0	[122211]^X	0	[222121]^X	0
[212211]^X	0			[122212]^X	0			[122212]^X	0	[222122]^X	0
[212212]^X	0			[122221]^X	0			[122221]^X	0	[222211]^X	0
				[122222]^X	0			[122222]^X	0	[222212]^X	0
				[211111]^X	0			[211111]^X	0	[222221]^X	0
				[211112]^X	0			[211112]^X	0	[222222]^X	0
				[211121]^X	0			[211121]^X	0	[222312]^X	0
Total: 10000 (100%)				**Total: 10000 (100%)**				**Total: 10000 (100%)**			

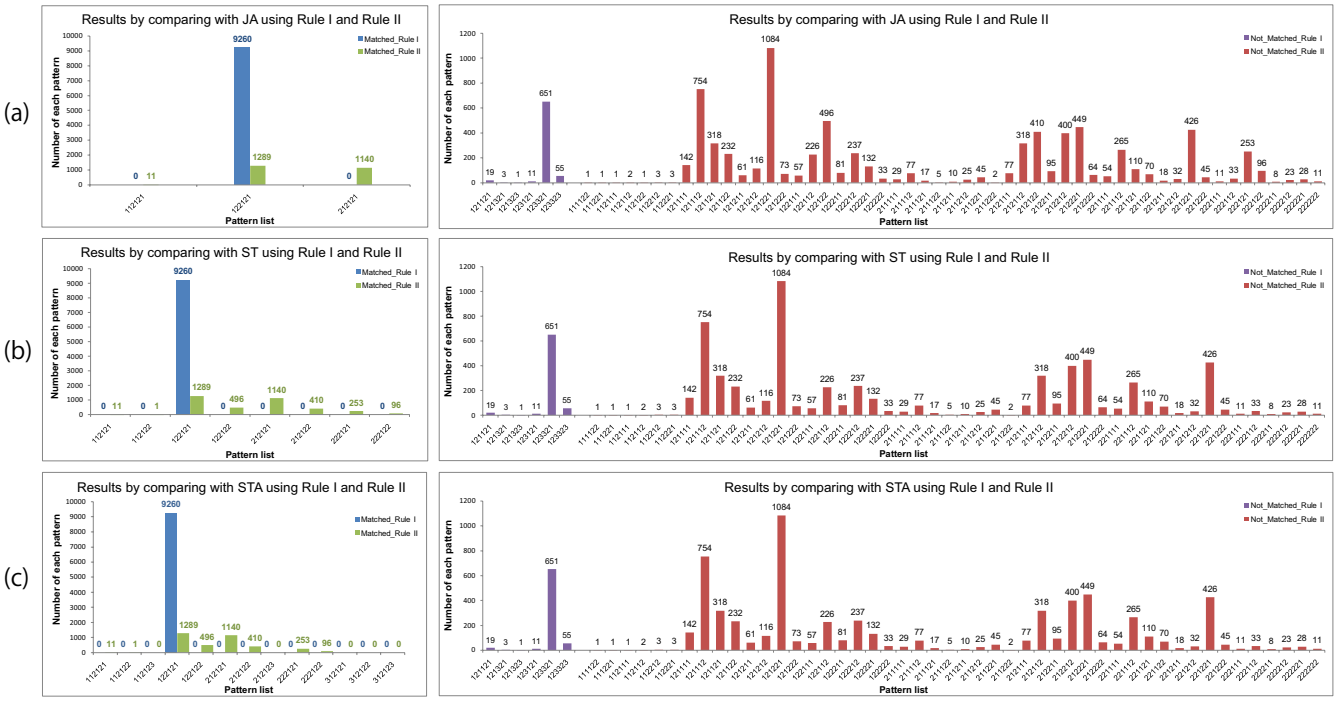


Figure 9
The simulation results of *lin-15ko* mutants (RowID 5).

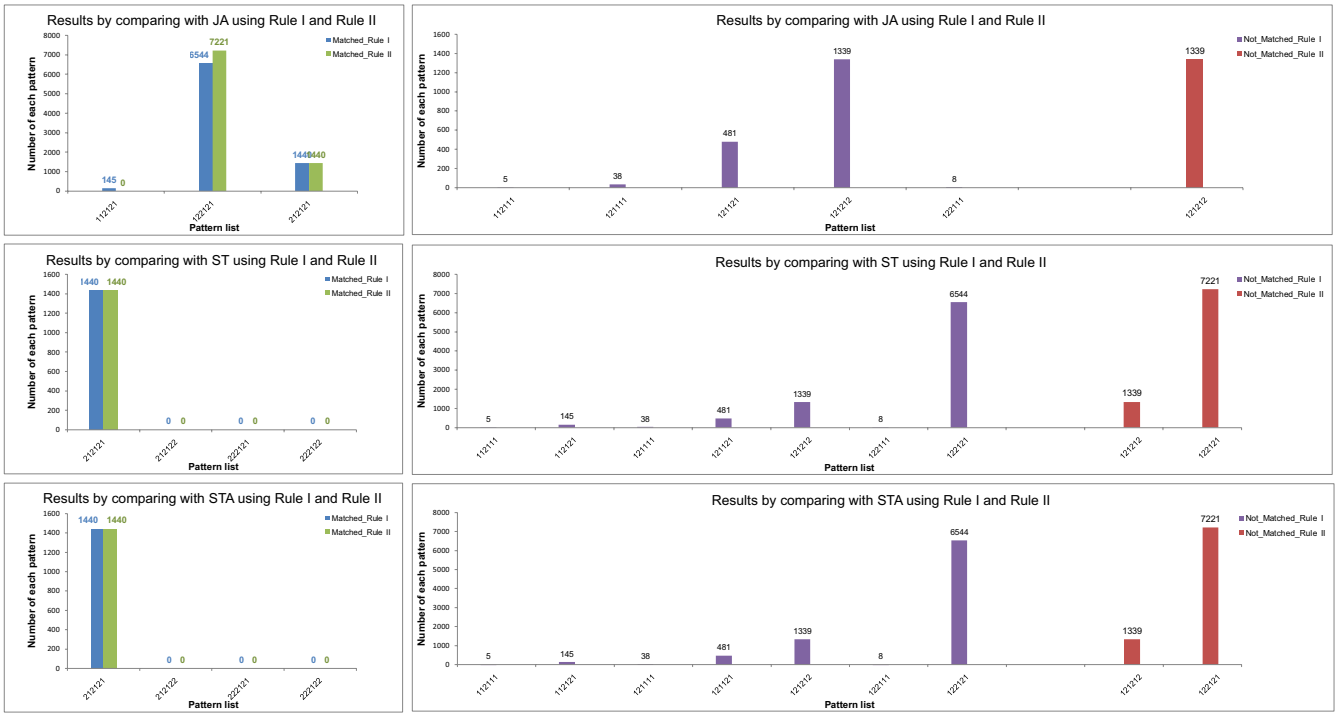


Figure 10
The simulation results of *lin-12gf; lin-15ko* double mutants (RowID 21).

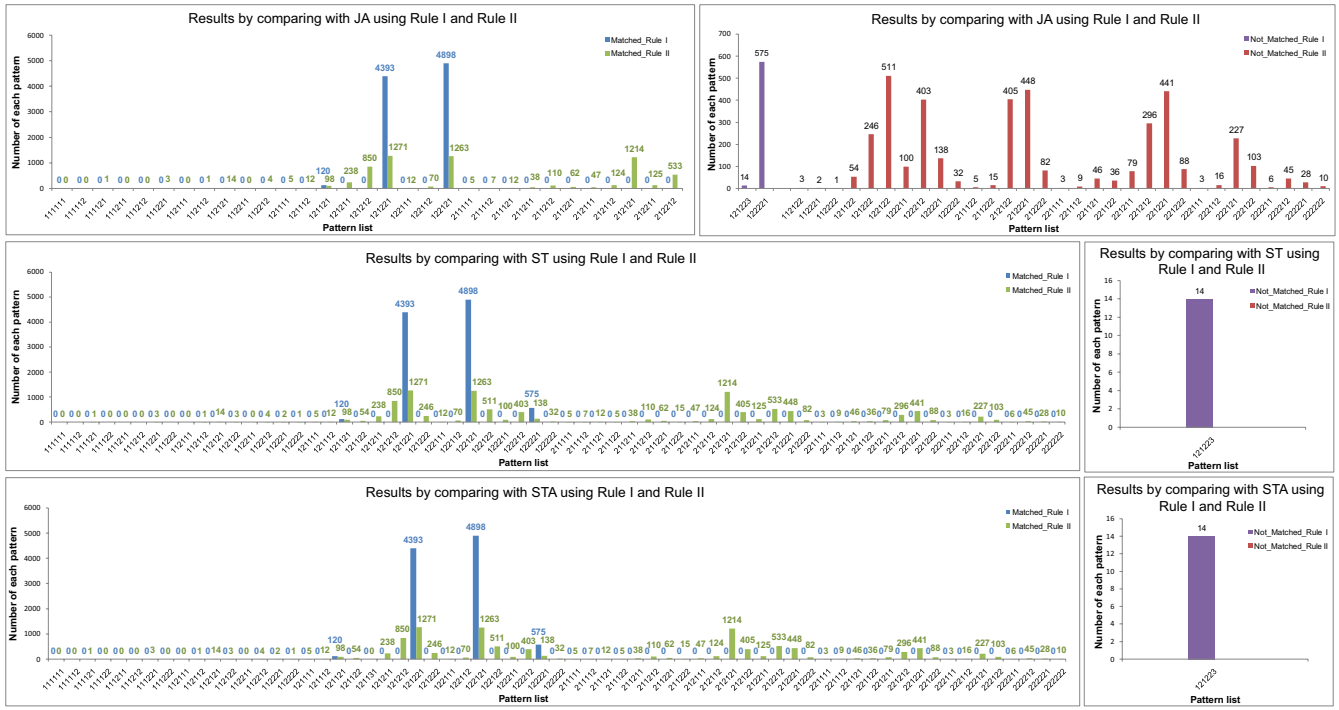


Figure 11
The simulation results of *ac*- and *lin-15ko* mutant (RowID 29).

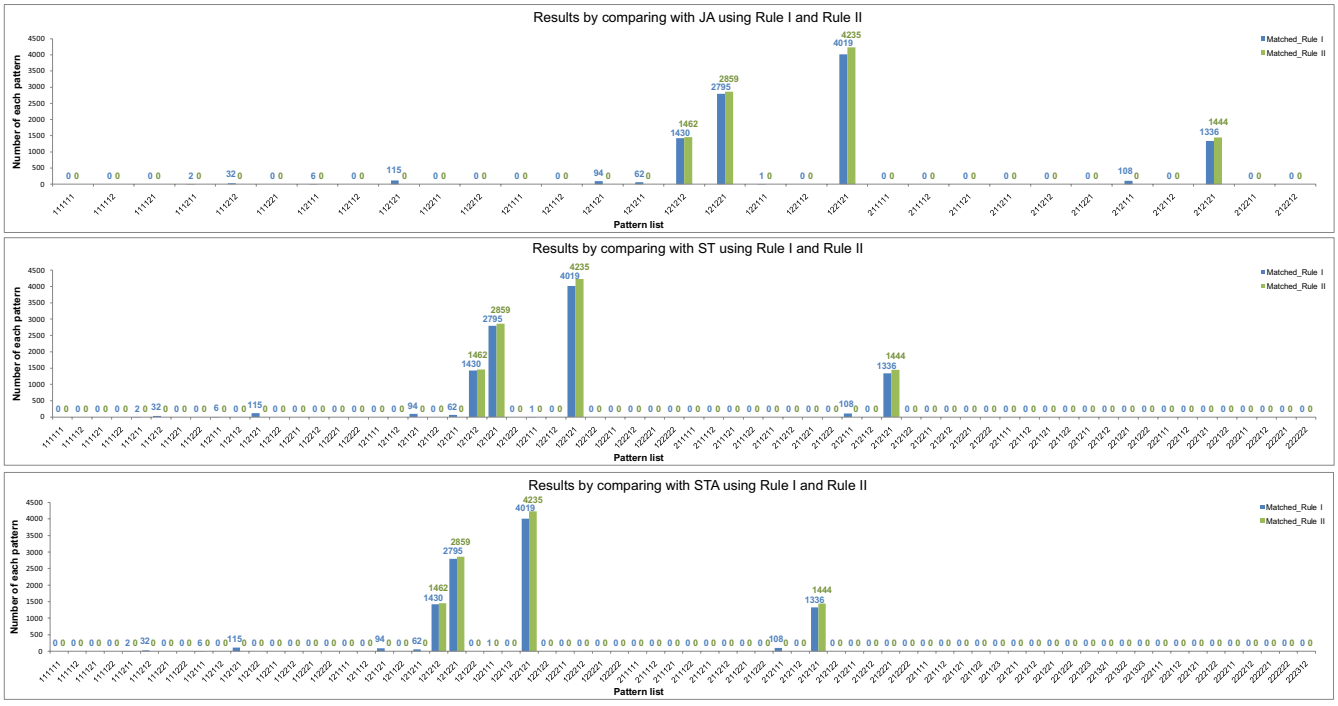


Figure 12
The simulation results of *ac*- and *lin-12gf; lin-15ko* double mutants (RowID 45).

be considered that P3.p has a greater possibility to adopt the 1° fate, if the number of animals for the *in vivo* experiments can be increased to 10,000. In this way, the fitting score can be exactly increased to exceed more than 81% from the present 15%.

(2) For *lin-15ko* (RowID 5) and *ac-;lin-15ko* (RowID 29) genotypes, we can see that nearly 10% patterns in JA, ST and STA are not matched (refer to Tables 11 and 13). These patterns still have the possibility to be examined in *in vivo* experiments by means of enlarging animal population.

(3) Since the experiment data about hybrid lineages has not been taken into account as we have pointed out, we interpret these uninterpretable lineages to three fate pattern extensions as listed in the column of STA which includes more extensive possible fate patterns than ST. In the simulation results, the results of *lin-12ko* mutant (RowID 9) interest us (refer to the Additional file 1). We can find that the fate pattern [331133] appeared in our simulation results of STA, which is the extended fate pattern from the hybrid lineage data exhibited in [32]. This hybrid lineage gives new insights on the imprecise fate decision and the fate specification mechanisms when a system comes close to a certain threshold.

Taking these observations together, we conclude that **Rule I** relying on the temporal interval can be considered more reasonable and proper in evaluating the VPC fate specification than **Rule II**.

Conclusion

The contribution of this paper is a novel method of modeling and simulating biological systems with the use of model checking approach on the hybrid functional Petri net with extension. A quantitative HFPNe model for the vulval development is constructed based on the literature. Then we employ two major biological fate determination rules to the quantitative model. These two rules are investigated by applying model checking approach in a quantitative manner. Three simulation targets of this model are considered: The first one is the fate patterns obtained by improving the qualitative method of Fisher *et al*. [9]; the second target is the fate patterns summarized by Sternberg and Horvitz [32]; and the last one is derived from the biological experiments in [32] including the hybrid lineage data. We have performed 480,000 simulations on the quantitative HFPNe model by using Cell Illustrator. We have examined the consistency and the correctness of the model, and evaluated the two rules of VPC fate specification. We consider that this computational experiment and the biological evaluation could not be easily put into practice without the HFPNe modeling method and the functions of Cell Illustrator, especially, the "High-Speed Simulation Module". Finally, *in silico* simulation results have been given in the form of the fitting score and the variation frequency of each pattern. We have discussed the results and summarized several plausible explanations.

Appendix: Abbreviations

AC: (gonadal anchor cell); ac-: (absence of an anchor cell); EGFR: (the epidermal growth factor receptor); HFPNe: (hybrid functional Petri net with extension); JA: (fate patterns obtained with our extended method from the discrete model of Fisher *et al*. [9]); LBS: (LAG-1 binding site); lst: (lateral signal target); MAPK: (mitogen-activated protein kinases); ST: (fate patterns summarized by Sternberg and Horvitz [32]); STA: (fate pattern combinations consisting of ST and the pattern extensions); VPC:(vulval precursor cell); synMuv: (synthetic Multivulva);

Authors' contributions

The basic idea was considered by MN and further developed by CL and MN. CL created biological diagram of multiple signaling pathways of *C. elegans* depicting VPC fate specification in Figure 4. CL and KU constructed the original HFPNe model in Figures 5 and 6. The whole HFPNe model was refined by CL and MN. CL and MN evaluated the results of simulation and validated the performance of HFPNe model. SM supervised the whole study. The final manuscript was read and approved by all authors.

Additional material

Additional File 1

Preliminary notations and mathematical definitions of Rule I and Rule II. *The data provide the details of preliminary notations and mathematical definitions of two rules to determine the cell fate.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-0509-3-42-S1.pdf]

Additional File 2

Simulation results of 44 genotypes. *The data give the 10,000 simulation results without unstable fate patterns. Yellowish region means that corresponding genotype possesses experimental evidences exhibited in [32]. Bright yellow cells give the fitting scores of predicted fate patterns.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-0509-3-42-S2.pdf]

Acknowledgements

We are grateful to the anonymous reviewers for their valuable hints and suggestions. This work was supported by KAKENHI (Grant-in-Aid for Scientific Research) on Priority Areas "Systems Genomics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

1. Clarke EM, Grumberg O, Peled DA: *Model Checking* The MIT Press; 1999.
2. Bérard B, Bidoit M, Finkel A, Laroussinie F, Petit A, Petrucci L, Schnoebelen P, McKenzie P: *Systems and Software Verification: Model-Checking Techniques and Tools* Springer; 2001.
3. Clarke EM, Emerson EA: **Design and synthesis of synchronization skeletons using branching time temporal logic.** In *Logic of Programs* New York: Springer Berlin/Heidelberg; 1982:52-71.
4. Queille JP, Sifakis J: **Specification and verification of concurrent systems in CESAR.** *Lecture Notes in Computer Science* 1982, **137:**337-351.
5. Chabrier N, Fages F: **Symbolic model checking of biochemical networks.** In *Proceedings of CMSB 2003 (Computational Methods in Systems Biology)* Italy: Springer Berlin/Heidelberg; 2003:149-162.
6. Antoniotti M, Policriti A, Ugel N, Mishra B: **Model building and model checking for biochemical processes.** *Cell Biochem Biophys* 2003, **38(3):**271-286.
7. Batt G, Ropers D, de Jong H, Geiselmann J, Mateescu R, Page M, Schneider D: **Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia coli.** *Bioinformatics* 2005, **21(Suppl 1):**i19-i28.
8. Calder M, Vyshemirsky V, Gilbert D, Orton R: **Analysis of signalling pathways using the prism model checker.** In *Proceedings of CMSB 2005 (Computational Methods in Systems Biology)* Edited by: Plotkin G. University of Edinburgh; 2005:179-190.
9. Fisher J, Piterman N, Hajnal A, Henzinger TA: **Predictive modeling of signaling crosstalk during C. elegans vulval development.** *PLoS Comput Biol* 2007, **3(5):**e92.
10. Heath J, Kwiatkowska M, Norman G, Parker D, Tymchyshyn O: **Probabilistic model checking of complex biological pathways.** *Theor Comput Sci* 2008, **391(3):**239-257.
11. Kwiatkowska MZ, Norman G, Parker D: **Using probabilistic model checking in systems biology.** *SIGMETRICS Performance Evaluation Review* 2008, **35(4):**14-21.
12. Monteiro PT, Ropers D, Mateescu R, Freitas AT, de Jong H: **Temporal logic patterns for querying dynamic models of cellular interaction networks.** *Bioinformatics* 2008, **24(16):**i227-i233.
13. Alur R, Henzinger TA: **Reactive modules.** *Formal Methods in System Design* 1999, **15:**7-48.
14. Regev A, Silverman W, Shapiro E: **Representation and simulation of biochemical processes using the pi-calculus process algebra.** *Pac Symp Biocomput* 2001:459-470.
15. Hatakeyama M, Kimura S, Naka T, Kawasaki T, Yumoto N, Ichikawa M, Kim JH, Saito K, Saeki M, Shirouzu M, Yokoyama S, Konagaya A: **A computational model on the modulation of mitogen-activated protein kinase (MAPK) and Akt pathways in heregulin-induced ErbB signalling.** *Biochem J* 2003, **373(2):**451-463.
16. Matsuno H, Tanaka Y, Aoshima H, Doi A, Matsui M, Miyano S: **Biopathways representation and simulation on hybrid functional Petri net.** In *Silico Biol* 2003, **3(3):**389-404.
17. Nagasaki M, Doi A, Matsuno H, Miyano S: **Computational modeling of biological processes with Petri net based architecture.** In *Bioinformatics Technologies* Edited by: Chen YP. Springer Berlin Heidelberg; 2005.
18. Nagasaki M, Doi A, Matsuno H, Miyano S: **A versatile Petri net based architecture for modeling and simulation of complex biological processes.** *Genome Inform* 2004, **15(1):**180-197.
19. Pinney JW, Westhead DR, McConkey GA: **Petri net representations in systems biology.** *Biochem Soc Trans* 2003, **31(Pt 6):**1513-1515.
20. Matsuno H, Li C, Miyano S: **Petri net based descriptions for systematic understanding of biological pathways.** *IEICE Trans Fundamentals* 2006, **E89-A(11):**3166-3174.
21. Li C, Ge QW, Nakata M, Matsuno H, Miyano S: **Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets.** *J Biosci* 2007, **32(1):**113-127.
22. Chaouiya C: **Petri net modelling of biological networks.** *Brief Bioinform* 2007, **8(4):**210-219.
23. Koch I, Heiner M: **Petri nets.** In *Analysis of Biological Networks* Edited by: Junker BH, Schreiber F. A Wiley Interscience Publication; 2008:139-180.
24. **Cell Illustrator** [http://www.cellillustrator.com/]
25. **Cell Illustrator Online** [http://cionline.hgc.jp/]
26. Nagasaki M, Doi A, Matsuno H, Miyano S: **Genomic Object Net: I. A platform for modelling and simulating biopathways.** *Appl Bioinformatics* 2003, **2(3):**181-184.
27. Doi A, Fujita S, Matsuno H, Nagasaki M, Miyano S: **Constructing biological pathway models with hybrid functional Petri nets.** In *Silico Biol* 2004, **4(3):**271-291.
28. Doi A, Nagasaki M, Fujita S, Matsuno H, Miyano S: **Abstract Genomic Object Net: II. modelling biopathways by hybrid functional Petri net with extension.** *Appl Bioinformatics* 2003, **2(3):**185-188.
29. Matsuno H, Doi A, Nagasaki M, Miyano S: **Hybrid Petri net representation of gene regulatory network.** *Pac Symp Biocomput* 2000:341-352.
30. Matsuno H, Murakami R, Yamane R, Yamasaki N, Fujita S, Yoshimori H, Miyano S: **Boundary formation by notch signaling in *Drosophila multicellular* systems: experimental observations and gene network modeling by Genomic Object Net.** *Pac Symp Biocomput* 2003:152-163.
31. Troncale S, Tahi F, Campard D, Vannier JP, Guespin J: **Modeling and simulation with hybrid functional Petri nets of the role of interleukin-6 in human early haematopoiesis.** *Pac Symp Biocomput* 2006:427-438.
32. Sternberg PW, Horvitz HR: **The combined action of two intercellular signaling pathways specifies three cell fates during vulval induction in C. elegans.** *Cell* 1989, **58(4):**679-693.
33. Peterson JL: *Petri Net Theory and the Modeling of Systems* Prentice Hall; 1981.
34. Greenwald I: **LIN-12/Notch signaling in C. elegans.** *Worm Book* 2005, **8:**1-16.
35. Yoo A, Bais C, Greenwald I: **Crosstalk between the EGFR and LIN-12/Notch pathways in C. elegans vulval development.** *Science* 2004, **303(5658):**663-666.
36. Christensen S, Kodoyianni V, Bosenberg M, Friedman L, Kimble J: **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* 1996, **122(5):**1373-1383.
37. Cui M, Chen J, Myers TR, Hwang BJ, Sternberg PW, Greenwald I, Han M: **SynMuv genes redundantly inhibit lin-3/EGF expression to prevent inappropriate vulval induction in C. elegans.** *Dev Cell* 2006, **10(5):**667-672.
38. Miller LM, Gallegos ME, Morisseau B, Kim S: **lin-31, a Caenorhabditis elegans HNF-3/fork head transcription factor homolog, specifies three alternative cell fates in vulval development.** *Genes Dev* 1993, **7(6):**933-947.
39. Sternberg PW, Horvitz HR: **Pattern formation during vulval development in C. elegans.** *Cell* 1986, **44(5):**761-772.
40. Lackner M, Kornfeld K, Miller LM, Horvitz HR, Kim S: **A MAP kinase homolog, mpk-1, is involved in ras-mediated induction of vulval cell fates in Caenorhabditis elegans.** *Genes Dev* 1994, **8(2):**160-173.
41. Shaye DD, Greenwald I: **Endocytosis-mediated downregulation of LIN-12/Notch upon Ras activation in Caenorhabditis elegans.** *Nature* 2002, **420(6916):**686-690.
42. Sternberg PW: **Vulval development.** *Worm Book* 2005, **25:**1-28.
43. Kohara Y: *Caenorhabditis elegans – Symphony of 1000 Cells* Kyoritsu Shuppan; 1997. In Japanese
44. Riddle DL, Blumenthal T, Meyer BJ, Priess JR: *C. Elegans II: Monograph 33* Cold Spring Harbor Laboratory Press; 1998.
45. Tan PB, Lackner MR, Kim SK: **MAP kinase signaling specificity mediated by the LIN-1 Ets/LIN-31 WH transcription factor complex during C. elegans vulval induction.** *Cell* 1998, **93(4):**569-580.
46. Berset T, Hoier EF, Battu G, Canevascini S, Hajnal A: **Notch inhibition of RAS signaling through MAP kinase phosphatase LIP-1 during C. elegans vulval development.** *Science* 2001, **291(5506):**1055-1058.
47. Jeong E, Nagasaki M, Saito A, Miyano S: **Cell System Ontology: Representation for modeling, visualizing, and simulating biological pathways.** In *Silico Biol* 2007, **7(6):**623-638.
48. **Pnuts** [http://www.pnuts.org/]
49. Sulston JE, Horvitz HR: **Post-embryonic cell lineages of the nematode, Caenorhabditis elegans.** *Dev Biol* 1977, **56(1):**110-156.
50. Berset TA, Hoier EF, Hajnal A: **The C. elegans homolog of the mammalian tumor suppressor Dep-1/Scc1 inhibits EGFR sig-**

naling to regulate binary cell fate decisions. *Genes Dev* 2005, **19(11):**1328-1340.
51. Ferguson EL, Sternberg PW, Horvitz HR: **A genetic pathway for the specification of the vulval cell lineages of Caenorhabditis elegans.** *Nature* 1987, **326(6110):**259-267.
52. Kimble J: **Alterations in cell lineage following laser ablation of cells in the somatic gonad of Caenorhabditis elegans.** *Dev Biol* 1981, **87(2):**286-300.
53. Han M, Aroian RV, Sternberg PW: **The let-60 locus controls the switch between vulval and nonvulval cell fates in Caenorhabditis elegans.** *Genetics* 1990, **126(4):**899-913.
54. **VPC fate specification of C. elegans in CSML** [http://www.csml.org/models/csml-models/vulvaldev/]
55. Alur R, Henzinger TA, Mang FYC, Qadeer S, Rajamani SK, Tasiran S: **MOCHA: modularity in model checking.** *Lecture Notes in Computer Science* 1998, **1427:**521-525.