Process Algebra with Chaos Executing Policy for Unhealthy Systems

Guang Zheng[1], Lian Li[1], Wenbo Chen[1], Anping He[1] and Jinzhao Wu[2]
[1]School of Information Science and Engineering
Lanzhou University
Lanzhou University, Lanzhou 730000, China
{zhengguang,lil,chenwb}@lzu.edu.cn, hapetis@gmail.com
[2]School of Computer and Information Technology,
Beijing Jiaotong University,
Beijing 100044, China
himrwujz@yahoo.com.cn

***Abstract*—Healthy systems are important for us, but unhealthy systems exist everywhere. Healthy systems behave the manner they are designed, but not the unhealthy ones. It is a meaningful thing for us to turn an unhealthy system into a healthy one. We propose a process algebra with chaos executing policy to specify the behavior of unhealthy systems. For unhealthy systems, we also propose the equivalent relationships which are vital for the cure of them.**

***Key words:* process algebra, chaos, executing policy, equivalent relation.**

I. Introduction

Healthy systems are systems that function properly. They are our tools and important for us. We all have the experience that when a personal computer system is built up, it is *healthy* and runs smoothly and quickly. As time goes by, more and more programs are installed without discrimination, and the system slows down. This happens all the time until to the point unbearable for the user e.g. it takes too much time to respond or even collapse while loading, and then, the system becomes *unhealthy*.

If you investigate some key parameters of the system like CPU occupation, memory allocation, the busyness of ethernet and hard disk drive from the beginning of the system, you will find that the parameters of healthy systems are not very busy at the beginning of the system. When the system reaches its unhealthy point, check those parameters again and you will find at least one of the parameters of CPU, memory, ethernet and HDD are very busy.

Why does this happen? The reason is: computer system is under attack of virus and vicious codes which reproduce themselves and consume too much resource i.e. CPU, memory, ethernet and HDD. One way to settle this is to install security software, which can help to some extent. Apart from this, we think that the system can also do something by its own i.e. it should keep an eye on those key parameters when the system is built and healthy. When some program/process occupies too much system resources and takes a long time, the system can inform the user if it is wanted or expected. Continue if it is positive, and kill the processes if it is negative. By doing this, the administrator/user knows what the system is busy for so as to prolonging the system's healthy time.

This work was partially supported by the National Natural Science Foundation of China (No. 90612016 and 60473095), China Next Generation Internet Cernet2 (No. 432353), National Key Basic Research Project of China (2004CB318000 and 2007CB310800), and The National High Technology Research and Development Program of China (No. 2007AA01Z143).

The system can also build its blacklists of viruses, Trojan horses and a white-list of healthy programs, and then acts according to the lists. When some programs/processes are activated, the system can be strong enough to deny requests of dangerous programs and activate those healthy ones, which can be called as healthiness of the system. All these are done under the operator of *choice composition*.

Most actions in system runs have their time limitation and priorities. Thus we add parameter t and w into the action a and form action structure $a(t, w)$. t represents the time limitation of a and w represents the priority of a. We can get the information of a directly from its parameters in $a(t, w)$.

Example 1.1 *Suppose there is a web server providing three kinds of services:* file *service,* data *service and* internal *service. Accordingly, there are three kinds of requests. They are file requests denoted by actions $a_i(t_i, w_i)$ $(i \geq 1)$, data requests denoted by actions $b_j(t_j, w_j)$ $(j \geq 1)$, and internal requests denoted by actions $c_k(t_k, w_k)$ $(k \geq 1)$. What's more, there is also another kind of actions denoted by I_i $(i \geq 1)$ representing the virus and vicious codes consuming the resources and making damages to the server and clients.*

When a web server is infected with virus and vicious codes, its behaviors will be changed by the internal *actions I_i $(i \geq 1)$. The executing policy deployed onto this server will not function well. As time goes by, more and more virus intrude and be triggered to fire, the server's resources will be consumed up by the internal vicious actions I_i $(i \geq 1)$ which is shown by figure 1.*

In figure 1, we use $\Rrightarrow$ to denote the weightest *actions line, use $\longrightarrow$ to denote the weighter actions line, and use $\Longrightarrow$ to denote the* internal *actions line. As to the*

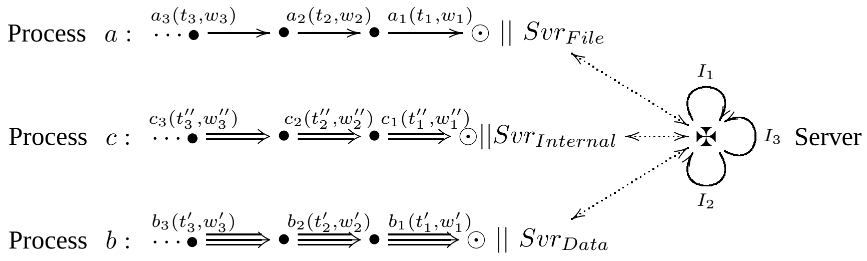

Fig. 1. Executing policy of *chaos*

configuration of resources, we used $\Longleftrightarrow$ *to denote the heavier resources for* data *service, and* $\leftrightarrow$ *to denote the resources allocated to* file *service. In this example, we change them to* $\dashleftarrow\dashrightarrow$ *for no policy can perform properly and no resources left to configure for useful services. The dashed arrows stand for some resources should be there but failed.*

Under the execution policy of *maximum process* in process algebras [1], [10], [2], [7], [12], the framework does not consider the result of actions and just lets actions be executed as quickly as possible. This policy is right for healthy systems, but not for unhealthy systems. Consider the operating system of computers used today. If it is an healthy one, this policy is helpful in the performance. Otherwise, computer system is infected with virus and vicious codes, this *maximum process* policy will slow down the system, destroy or even leak important data. Based on this, we know that one policy for execution in process algebra is not enough. In this paper, we propose *chaos* executing policy for unhealthy systems.

In example 1.1, the server encounters the problem of selecting the proper actions ($a_i(t_i, w_i)$, $b_j(t_j, w_j)$, $c_k(t_k, w_k)$, or $I_l(t_l, w_l)$) to execute during its run. There is a need for internal choice composition $\oplus$, and we propose it in our language together with external choice $\uplus$ which represent the external choice.

There is little change in the composition of *non-deterministic choice* in the development of process algebras [12], [10], [1] under the assumption of maximal progress. In PCCS (Probabilistic CCS[4], [5]), components under choice composition are equipped with probabilistic variables. These probabilistic variables make the decision among processes from non-deterministic to a more accurate form.

In PEPA [7], [6], [8], performance modeling is concerned with the capture and analysis of the dynamic behavior of computer and communication systems. Actions in PEPA are associated with exponential distributions which means the durations of the activities. *Race condition* is used in PEPA that governs the dynamic behavior of a model whenever more than one activity is enabled, which means that of all the activities attempting to proceed only the "fastest" succeeds. The policy in PEPA is an variation of *maximal progress*: the system selects the action with shortest duration for the coming execution.

In [11], Jou and Smolka describe a language in PCCS which is similar to SCCS, but with probabilistic choice replacing nondeterministic choice. Another extension of SCCS is Tofts' WSCCS [14] which uses weights to assign probabilities.

We focus on the *choice* composition, which is non-deterministic choice operator in CCS [12]. We classify the choice composition into two groups. One group is internal choice $\oplus$, through which the system can decide which action is to be executed by its execution policy. Another group is external choice $\uplus$ which is the choice exposed to the environment. The system cannot predict which one is selected for the coming execution. Apart from the choice composition, we also study the parallel composition because system can also execute actions with its decision of picking up actions through the components under parallel composition.

After defining the language of process algebra with execution policies, we show the operational semantics of the operators in this language. By analyzing the behavior of the operators in the intuitive way of operational semantics, we give out the axioms of them. As bisimulation relationships play an important role in the theory of process algebra, we define the strong and weak bisimulations.

This paper is organized as follows. Section 2 defines the language of process algebra with chaos executing policy. Section 3 defines the operational semantics and the axioms of the operators in the language. Section 4 introduces the equivalent relations including strong bisimulation, weak bisimulation. Section 5 concludes this paper.

II. LANGUAGE

In this section, we define the language. Process algebras have operators as prefix, parallel, termination, deadlock and recurrence. All these operators above have little relationship with intelligent behaviors, the operator which can represent the intelligence is *choice* which is divided into two in this paper: the external choice $\uplus$ and internal choice $\oplus$.

The grammar of our process algebra is:

$$\begin{aligned} P :: &= 0 \mid \checkmark \mid \delta \mid a(t,w).P \mid P \uplus P \mid P \oplus P \mid P;P \\ &\mid P||_S P \mid P \setminus L \mid P[f] \mid X \mid fix(X = E) \end{aligned}$$

0 is the constant named *empty process* indicating inactive process capable of doing nothing.

$\checkmark$ is the constant named *successful termination* indicating a process terminate successfully.

δ is the constant named *deadlock* indicating unsuccessful termination of a process capable of doing nothing.

$a(t,w).P$ is action prefixing. Only when the action $a(t,w)$ is executed, the term behaves as P. There are two parameters indicating information of action a: t indicates the time limitation of a and w indicates its priority.

$P \uplus P$ is the external choice composition for the interaction between system and environment. System just provides options for the environment. This operator is totally non-deterministic for system.

$P \oplus P$ is the internal choice which is decided by the system. Through this operator, the system can decide which action is allowed for execution according to its executing policy.

$P;Q$ is the sequential operator. Only when P successfully terminates, Q can get its turn for execution.

$P||_S Q$ is the parallel operator. It represents a system in which components P and Q work together to perform activities in the set S. The set S is called the *synchronising* or *cooperation* set. Both components proceed independently with any activities whose types do not belong to set S. However, activities with action types in set S are assumed to require the simultaneous involvement of both components. The resulting activity will have the same action type as the two contributing activities and a rate reflecting their rates. As to the rate, which is decided by action in the form of $a(t,w)$ or $a(p)$.

$P \setminus L$ is the hiding operator. It behaves as P except that any activities are in set L are *hidden*. They are not visible from outside and appear as internal action τ.

$P[f]$ is the relabelling operator. It behaves like P with its actions are relabelled by function f.

X is a bound process variable. It is used in the definition of the recursive expression.

$fix(X=E)$ is the recursive expression. We treat recursive expression as fixed-point to express the recursive process in the real world. For example, the expression $fix(X=a.X)$ represents a process that can perform infinite number of actions a.

Example 2.1 *Revisit example 1.1, we can specify the behavior of the web server formally by the language as:*

$$Server := Svr_{File}||Svr_{Internal}||Svr_{Data}||I_1||I_2||I_3$$

From the above formula, we can see that services (Svr_{File}, $Svr_{Internal}$, Svr_{Data}, I_1, I_2 and I_3) run synchronously under parallel composition. As to the priority of these services, we assume $I_1 \geq Svr_{Internal} \geq I_2 \geq Svr_{Data} \geq Svr_{File} \geq I_3$. They can be specified by parameter w, thus we have $w_{I1} \geq w''_i \geq w_{I2} \geq w'_i \geq w_i \geq w_{I3}$ with $i \geq 1$.

As to the service of file, we know that the web server transfers files from itself to the clients when it is requested. We assume this action is $Trans$, and this service can be specified as $Svr_{File} = Trans.Svr_{File}$. As there are countable files (n) contained in the server, the action $Trans$ can be refined to

$$\begin{aligned} Trans &= (File_1(t_1,w_1) \uplus File_2(t_2,w_2) \uplus \cdots \uplus \\ &\quad File_n(t_n,w_n)).Trans \end{aligned}$$

As to the service of data, we assume there is a database inside the web server. There are four kinds of actions associated with database behavior: $Select(t'_i,w'_i)$, $Delete(t'_i,w'_i)$, $Update(t'_i,w'_i)$ and $Insert(t'_i,w'_i)$ $(i \geq 1)$. Accordingly, the behavior of the Svr_{Data} can be specified as

$$\begin{aligned} Svr_{Data} &= (Select(t'_i,w'_i) \uplus Delete(t'_i,w'_i) \uplus \\ &\quad Update(t'_i,w'_i) \uplus Insert(t'_i,w'_i)).Svr_{Data} \end{aligned}$$

As to the internal service, based on Fig. 1., we know that $Svr_{Internal}$ have two kinds of actions ($Update_{File}(t''_i,w''_i)$ and $Update_{Data}(t''_i,w''_i)$ $(i \geq 1)$). Thus, the general action $c_i(t''_i,w''_i)$ can be specified as

$$\begin{aligned} Svr_{Internal} &= (Update_{File}(t''_i,w''_i) \uplus \\ &\quad Update_{Data}(t''_i,w''_i)).Svr_{Internal} \end{aligned}$$

As to vicious processes I_1, I_2 and I_3, they all recursive processes consuming system resources, reproducing themselves and making damages. All of them run continuously after being triggered.

Now, the web server's behaviors can be specified as:

$$\begin{aligned} Server &= (File_1(t_1,w_1) \uplus File_2(t_2,w_2) \uplus \cdots \uplus \\ &\quad File_n(t_n,w_n))|| \\ &\quad (Select(t'_i,w'_i) \uplus Delete(t'_i,w'_i) \uplus \\ &\quad Update(t'_i,w'_i) \uplus Insert(t'_i,w'_i))|| \\ &\quad (Update_{File}(t''_i,w''_i) \uplus Update_{Data}(t''_i,w''_i))|| \\ &\quad I_1||I_2||I_3. \end{aligned}$$

As virus can reproduce themselves, the parallel behaviors of the $Server$ will be changed into choice composition no matter how many processors inside it. Under this situation, the parallel composition will be replaced by internal choice composition:

$$\begin{aligned} Server &= (File_1(t_1,w_1) \uplus File_2(t_2,w_2) \uplus \cdots \uplus \\ &\quad File_n(t_n,w_n)) \oplus \\ &\quad (Select(t'_i,w'_i) \uplus Delete(t'_i,w'_i) \uplus \\ &\quad Update(t'_i,w'_i) \uplus Insert(t'_i,w'_i)) \oplus \\ &\quad (Update_{File}(t''_i,w''_i) \uplus Update_{Data}(t''_i,w''_i)) \oplus \\ &\quad I_1 \oplus I_2 \oplus I_3. \end{aligned}$$

Thus, the discrete services ($Svr_{Internal}$, Svr_{Data} and Svr_{File}) are blocked by the continues vicious processes (I_1, I_2 and I_3). The web server cannot perform its services smoothly as designed. Thus, the behaviors of unhealthy web server can be specified by the language defined above.

III. Operational Semantics and Axioms

Operational semantics gives out the intuitive evolution rules of the system the process algebra. We propose the operational semantics and the axioms of operators in this section.

A. Operational semantics

We show the operational semantics of process algebra in Table 1. All deduction rules of the language listed in this table are in form of $a(t,w)$

All deduction rules of the language listed in Table III-A are in form of $a(t,w)$.

Action prefix	$\frac{}{a(t,w).P \xrightarrow{a(t,w)} P}$	
External Choice	$\frac{P \xrightarrow{a(t,w)} P'}{P \uplus Q \xrightarrow{a(t,w)} P'}$	$\frac{Q \xrightarrow{a(t,w)} Q'}{P \uplus Q \xrightarrow{a(t,w)} Q'}$
Internal Choice	$\frac{P \xrightarrow{a(t,w)} P', Q \xrightarrow{a(t',w')} Q'}{P \uplus Q \xrightarrow{a(t,w)} P'}(w > w')$	$\frac{P \xrightarrow{a(t',w')} P', Q \xrightarrow{a(t,w)} Q'}{P \uplus Q \xrightarrow{a(t,w)} Q'}(w = w', t < t')$
Parallel	$\frac{P \xrightarrow{a(t,w)} P'}{P\|\|_S Q \xrightarrow{a(t,w)} P'\|\|_S Q}(a \notin S)$	$\frac{Q \xrightarrow{a(t,w)} Q'}{P\|\|_S Q \xrightarrow{a(t,w)} P\|\|_S Q'}(a \notin S)$
	$\frac{P \xrightarrow{a(t',w')} P', Q \xrightarrow{a(t'',w'')} Q'}{P\|\|_S Q \xrightarrow{\tau(t,w)} P'\|\|_S Q'}(a \in S)$	where $(t,w) = (max(t',t''), min(w',w''))$
Hiding	$\frac{P \xrightarrow{a(t,w)} P'}{P \setminus L \xrightarrow{a(t,w)} P' \setminus L}(a \notin L)$	$\frac{P \xrightarrow{a(t,w)} P'}{P \setminus L \xrightarrow{\tau} P' \setminus L}(a \in L)$
Relabelling	$\frac{P \xrightarrow{a(t,w)} P'}{P[f] \xrightarrow{f(a)(t,w)} P'[f]}$	
Recursion	$\frac{E\{fix(X=E)/X\} \xrightarrow{a(t,w)} E'}{fix(X=E) \xrightarrow{a(t,w)} E'}$	

TABLE I
THE OPERATIONAL SEMANTICS

Action prefix: Process term $a(t,w).P$ will evolve to P by executing action $a(t,w)$ within time t. w does not count in term $a(t,w).P$ since there have no other competitive actions involved.

Sequential composition: As to sequential composition $P;Q$, the execution of P or Q can be showed in the way of a serious of actio prefix.

Internal action: During system runs, there are internal actions denoted as τ. τ cannot be observed from outside.

Termination predicator: We present $\checkmark$ to indicate the successful termination of a process which can be distinguished from deadlock (δ).

$\checkmark$ indicates whether or not a process has a termination option. If so, then the result will be 0, and otherwise δ. For example, $\checkmark(P+0) = 0$ where as $\checkmark(P+Q) = \delta$.

External Choice: External choice $\uplus$ is an interface between system and the environment. System provides external choices for its environment. The external choice is completely non-deterministic for the system, but the system can also influence the environment by providing certain amount services.

For example, formula $\frac{P \xrightarrow{a(t,w)} P'}{P \uplus Q \xrightarrow{a(t,w)} P'}$ indicates that the environment picks up the action $a(t,w)$ to execute among actions available provided by the system.

Parallel composition: There are some changes in parallel composition.

For $a(t,w)$, the rule $\frac{P \xrightarrow{a(t,w)} P'}{P||_S Q \xrightarrow{a(t,w)} P'||_S Q}(a \notin S)$ indicates that process P evolves to P' by executing $a(t,w)$ with $a \notin S$. This means there is no communication between P and Q, and Q just remains unchanged. It is similar with rule $\frac{Q \xrightarrow{a(t,w)} Q'}{P||_S Q \xrightarrow{a(t,w)} P||_S Q'}(a \notin S)$. As for rule $\frac{P \xrightarrow{a(t',w')} P', Q \xrightarrow{a(t'',w'')} Q'}{P||_S Q \xrightarrow{\tau} P'||_S Q'}(a \in S)$ there is a communication action τ between process P and process Q (should be $\tau(t,w)$, but we omit parameter (t,w) here for internal action τ). System $P||_S Q$ evolves to $P'||_S Q'$ by executing an internal action τ. All can be observed is the time elapsed which is no more than t.

For synchronization, the executing time is $t = min(t', t'')$ for the system $P||_S Q$ need corporation of both process P and process Q to perform the internal action τ. As to the priority of τ, it is the lighter one: $w = min(w', w'')$. Then, we get the condition: $(t, w) = (min(t', t''), min(w', w''))$.

Hiding: For hiding operator $P \setminus L$, it behaves like P except that any activities of types within the L are hidden. Actions within the L are turned into internal action τ.

Relabelling: $P[f]$ is the relabeling operator. $P[f]$ behaves like P with its actions relabeled by function f.

Recursion: The meaning of recursion operator is given by equation such as $E\{fix(X = E)/X\}$. Process variable X is guarded in expression E, and the system run of $E\{fix(X = E)/X\}$ can be taken as a fixed-point. It means the the final step of one cycle run leads to the starting point of another cycle.

The deduction rule of constant can be expressed in action form $a(t, w)$ like

$$\frac{E\{fix(X = E)/X\} \xrightarrow{a(t,w)} E'}{fix(X = E) \xrightarrow{a(t,w)} E'}.$$

As to the rule

$$\frac{P \xrightarrow{a(t_1,w_1)} P', Q \xrightarrow{a(t_2,w_2)} Q'}{P||_S Q \xrightarrow{\tau} P'||_S Q'}(a \in S),$$

there is an internal communication τ between P and Q by action $a(t_i, w_i)$ for $i = 1, 2$. As to this kind of parallel composition of two processes, no execution policy can be showed for there have no choice. What we consider here is parallel composition with more than 2 processes to execution with the same synchronizing composition. The system will select two processes with highest two parameters w in action $a(t, w)$ i.e., process term

$$\frac{P_i \xrightarrow{a(t_i,w_i)} P_i', \ P_j \xrightarrow{a(t_j,w_j)} P_j'}{P_1||_S P_2||_S \cdots ||_S P_n \xrightarrow{\tau} P_1||_S \cdots P_i'||_S \cdots P_j'||_S \cdots ||_S P_n}$$

where $1 \le i < j \le n$ and $min(w_i, w_j) \ge (w_k \mid k \notin \{i, j\} \wedge 1 \le k \le n)$.

B. Axioms

Providing sound and complete axiomatizations for various equivalence relations has been one of the major research topics in the development of process theories [2], [12]. A complete axiomatization not only allows us to reason about process behaviors by syntactic manipulation, but also helps to understand the properties of the operators used to build complex processes from simpler components.

In table II, we use $+$ to stands for the set $\{\oplus, \uplus\}$ when there have no confusion, and use $\oplus$ and $\uplus$ to stands for the axioms for themselves.

There are two axioms about the distributive law among internal choice $\oplus$ and external choice $\uplus$. Each of them can distribute over the other by **D1** and **D2**.

In some situations there is no exact value on the parameter t and w of action $a(t, w)$, under this situation, we can degenerate $a(t, w)$ back to a. For internal choice $\oplus$, in term like $a.P \oplus_r a.Q$, there would be a randomize variable r that is not determined by a distribution. It just appears to be deterministic and is not possible to decide which one to be selected for the further execution. This can be showed as $\dfrac{a.P \xrightarrow{a} P, \ a.Q \xrightarrow{a} Q}{a.P \oplus_r a.Q \xrightarrow{a} P \oplus_r Q}$.

What's more, rules of internal choice under the policy of *chaos* are A1-A5 and A7-A10 which are the same as classic process algebras.

As to the parallel composition, under the executing policy of chaos, it also obey the rules as C1-C5 in the table II.

When a system is infected with a virus, and the virus is triggered. The system can still work, but not in the *healthy* way, and this kind of execution policy is useful in the checking and ascertain of *unhealthy* behavior of systems.

IV. Equivalent relations

Equivalence relations have been widely used in process algebras [2], [7], [6], [12], [9], [4], [8]. We can compare components and to replace a component with another which exhibits an equivalent behavior, but has a simpler representation.

A. Strong bisimulations

We introduce the strong bisimulation in this section. There is no difference between observable action and internal action under strong bisimulation. All actions can be observed together with their parameters.

Definition 4.1 *A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over processes with action in form of $a(t, w)$ is a strong bisimulation under policy of* chaos *if $(P, Q) \in S_R$ implies, for all $a(t, w) \in Act$,*

- *Whenever $P \xrightarrow{a(t,w)} P'$ then, for some Q', $Q \xrightarrow{a(t,w)} Q'$ and $(P', Q') \in S_R$;*
- *Whenever $Q \xrightarrow{a(t,w)} Q'$ then, for some P', $P \xrightarrow{a(t,w)} P'$ and $(P', Q') \in S_R$.*

We use $P \sim_R Q$ to denote processes P and Q in the relationship of strong bisimulation under chaos policy.

This equivalence relationship is useful in the checking and ascertaining the "*healthiness*" of a system. For example, healthy systems P' and Q' are in the relationship of strong bisimulation, if they both infected with a virus and evolved into P and Q, through the bisimulation under policy of chaos, it is easy to check if they are infected with the virus of the same kind for the same virus will behave the same and be activated by the same trigger. If we have the relationship of $P \sim_R Q$, then, we can tell that they are infected with the *same* virus. If there have a action r through which we can change P back to P' as $P \xrightarrow{r} P'$ and $Q \xrightarrow{r} Q'$. Through the execution of r, the systems P and Q are repaired to "*healthy*" state P' and Q' separately. This *cure* action r is also of strong bisimulation according the labeled transition according to the definition.

Based on the above analysis, we have the following proposition.

For simplicity, we use $+$ to stands for $\{\oplus, \uplus\}$ in general.			
$P+Q=Q+P$	**A1**	$a\cdot\tau\cdot P=a\cdot P$	**T1**
$P+(Q+R)=(P+Q)+R$	**A2**	$\tau\cdot P=\tau\cdot P+P$	**T2**
$P+P=P$	**A3**	$P\cdot(\tau\cdot Q+R)=P\cdot(\tau\cdot Q+R)+P\cdot Q$	**T3**
$(P+Q)\cdot R=P\cdot R+Q\cdot R$	**A4**		
$(P\cdot Q)\cdot R=P\cdot(Q\cdot R)$	**A5**	$P\|\|_S 0=P$	**C1**
$P\uplus\delta=P$	**A6**	$P\|\|_S\delta=P$	**C2**
$\delta\cdot P=\delta$	**A7**	$P\|\|_S Q=Q\|\|_S P$	**C3**
$P+0=P$	**A8**	$(P+Q)\|\|_S R=P\|\|_S R+Q\|\|_S R$	**C4**
$P\cdot 0=P$	**A9**	$R\|\|_S(P+Q)=R\|\|_S P+R\|\|_S Q$	**C5**
$0\cdot P=P$	**A10**		
		$p[f]=p$ if $p=\{0,\checkmark,\delta\}$	**L1**
$\delta_H(\tau)=\tau$	**H0**	$p[f]=f(p)$	**L2**
$\delta_H(a)=a$ if $a\notin H$	**H1**	$P[id]=P$	**L3**
$\delta_H(a)=\delta$ if $a\in H$	**H2**	$(P+Q)[f]=P[f]+Q[f]$	**L4**
$\delta_H(P+Q)=\delta_H(P)+\delta_H(Q)$	**H3**	$(P\cdot Q)[f]=(P[f])\cdot(Q[f])$	**L5**
$\delta_H(P\cdot Q)=\delta_H(P)\cdot\delta_H(Q)$	**H4**	$P[f][g]=P[f\circ g]$	**L6**
$\delta_H\delta_K(P)=\delta_{H\cup K}(P)$	**H5**	$(P\|\|_S Q)[f]=(P[f])\|\|_{S[f]}(Q[f])$	**L7**
$P\oplus(Q\uplus R)=(P\oplus Q)\uplus(P\oplus R)$	**D1**	$P\uplus(Q\oplus R)=(P\uplus Q)\oplus(P\uplus R)$	**D2**
$fix(X=E)=E\{fix(X=E)/X\}$			**R1**
If $F=E\{F/X\}$ then $F=fix(X=E)$, with X is guarded in E			**R2**
$fix(X=X+E)=fix(X=E)$			**R3**
$fix(X=\tau\cdot X+E)=fix(X=\tau\cdot E)$			**R4**
$fix(X=\tau\cdot(X+E)+F)=fix(X=\tau+X+E+F)$			**R5**

TABLE II
THE AXIOMS OF PROCESS ALGEBRA WITH CHAOS EXECUTING POLICY

Proposition 4.2 *If P and Q are healthy systems, and $P\sim_R Q$. P' and Q' are unhealthy systems, and $P'\sim_R Q'$. If $r\in Act$ and $P'\xrightarrow{r(t,w)}P$, then we have $Q'\xrightarrow{r(t,w)}Q$.*

Proof: *From $P\sim_R Q$, we know that if there is a vicious action $\bar{r}(t',w')$ that caused $P\xrightarrow{\bar{r}(t',w')}P'$, then we have $Q\xrightarrow{\bar{r}(t',w')}Q'$.*

What's more, if there is a cure action $r(t,w)$ which is the reverse of $\bar{r}(t',w')$, and $P'\xrightarrow{r(t,w)}P$. Then, based on the definition 4.2, we have the $Q'\xrightarrow{r(t,w)}Q$. $\square$

This proposition tells us in the most concrete level that we can *cure* unhealthy system if there is a way to *cure* another unhealthy system which is in strong bisimulation with this one.

B. Weak bisimulations

In the previous section, we introduced the notion of strong bisimulation, in which every α action of a process must be matched by an α action of the other - even every internal action τ. But in some cases, we should loosen the requirement: if the observer do not has the ability of observing the internal action τ, this yields a weaker notion of bisimulation called *weak bisimulation.* More precisely, we merely require that each internal action τ can just be omitted from the process.

Definition 4.3 *If $t\in Act^*$, then $\widehat{t}\in\mathcal{L}^*$ is the sequence gained by deleting all occurrences of τ from t.*

Note: $\widehat{\tau^n}=0$ (the empty sequence).

Definition 4.4 *If $t=a_1\cdots a_n\in Act^*$, then we write $E\xrightarrow{t}E'$ if $E\xrightarrow{a_1}\cdots\xrightarrow{a_n}E'$. We shall also write $E\xrightarrow{t}$ to mean that $E\xrightarrow{t}E'$ for some E'.*

We now define a new labeled transition system

$$(\mathcal{E},\mathcal{L}^*,\{\overset{s}{\Rightarrow}\ \mid s\in\mathcal{L}^*\})$$

over process expressions, in which the transition relations $\overset{s}{\Rightarrow}$ are defined as follows. For convenience we actually define $\overset{s}{\Rightarrow}$ for all $t\in Act^*$, i.e. for sequences which may contain internal action τ:

Definition 4.5 *If $t=a_1\cdots a_n\in Act^*$, then $E\overset{t}{\Rightarrow}E'$ if*

$$E(\xrightarrow{\tau})^*(\xrightarrow{a_1})(\xrightarrow{\tau})^*\cdots(\xrightarrow{\tau})^*(\xrightarrow{a_n})(\xrightarrow{\tau})^*E'$$

We also write $E\overset{t}{\Rightarrow}$ to mean that $E\overset{t}{\Rightarrow}E'$ for some E'.

Thus $E\overset{ab}{\Rightarrow}E'$ means that $E(\xrightarrow{\tau^p})(\xrightarrow{a})(\xrightarrow{\tau^q})(\xrightarrow{b})(\xrightarrow{\tau^r})E'$ for some $p,q,r\geq 0$. Note also that $E\overset{0}{\Rightarrow}E'$ iff $E\overset{\tau^n}{\Rightarrow}E'$ for some $n \geq 0$.

Definition 4.6 *If $t \in Act^*$, then E' is a t-descendant of E iff $E \overset{\widehat{t}}{\Rightarrow} E'$.*

Note that if $t \in \mathcal{L}^*$ this just means $E \overset{t}{\Rightarrow} E'$, since $t = \widehat{t}$ in this case. But notice that E' is a τ-descendent of E iff $E \overset{\tau^*}{\Rightarrow} E'$ for some $n \geq 0$, and this includes the case $n = 0$ in which $E' \equiv E$.

Now, we may summarize the difference between the three relations $\xrightarrow{t}$, $\overset{t}{\Rightarrow}$, and $\overset{\widehat{t}}{\Rightarrow}$ for $t \in Act^*$. Each specifies an action-sequence with exactly the same observable content as t, but the possibilities for intervening τ actions are different:

$\xrightarrow{t}$ specifies exactly the τ actions occurring in t;
$\overset{t}{\Rightarrow}$ specifies at least the τ actions occurring in t;
$\overset{\widehat{t}}{\Rightarrow}$ specifies nothing about τ actions.

Thus $P \xrightarrow{t} P'$ implies $P \overset{t}{\Rightarrow} P'$, and $P \overset{t}{\Rightarrow} P'$ implies $P \overset{\widehat{t}}{\Rightarrow} P'$.

So, keep in mind what we said about matching a τ by zero or more τ actions, we want a notion of equivalence – with the following definitions under different execution policies.

Definition 4.7 *A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over processes with action in form of $a(t,w)$ is a weak bisimulation under policy of* chaos, *if $(P,Q) \in S$ implies, for all $a(t,w) \in Act$,*

- *Whenever $P \xrightarrow{a(t,w)} P'$ then, for some Q', $Q \overset{\widehat{a(t,w)}}{\Longrightarrow} Q'$ and $(P',Q') \in S$;*
- *Whenever $Q \xrightarrow{a(t,w)} Q'$ then, for some P', $P \overset{\widehat{a(t,w)}}{\Longrightarrow} P'$ and $(P',Q') \in S$.*

where the $\overset{\widehat{a(t,w)}}{\Longrightarrow}$ means $\overset{\widehat{\tau_R^n}}{\Longrightarrow}\xrightarrow{a(t,w)}\overset{\widehat{\tau_R^n}}{\Longrightarrow}$, and $\widehat{\tau_R^n} = 0$ (the empty sequence).

We write this formally as follows using $\approx_R$ to stand for *weak bisimulation* (also as *observation equivalence*) under policy *chaos*.

Definition 4.8 *P and Q are observation equivalent or weakly bisimilar under chaos policy, written $P \approx_R Q$, if $(P,Q) \in \mathcal{S}$. That is,*

$$\approx_R = \cup\{\mathcal{S} \mid \mathcal{S} \text{ is a bisimulation under policy of } \textit{chaos}\}$$

Based on the above analysis, we have the following proposition.

Proposition 4.9 *If P and Q are healthy systems, and $P \approx_R Q$. P' and Q' are unhealthy systems, and $P' \approx_R Q'$. If $r(t,w) \in Act$ and $P' \xrightarrow{r(t,w)} P$, then we have $Q' \xrightarrow{\hat{r}(t,w)} Q$. Vice versa, if $r(t,w) \in Act$ and $Q' \xrightarrow{r(t,w)} Q$, then we have $P' \xrightarrow{\hat{r}(t,w)} P$.*

Proof: *From $P \approx_R Q$, we know that if there is a vicious action $\bar{r}(t',w')$ that caused $P \xrightarrow{\bar{r}(t',w')} P'$, then we have $Q \xrightarrow{\hat{\bar{r}}(t',w')} Q'$.*

What's more, if there is a cure action $r(t,w)$ which is the reverse of $\bar{r}(t',w')$, and $P' \xrightarrow{\bar{r}(t,w)} P$. Then, based on the definition 4.4, we have the $Q' \xrightarrow{\hat{r}(t,w)} Q$. Vice versa. □

Compared with proposition 4.3, one may wonder if this proposition is strong enough to handle the *cure*. The difference between proposition 4.3 and 4.5 is internal action τ. We believe that internal actions τ contain information not necessary and abstracted away from observation.

C. Recursiveness

It is very common for system to behave recursively. In this section, we will study the equivalent relationship about recursive behavior in our language.

Definition 4.10 *For strong bisimulation relationships under execution policy as $\sim_R$, let E and F contain variables $\widetilde{X}$ at most. Then $E \sim F$ if, for all indexed sets $\widetilde{P}$ of processes, $E\{\widetilde{P}/\widetilde{X}\} \sim F\{\widetilde{P}/\widetilde{X}\}$.*

We shall also use $\widetilde{E} \sim \widetilde{F}$ to mean component-wise congruence between $\widetilde{E}$ and $\widetilde{F}$.

One may wonder why we define the equivalence relationship about strong bisimulation on recursive operator under chaos policy. The reason is that all the operators in our language under chaos policy obey the same group of axioms as [1], [10], [12].

Proposition 4.11 *If $\widetilde{A} \overset{def}{=} \widetilde{P}$, then $\widetilde{A} \sim \widetilde{P}$*

Proof By the operational semantics of *Congruence* we see that for each i, A_i and P_i have exactly the same derivatives, and the result follows directly. □

Now we are ready to show that $\sim_R$ is preserved by recursive definition.

Proposition 4.12 *Let $\widetilde{E}$ and $\widetilde{F}$ contain variables $\widetilde{X}$ at most. Let $\widetilde{A} \overset{def}{=} \widetilde{E}\{\widetilde{A}/\widetilde{X}\}$, $\widetilde{B} \overset{def}{=} \widetilde{F}\{\widetilde{B}/\widetilde{X}\}$ and $\widetilde{E} \sim \widetilde{F}$. Then $\widetilde{A} \sim \widetilde{B}$.*

Proof We shall deal only with the case of single recursion equations, thus replacing $\widetilde{E}, \widetilde{F}, \widetilde{A}, \widetilde{B}$ by E, F, A, B. So assume

$$E \sim F,\ A \overset{def}{=} E\{A/X\}, \text{ and } B \overset{def}{=} F\{B/X\}$$

It will be enough to show that $\mathcal{S}$ is a strong bisimulation up to $\sim$, where

$$\mathcal{S} = \{(G\{A/X\}, G\{B/X\}) \mid G \text{ contains at most the variable } X\}$$

For then, by taking $G \equiv X$, it follows that $A \sim B$.

To show this, it will be enough to prove that
If $G\{A/X\} \xrightarrow{a} P'$ then, for some Q' and Q'',
$G\{B/X\} \xrightarrow{a} Q'' \sim Q'$, with $(P',Q') \in \mathcal{S}$

We shall prove the above formula by transition induction, on the depth of the inference by which the action $G\{A/X\} \xrightarrow{a} P'$ is inferred. We argue by cases on the form of G:

Case 1 $G \equiv X$.

Then $G\{A/X\} \equiv A$, so $A \xrightarrow{a} P'$, hence also $E\{A/X\} \xrightarrow{a} P'$ by a shorter inference. Hence, by induction

$$E\{B/X\} \xrightarrow{a} Q'' \sim Q', \text{ with } (P',Q') \in \mathcal{S}$$

But $E \sim F$, so $F\{B/X\} \xrightarrow{a} Q''' \sim Q'$, and since $B \overset{def}{=} F\{B/X\}$

$$G\{B/X\} \equiv B \xrightarrow{a} Q''' \sim Q', \text{ with } (P',Q') \in \mathcal{S}$$

are required.

Case 2 $G \equiv a \cdot G'$.

Then $G\{A/X\} \equiv a \cdot G'\{A/X\}$, so $P' \equiv G'\{A/X\}$; also

$$G\{B/X\} \equiv a \cdot G'\{B/X\} \xrightarrow{a} G'\{B/X\}$$

and clearly $(G'\{A/X\}, G'\{B/X\} \in \mathcal{S})$ as required.

Case 3 $G \equiv G_1 \oplus G_2$ and $G \equiv G_1 \uplus G_2$.

This is simpler than the following case, and we omit the proof.

Case 4 $G \equiv G_1||_S G_2$.

Then $G\{A/X\} \equiv G_1\{A/X\}||_S G_2\{A/X\}$. There are three cases for the action $G\{A/X\} \xrightarrow{a} P'$, according to whether it arises from one or other component alone or from a communication. We shall treat only the case in which $a \in S$, and

$$G_1\{A/X\} \xrightarrow{a} P_1',\ G_2\{A/X\} \xrightarrow{a} P_2'$$

where $P' \equiv P_1'||_S P_2'$. Now each component action has a shorter inference, so by induction

$$G_1\{A/X\} \xrightarrow{a} Q_1'' \sim Q_1', \text{ with } (P_1', Q_1') \in \mathcal{S}$$

$$G_2\{A/X\} \xrightarrow{a} Q_2'' \sim Q_2', \text{ with } (P_2', Q_2') \in \mathcal{S}$$

Hence, setting $Q' \equiv Q_1'||_S Q_2'$ and $Q'' \equiv Q_1''||_S Q_2''$

$$G\{B/X\} \equiv G_1\{B/X\}||_S G_2\{B/X\} \xrightarrow{\tau} Q'' \sim Q'$$

It remains to show that $(P', Q') \in \mathcal{S}$. But $(P_i', Q_i') \in \mathcal{S}(i = 1, 2)$ so for some H_i, $P_i' \equiv H_i\{A/X\}$ and $Q_i' \equiv H_i\{B/X\}(i = 1, 2)$; thus if we set $H \equiv H_1||_S H_2$ we have

$$(P', Q') \equiv (H\{A/X\}, H\{B/X\}) \in \mathcal{S}$$

Case 5 $G \equiv G_1 \setminus L$, or $G_1[R]$

These cases are simpler than **Case 4**, and we omit the proof.

Case 6 $G \equiv C$, a process Constant with associated definition $C \overset{def}{=} R$. then, since X does not occur, $G\{A/X\}$ and $G\{B/X\}$ are identical with C and hence both have a-derivative P'; clearly $(P', P') \equiv (P'\{A/X\}, P'\{B/X\}) \in \mathcal{S}$ □

V. CONCLUSIONS

Healthy systems can provide us with smooth services as expected, but unhealthy systems cannot provide services properly and smoothly. However, unhealthy systems widely exist in the real world. How to identify and cure the unhealthy system is an interesting and meaningful question. In this paper, we proposed a process algebra with *chaos* executing policy that can identify and cure the unhealthy system in the algebraic way.

We use the structure $a(t, w)$ to stand for actions of systems with parameters t for time limitation and w for priority. This structure is intuitive and easy to put into use.

Process algebras have been studied over 20 years, and researchers have extended them in some aspects, e.g., PEPA [8], SPAs [7], [9], probabilistic PAs [11], [13], [15] and so on. Among all of them, there is only one common executing policy: *maximum progress*. What's more, this policy is not enough to specify all the situations esp. for unhealthy complex systems. So we propose a process algebra with chaos executing policy for them.

Action structure $a(t, w)$ does not mean much in the sequential systems. It is useful in the smart system with multi-tasks, system of this kind can make its own decisions according to different situations.

We defined the language of chaos execution policy in this paper. There are some changes in this languages: we replace the traditional choice composition $+$ with internal choice $\oplus$ and external choice $\uplus$. After the definition of the language, we gave out the operational semantics of our language. Then, according to the intuitive meaning of the operators, we listed the axioms of the operators.

As equivalent relations are very important in process algebras, we defined strong and weak bisimulation.

We had a interesting result: we use P and Q represent the states of unhealthy systems and bisimulate (strong or weak), if there is an action $a(t, w)$ which can cure P (or Q) to healthy state, then, it can also cure Q (or P).

REFERENCES

[1] J. A. Bergstra and J.W. Klop, Algebra of Communitating Processes with Abstraction, *TCS 37,1*, pp. 77-121, 1985.

[2] M. Bravetti and M. Bernardo. *Compositional asymmetric cooperations for process algebras with probabilities, priorities, and time*. In *Proc. of the 1st International Workshop on Models for Time Critical Systems*, volume 39 (3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2000.

[3] E. Brinksma, H. Hermanns. Process Algebra and Markov Chains *J.-P. Katoen (Eds.) FMPA2000*, LNCS 2090, pp. 183C231, 2001. Springer-Verlag Berlin Heidelberg 2001.

[4] P.R. DArgenio, B. Jeannet, H.E. Jensen, and K.G. Larsen, Reachability analysis of probabilistic systems by successive refinements, *PAPM-PROBMIV 2001*, Aachen, Germany (L. de Alfaro and S. Gilmore, eds.), LNCS, vol. 2165, Springer-Verlag, 2001, pp. 29C56.

[5] A. Giacalone, C.-C. Jou and S.A. Smolka. *Algebraic reasoning for probabilistic concurrent systems*. In M. Broy and C.B. Jones, eds, Proc. of the Working Conf. on Programming Concepts and Methods, pages 443–458. North-Holland, 1990.

[6] Holger Hermanns, Ulrich Herzog and Joost-Pieter Katoen, *Process algebra for performance evaluation*, Theoretical Computer Science, 274, pp 43-87, 2002.

[7] H. Hermanns and M. Rettelbach. *Syntax, Semantics, Equivalences, and Axioms for MTIPP*. In U. Herzog and M. Rettelbach, editors, Proc. of the 2nd Workshop on Process Algebras and Performance Modelling, Erlangen-Regensberg, July 1994. IMMD, Universitat Erlangen-Nurnberg.

[8] Jane Hillston, *A Compositional Approach to Performance Modelling*, 1996, Cambridge University Press.

[9] Jane Hillston, *Process algebras for Quantitative Analysis*, 2005, Proceedings of the 20^{th} Annual Symposium on Logic in Computer Science (LICS'05).

[10] C.A.R. Hoare. *Communicating Sequential Process*, Prentice-Hall, 1985.

[11] C-C. Jou and S.A. Smolka. *Equivalences, Congruences and Complete Axiomatizations of Probabilistic Processes*. In J.C.M. Baeten and J.W. Klop, editors, CONCUR'90, volume 458 of LNCS, pages 367-383. Springer-Verlag, August 1990.

[12] Robin Milner, Communication and Concurrency, *Prentice Hall*, 1989.

[13] M. Mislove, J. Ouaknine and J. Worrell. *Axioms for Probability and Nondeterminism*. In Proc. EXPRESS'03, ENTCS 91(3), 2003.

[14] C. Tofts. *Describing Social Insect Behaviour Using Process Algebra.* Transactions of the Society for Computer Simulation, 9(4):227-283, December 1992.

[15] Daniele Varacca, Glynn Winskel. *Distributing probability over non-determinism* Mathematical Structures in Computer Science archive Volume 16 , Issue 1 (February 2006) Pages: 87 - 113, 2006 ISSN:0960-1295, Cambridge University Press.