Process Algebra with Timed-Priority Executing Policy

Zhongxian Li[1], Miao Jiang[2], Gao Chen[3], Qinglin Zha[4], and Guang Zheng[1,2,*]
[1] School of Information Science-Engineering,
Lanzhou University, Lanzhou, 730000, China
[2] Institute of Basic Research in Clinical Medicine,
China Academy of Chinese Medical Sciences,
Dongzhimen, Beijing 100700, China
[3] Life Science School, Hubei University,
Wuhan 430062, China, Email: gc_npwr@yahoo.cn
[4] Data Analysis Center, Jiangxi University of Traditional Chinese Medicine
Nanchang 330006, Jiangxi Province, China
[*] Corresponding author: forzhengguang@163.con

***Abstract*—Equipped with powerful machines and complex softwares, web servers providing services are widely used all over the Internet. But, how to specify their behaviors are interesting and meaningful. However, process algebras nowadays cannot specify the behaviors of web servers with time limitations and different groups of clients who are belong to different groups/priorities. The behaviors of web servers can be expressed by actions equipped with parameters of time t and priority w. We present a process algebra with timed-priority executing policy which can specify the behaviors of web servers.**

***Key words:* process algebra, executing policy, web server, time limitation, priority.**

I. Introduction

Internet has infiltrated into common life world wide. Equipped with powerful machines and complex softwares, web servers form the core of the Internet. They provide services for the whole Internet. Their behaviors are composed by all kinds of actions they can perform. Thus, their behavior are meaningful and interesting for us to study. We can show this by the following example.

Example 1.1 *There is a web server providing three kinds of services including* file *service,* data *service and* internal *service at the same time. 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* $c_k(t_k, w_k)$ $(k \geq 1)$*. Fig. 1 demonstrates the workflow of the web server.*

During the system runs, web servers may face the situation that three kinds services are requested at the same time. They meet different needs, and intuitively, we have reason to assign the priority parameters $w_i'' > w_i$ *and* $w_i'' > w_i'$ *(*w_i *in* $a_i(t_i, w_i)$*,* w_i' *in* $b_i(t_i', w_i')$ *and* w_i'' *in* $c_i(t_i'', w_i'')$*) with* $w_i = w_i'$*. Actions* $c_i(t_i'', w_i'')$ *in process* c *are internal requests which update web server's information, and they are critical for the correctness of services.* $a_i(t_i, w_i)$ *in process* a *are external requests for files and they do not consume much server resources.* $b_i(t_i', w_i')$ *in process* b *are external requests for data which are stored in the web server in the form of large databases (e.g. DB2, MS SQL Server, Oracle etc.). They consume many server resources.*

In process algebras [22], [18], [3], [14], [16], [5], there is a default executing policy named "*Maximal Progress*". From the above example, we can see clearly that this default executing policy is too abstract for systems to schedule their services/actions. A more doable policy which can specify the behaviors of web servers is required, based on this, we propose a process algebra with *timed-priority* executing policy to meet the need which can be taken as a refinement of the "*Maximal Progress*" executing policy for web servers.

According to the timed-priority policy, we classify the choice composition $+$ into two groups: one is "internal choice composition" $\oplus$, through which system can decide which action is to be executed (viz. the choice for server encountered with processes a, b and c at the same time); the other is "external choice composition" $\uplus$, which is the options exposed for environment, and the system has no idea which one to be executed next (viz. the choice for clients to request either $file$ or $data$ from the server). We also focus on the parallel composition with which systems can execute actions on its decision i.e. pick up actions through components under parallel composition.

After defining the language for process algebra with timed-priority execution policy, we show the operational semantics for its operators. By analyzing the operators' behaviors, we give out the axioms operators in our language. As there are some changes in both choice and parallel compositions, we construct intuitive algorithms for them under timed-priority executing policy. Bisimulation relationships play an important role in the study of process algebras. According to the operators in process algebra with timed-priority, we define strong bisimulation, weak bisimulation and the expansion law as equivalent relationships in this language.

This paper is organized as follows. Section 2 defines the language for process algebra with timed-priority executing policy. Section 3 defines the operational semantics. Section

$$\begin{tikzcd}
Process\ \ a\ : & \cdots\bullet \arrow[r, "a_3(t_3{,}w_3)"] & \bullet \arrow[r, "a_2(t_2{,}w_2)"] & \bullet \arrow[r, "a_1(t_1{,}w_1)"] & \odot \parallel Svr_{File} \arrow[dr, leftrightarrow] & \\
Process\ \ c\ : & \cdots\bullet \arrow[r, Rightarrow, "c_3(t''_3{,}w''_3)"] & \bullet \arrow[r, Rightarrow, "c_2(t''_2{,}w''_2)"] & \bullet \arrow[r, Rightarrow, "c_1(t''_1{,}w''_1)"] & \odot || Svr_{Internal} \arrow[r, leftrightarrow] & Server \\
Process\ \ b\ : & \cdots\bullet \arrow[r, Rrightarrow, "b_3(t'_3{,}w'_3)"] & \bullet \arrow[r, Rrightarrow, "b_2(t'_2{,}w'_2)"] & \bullet \arrow[r, Rrightarrow, "b_1(t'_1{,}w'_1)"] & \odot \parallel Svr_{Data} \arrow[ur, leftrightarrow] &
\end{tikzcd}$$

Fig. 1. Web server provids three kinds of services

4 lists out axioms for all operators. Section 5 proposes algorithms for internal choice composition and parallel composition under timed-priority policy. Section 6 proposes the equational relationships including strong bisimulation, weak bisimulation and the expansion law. Section 7 concludes this paper.

II. LANGUAGE

We introduce the language of process algebra with *timed-priority* executing policy in this section. The grammar is:

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

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

✓ 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 $a(t,w)$ is executed, system can behave process P. There are two parameters in $a(t,w)$: t indicates the time left before action a had to be finished, and w indicates the priority of the action a in the system runs.

$P \uplus P$ is the *external* choice composition for interaction between system and environment. It is totally non-deterministic for system.

$P \oplus P$ is the *internal* choice composition which is decided by the system itself.

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

$P||_S Q$ is the parallel composition. It represents a situation when P and Q work together to perform activities in the set S. Or executes other actions independently.

$P \setminus L$ is the hiding operator. It behaves as P except that any activities of in set L are *hidden*.

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

X is a bound process variable. It is used in recursive expressions.

$fix(X = E)$ is the recursive expression, we treat recursive expression as fixed-point to express the recursive process in the real world.

Example 2.1 *Revisit example 1.1, we can specify the behaviors of the server as*

$$Server := Svr_{File}||Svr_{Internal}||Svr_{Data}$$

From the above formula, we can see that three services (Svr_{File}, $Svr_{Internal}$ and Svr_{Data}) run synchronously under parallel composition.

For file service, the web server transfers files to its clients on requests. We use $Trans$ to denote it and it can be specified as $Svr_{File} = Trans.Svr_{File}$. As there are n countable files in the server, $Trans$ can be refined to

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

For data service, it based on databases in $Server$. There are four kinds of actions associated with it: $Select(t'_i, w'_i)$, $Delete(t'_i, w'_i)$, $Update(t'_i, w'_i)$ and $Insert(t'_i, w'_i)$ $(i \geq 1)$. Accordingly, the behaviors of the Svr_{Data} can be specified as

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

For internal service based on Fig. 1., $Svr_{Internal}$ has two kinds of actions: ($Update_{File}(t''_i, w''_i)$ and $Update_{Data}(t''_i, w''_i)$ $(i \geq 1)$) representing general action $c_i(t''_i, w''_i)$ which can be specified as $Svr_{Internal} = (Update_{File}(t''_i, w''_i) \uplus Update_{Data}(t''_i, w''_i)).Svr_{Internal}$.

Thus, the behaviors of web server $Server$ can be specified as

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

Now, the behaviors of web server are specified by the language defined above.

A. Operational semantics and axioms

Based on the informal expressions of the language in the previous section, we give out its formal operational semantics now. All deduction rules of the language listed in this table are in form of $a(t,w)$.

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

TABLE I
THE OPERATIONAL SEMANTICS OF PROCESS ALGEBRA WITH TIMED-PRIORITY EXECUTING POLICY

Action prefix: $a(t,w).P$ can evolve to P by executing action $a(t,w)$ within time limitation t. Parameter w representing priority does not take effect in term $a(t,w).P$ for there is no other competitive actions involved.

Sequential composition: $P;Q$. Q can get its turn for execution only when P terminates successfully.

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

External Choice: $\uplus$ is an interface between system and environment. System provides external choices for its environment, and the environment can accomplish its purposes by interacting with them. $\uplus$ is completely non-deterministic for system.

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

Parallel composition: Rule $\dfrac{P \xrightarrow{a(t,w)} P'}{P||_S Q \xrightarrow{a(t,w)} P'||_S Q}(a \notin S)$ indicates that P evolves to P' by executing $a(t,w)$ with $a \notin S$ while Q remains unchanged. It is similar with $\dfrac{Q \xrightarrow{a(t,w)} Q'}{P||_S Q \xrightarrow{a(t,w)} P||_S Q'}(a \notin S)$.

As to rule $\dfrac{P \xrightarrow{a(t',w')} P', Q \xrightarrow{a(t'',w'')} Q'}{P||_S Q \xrightarrow{\tau} P'||_S Q'}(a \in S)$, there is an internal action τ (can also be $\tau(t,w)$). System $P||_S Q$ evolves to $P'||_S Q'$ by executing an internal action τ.

System waits until to the point where neither P or Q can wait any longer. That point is $t = min(t',t'')$. $P||_S Q$ needs corporation of both P and Q to perform internal action τ. The priority of the action τ is the lighter one which is $w = min(w',w'')$, then we get the condition $(t,w) = (min(t',t''), min(w',w''))$.

Hiding: $P \setminus L$ behaves like P except that any activities of types within set L are hidden.

Relabeling: $P[f]$ 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 $E\{fix(X=E)/X\}$, provided that process variable X is guarded in the expression E, and the system run of $E\{fix(X=E)/X\}$ can be taken as a fixed-point, which means the the final step of one cycle run of the system leads to the starting point of the system run.

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.

In table II, we use $+$ to stands for $\oplus$ and $\uplus$ when there is no confusion.

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

III. Algorithms

Under timed-priority executing policy, there are some changes in *internal choice* $\oplus$ and *parallel composition* $||_S$, and we propose algorithms for them in this section.

A. Algorithm for internal choice composition

If there is only one action $a_i(t_i, w_i)$ available, the system executes it. If there are more than one actions ready for execution, the system can schedule their order by certain algorithm under policy of timed-priority.

Example 4.1: *Revisit example 1.1, there are three lines of sequential actions representing clients with different priorities. Process c owns the biggest priority, the system will halt a and b when c is ready for execution.*

Based on the analysis, we design the following algorithm for scheduling:

```
1)  while $\exists a_i(t_i, w_i)$
2)    $Action\_Number = count(a_i(t_i, w_i))$   //the number of actions
3)    if $Action\_Number = 1$
4)       do $a_i(t_i, w_i)$
5)       else $Priority\_Action\_Number = count(max(w_i))$   //the number of actions belong to the maximal priority
6)         if $Priority\_Action\_Number = 1$
7)            do $a_i(t_i, w_i)$ with $max(w_i)$
8)         else do $a_i(t_i, w_i)$ with $max(w_i) \wedge min(t_i)$
9)         endif
10)   endif
11) endwhile
```

B. Algorithm for parallel composition

Under *timed-priority* executing policy, system can also schedule the executing for parallel composition.

Example 4.2: *Revisit example 1.1, we know that a, b and c are parallel processes. In the system runs, processes a, b and c apply CPU resources in the form of $a||b||c||CPU$. According to timed-priority policy, we know that c gets the CPU resource and executes, $a \cup b$ gets its turn after c terminates.*

Based on the analysis, we have the following algorithm:

```
1)  while $\exists a_i(t_i, w_i)||CPU$
2)    $Action\_Number = count(a_i(t_i, w_i))$   //the number of actions
3)    if $Action\_Number = 1$
4)       do $a_i(t_i, w_i)||CPU$
5)       else $Priority\_Action\_Number = count(max(w_i))$   //the number of actions belong to the maximal priority
6)         if $Priority\_Action\_Number = 1$
7)            do $a_i(t_i, w_i)||CPU$ with $max(w_i)$
8)         else do $a_i(t_i, w_i)||CPU$ with $max(w_i) \wedge min(t_i)$
9)         endif
10)   endif
11) endwhile
```

IV. Equational theories

Equivalence relations have been studied in classic [3], [18], [2], probabilistic [4], [7], [9], [11], [19], [20], [23], [25], [27], timed and stochastic process algebra [24], [5], [13], [14], [17] to compare components and to replace a component with another which exhibits an equivalent behavior, but has a simpler representation. In process algebra with timed-priority executing policy, we also study its equational theories.

A. Strong bisimulation

In strong bisimulation, there is no difference between observable action and internal action. All actions can be observed and compared together with their parameters.

Definition 5.1 *A binary relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ over processes with action in the form of $a(t, w)$ is a strong bisimulation under policy of* timed-priority *if $(P, Q) \in S$ implies, for all $a(t, w) \in Act$,*

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

Based on definition 5.1, we define the strong equivalence relation.

Definition 5.2 *P and Q are strongly equivalent or strongly bisimilar written as $P \sim_{(t,w)} Q$ w.r.t. $a(t, w)$ under execution policies of* timed-priority. *This may be equivalently expressed as follows:*

$$\sim_{(t,w)} \; = \; \cup\{\mathcal{S}|\mathcal{S} \text{ is a strong bisimulation under policy of } \textit{timed-priority}\}$$

Theorem 5.3 $\sim_{(t,w)}$ is orthogonal extension of process algebras both on timed and priority.

Proof: If we omit the parameter of t in $\sim_{(t,w)}$, $\sim_{(t,w)}$ becomes $\sim_w$ which represents the weak bisimulation of priority. Otherwise, if we omit the parameter of w in $\sim_{(t,w)}$, $\sim_{(t,w)}$ becomes $\sim_t$ which represents the weak bisimulation of real time. Thus, we know that $\sim_{(t,w)}$ is orthogonal extension of process algebras both timed and priority. $\square$

$P + Q = Q + P$	**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 TIMED-PRIORITY EXECUTING POLICY

B. Weak bisimulation

Under situation that internal action τ is unobservable, thus only observable actions can be compared between processes which yields *weak bisimulation*. More precisely, we merely require that each internal action τ can just be omitted from the process.

Definition 5.4 *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).

Now we 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 5.5 *If $t = a_1 \cdots a_n \in Act^*$, then $E \overset{t}{\Rightarrow} E'$ if $E(\overset{\tau}{\rightarrow})^*(\overset{a_1}{\rightarrow})(\overset{\tau}{\rightarrow})^* \cdots (\overset{\tau}{\rightarrow})^*(\overset{a_n}{\rightarrow})(\overset{\tau}{\rightarrow})^* E'$ 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 $E(\overset{\tau^p}{\rightarrow})(\overset{a}{\rightarrow})(\overset{\tau^q}{\rightarrow})(\overset{b}{\rightarrow})(\overset{\tau^r}{\rightarrow})E'$ 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 5.6 *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 summarize the differences between three relations $\overset{t}{\rightarrow}$, $\overset{t}{\Rightarrow}$, 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: $\overset{t}{\rightarrow}$ 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 $P \overset{t}{\rightarrow} P'$ implies $P \overset{t}{\Rightarrow} P'$, and $P \overset{t}{\Rightarrow} P'$ implies $P \overset{\widehat{t}}{\Rightarrow} P'$.

Definition 5.7 *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* timed-priority, *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 $\overset{\widehat{\tau^n}}{\Longrightarrow}\xrightarrow{a(t,w)}\overset{\widehat{\tau^n}}{\Longrightarrow}$, and $\widehat{\tau^n} = 0$ (the empty sequence).

Definition 5.8 *P and Q are observation equivalent or weakly bisimilar under timed-priority execution policy, written* $P \approx_{(t,w)} Q$, *if* $(P, Q) \in \mathcal{S}$. *That is,*

$$\approx_{(t,w)} \;=\; \cup\{\mathcal{S} \mid \mathcal{S} \text{ is a bisimulation under policy of } \textit{timed-priority}\}$$

Theorem 5.9 $\sim_{(t,w)} \Rightarrow \approx_{(t,w)}$

Proof: straight forward. □

Theorem 5.10 $\approx_{(t,w)}$ is orthogonal extension of process algebras both timed t and priority w.

Proof: Parameter t represents the time left for action to accomplish, and parameter w represents the weight/priority of the action. So, t and w are independent parameters. If we add t to process algebra, we get real-time process algebra. If we add w to process algebra, we get a process algebra with priority. Now, we add both t and w into process algebra, then we get the process algebra with timed-priority executing policy. This process algebra can tackle the behaviors of intelligence systems. Then, $\approx_{(t,w)}$ is orthogonal extension of both t and w. □

Note: If we omit parameter t, $\approx_{(t,w)}$ becomes $\approx_w$ which is the weak bisimulation of priority. What's more, if we omit the parameter w, $\approx_{(t,w)}$ becomes $\approx_t$ which is the weak bisimulation of real time. Thus, we know that $\approx_{(t,w)}$ is orthogonal extension of process algebras both timed and priority.

C. The expansion law

The expansion law shows all possible executions of concurrent systems. The nondeterminism and concurrency of the executions in complex systems can be showed clearly by this law.

Definition 5.11 The expansion law under *timed-priority* policy

Let $P \equiv (P_1||_S \cdots ||_S P_n)$, *with* $n \geq 1$. *Then*

$$\begin{aligned} P \;\sim\; & \sum_{\oplus}\{a(t,w).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a} P_i', a(t,w) \notin S, t = t_{min}, w = w_{max}\} \\ & + \sum_{\uplus}\{a(t,w).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a(t,w)} P_i', a(t,w) \notin S\} \\ & + \sum\{\tau.(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_j'\cdots||_S P_n) \mid \\ & P_i \xrightarrow{l} P_i', \quad P_j \xrightarrow{l} P_j', i < j, l \in S\} \end{aligned}$$

Under internal choice composition $\oplus$, action $a(t,w)$ is executed among P_i $(1 \leq i \leq n)$. System selects a action with the biggest priority w_{max} and the least time t_{min}. As for external choice composition $\uplus$, system responds according to the choices of its clients. For internal action τ between two processes, though unobservable, we have reason to believe that the system behaves in the same way.

Theorem 5.12 The expansion law under timed-priority policy is orthogonal extension of process algebras both on timed and priority.

Proof: If we omit the parameter of t, by dropping all t from the formula, the expansion law becomes

$$\begin{aligned} P \;\sim\; & \sum_{\oplus}\{a(w).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a} P_i', a(w) \notin S, w = w_{max}\} \\ & + \sum_{\uplus}\{a(w).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a(w)} P_i', a(w) \notin S\} \\ & + \sum\{\tau.(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_j'\cdots||_S P_n) \mid \\ & P_i \xrightarrow{l} P_i', P_j \xrightarrow{l} P_j', i < j, l \in S\} \end{aligned}$$

Otherwise, if we omit the parameter of w by dropping all w from the formula, the expansion law becomes

$$\begin{aligned} P \;\sim\; & \sum_{\oplus}\{a(t).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a} P_i', a(t) \notin S, t = t_{min}\} \\ & + \sum_{\uplus}\{a(t).(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_n) \mid \\ & P_i \xrightarrow{a(t)} P_i', a(t) \notin S\} \\ & + \sum\{\tau.(P_1||_S\cdots||_S P_i'||_S\cdots||_S P_j'\cdots||_S P_n) \mid \\ & P_i \xrightarrow{l} P_i', P_j \xrightarrow{l} P_j', i < j, l \in S\} \end{aligned}$$

Thus, we know that $\sim_{(t,w)}$ is orthogonal extension of process algebras both timed and priority. □

D. Recursiveness

In this section, we study the equivalent relationship about recursive behavior in our language.

Definition 5.13 *For strong bisimulation relationships under timed-priority execution policy.* $E \sim_{(t,w)} F$ *if, for all indexed sets* $\widetilde{P}$ *of processes,* $E\{\widetilde{P}/\widetilde{X}\} \sim_{(t,w)} F\{\widetilde{P}/\widetilde{X}\}$.

We also use $\widetilde{E} \sim_{(t,w)} \widetilde{F}$ to represent component-wise congruence between $\widetilde{E}$ and $\widetilde{F}$.

Proposition 5.14 *If* $\widetilde{A} \overset{def}{=} \widetilde{P}$, *then* $\widetilde{A} \sim_{(t,w)} \widetilde{P}$.

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

Now we are ready to show that $\sim_{(t,w)}$ is preserved by recursive definition.

Proposition 5.15 *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* $\widetilde{E} \sim_{(t,w)} \widetilde{F}$. *Then* $\widetilde{A} \sim_{(t,w)} \widetilde{B}$.

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

$$E \sim_{(t,w)} F,\ A \overset{def}{=} E\{A/X\}, \text{ and } B \overset{def}{=} F\{B/X\}$$

It will be enough to show that $\mathcal{S}$ is a strong bisimulation up to $\sim_{(t,w)}$, 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 $A \sim_{(t,w)} B$.

To show this, it will be enough to prove that

If $G\{A/X\} \xrightarrow{a} P'$ then, for some Q' and Q'',

$G\{B/X\} \xrightarrow{a} Q'' \sim_{(t,w)} Q'$, 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 $E\{B/X\} \xrightarrow{a} Q'' \sim_{(t,w)} Q'$, with $(P', Q') \in \mathcal{S}$.

But $E \sim F$, so $F\{B/X\} \xrightarrow{a} Q''' \sim_{(t,w)} Q'$, and since $B \overset{def}{=} F\{B/X\}$, $G\{B/X\} \equiv B \xrightarrow{a} Q''' \sim_{(t,w)} Q'$, 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 $G_1\{A/X\} \xrightarrow{a} Q_1'' \sim_{(t,w)} Q_1'$, with $(P_1', Q_1') \in \mathcal{S}$ and $G_2\{A/X\} \xrightarrow{a} Q_2'' \sim_{(t,w)} Q_2'$, with $(P_2', Q_2') \in \mathcal{S}$.

Hence, setting $Q' \equiv Q_1'||_S Q_2'$ and $Q'' \equiv Q_1''||_S Q_2''$ $G\{B/X\} \equiv G_1\{B/X\}||_S\ G_2\{B/X\} \xrightarrow{\tau} Q'' \sim_{(t,w)} Q'$.

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

Process algebra with *timed-priority* executing policy is equipped with the ability of specifying complex behaviors of concurrent systems, esp. for web servers.

Researchers of process calculi have extended in some aspects, e.g., PEPA [16], SPAs [15], [17], probabilistic PAs [19], [25], [23], [27] and so on. Among all of them, there is only one common default executing policy: *maximum progress*. However, one default policy is not enough to specify all behaviors of complex systems. Based on this, we proposed a process algebra with timed-priority executing policy. Actions in this process algebra are equipped with parameters of priority w and time t.

For $a(t, w)$, parameter t represents time limitation, and w represents its priority. New action structure $a(t, w)$ makes it more flexible for system to schedule its services/behaviors.

Compared with other process languages, there are some changes in our language: we refined choice composition $+$ into internal choice $\oplus$ and external choice $\uplus$. As timed-priority executing policy affects the behaviors of internal choice composition and parallel composition, we construct algorithms for them under the definitions.

After giving out operational semantics of our language, we proposed axioms for its operators. Equivalence relations are important in the study of process algebras, based on the analysis of timed-priority, we defined strong bisimulation, weak bisimulation and expansion law for the process language in this paper.

Through the study of process algebra with *timed-priority* executing policy, we found out that there were some changes in the operational semantics according to the timed-priority executing policy. However, all axioms for operators in this language remained unchanged.

In this paper, we also proved that equivalent relations as strong bisimulation, weak bisimulation and expansion law are all orthogonal extensions of process algebras both on time and priority.

VI. Acknowledgement

This work was partially supported by the Fundamental Research Funds for the Central Universities (No. lzujbky-2009-127), National Eleventh Five Year Support Project of China (2006BAI04A10), the Innovative Methodology Project supported by MOST of China(2008IM020900), National Science Foundation of China (No. 30902003, 30973975, and 90709007).

References

[1] Luca Aceto, Taolue Chen, Wan Fokkink and Anna Ingolfsdottir, *On the Axiomatizability of Priority*, Lecture Notes in Computer Science, Volume 4052, pp. 480-491, 2006.

[2] J.C.M. Beaten and W.P. Weijiland, *Process Algebra*, Cambridge University Press, 1990.

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

[4] 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.

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

[6] Eddy Caron, Pushpinder Kaur Chouhan, Frederic Desprez, *Deadline Scheduling with Priority for Client-Server Systems on the Grid*, Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing, Pages: 410 - 414, 2004

[7] L. Christoff. *Specification and Verification Methods for Probabilistic Process*. PhD thesis, Department of Computer Science, Uppsala University, March 1993.

[8] Rance Cleaveland, Gerald Lüttgen, V. Natarajan, *Priority and abstraction in process algebra*, Information and Computation, Volume 205 , Issue 9, Pages 1426-1458, 2007.

[9] 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.

[10] Harald Fecher, Heiko Schmidt, *Process Algebra Having Inherent Choice: Revised Semantics for Concurrent Systems*, Electronic Notes in Theoretical Computer Science (ENTCS), Volume 192, Issue 1, Pages 45-60, 2007 .

[11] 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.

[12] Rob van Glabbeek, Frits Vaandrager. Bundle Event Structures and CCSP *Lecture Notes in Computer Science* Volume 2761, 57-71, 2003.

[13] N. Götz, U. Herzog, and M. Rettelbach. *TIPP - A Stochastic Process Algebra*. In J. Hillston and F. Moller, editors, Proc. of the Workshop on Process Algebra and Performance Modelling. Department of Computer Science, University of Edinburgh, May 1993.

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

[15] 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.

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

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

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

[19] 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.

[20] K. Larsen and A. Skou. *Bisimulation through Probabilistic Testing.* Information and Computation, 94(1):1-28, September 1991.

[21] J-P. Katoen. *Quantitative and qualitative extensions of event structures*, PhD thesis, University of Twente, 1996.

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

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

[24] Anna Philippou, Rance Cleaveland, Insup Lee, Scott Smolka, Oleg Sokolsky, *Probabilistic Resource Failure in Real-Time Process Algebra*, Lecture Notes in Computer Science, Springer Berlin, Volume 1466, pp. 465-472, 1998

[25] M.I.A. Stoelinga and F.W. Vaandrager. *A testing scenario for probabilistic automata.* In J.C.M. Baeten, J.K. Lenstra, J. Parrow, and G.J. Woeginger, editors, Proceedings 30 ICALP, volume 2719 of Lecture Notes in Computer Science, pages 407-418. Springer-Verlag, 2003.

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

[27] 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.