A Versatile Petri Net Based Architecture for Modeling and Simulation of Complex Biological Processes

Masao Nagasaki[1] masao@ims.u-tokyo.ac.jp
Atsushi Doi[2] atsushi@ib.sci.yamaguchi-u.ac.jp
Hiroshi Matsuno[3] matsuno@sci.yamaguchi-u.ac.jp
Satoru Miyano[1] miyano@ims.u-tokyo.ac.jp

[1] Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokane-dai, Minato-ku, Tokyo 108-8639, Japan
[2] Graduate School of Science and Engineering, Yamaguchi University, 1677-1 Yoshida, Yamaguchi 753-8512, Japan
[3] Faculty of Science, Yamaguchi University, Yoshida, Yamaguchi 753-8512, Japan

Keywords: biopathway, hybrid functional Petri net with extension, biological process modeling, simulation, p53, frameshift, alternative splicing, Huntington's Disease

Abstract

The research on modeling and simulation of complex biological systems is getting more important in Systems Biology. In this respect, we have developed Hybrid Function Petri net (HFPN) that was newly developed from existing Petri net because of their intuitive graphical representation and their capabilities for mathematical analyses. However, in the process of modeling metabolic, gene regulatory or signal transduction pathways with the architecture, we have realized three extensions of HFPN, (i) an entity should be extended to contain more than one value, (ii) an entity should be extended to handle other primitive types, *e.g.* boolean, string, (iii) an entity should be extended to handle more advanced type called object that consists of variables and methods, are necessary for modeling biological systems with Petri net based architecture. To deal with it, we define a new enhanced Petri net called hybrid functional Petri net with extension (HFPNe). To demonstrate the effectiveness of the enhancements, we model and simulate with HFPNe four biological processes that are difficult to represent with the previous architecture HFPN.

1 Introduction

Our knowledge about cellular processes with biological experiments is increasing at a rapidly growing pace. With the knowledge, one of the most important topics in the Systems Biology is to understand and predict a cell's behavior. Related to this, particularly needed is a software environment with which biological and medical scientists (users) can comfortably model and simulate biological processes in the cell. Especially among biological processes, modelings and simulation of gene regulations, metabolic pathways, and signal-transduction cascades (say biopathways) are usually focused on the targets of software platforms.

First of all, to model and simulate biopathways, architecture is necessary. Thus, in 1999, we surveyed which architecture is suitable when modeling and simulating biopathway for biological and medical scientists. At that time, there were ODE-based attempts for modeling and simulating chemical reactions - *e.g.* Gepasi [26], E-Cell [35] - and others - *e.g.* the Lisp [25] based architecture, QSIM [17] and our other work, the π-Calculus based architecture, Bio-Calculus [28]. Unfortunately, applications based on these architectures are not acceptable in their fields. This is due to poor GUI interfaces, *e.g.* lacking biopathway editors, or their architectures themselves. To overcome this situation, we came to the conclusion that an architecture based on Petri nets should be suitable because of their intuitive graphical representation and their capabilities for mathematical analyses [29, 30, 31]. Then we have newly developed HFPN for easy biopathway modeling [22]. With HFPN, we can model rule based biological processes in biopathways, *e.g.* gene regulation and also ODEs-based kinetics, *e.g.* chemical processes in biopathways. In fact, with HFPN, we have modeled and simulated: the glycolytic pathway of *Escherichia coli*, gene regulation of circadian rhythms in *Drosophila*, boundary formation by notch signaling in *Drosophila* [23], and apoptosis induced by *Fas* ligand [21].

However, when modeling biopathways with HFPN, we have realized that three extensions will be useful 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 biopathways (*e.g.* alternative splicings, frameshiftings). The first is, an entity should be extended to contain more than one value, such as `list` and `pair`, because, (i) every part in a cell should contain 3D information, *e.g.* position and speed, (ii) proteins often have many modified states, *e.g. p53* has known sixteen phosphorylation positions and two acetylation positions as in Table 8 and modified states of *p53* can be 2^{18} [16]. In HFPN, an entity can contain only one value. The second is, HFPN should be extended to handle other primitive types, *e.g.* `boolean`, `string`, because major parts in a cell contain information similar to strings such as DNA sequences, mRNA sequences and protein sequences. In HFPN, an entity has only two types, `discrete` (non-negative integers) and `continuous` (non-negative real numbers). The third is, HFPN should be extended to handle more advanced type, `object` that consists of variables and methods. Many parts in a cell, *e.g.* DNA, mRNA, protein, have known functions, *e.g.* translation, transcription, degradation, and modification. Thus, if an entity takes the type `object` that has the methods with these known functions, each process that connects to the entity only needs to call a method of the `object` in the entity. To realize these three extensions, we have defined two special components; entity and process, called *generic entity* and *generic process*, respectively. We name this extension of Petri net *hybrid functional Petri net with extension* (HFPNe) that is defined in Section 2. By extensively using the new features of HFPNe, we model two biological processes that are not normally treated in biopathway modelings, alternative splicing and frameshifting, in Sections 3.1 and 3.2 and two biopathways that contain complicated processes, the pathway of Huntington's disease and the pathway of multi domain modifiable protein *p53*, in Sections 3.3 and 3.4. All these models with HFPNe demonstrate the new features of HFPNe are important for easy modeling and simulation. In Section 4 discusses the related works of HFPNe by comparing other object based Petri nets, OCPN [19] and Reference nets [18] and our ongoing works.

2 HFPNe: Hybrid Functional Petri Net with Extension

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:

$$\begin{aligned}\langle type\rangle ::= \;& \texttt{boolean} \;||\; \texttt{integer} \;||\; \texttt{integer+} \;||\; \texttt{real} \;||\; \texttt{real+} \;||\; \texttt{string} \;||\\ & \texttt{pair(}\langle type\rangle\texttt{,}\langle type\rangle\texttt{)} \;||\; \texttt{list}\ \langle type\rangle \;||\; \texttt{object(}\langle type\rangle\texttt{,}\cdots\texttt{,}\langle type\rangle\texttt{)}.\end{aligned}$$

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

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

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

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

connector type		process connector		
process type		discrete	continuous	generic
entity type	discrete	$\surd$	$-$	$\surd$
	continuous	$\surd$	$\surd$	$\surd$
	generic	$-$	$-$	$\surd$

connector type		associate or inhibitory connector		
process type		discrete	continuous	generic
entity type	discrete	$\surd$	$\surd$	$\surd$
	continuous	$\surd$	$\surd$	$\surd$
	generic	$\surd$	$\surd$	$\surd$

(a)

connector type		process connector		
process type		discrete	continuous	generic
entity type	discrete	$\surd$	$-$	$\surd$
	continuous	$\surd$	$\surd$	$\surd$
	generic	$\surd$	$\surd$	$\surd$

(b)

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 $\surd$ 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 $\surd$ means that the connection is allowed and $-$ means that the connection is not allowed.

4. $D(\texttt{object}(\theta_1, \cdots, \theta_n)) = D(\theta_1) \times \cdots \times D(\theta_n)$.

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

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 $\mathcal{M}$ the set of all markings of E. We can regard $\mathcal{M}$ as the set $\prod_{e \in E} D(\tau(e))$. Consider a function $f : \mathcal{M} \to \mathbf{R}$. For a subset $F \subseteq E$ and an element $v \in \prod_{e \in F} D(\tau(e))$, let $f[F = v] : \prod_{e \in E-F} D(\tau(e)) \to \mathbf{R}$ be the function obtained from f by restricting the value for F to v, i.e. $f[F = v](z) = f(z, v)$ for $z \in \prod_{e \in E-F} D(\tau(e))$. 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 $v \in \prod_{e \in E-F} D(\tau(e))$.

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

2. $h : E \cup P \to \{\texttt{discrete}, \texttt{continuous}, \texttt{generic}\}$ is a mapping called the *hybrid function* [†] A process $p \in P$ with $h(p) = \texttt{discrete}$ (resp., `continuous`, `generic`) is called a *discrete process* (resp., *continuous process*, *generic process*). An entity $e \in E$ with $h(e) = \texttt{discrete}$ (resp., `continuous`, `generic`) is called a *discrete entity* (resp., *continuous entity*, *generic entity*).

[*]Terms "entity" and "process" correspond to *place* and *transition*, respectively.

[†]Terms "discrete" and "continuous" correspond to those in hybrid Petri net [1] and "generic" is a newly introduced name which can be of any type in T.

3. $\tau : E \to T$ is a type function for E such that $\tau(e) = \texttt{integer+}$ if e is a discrete entity, and $\tau(e) = \texttt{real+}$ 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 $a : EP \cup PE \to \{\texttt{process}, \texttt{associate}, \texttt{inhibitor}\}$ called the *connector type function* which satisfies the conditions: (i) $a(c) = \texttt{process}$ 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 $a(c) = \texttt{process}$ (resp., $\texttt{associate}$, $\texttt{inhibitor}$)[§]. For a connector $c = (p,e) \in PE$, $a(c) = \texttt{process}$ 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*) if p is a discrete process (resp., continuous process, generic process). Let $\mathcal{M}$ be the set of all markings of E and let F be the set of continuous entities in E. Then we denote $\mathcal{D}_{\texttt{discrete}} = \{f \mid f : \mathcal{M} \to \mathbf{N}\}$, $\mathcal{D}_{\texttt{continuous}} = \{f \mid f : \mathcal{M} \to \mathbf{R}^{\geq 0}$ is continuous for $F\}$, $\mathcal{D}_{\texttt{generic}} = \{f \mid f : \mathcal{M} \to D^*\}$, and $\mathcal{D}_{\texttt{boolean}} = \{f \mid f : \mathcal{M} \to \{\texttt{true}, \texttt{false}\}\}$. Then w and u are given as follows:

   (a) $w : EP \to \mathcal{D}_{\texttt{discrete}} \cup \mathcal{D}_{\texttt{continuous}} \cup \mathcal{D}_{\texttt{boolean}}$ is a function called the *activity function* such that for a connector $c \in EP$ (i) $w(c) \in \mathcal{D}_{\texttt{discrete}}$ if c is discrete, (ii) $w(c) \in \mathcal{D}_{\texttt{continuous}}$ if c is continuous, (iii) $w(c) \in \mathcal{D}_{\texttt{boolean}}$ 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) $u : EP \cup PE \to \mathcal{D}_{\texttt{discrete}} \cup \mathcal{D}_{\texttt{continuous}} \cup \mathcal{D}_{\texttt{generic}}$ 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) $u(c) \in \mathcal{D}_{\texttt{discrete}}$ if c is discrete. (ii) $u(c) \in \mathcal{D}_{\texttt{continuous}}$ if c is continuous. (iii) If c is generic, then $u(c)$ is a function in $\mathcal{D}_{\texttt{generic}}$ such that $u(c)(M)$ is in $D(\tau(e))$ for any marking $M \in \mathcal{M}$. 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. $d : P_{\texttt{discrete}} \to \mathcal{D}_{\texttt{continuous}}$ is a mapping called the *delay*, where $P_{\texttt{discrete}}$ is the set of discrete processes in P. For a discrete process p, $d(p) : \mathcal{M} \to \mathbf{R}^{\geq 0}$ 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.

For graphical representation, HFPNe inherits the tradition of other Petri nets as in Figure 1.

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, $M_r(e,t) = \texttt{null}$ (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 *unenabled* at time t.

[‡] "Connector" corresponds to *arc.*

[§] "Process connector", "associate connector" and "inhibitory connector" correspond to *arc with weight*, *test arc* and *inhibitory arc*, respectively.

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 $a(c) \neq \texttt{inhibitor}$.
   (b) $\tilde{M}(e, t) < w(e, p)(M(t))$ if $a(c) = \texttt{inhibitor}$.
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 $a(c) \neq \texttt{inhibitor}$.
   (b) $\tilde{M}(e, t) \leq w(e, p)(M(t))$ if $a(c) = \texttt{inhibitor}$.
3. If p is a generic process, then for all connectors $c = (e, p) \in EP$ the following conditions hold:
   (a) $w(e, p)(\tilde{M}(t)) = \texttt{true}$ if $a(c) \neq \texttt{inhibitor}$.
   (b) $w(e, p)(\tilde{M}(t)) = \texttt{false}$ if $a(c) = \texttt{inhibitor}$.

If an unenabled 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 unenabled or an unenabled 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 $E - E_{\texttt{continuous}}$ 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 $E_{\texttt{continuous}} = \{e \in E \mid e \text{ is continuous}\}$.
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 $E_{\texttt{continuous}}$ and $\tilde{M}(t')$ is constant on $E - E_{\texttt{continuous}}$ 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 $a(e, p) = \texttt{process}$
                $Tmp'(e) \leftarrow u(e, p)(Tmp)$
            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 $a(e,p) = \texttt{process}$
            $Tmp'(e) \leftarrow Tmp'(e) - \int_{t-\varepsilon_t}^{t} u(e,p)(\tilde{M}(x))dx$
        **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 $a(e,p) = \texttt{process}$
            $Tmp'(e) \leftarrow Tmp'(e) - u(e,p)(\tilde{M}(q(p,t)))$
        **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 $h(e) = \texttt{discrete}$ or $\texttt{continuous}$
        $Tmp_r(e) \leftarrow Tmp_r(e) - \sum_{\substack{p \text{ with } p \in S_d(t) \\ \text{and } (e,p) \in EP}} u(e,p)(\tilde{M}(q(p,t))) + \sum_{\substack{p \text{ with } p \in U_d(t) \\ \text{and } (e,p) \in EP}} u(e,p)(\tilde{M}(t-\varepsilon_t)).$
7.  $M_r(t) \leftarrow Tmp_r.$
```

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

2.1 Remarks in Implementation of HFPNe

In Genomic Object Net, we have implemented a simulator of HFPNe by approximating the time t by $t = 0, \delta, \cdots, k\delta, \cdots$ for integers k by using an appropriately small real number $\delta > 0$. Furthermore, the generic time α is also set to be $\delta = \alpha$ for simplicity.

Another issue in implementation is the problem of conflict resolution. In the above procedure, Step 2 for generic processes and Step 4 for discrete processes may have conflicts for execution. Let $p_{i_1}, \ldots, p_{i_t}$ be processes which can be fired in Step 2 or Step 4. In our implementation, we arrange these processes in a random order and execute the processes according to this order. During this execution, we will skip the processes which cannot be fired anymore due to the changes of marks of entities.

2.2 Relationships with Other Petri Nets

HFPN $H = (E, P, h, \tau, C, d)$ is defined from HFPNe by deleting all matters with "generic" and by adding the restriction that $u(c) = w(c)$ for any discrete connector $c = (e,p) \in EP$. This condition means that the weight $w(c)$ of the connector is the same as the number of tokens $u(c)$ removed from the entity e by firing. This convention is traditionally employed in Petri net as a weight. In our definition of HFPNe, however, we have separated these two notions.

HPN $H = (E, P, h, \tau, C = (EP, PE, a, w, u), d)$ [1] is defined by adding the following restriction to HFPN: (i) $w(c)$, $u(c)$, and $d(p)$ are constants for any connector $c \in EP \cup PE$ and any process $p \in P$. HDN $H = (E, P, h, \tau, C = (EP, PE, a, w, u), d)$ [10, 11] is also defined by adding the following

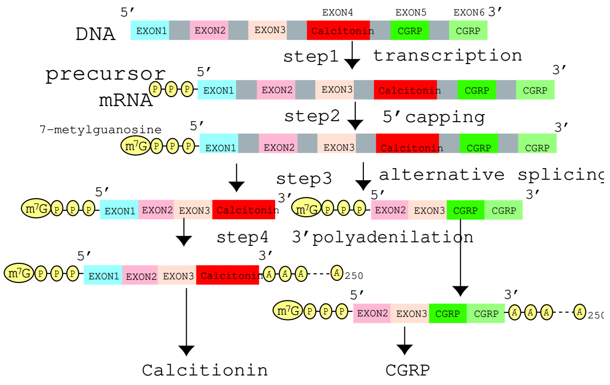


Figure 2: Alternative RNA splicing model of the *Calcitonin/CGRP* gene in Figure 3.

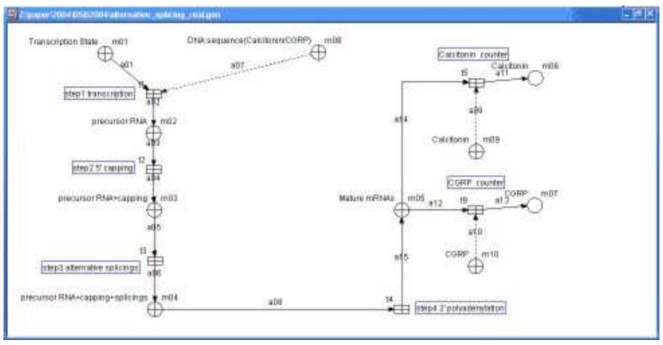


Figure 3: An alternative RNA splicing model of the *Calcitonin/CGRP* gene with HFPNe.

restriction to HFPN: (ii) For any continuous process p, it is assumed that $w(e,p) = u(p,e')$ for any process connectors $(e,p) \in EP$ and $(p,e') \in PE$.

If we delete all matters with "discrete" from HPN, we have the definition of CTPN. If we delete all matters with "continuous" from HPN, we have the definition of TPN. Furthermore, if we delete the matters with "delay" from TPN, we have the definition of the original Petri net. Thus, HFPNe is a highly abstract extension of Petri net for biological process modeling which can involve PN, TPN, CTPN, HPN and HFPN as its special cases.

3 Modeling of Biological Processes with HFPNe

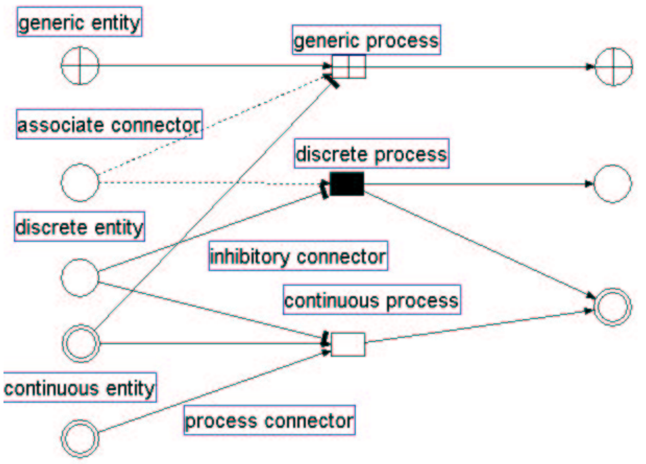


Figure 1: Graphical notations of HFPNe components with conventional way.

In this section, we demonstrate that more biological processes can be easily modeled with HFPNe than HFPN. For this purpose, we selected four biological processes for modeling that extensively use the HFPNe features. These biological processes are important activities in living cells and should be handled with application tools that aim to model and simulate biological systems. Our aim is not only to theoretically describe how to model biological processes but also to provide a useful application tool to users in biology and medicine. We have developed GON [9, 27] which is based on the HFPNe architecture. In the following sections, we exemplify with GON that complex biological processes can be modeled while illustrating snapshots of HFPNe models as in Figures 3, 4(a), 5 and 6.

3.1 From DNA to mRNA in Eucaryotes - Alternative Splicing

The mechanism from DNA to mRNA in eucaryote is more complicated than that in bacteria. The major difference derives from eucaryotic genes which consist of two regions, i.e. exons and introns. As in Figure 2, the mechanism mainly consists of four steps; **step1**: transcription, **step2**: 5' capping, **step3**: RNA splicing, and **step4**: 3' polyadenylation. In **step1**, DNA is transcribed into precursor-mRNA (specially named to distinguish from mRNA (mature-mRNA) after **step4**). In **step2**, 5' end of

Table 2: Update functions of connectors for the *Calcitonin/CGRP* transcription model in Figure 3. External Java classes `Transcription`, `Splicing` and `Splicing_CalcitoninCGRP` are in our site [43].

conn-ector	**connector type**	**update function**

a01	process	import("gon.Transcription");
		totalnum = m08.length(); num = m01.length();
		if(totalnum > num){
		nextcode = m08.substring(totalnum-num-1,totalnum-num);
		newsequence = m01 + Transcription::Trans(nextcode);}
		else{new_sequence = "";} return newsequence;

a02	process	import("gon.Transcription");
		if(Transcription::Finish(m01,m08)){return m01;} else{return "";}

a03	process	return m02;

a04	process	import("gon.Transcription");
		if(!m02.equals("")){return Transcription::Capping(m02);} else{return "";}

a05	process	return "";

a06	process	import("gon.Splicing_CalcitoninCGRP"); if(m03.equals("")){return m03;}
		else{return Splicing_CalcitoninCGRP::AlternativeSplicing(m03);}

a08	process	return "";
a10	process	return m10;

a11	process	num = 0; for(i=0;i<m05.length;++i){if(m05[i].equals(m09)){num++;}}
		return num;

a12	process	return m05;

a13	process	num = 0; for(i=0;i<m05.length;++i){if(m05[i].equals(m10)){num++;}}
		return num;

a14	process	return m05;

a15	process	import("gon.Transcription"); if(m04.equals("")){return m05;}
		else{return m05+[Transcription::Polyadenylation(m04)];}

entity	**entity type**	**type**	**initial mark**
m01/m02/m03/m04	generic	string	“”
m05	generic	list string	()
m06/m07	discrete	integer+	0
m08/m09/m10	generic	string	see [43]

Table 3: Properties of entities for the *Calcitonin/CGRP* transcription model in Figure 3.

the precursor-mRNA is modified. In **step3**, each splicing event removes one intron and accordingly all introns of the precursor-mRNA are removed. Finally in **step4**, 3' end of precursor-mRNA is modified to produce mature-mRNA in order to allow the cell to assess whether both ends of the mRNA are present before it exports the RNA sequence from the nucleus for translation into proteins.

In **step3**, there is a eucaryote specific splicing process named alternative RNA splicing. The alternative RNA splicing is to produce different mRNA from the same precursor-mRNA by splicing it in different ways. The DNA to mRNA transcription with alternative RNA splicing was found in 1982 for *Calcitonin/CGRP* gene expression [2]. We draw focus to model *Calcitonin/CGRP* gene expression while combining other recent biological knowledge [7, 36].

As in Figure 2, the *Calcitonin* gene expression consists of four introns and five exons, and the transcription process progresses with **step1**, **step2**, **step3** and **step4**. In **step3**, by alternative splicing events, one of exon1/exon2/exon3/exon4 (say set1) or exon2/exon3/exon5/exon6 (say set2) is selected. If set1 (set2) is selected, the mature-mRNA is translated into *Calcitonin* (*CGRP*), respectively. The HFPNe model can faithfully realize these steps as generic processes and realize precursor-mRNA and mature-mRNA states as generic entities with the type `string` (see Figure 3). In the model, the mature-RNAs, *Calcitonin* mRNA and *CGRP* mRNA, are represented by the generic entity `m09` with the type `list string`. The generic entities, `m12`,`m13`,`m14`, and `m15` are used to inform which sequence should be spliced in the generic processes, `t03`,`t04`,`t05`, and `t06`, respectively. These generic notions are necessary for modeling these four steps from DNA to mRNA. It is hard to model with HFPN [9] and other simulation tools [26, 35]. The detailed parameters and functions are described in Tables 2 and 3.

As another example of alternative splicing, *DSCAM* gene in *Drosophila*, where for exons (named exon A, B, C, and D) are selected and combined. Each precursor-mRNA contains 12 alternatives for exon A, 48 alternatives for exon B, 33 alternatives for exon C, 2 alternatives for exon D [38]. Thus, there are 38,016 possible mature mRNAs for the DSCAM gene. If the model is created with HFPN, we have to deal with 38,016 entities for this DSCAM transcription process. If the model is created with HFPN, 38016 entities are necessary for the simple *DSCAM* transcription model.

3.2 Translation of mRNA - Frameshift

Frameshift is also an important biological process that occurs during RNA translations. The frameshift is to skip or reread some ribonucleotides when translating RNAs and it is commonly utilized by many RNA viruses as programmed ribosomal frameshifting [14].

For example, a model of ribosomal frameshifting in the human immunodeficiency virus (HIV-1) (*gag-pol* expression in Figure 4(b)) can be modeled with HFPNe as in Figure 4(a). As in Figure 4, two proteins, *gag* and *pol*, are produced from one RNA sequence with the frameshift by rereading a ribonucleotide, adenine (A), twice. In HFPNe, the generic entity `e3` has the type `pair(integer (i1) ,string (s1))`, in order to express frameshifting states; `i1` denotes how many ribonucleotides are skipped (+1) or reread (-1) and `s1` denotes the length of current translated amino-acid sequence. The frameshifting occurs at a specific point. In addition the probability of reread at the point is 0.1 in wild type [14]. Thus, in the model we assign an mRNA translation skip function with stochastic behavior to this specific point. The stochastic feature is already realized in HFPN and HFPNe also inherits this feature. The detailed update functions are summarized in Table 4(c) and the initial marks and types of entities are summarized in Figure 4(d). It is also not straightforward to represent the ribosomal frameshifting with HFPN.

3.3 Huntington's Disease

As an advanced pathway model with HFPNe, we selected a genetic disease called Huntington's disease. Because in biology and medical sciences disease pathway modeling and simulation are becoming one of the most important topics to treat the disease from the pathway level. Huntington's disease is an autosomal dominant progressive neurodegenerative disorder that is characterized by chorea, psychiatric disturbances, and dementia [20]. The disease is in short the generic defects of *Huntingtin. Huntingtin* is a multi-domain protein with a polymorphic glutamine/proline (G/P)-rich domain at the N terminus [37]. Polyglutamine (polyQ) sequences in unaffected individuals range from 11 to 34 glutamine residues, whereas those of *Huntingtin* disease patients contain 37 or more glutamine residues (more than 90)[4]. The disease appears when a specific polyQ length is exceeded.

Based on the disease model proposed by Wellington *et al.* [13], we created a HFPNe model for this disease in Figure 5. Our model uses following known experimental facts in the literature.

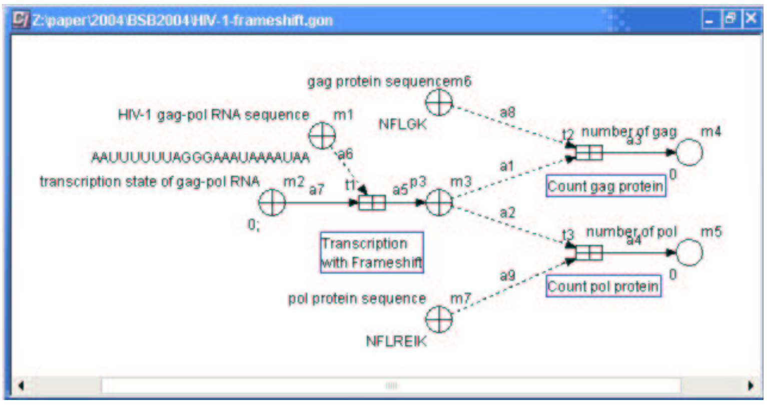
fact1 *Huntingtin* can be cleaved by *caspase-3* and yields two fragments, N terminal region (*NT*) that contains polyQ repeats in *Huntingtin* and C terminal region (*CT*) [13].

fact2 Both of *Huntingtin* disease and normal *Huntingtin* can cleave and the rates of fragmentation are the same [40].

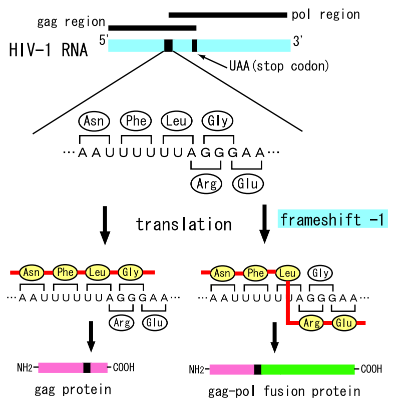
fact3 *Procaspase-3* has low-level catalytic activity for disease *Huntingtin* and is capable of cleaving the same substrates, i.e. *NT* and *CT*, as activated *caspase-3* [32].

fact4 When cleaved by *caspase-3*, *NT* has a responsibility for cytotoxicity [15, 41].

fact5 *NT* has the ability to form protein aggregates and can be found in the nucleus and in the cytoplasm [33].



(a)



(b)

connector	**update function**
a3	`m4 = 0;for(i=0;i<m3.length;++i){if(m3[i].equals(m6)){m4++;}}return m4;`
a4	`m5 = 0;for(i=0;i<m3.length;++i){if(m3[i].equals(m7)){m5++;}}return m5;`

a5	`import("gon.Translation");`
	`num=m2[1].length(); n=m1.substring(num*3+m2[0],(num+1)*3+m2[0]);`
	`if(Translation::EndCode(n)){ m3 += [m2[1]]; }return m3;`

a7	`import("gon.Translation");`
	`num=m2[1].length(); n=m1.substring(num*3+m2[0],(num+1)*3+m2[0]);`
	`if(!Translation::EndCode(n)){m2[1] += Translation::Trans(n);`
	` if(num==2 && Math::random()<=0.1){m2[0]=-1;}}`
	`else{m2[0] = 0;m2[1] = "";}return m2;`

(c)

entity	**entity type**	**type**	**initial mark**
`m1/m6/m7`	`generic`	`string`	see [43]
`m2`	`generic`	`pair(integer,string)`	(0,“”)
`m3`	`generic`	`list string`	()
`m4/m5`	`discrete`	`integer+`	0

(d)

Figure 4: (a) Ribosomal frameshifting in HIV-1 *gag-pol*. (b) Ribosomal frameshifting in HIV-1 *gag-pol* expressed with HFPNe. The characterization of ribosomal frameshifting was first described by Tyler [14]. (c) Update functions in (b). (d) Properties of entities in (b).

fact6 The aggregate number of *NT* increases in proportion to the length of polyQ [34].

fact7 *NT* fragments of mutant *Huntingtin* have the ability to induce *caspase-3* [6].

fact8 Both of disease and normal *NT* can cross the nuclear membrane. The smaller the cleavage product, the greater the tendency of *Huntingtin* cross [12, 39].

Huntingtin and *NT* change their function depending on their polyQ length (**fact3**, **fact6**, **fact7**). Thus, in our model, these proteins are represented with generic entities with the type: `pair(integer+, real+)`. The first attribute `integer+` corresponds to the length of polyQ and the second attribute `real+` corresponds to the quantity of the protein, i.e. *NT* or *Huntingtin*. In our model, *NT* and *Huntingtin* are represented with `m1` and `m4`, respectively. The properties of processes and entities in the model are summarized in Tables 4 and 5. *Huntingtin* `m4` is translated by the generic process `t09` while *procaspase-3* `m6` is translated by the continuous process `t10`. The *procaspase-3* `m6` can cleave `t12` disease *Huntingtin* (**fact3**). Thus, the weight function of associate connector `a14` has a function that depends on the length of huntingtin `m4` and cleavage process `t12` is not enabled for normal *Huntingtin*.

process	**process type**
`t01/t02/t04/t06/t07/t08/t10/t17/t18`	`continuous`
`t03/t05/t09/t11/t12/t13/t14/t15/t16/t19`	`generic`

Table 4: Properties of processes for Huntington's disease model in Figure 5.

entity	**entity type**	**type**	**initial mark**
`dt`	`continuous`	`real+`	0.0
`m1/m2/m4`	`generic`	`pair(integer+ ,real+)`	(n,0.0)
`m3/m5/m6/m7/m8/m9`	`continuous`	`real+`	0.0

Table 5: Properties of entities for the huntingtin disease model in Figure 5. Initial value n in `m1`, `m2`, and `m4` denotes the length of Q repeats in the huntingtin protein and must be assigned an integer value when simulating the model.

The activity functions and types of connectors are summarized in Table 6. On the other hand, the *caspase-3* `m5` can cleave normal *Huntingtin* and disease *Huntingtin* that is modeled with the process `t11` (**fact2**). Thus, the weight of associate connector `a21` has a function that does not depend on the length of *Huntingtin* `m4`. *Huntingtin* `m4` is cleaved and separated into *CT* `m3` and *NT* `m1` (**fact1**). *CT* does not contain the polyQ region and is represented with continuous entity `m3` (**fact1**).

NT inherits the polyQ region. The length of *NT* relates to the activity of the process `t13` that changes *procaspase-3* `m6` into *caspase-3* `m5` (**fact7**). To model it, the weight function of associate connector `a15` depends on the length of *NT* `m1` and process `t13` can not be activated by normal huntingtin.

NT in cytoplasm migrates to nucleus (**fact8**), and *NT* in nucleus `m2` causes cell death `t19` with the cytotoxicity (**fact4**). From **fact5**, cell death can be prevented by aggregation of *NT*. Moreover, the aggregation rate depends on the length of polyQ (**fact6**). *NT* in nucleus `m2` should therefore be denoted by generic entities such as *Huntingtin* `m4` and *NT* `m1` in cytoplasm. *NT* in the cytoplasm `m1` and *NT* in the nucleus `m2` connect to the aggregation process `t15` and `t16`, respectively. The update functions of process connectors `a23` and `a26` represent aggregate speeds and depend on the length of polyQ in *NT* in nucleus `m2` (**fact6**). The update functions of connectors are summarized in Table 7.

With the model, we can simulate the n polyQ length huntingtin ($0 \leq n \leq 100$ [4]) by only changing the initial marks of `m1`, `m2`, and `m4` as in Table 5. If the same disease model is created with HFPN, at least extra (i) one entity and (ii) many connectors are necessary; (i) an entity (say `e1`) that denotes the length of huntingtin, (ii) associate connectors that point from `e1` to every process that has more than one connector whose weight function or speed function depends on the value of `e1`. Thus, the HFPNe huntingtin disease model is simpler than the HFPN model.

3.4 Protein Modification - *p53*

The previous HFPNe models extensively use generic entities and generic processes. However, their types of generic entities are only simple ones, *e.g.* `string`, `list real`, `list string`, `pair(integer+, string+)`, `pair(integer+,real+)`. A generic entity can take more advanced type `object` as its type. Under HFPNe application GON, an object corresponds to a Java *class* and a *method* of an object is a specific function assigned to the object. If a generic entity has the type `object`, the entity can be initialized with an object with methods and variables and these methods can be applied to update functions of connectors that join the generic entity and generic processes, and change the mark by updating variables of the entity.

To show the effectiveness of generic entities with the type `object`, we deal with protein modification biological processes - *e.g.* phosphorylation, acetylation, and methylation - with enzymes. Specifically, the protein *p53* that can be modified by enzymes - *CK1*, *ATM*, *DNAPK*, *TFIIH*, *MAPK*, *PCAF*, *PKC*,

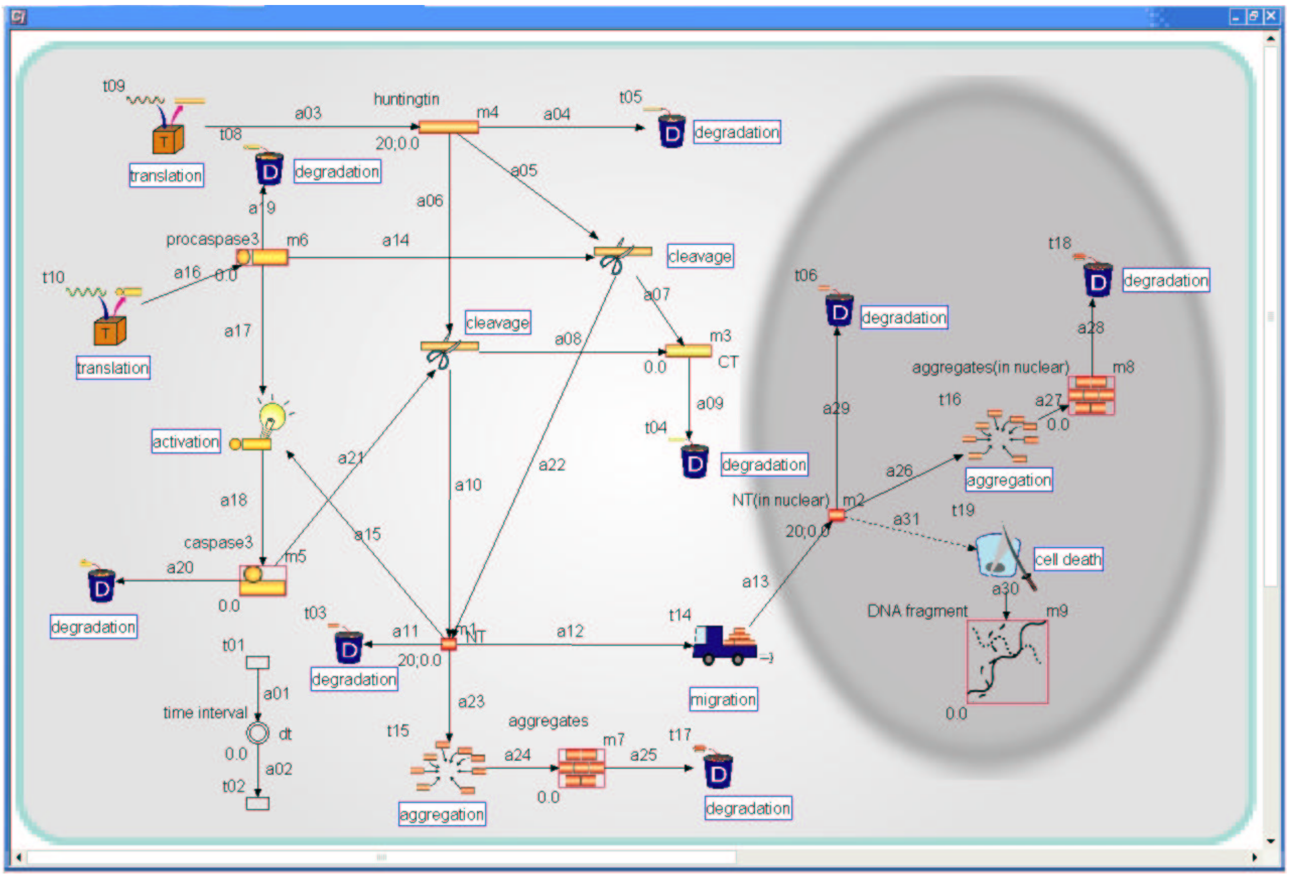


Figure 5: Huntington's disease model with HFPNe. HFPNe components in this figure are replaced with suitable images that represents biological information.

p300, *CK2*, and *JNK* - is modeled [16]. By these enzymes, eighteen positions of *p53* are modified. There are 2^{18} patterns. Thus, it is again hard to create a model with HFPN because 2^{18} entities are necessary for the model with HFPN.

To model with HFPNe while using the type `object` for generic entities, `Protein` and `Enzyme` classes are created and these classes are inherited in order to realize specific `Protein` class, *e.g.* `p53`, and `Enzyme` classes, *e.g.* `CK1`, `ATM`, `DNAPK`, `TFIIH`, `MAPK`, `PCAF`, `PKC`, `p300`, `CK2`, and `JNK`. As in Table 9, (i) the generic entity `m01` takes a specific object `ProteinSet` that is initialized with `p53` object that corresponds to `p53` class and inherits `Protein` class, (ii) the generic entities `m02`, `m03`, `m04`, `m05`, `m06`, `m07`, `m08`, `m09`, `m10`, and `m11` take specific objects whose classes inherit `Enzyme` classes `CK1`, `ATM`, `DNAPK`, `TFIIH`, `MAPK`, `PCAF`, `PKC`, `p300`, `CK2`, and `JNK`, respectively (a class name corresponds to an enzyme name). These objects are implemented with Java classes on GON. Full codes of these classes are available from [43].

To describe modification biological processes by enzymes, the generic processes `t01`,$\cdots$, `t10` are used. As in Table 10, the update functions `a13`,`a14`,`a15`,`a16`,`a17`, `a18`,`a19`,`a20`,`a21`,`a22` of the generic processes call the `Modify` method of `p53` class to simulate the modification biological process by CK1, ATM, DNAPK, TFIIH, MAPK, PCAF, PKC, p300, CK2, JNK, respectively. To describe translation and degradation biological processes of p53, the generic processes `t11` and `t12` are created, respectively. As in Table 10, processes `t11` and `t12` follow connectors with update functions `a21` and `a22` that call `Translation` and `Degradation` methods of `p53` class to simulate the translation and degradation biological processes, respectively.

At the first glance, HFPNe model of *p53* modification biological processes seems complicated. However, as in Table 10, in the *p53* model, all update functions just call a specific method, *e.g.* `Modify`,

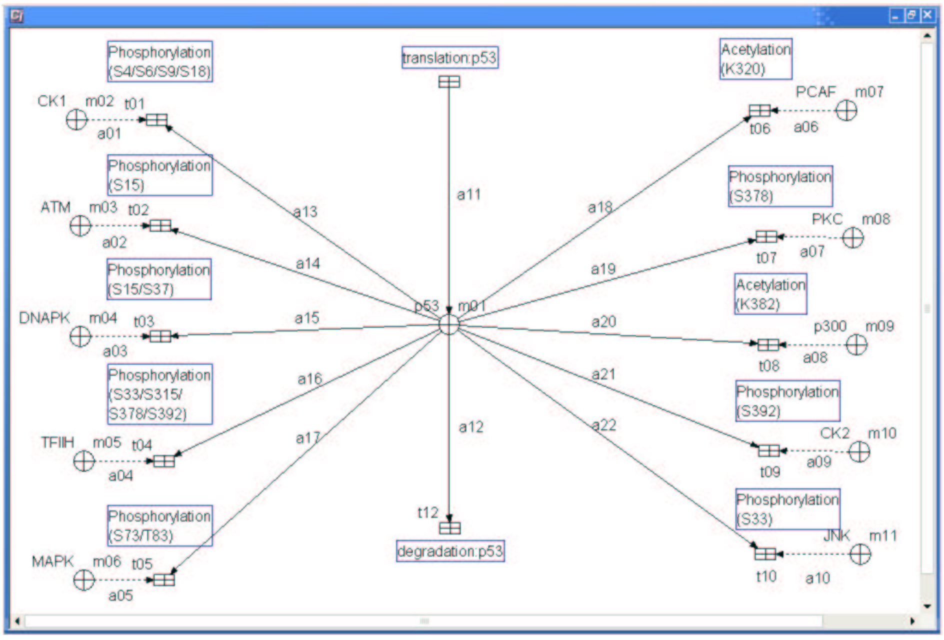


Figure 6: A protein modification model of the *p53* protein with HFPNe. The generic entity in the center `m01` is modification target *p53* and other generic entities `m02`,$\cdots$,`m11` are enzymes that modify the *p53* entity. Generic processes `t01`,$\cdots$,`t10` have connectors `a01`,$\cdots$,`a10` and by their update functions, the modified state of *p53* in `m01` is updated.

`Translation`, and `Degradation`. As in Table 9, all initial marks of generic entities are defined by just calling the specific method, i.e. `Initialize`. Thus, if objects with suitable methods that correspond to biological processes are created as a biopathway library, the majority of users, biological and medical scientists without programming skills, can create biological models. But with their expert knowledge of biological processes, they can create models by simply assigning a suitable object to a generic entity and a suitable method to the update function of the generic connector.

We can apply the same approach to the previous biological processes. In the previous example, generic entities can take type `object` whose class is `DNA` with `transcription` method or `RNA` with `translation` method instead of `string` primitive type. Thus, owing to the `object` extension of HFPNe, more complicated biological processes can be easily modeled than with HFPN.

On the other hand, the Gene Ontology (GO) Consortium is trying to produce a controlled vocabulary that can be applied to all organisms as knowledge of gene and protein roles in cells [3]. By applying these gene ontology related vocabularies to names of objects and their methods, we can create a more standardized library for biological processes with HFPNe.

Table 6: Properties of connectors for Huntington's disease model in Figure 5. A property of connectors, update function is summarized in Table 7.

connector	**connector type**	**activity function**
`a02/a09/a19/a20`	`process`	0.0
`a04/a05/a06/a11`	`process`	`return true;`
`a12`	`process`	`return {(m4[1]>20.0) ? true : false;}`
`a14`	`associate`	`return {(m4[0]>37&&m6>0.0) ? true : false;}`
`a15`	`associate`	`return {(m4[0]>37&&m1>0.0) ? true : false;}`
`a17`	`process`	20.0
`a21`	`associate`	`return {(m5[1]>0.0) ? true : false;}`
`a23/a26/a29`	`process`	`return true;`
`a25/a28`	`process`	1.0
`a31`	`process`	`return {(m2[1]>10.0) ? true : false;}`

Table 7: Update functions of connectors for the huntingtin disease model in Figure 5. Update function must be defined for all process connectors that are connected with at least one generic process.

connector	**connector type**	**update function**
a01/a02	process	1.0
a03	process	`m4[1] = m4[1]+1.0*dt; return m4;`
a04	process	`m4[1] = m4[1] - (m4[1]*dt)/200; return m4;`
a05	process	`m4[1] = m4[1]-0.0001*m4[1]*m6*dt; return m4;`
a06	process	`m4[1] = m4[1]-0.001*m4[1]*m5*dt; return m4;`
a07	process	`m3 = m3+0.0001*m4[1]*m6*dt; return m3;`
a08	process	`m3 = m3+0.001*m4[1]*m5*dt; return m3;`
a09	process	`m3/50`
a10	process	`m1[1] = m1[1]+0.001*m4[1]*m5*dt; return m1;`
a11	process	`m1[1] = m1[1]-(m1[1]/100)*dt; return m1;`
a12	process	`m1[1] = m1[1]-0.001*m1[1]*dt; return m1;`
a13	process	`m2[1] = m2[1]+0.001*m1[1]*dt; return m2;`
a16	process	1.0
a17	process	`m6 = m6 - 0.001*m1*m6*dt; return m6;`
a18	process	`m5 = m5 + 0.001*m1*m6*dt; return m5;`
a19	process	`m6/100`
a20	process	`m5/100`
a22	process	`m1[1]=m1[1]+0.0001*m4[1]*m6*dt;return m1;`
a23	process	`m1[1]=m1[1]-(m1[1]*(m1[0]/1000))*dt;return m1;`
a24	process	`m7=m7+(m1[1]*(m1[0]/1000))*dt;return m7;`
a25	process	`m7/100`
a26	process	`m2[1]=m2[1]-(m2[1]*(m2[0]/1000))*dt;return m2;`
a27	process	`m8=m8+(m2[1]*(m2[0]/1000))*dt;return m8;`
a28	process	`m8/100`
a29	process	`m2[1]=m2[1]-(m2[1]/100)*dt;return m1;`
a30	process	`m9+=0.001*m2[1];return m9;`

Table 8: The protein modification positions and modification type of *p53* enzymes [16].

enzyme	**modification type**	**modification positions**
CK1	Phosphorylation	S4,S6,S9,S18
ATM	Phosphorylation	S15
DNAPK	Phosphorylation	S15,S37
TFIIH	Phosphorylation	S33,S315 S378,S392
MAPK	Phosphorylation	T73,T83
PCAF	Acetylation	K320
PKC	Phosphorylation	S378
p300	Acetylation	K382
CK2	Phosphorylation	S392
JNK	Phosphorylation	S33

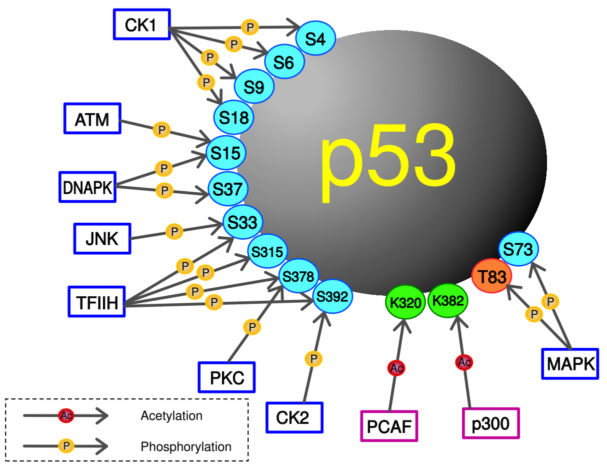


Table 9: Properties of entities for the *p53* modification model in Figure 6.

entity	**entity type**	**type**	**initial mark**
`m01`	`generic`	`object`	`ProtenSet::Initialize(m01,p53::Initialize())`
`m02`	`generic`	`object`	`CK1::Initialize(m02)`
`m03`	`generic`	`object`	`ATM::Initialize(m03)`
`m04`	`generic`	`object`	`DNAPK::Initialize(m04)`
`m05`	`generic`	`object`	`TFIIH::Initialize(m05)`
`m06`	`generic`	`object`	`MAPK::Initialize(m06)`
`m07`	`generic`	`object`	`PCAF::Initialize(m07)`
`m08`	`generic`	`object`	`PKC::Initialize(m08)`
`m09`	`generic`	`object`	`p300::Initialize(m09)`
`m10`	`generic`	`object`	`CK2::Initialize(m10)`
`m11`	`generic`	`object`	`JNK::Initialize(m11)`

conn-ector	**connector type**	**update function**
`a11`	`process`	`m01.Translate(); return m01;`
`a12`	`process`	`m01.Degradation();return m01;`
`a13`	`process`	`m01.Modify(m02); return m01;`
`a14`	`process`	`m01.Modify(m03); return m01;`
`a15`	`process`	`m01.Modify(m04); return m01;`
`a16`	`process`	`m01.Modify(m05); return m01;`
`a17`	`process`	`m01.Modify(m06); return m01;`
`a18`	`process`	`m01.Modify(m07); return m01;`
`a19`	`process`	`m01.Modify(m08); return m01;`
`a20`	`process`	`m01.Modify(m09); return m01;`
`a21`	`process`	`m01.Modify(m10); return m01;`
`a22`	`process`	`m01.Modify(m11); return m01;`

Table 10: Update functions of the *p53* modification model with HFPNe in Figure 6. All update functions are written in Pnuts language [44]. External java classes, `CK1`, `ATM`, `DNAPK`, `TFIIH`, `MAPK`, `PCAF`, `PKC`, `p300`, `CK2`, `JNK`, `ProteinSet`, `Enzyme`, and `Protein` are described in [43]. Connectors `a01`,·,`a10` have connector type `associate` and **update function** do not exist.

4 Discussion and Conclusion

In this paper, new versatile Petri net based architecture HFPNe is demonstrated for easy biological process modeling and simulation. HFPNe introduces the notions of generic entities and generic processes to HFPN. In HFPNe architecture, generic entities can hold various kinds of types including `object`. With the feature and inherited features of HFPN, complex biological processes can be effectively modeled. Four biological processes, alternative splicing, frameshifting, Huntington's disease, and multi-domain modification process of *p53* are employed in order to show the effectiveness of HFPNe.

From the theoretical point of view in the Petri net community, HFPNe can be one of the object oriented Petri nets (OPN). This approach was first given by Becker and Colom [5]. In the community, two advanced Petri nets based on the OPN architectures have been proposed, Objective Colored Petri Nets (OCP-Nets) [19] and Reference nets [18]. OCP-Nets is based on the OPN enhanced by the fusion place concept of Hierarchical Colored Petri Nets (HCPN). With this concept, OCP-Nets can dynamically change (shrink or grow) their structure of the net in simulation, the feature does not exist in HFPNe. Reference nets is also based on the OPN enhanced by the reference entities concept. With the concept, (i) an entity in a net (net1) can hold another net (net2), (ii) the net2 can move to other entities in the net1 by firing, (iii) the net2 can also change its state by firing. The differences between them are as follows: Reference nets can move the net but OCP-Nets cannot move the net in simulation processes. Besides OCP-Nets can create new connection rules in simulation processes. The HFPNe does not support these two OCP-Nets and Reference nets features. On the other hand, the feature of delay function of HFPNe does not exist in these object oriented Petri nets. However, in a cell, various kinds of time scale biological processes exist and to model these biological processes in one model, the notion of delay is necessary. With delay functions, HFPNe can handle these biological processes efficiently. In theoretical point of view, it may be easy to extend HFPNe to support OCP-Nets or Reference nets features because an implementation of HFPNe GON has been already extended to treat the feature to add/remove entities and processes while simulation.

As the reason that is already described in Section 3.4, we are trying to create a biological process library with HFPNe on GON. However, it is difficult to create all biological processes at once. Thus, as the first step, all processes in Kyoto Model [24] will be recompiled as a part of the library. Kyoto Model is a ventricular cell model by compiling classical electro-physiological findings. In the process of reconstruction, we will persist efforts to locate the best approach to systematically reconstruct existing other biological process models like Kyoto Model. Then we will apply the systematic approach to these other biological process models for enriching the library.

References

[1] Alla, H. and David, R., Continuous and hybrid Petri nets, *Journal of Circuits, Systems, and Computers*, 8:159–188, 1998.

[2] Amara, S.G., Jonas, V., Rosenfeld, M.G., Ong, E.S., and Evans, R.M., Alternative RNA processing in calcitonin gene expression generates mRNAs encoding different polypeptide products, *Nature*, 298:240–244, 1982.

[3] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., and Sherlock, G., Gene ontology: tool for the unification of biology. The Gene Ontology Consortium, *Nat. Genet.*, 25:25–29, 2000.

[4] Bates, G., Harper, P., and Jones, L., eds., Huntington's Disease, 3rd edn, *Oxford University Press*, 2002.

[5] Becker, U. and Moldt, D., Object-oriented concepts for Coloured Petri nets, *IEEE International Conference on Systems, Man and Cybernetics*, 3:279–286, 1993.

[6] Chen, M., Ona, V.O., Li, M., Ferrante, R.J., Fink, K.B., Zhu, S., Bian, J., Guo, L., Farrell, L.A., Hersch, S.M., Hobbs, W., Vonsattel, J.P., Cha, J.H., and Friedlander, R.M., Minocycline inhibits caspase-1 and caspase-3 expression and delays mortality in a transgenic mouse model of Huntington disease, *Nat. Med.*, 6:797–801, 2000.

[7] Coleman, T.P., Tran, Q., and Roesser, J.R., Binding of a candidate splice regulator to a calcitonin-specific splice enhancer regulates calcitonin/CGRP pre-mRNA splicing, *Biochim. Biophys. Acta.*, 1625:153–164, 2003.

[8] David, R. and Alla, H., Continuous Petri nets, In: 8th European Workshop on Application and Theory of Petri Nets, *Lecture Notes in Computer Science*, Springer-Verlag 340:275–294, 1987.

[9] Doi, A., Nagasaki, M., Fujita, S., Matsuno, H., and Miyano, S., Genomic Object Net:II. Modeling biopathways by hybrid functional Petri net with extension, *Applied Bioinformatics*, 2:185–188, 2004.

[10] Drath, R., Hybrid object nets: An object oriented concept for modeling complex hybrid systems, *In: 3rd International Conference on Automation of Mixed Processes: Hybrid Dynamical Systems (ADPM)*, Shaker Verlag, 437–442, 1998.

[11] Drath, R., Engmann, U., and Schwuchow, S., Hybrid aspects of modeling manufacturing systems using modified Petri nets, *In: 5th Workshop on Intelligent Manufacturing Systems*, 1999, `http://www.systemtechnik.tu-ilmenau.de/~drath/Download/Brasil98.ps.zip`

[12] Hackam, A.S., Singaraja, R., Wellington, C.L., Metzler, M., McCutcheon, K., Zhang, T., Kalchman, M., and Hayden, M.R., The influence of huntingtin protein size on nuclear localization and cellular toxicity, *J. Cell. Biol.*, 141:1097–1105, 1998.

[13] Hickey, M.A.and Chesselet, M.F., Apoptosis in Huntington's disease, *Prog Neuropsychopharmacol Biol Psychiatry*, 27:255–265, 2003.

[14] Jacks, T., Power, M.D., Masiarz, F.R., Luciw, P.A., Barr, P.J., and Varmus, H.E., Characterization of ribosomal frameshifting in HIV-1 *gag-pol* expression, *Nature*, 331:280–283 1988.

[15] Kim, Y.J., Yi, Y., Sapp, E., Wang, Y., Cuiffo, B., Kegel, K.B., Qin, Z.H., Aronin, N., and DiFiglia, M., Caspase 3-cleaved N-terminal fragments of wild-type and mutant huntingtin are present in normal and Huntington's disease brains, associate with membranes, and undergo calpain-dependent proteolysis, *Proc. Natl. Acad. Sci. USA*, 98:12784–12789, 2001.

[16] Kohn, K.W., Molecular interaction map of the mammalian cell cycle control and DNA repair systems, *Mol. Biol. Cell.*, 10:2703–2734, 1999.

[17] Kuipers, B.J. and Shults, B., Reasoning in logic about continuous systems, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference KR94*, Morgan Kaufmann 391–402, 1994.

[18] Kummer, O., Introduction to Petri Nets and Reference Nets, *Sozionik Aktuell*, 1:1–9, 2001.

[19] Maier, C. and Moldt, D., Object Coloured Petri Nets - A Formal Technique for Object Oriented Modeling, In: Concurrent Object-Oriented Programming and Petri Nets, Advances in Petri Nets. *Lecture Notes in Computer Science*, Springer-Verlag, 2001:406–427, 2001.

[20] Martin, J.B. and Gusella, J.F., Huntington's disease. Pathogenesis and management, *N. Engl. J. Med.*, 315:1267–1276, 1986.

[21] Matsuno, H., Tanaka, Y., Aoshima, H., Doi, A., Matsui, M., Miyano, S., Biopathways Representation and Simulation on Hybrid Functional Petri Net, *In Silico Biol.*, 3:389–404, 2003. `http://www.bioinfo.de/isb/toc_vol_03.html/`

[22] Matsuno, H., Doi, A., Nagasaki, M., and Miyano, S., Hybrid Petri net representation of gene regulatory network, *Pacific Symposium on Biocomputing 2000*, 5:341–352, 2000.

[23] Matsuno, H., Murakami, R., Yamane, R., Yamasaki, N., Fujita, S., Yoshimori, H., and Miyano, S., Boundary formation by notch signaling in *Drosophila* multicellular systems: Experimental observations and gene network modeling by Genomic Object Net, *Pacific Symposium on Biocomputing 2003*, 8:152–163, 2003.

[24] Matsuoka, S., Sarai, N., Kuratomi, S., Ono, K., and Noma, A., Role of individual ionic current systems in ventricular cells hypothesized by a model study, *The Japanese Journal of Physiology*, 53:105-123, 2003.

[25] McCarthy, J., Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I, *ACM Conference on Symbolic Manipulation*, 3:184–195, 1960.

[26] Mendes, P., GEPASI: A software package for modelling the dynamics, steady states and control of biochemical and other systems, *Comput. Appl. Biosci.*, 9:563–571, 1993.

[27] Nagasaki, M., Doi, A., Matsuno, H., and Miyano, S., Genomic Object Net:I. A platform for modeling and simulating biopathways, *Applied Bioinformatics*, 2:181–184, 2004.

[28] Nagasaki, M., Onami, S., Miyano, S., and Kitano, H., Bio-calculus: Its concept and molecular interaction, *Genome Informatics*, 10:133–143, 1999.

[29] Reddy, V.N., Liebman, M.N., and Mavrovouniotis, M.L., Qualitative analysis of biochemical reaction systems, *Comput. Biol. Med.*, 26:9–24, 1996.

[30] Reddy, V.N., Mavrovouniotis, M.L., Liebman, M.N., Petri Net Representations in Metabolic Pathways, In *Proceedings First International Conference on Intelligent Systems for Molecular Biology*, Volume 1., MIT Press, 328–336, 1993.

[31] Reisig, W. and Rozenberg, G., eds., Lecture on Petri nets I: Basic models, *Lecture Notes in Computer Science*, Springer-Verlag, 1491, 1998.

[32] Roy, S., Bayly, C.I., Gareau, Y., Houtzager, V.M., Kargman, S., Keen, S.L., Rowland, K., Seiden, I.M., Thornberry, N.A., and Nicholson, D.W., Maintenance of caspase-3 proenzyme dormancy by an intrinsic "safety catch" regulatory tripeptide, *Proc. Natl. Acad. Sci. USA*, 98:6132–6137, 2001.

[33] Roizin, L., Stellar, S., and Liu, J.C., Neuronal nuclear-cytoplasmic changes in Huntington's chorea: electron microscope investigations, *Raven Press*, 1979.

[34] Senut, M.C., Suhr, S.T., Kaspar, B., and Gage, F.H., Intraneuronal aggregate formation and cell death after viral expression of expanded polyglutamine tracts in the adult rat brain, *J. Neurosci.*, 20:219–229, 2000.

[35] Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T.S., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J.C., and Hutchison, C.A.3rd., E-CELL: software environment for whole-cell simulation, *Bioinformatics*, 15:72–84, 1999.

[36] Tran, Q., Coleman, T.P., and Roesser, J.R., Human transformer 2beta and SRp55 interact with a calcitonin-specific splice enhancer, *Biochim. Biophys. Acta.*, 1625:141–152, 2003.

[37] The Huntington's Disease Collaborative Research Group, A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes, *Cell*, 72:971–83, 1993.

[38] Webster, S.D., Cason, Z., Lemos, L.B., and Benghuzzi, H., Cytohistologic correlation in patients with clinical symptoms of postmenopausal bleeding, *Biomed. Sci. Instrum.*, 36:367–372, 2000.

[39] Wheeler, V.C., White, J.K., Gutekunst, C.A., Vrbanac, V., Weaver, M., Li, X.J., Li, S.H., Yi, H., Vonsattel, J.P., Gusella, J.F., Hersch, S., Auerbach, W., Joyner, A.L., and MacDonald, M.E., Long glutamine tracts cause nuclear localization of a novel form of huntingtin in medium spiny striatal neurons in HdhQ92 and HdhQ111 knock-in mice, *Hum. Mol. Genet.*, 9:503–513, 2000.

[40] Wellington, C.L., Ellerby, L.M., Gutekunst, C.A., Rogers, D., Warby, S., Graham, R.K., Loubser, O., van R., J., Singaraja, R., Yang, Y.Z., Gafni, J., Bredesen, D., Hersch, S.M., Leavitt, B.R., Roy, S., Nicholson, D.W., and Hayden, M.R., Caspase cleavage of mutant huntingtin precedes neurodegeneration in Huntington's disease, *J. Neurosci.*, 22:7862–7872, 2002.

[41] Wellington, C.L., Ellerby, L.M., Hackam, A.S., Margolis, R.L., Trifiro, M.A., Singaraja, R., McCutcheon, K., Salvesen, G.S., Propp, S.S., Bromm, M., Rowland, K.J., Zhang, T., Rasper, D., Roy, S., Thornberry, N., Pinsky, L., Kakizuka, A., Ross, C.A., Nicholson, D.W., Bredesen, D.E., and Hayden, M.R., Caspase cleavage of gene products associated with triplet expansion disorders generates truncated fragments containing the polyglutamine tract, *J. Biol. Chem.*, 273:9158–9167, 1998.

[42] GON: Genomic Object Net project, `http://GenomicObject.net/`

[43] HFPNe Generic Models, `http://genomicobject.net/public/BSB2004/code/`

[44] Pnuts, `http://www.pnuts.org/`