시스템 오류에 대한 확률적 분석

성순용*

Probabilistic Analysis of System Failure

Soonyong Seong*

이 논문은 2008학년도 부산외국어대학교 교수연구년 제도에 의하여 작성되었음

요 약

자원할당 시스템에서는 자원에 대한 요구연산과 반환연산이 반복적으로 이루어진다. 자원을 요구한 프로세스는 우선순위에 따라 할당받은 뒤, 일정 기간 사용 후 다시 반납하게 된다. 이때 자원에 오류가 발생하면 그 오류로부터 회복될 때까지 할당이 지연되거나, 할당받은 프로세스를 중단하는 사태가 발생한다. 이 논문은 이와 같은 처리 과정을 효과적으로 분석하기 위해, 기존의 프로세스 대수학 ACSR에 확률적 선택연산 개념을 추가한 확률적 ACSR을 설계하였다. 확률적 ACSR을 이용하여 요구연산과 반환연산이 발생하는 비율과, 오류가 발생하고 그 오류로부터 복구하는 비율을 확률적으로 표기하고 분석할 수 있음을 보였다.

ABSTRACT

Request operations and release operations occur repeatedly in resource allocation systems. The process requesting a resource acquires one by any priority-based mechanism, and returns the resource after some periods. In this system, resource failures lead to delay of resource allocation, or to termination of process holding the failed resource. To analyze this process effectively, this paper designs a probabilistic ACSR, a process algebra that extends ACSR with the probabilistic choice operation. The ability to express/analyze both request-release rates and failure-recovery rates is illustrated using probabilistic ACSR.

키워드

자원할당시스템, 프로세스 대수학, 오류발생률, 확률적 선택연산

Key word

resource allocation system, process algebra, fault rate, probabilistic choice operation

* 부산외국어대학교 컴퓨터공학과

접수일자 : 2009. 12. 30
심사완료일자 : 2010. 01. 29

I. 서 론

실시간 시스템의 기술 및 분석에 사용되는 정형적 방법 중에 시간과 확률 개념을 도입하고 시간 및 확률 연산자를 추가한 프로세스 대수학들이 많이 발표되어지고 있다[1,2,3]. 그 중에서도 ACSR(Algebra of Communicating Shared Resources)[4]은 프로세스 동기화에 따른 처리시간 지연만을 다루는 다른 기법들에 비해 공유 자원 경합으로 인한 시간 지연도 동시에 기술할 수 있는 프로세스 대수학으로서 실시간 시스템을 보다 구체적이고 현실적으로 표현할 수 있는 기법으로 평가할 수 있다.

ACSR에 자원의 오류 발생 확률을 부여함으로써 정상자원과 결함자원 두 가지 형태의 자원으로 시스템을 기술한 PACSR[5]도 발표된 바 있다. 그러나 PACSR에서는 자원의 오류 발생 확률을 제외한 다른 확률적 기술이 불가능하며, 마치 결함이 생긴 자원을 보유하고 동작이 수행되는 것처럼 기술하여 일반적인 시스템 기술이 부자연스럽게 이루어진다.

본 논문에서는 ACSR의 선택연산에 확률개념을 추가한 확률적 ACSR을 설계하고자 한다. ACSR에서 선택연산을 수행할 때 두 가지 프로세스가 모두 가능한 경우, 선점관계에 의해서 한 동작이 다른 동작을 선점하지 않는 한, 같은 확률로 두 동작이 모두 가능하도록 처리하고 있다. 그러나 일반적인 자원할당 시스템에서는 자원의 요구나 반납 비율이 시스템 상태에 따라 영향을 받으므로 이러한 문제를 제어하기 위한 확률적 선택 기능을 ACSR에 추가하는 것이 필요하게 되었다. 이와 같은 확률 개념이 추가된 확률적 ACSR을 정의하고, 확장된 확률적 ACSR을 이용하여 자원할당시스템의 기술이 보다 효과적으로 이루어질 수 있음을 보이고자 한다.

II. 확률적 ACSR

확률적 ACSR은 시간과 자원을 소모하며 발생하는 시간동작(timed action)과 동기화를 위해 필요한, 순간적으로 발생하는 이벤트(event), 이 두 가지 형태의 동작을 기본으로 하여 프로세스를 기술한다.

시간동작은 순차적으로 재사용 가능한 자원과 우선순위로 이루어진 순서쌍의 집합으로 한 단위시간을 소모하는 동작을 표기한다. 이때 자원 집합 R의 각 자원은 한 단위씩으로 구성되며, 그 결과 어느 시점에 있어서 최대 한 번만 표기 가능하다. 시간동작 A에 대해 $\rho(A)$는 A가 사용하는 자원의 집합을, $\pi_r(A)$는 A의 자원 r에 대한 우선순위 값을 나타낸다. {}는 자원은 전혀 사용하지 않으면서 한 단위시간을 소모하는 시간동작이다.

이벤트는 그 실행에 시간을 소모하지 않으며 (a, p)와 같은 쌍으로 표기한다. 이때 a는 이벤트의 라벨이고, p는 그 실행 우선순위이다. 이벤트 e에 대해 $l(e)$는 e의 라벨을, $\pi(e)$는 우선순위를 나타낸다. 이벤트 발생 시 값의 전송도 가능하다[6]. 예를 들어 (a, p)?x는 입력 이벤트로서 전송 받은 값을 x에 주는 것을 표기하고, (a, p)!x는 출력 이벤트로서 x의 값을 전송함을 표기한다. 특수 라벨 τ를 사용하여 같은 라벨을 갖는 입력 이벤트와 출력 이벤트가 동기화되어 동시에 실행될 때 그 결과의 이벤트를 표시한다.

이 두 가지 실행 동작으로부터 프로세스의 기술이 가능하다. 확률적 ACSR의 프로세스P는 다음과 같은 문법으로 설명할 수 있다.

$$
\begin{aligned}
P ::= & \ NIL \mid A : P \mid e \,.\, P \mid be \rightarrow P \mid P + Q \\
& \mid P \parallel Q \mid [P]I \mid P\backslash F \mid C(x)
\end{aligned}
$$

NIL은 아무런 실행 동작도 취하지 않는, 교착상태의 프로세스다. A : P는 한 단위 시간을 소모하는 시간동작 A를 실행하고 다음 프로세스 P로 진행하고, e . P는 이벤트 e를 실행하고 P로 진행한다.

$be \rightarrow P$ 는 부울 식 be가 참일 때만 P를 실행한다.

P + Q는 P과 Q 중에 하나를 선택하여 실행한다.

$P \parallel Q$ 는 두 프로세스의 실행이 동시에 병렬로 진행된다. 즉 $P \parallel Q$ 에서 이벤트는 시간을 소모하지 않으니까 각각 순서적으로 발생하든지, 또는 같은 라벨을 갖는 입, 출력 이벤트라면 동기적으로 동시 발생 가능하며, 시간동작은 서로 같은 자원을 사용하지 않는 한, 각각 한 단위의 시간을 소모하며 동시에 병렬 실행된다.

$[P]_I$는 P가 집합 I에 있는 자원을 독점적으로 사용한다. $P\backslash F$는 P의 실행에 있어서 집합 F에 포함된 라벨을 갖

는 이벤트의 실행이 제한된다. 즉 $a \in F$라면 (a, 1)?, (a, 1)!과 같은 이벤트 실행은 불가능하고, 두 이벤트가 동기화된 (τ, 2) 이벤트만 실행 가능해져 프로세스 동기화를 제어할 때 유용하게 이용될 수 있다.

C(x)는 C(x) ≡ P와 같이 프로세스를 재귀적으로 정의할 때 사용할 수 있다.

우선순위를 고려하여 실행을 제어하고자 할 때 선점관계(preemption relation) " $<$ "을 다음과 같이 정의한다. 이때 두 실행 동작 α, β에 대해 $\alpha < \beta$라 함은 어느 시점에 α 또는 β를 선택해야 하는 경우 항상 β를 우선적으로 실행하도록 제어함을 의미한다.

정의) 선점관계 : 두 실행동작 α, β에 대해 다음 세 조건 중의 하나를 만족하면 $\alpha < \beta$ (β가 α를 선점한다)라고 한다.

1) α, β가 모두 시간동작이고
$\rho(\beta) \subseteq \rho(\alpha)) \wedge (\forall r \in \rho(\alpha), \pi r(\alpha) \leq \pi r(\beta))$
$\wedge (\exists r \in \rho(\beta), \pi r(\alpha) < \pi r(\beta))$
2) α, β가 모두 이벤트고
$\pi(\alpha) < \pi(\beta) \wedge l(\alpha) = l(\beta)$
3) α는 시간동작, 또는 확률적 선택,
β는 이벤트고 $l(\beta) = \tau \wedge \pi(\beta) > 0$

확률적 ACSR에서 선택연산 P + Q 는 다음과 같이 확률적 선택 기능이 추가되어 재정의된다.

$$p_p \bullet P + p_q \bullet Q \ , \ p_p + p_q = 1$$

위의 식에서 p_p와 p_q는 각각 프로세스 P와 프로세스 Q를 선택할 확률을 나타낸다. 이 확률들에 의해 두 프로세스 중 하나의 동작을 시도하게 된다는 것으로, 이 확률적 선택이 우선순위에 의한 선점관계보다 우선하는 것으로 정의한다.

여러 개의 프로세스 중에서 선택해야 하는 경우도 비슷하게 처리 가능하다.

$$p_1 \bullet P_1 + p_2 \bullet P_2 + \cdots + p_n \bullet P_n \ , \ \sum p_i = 1$$

확률이 명시되지 않은 선택연산의 경우에는 우선순위로 인한 선점관계가 존재할 경우 그 프로세스를 시도하고, 선점관계가 존재하지 않을 경우에는 작동 가능한 모든 프로세스 중에서 하나를 같은 확률로 선택하게 하는 것으로 정의한다.

예를 들어보면 다음과 같다.

1) JOB = {} : JOB + {(CPU, 1)} : P

CPU 자원이 사용가능하면 선점관계에 의해 시간동작 {(CPU, 1)}이 실행되고, CPU 자원이 사용가능하지 않으면 시간동작 {}이 실행된다.

2) JOB = ({} : JOB + {(CPU, 1)} : P1
+ (a, 1)?.P2) $\backslash_{\{a\}}$

이벤트 (a, 1)?가 이벤트 (a, 1)!와 동기화 되면 이벤트 (τ, 2)가 발생하여 프로세스 P2로 가고, 동기화가 불가능하면 1)과 같이 선택적으로 실행된다.

3) JOB = $\frac{1}{2}$ • {} : JOB + $\frac{1}{2}$ • {(CPU, 1)} : P

이때는 확률이 명시되어 있으므로 자원 CPU의 사용가능성과 관계없이 먼저 $\frac{1}{2}$의 확률로 두 시간동작 중에서 하나를 선택한 후, 시간동작 {}을 실행하거나 혹은 시간동작 {(CPU, 1)}을 실행한다. 시간동작 {(CPU, 1)}을 실행하고자 할 때 CPU 자원이 사용불가능하면 교착상태 NIL이 되는 것이다.

4) JOB = (($\frac{1}{2}$ • {}:JOB + $\frac{1}{2}$ • {(CPU, 1)}:P1)
+ (a, 1)?.P2) $\backslash_{\{a\}}$

이벤트 (a, 1)?의 동기화 연산이 가능하면 동기화 연산이 확률적 선택연산을 선점하므로 동기화 연산이 발생하고, 동기화가 불가능한 상태에서만 $\frac{1}{2}$ 확률로 선택연산을 실행한다.

ACSR에서는 C 단위만큼의 시간 동안 CPU를 사용하는 JOB을 다음과 같이 기술할 수 있다.

JOB = {} : JOB + $P_1(0)$
$P_1(S) = (S < C) \rightarrow$ {(CPU, 1)} : $P_1(S+1)$
+ $(S = C) \rightarrow$ NIL

JOB은 자원 CPU가 가능해질 때까지 기다리다, CPU를 할당받으면 그 실행시간 S가 C가 될 때까지 계속 CPU를 사용하고 멈추게 된다. 확률적 ACSR을 사용하면, 위의 JOB 문제를 일반화된 CPU 사용시간을 갖는 JOB'으로 확장하여 기술할 수 있다.

$JOB' = \{\} : JOB' + \{(CPU, 1)\} : P$

$P = p_p \bullet \{(CPU, 1)\} : P + (1 - p_p) \bullet Q$

즉 일단 CPU를 차지하여 작업을 시작한 프로세스 P가 매번 확률 p_p로 CPU 작업을 계속 반복 선택하다가 어느 순간 확률 $(1 - p_p)$로 CPU를 떠나 다른 프로세스 Q로 이동하는 과정을 기술하고 있다. 프로세스 P에서 다시 시간동작 {(CPU, 1)}을 선택할 확률이 이전 선택에 전혀 영향을 받지 않고 매번 독립적으로 선택과정이 이루어지므로, 이러한 기술은 CPU에서 서비스 받는 시간이 평균 $1/(1 - p_p)$인 지수분포를 따르는 Markov 프로세스를 표현할 수 있게 한다.

지금까지 확률적 ACSR을 이용하여 프로세스를 기술하는 문법에 대해 살펴보았다. 이와 같이 기술된 프로세스는 간단한 대수학 법칙을 사용하여 보다 표준화된 형태로 전이 가능하며 이와 같은 전이를 통하여 시스템 분석이 가능해진다[4].

Ⅲ. 시스템 오류 기술 및 분석

앞에서 기술한 확률적 ACSR을 이용하여 시스템의 오류발생 현상을 기술하고 이를 분석하고자 한다. 앞으로 분석할 대상 시스템을 다음과 같이 정의한다.

시스템에는 n 개의 동종 프로세스(homogeneous process)와 서로 다른 r 종류의 재사용 가능 자원이 한 단위씩 존재한다. 시스템 전체에서 프로세스들의 자원 요구 및 반환 연산이 반복 수행된다. 이때 연산을 수행하는 프로세스는 자신이 현재 가지고 있지 않은 자원들을 한 번에 하나씩 요구하며, 자신이 갖고 있는 자원들을 한 번에 하나씩 반환한다. 요구연산에서 요구하는 자원의 선택은 가지고 있지 않는 자원 모두에게 같은 확률로 주어지며, 반환연산에서도 반환하는 자원의 선택은 갖고 있는 자원 모두에게 같은 확률로 주어진다. 각 프로세스의 자원에 대한 요구연산은 현재 보유한 자원의 수가 r 보다 적을 때 λ의 비율로, 반환연산은 보유한 자원의 수가 하나 보다 많을 때 μ의 비율로 발생한다.

각 자원 r_i는 현재 임의의 프로세스에게 할당된 상태 R_i'이거나 할당이 가능한 상태 R_i로 존재한다. 또한 결함이 발생하면 상태 $\underline{R_i}$로 전이한다.

자원 r_i의 결함 발생 비율은 현재 결함이 없는 경우에 매 단위시간마다 p_i의 확률로 발생한다고 가정한다. 그리고 일단 발생한 결함은 매 단위시간마다 q_i의 확률로 치료가능하다고 가정한다. 이와 같은 시스템에서 자원 r_i의 상태는 다음과 같이 진행한다.

$R_i = (r_i, 1)?.R_i' + (p_i \bullet \{\} : \underline{R_i} + (1-p_i) \bullet \{\} : R_i)$

$R_i' = (r_i', 1)?.R_i + (p_i \bullet (\underline{r_i}, 1)!.\{\} : \underline{R_i}$

$\quad + (1-p_i) \bullet \{\} : R_i')$

$\underline{R_i} = q_i \bullet \{\} : R_i + (1-q_i) \bullet \{\} : \underline{R_i}$

상태 R_i에서 요구연산을 나타내는 $(r_i, 1)!$ 이벤트와 동기화되면 R_i'로 전이한다. 요구연산 이벤트가 없을 경우엔 결함이 발생하느냐의 여부에 따라 R_i, 또는 $\underline{R_i}$로 이동한다. R_i'에서도 반환연산 이벤트 $(r_i', 1)!$과 동기화되어 R_i로 전이되어진다. 역시 결함 여부에 따라 R_i', 또는 $\underline{R_i}$로 전이하게 된다. R_i'에서 $\underline{R_i}$로 전이하게 되는 경우 이 자원을 소유한 프로세스에게 결함이 발생했다는 사실을 $(\underline{r_i}, 1)!$ 이벤트로 알려주면, 해당 프로세스를 중단시키고, 중단된 프로세스가 보유한 모든 자원을 반납하게 한 후 다시 초기 상태에서 실행을 재개하게 한다. 상태 $\underline{R_i}$는 결함이 존재하는 경우이므로 결함이 복구될 때까지 같은 상태를 유지하며, 이 상태에서는 할당이 이루어지지 못한다.

자원의 요구연산 시 해당 자원이 다른 프로세스에 이미 할당된 상태이면 그 자원이 반환될 때까지 기다려야 한다. 또한 그 자원에 결함이 발생했을 때도 그 결함이 고쳐질 때까지 기다려야 한다.

시스템의 자원 할당은 즉시 할당 상태(expedient state)로 가정한다. 즉 요구한 자원이 현재 할당된 상태가 아니고 결함상태가 아니라면 바로 할당받는다.

자원 r_i에 대한 요구연산은 아래와 같이 기술될 수 있다. 현재 시스템에는 자원이 r_i 하나만 존재한다는 가정하에 프로세스 P가 자원 r_i를 요구하고 반납하는 과정을 기술한다.

$P = (1-\lambda) \bullet \{\} : P + \lambda \bullet P^j$

$P^j = (r_i, 1)!.\{(r_i,1)\} : P' + \{\} : P^j$

$P' = (\underline{r_i}, 1)?.P$

$\quad + ((1-\mu) \bullet \{(r_i,1)\} : P' + \mu \bullet (r_i', 1)!.\{\} : P)$

상태 P는 요구연산이 발생하면 P^i로 전이한다. P^i에서는 현재 자원이 상태 R_i에 있을 때에만 할당 가능하며, 이때 $(r_i, 1)?$ 이벤트와 동기화되어 자원을 할당받고 P'으로 전이된다. P'에서는 자원에 결함이 생겼다는 이벤트 $(\underline{r_i}, 1)!$와 동기화되면 자원을 반납하고 P로 전이한다. 상태 P'에서 자원을 반납할 때는 $(r_i', 1)!$ 이벤트를 이용해 R_i' 상태의 자원에게 알려준다.

위에서 기술한 자원과 프로세스의 상태전이를 도식화한 것이 각각 <그림 1>과 <그림 2>다.

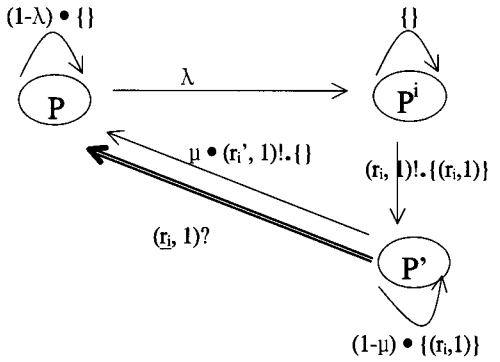

그림 1. 자원의 결함 상태 전이
Fig. 1 Transition of resource failure states

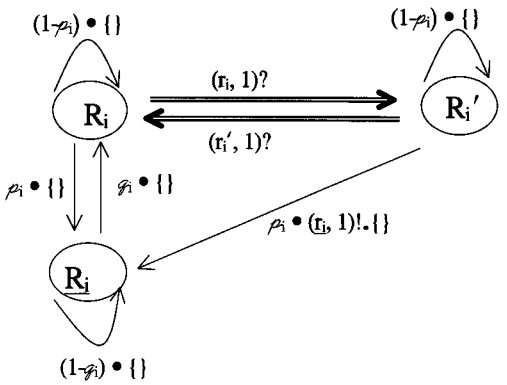

그림 2. 자원할당 상태 전이
Fig. 2 Transition of resource allocation states

그림에서 상태의 진행방향을 나타내는 화살표에 표기되어 있는 라벨에서 수치 pp 등은 선택 확률을 나타내고. 반면에 $(r_i, 1)?$와 같은 이벤트와 {}와 같은 시간동작은 각각 그 동작을 수행하고 상태가 전이됨을 나타낸다. 이벤트의 경우에는 그 상대 이벤트와 동기화되어야만 전이가 일어난다. 화살표 ⇒는 해당 동기화 연산이 화살표 →의 확률적 선택 연산을 선점함을 표기한다.

두 종류의 자원 r_1과 r_2가 존재하는 오류 발생 시스템을 확률적 ACSR로 표기하면 아래와 같다.

r 종류의 자원을 갖는 시스템도 같은 방식으로 기술 가능하다. 전체 시스템은 다음과 같이 자원 r_1과 r_2를 경합하는 n 개의 프로세스 P_i로 구성된다.

$$\boxed{\begin{aligned}&\mathrm{SYSTEM} = [(P_1 \parallel \ldots \parallel P_i \parallel \ldots \parallel P_n \\ &\quad \parallel R_1 \parallel R_2) \setminus \{r_1, r_2, r_1', r_2', \underline{r_1}, \underline{r_2}\}]_{\{r1,r2\}}\end{aligned}}$$

$$P_i = P_{i\{\}}$$

$$P_{i\{\}} = (1-\lambda) \bullet \{\} : P_{i\{\}} + \lambda/2 \bullet P_{i\{\}}^{(1)} + \lambda/2 \bullet P_{i\{\}}^{(2)}$$

$$P_{i\{\}}^{(m)} = (r_m, \pi_i)!.\{(r_m,\pi_i)\} : P_{i\{m\}} + \{\} : P_{i\{\}}^{(m)}$$

$$\begin{aligned}P_{i\{m\}} = (\underline{r_m}, 1)?.P_{i\{\}} + (\ &(1-\lambda-\mu) \bullet \{(r_m,\pi_i)\} : P_{i\{m\}} \\ &+ \lambda \bullet P_{i\{m\}}^{(n)} + \mu \bullet (r_m', 1)!.\{\} : P_{i\{\}}\ )\end{aligned}$$

$$\begin{aligned}P_{i\{m\}}^{(n)} = (r_n, \pi_i)!.\{(r_m,\pi_i), (r_n,\pi_i)\} &: P_{i\{m,n\}} \\ + \{(r_m,\pi_i)\} &: P_{i\{m\}}^{(n)}\end{aligned}$$

$$\begin{aligned}P_{i\{m,n\}} = (\underline{r_m}, 1)?.P_{i\{\}} &+ (\underline{r_n}, 1)?.P_{i\{\}} \\ &+ (\ (1-\mu) \bullet \{(r_m,\pi_i), (r_n,\pi_i)\} : P_{i\{m,n\}} \\ &+ \mu/2 \bullet (r_m', 1)!.\{(r_n,\pi_i)\} : P_{i\{n\}} \\ &+ \mu/2 \bullet (r_n', 1)!.\{(r_m,\pi_i)\} : P_{i\{m\}}\ )\end{aligned}$$

$$\begin{aligned}R_m = (r_m, 1)?.R_m' &+ (p_m \bullet \{\} : \underline{R_m} \\ &+ (1-p_m) \bullet \{\} : R_m)\end{aligned}$$

$$\begin{aligned}R_m' = (r_m', 1)?.R_m \\ + (p_m \bullet (\underline{r_m}, 1)!.\{\} : \underline{R_m} + (1-p_m) \bullet \{\} : R_m')\end{aligned}$$

$$\underline{R_m} = q_m \bullet \{\} : R_m + (1-q_m) \bullet \{\} : \underline{R_m}$$

위의 기술에서 집합 A에 대해 P_{iA}는 $m \in A$인 경우 프로세스 P_i가 자원 R_m을 보유하고 있는 상태이고, $P_{iA}^{(m)}$은 프로세스 P_i가 $m \notin A$인 자원 R_m을 요구하고 있는 상태이다. π_i는 프로세스 P_i의 우선순위이다.

우선 각 프로세스의 초기상태는 아무 자원도 보유하지 않은 상태 $P_{i\{\}}$에서 시작한다. $P_{i\{\}}$는 현재 자원을 보유하지 않은 상태이므로 반환연산은 발생하지 않으며, λ의 확률로 자원을 요구하거나, 1-λ의 확률로 자원을 보유하지 않은 상태에서 그대로 한 단위시간만큼 작업을 수행한다. 요구 자원은 R_1, R_2에 대해 같은 확률을 가지므로

각각 ½의 확률로 선택하게 된다. 위의 기술에서 m=0이면 n=1이고, m=1이면 n=0이다.

$P_{i\{\}}^{(m)}$은 자원 R_m을 사용 가능하여 얻게 되면 시간동작 $\{(r_m,1)\}$을 한 단위시간만큼 수행한 후 $P_{i\{m\}}$으로 진행되고, 자원 r_m을 얻지 못하면 얻을 때까지 그 상태에서 계속 기다린다.

$P_{i\{m\}}$은 자원 r_m을 보유하고 있는 상태이므로 다른 자원 r_n을 요구할 수도 있고, 보유자원 r_m을 반환할 수도 있다. 그 결과 각각 λ와 μ의 확률로 $P_{i\{m\}}^{(n)}$, $P_{i\{\}}$로 진행하고, 나머지 1-λ-μ의 확률로 보유자원을 갖고 시간동작 $\{(r_m,1)\}$을 한 단위만큼 수행한 후 같은 과정을 반복한다. 이때 r_m에 오류가 발생하면 그 자원을 반납하고 초기상태 $P_{i\{\}}$로 돌아간다.

$P_{i\{m\}}^{(n)}$은 자원 r_n도 얻어 시간동작 $\{(r_m,1), (r_n,1)\}$을 수행할 수도 있고, r_n을 얻을 때까지 r_m을 보유한 상태로 대기상태에 머문다.

$P_{i\{m,n\}}$은 모든 자원을 보유하고 있는 프로세스 P_i가 더 이상의 요구연산은 수행하지 못하고, μ의 확률로 r_m 또는 r_n 중의 하나를 반환한다. 그리고 1-μ의 확률로 자원 r_m, r_n을 보유한 시간동작을 한 단위시간만큼 수행한다. 여기서 하나 이상의 자원에 결함이 생기는 경우에는 그 자원을 포함하여 모든 자원을 반납하고 중단됨으로써 초기상태 $P_{i\{\}}$로 돌아가게 된다.

지금까지 기술한 SYSTEM의 상태전이를 도식화한 것이 <그림 3>이다.

지금까지 전형적인 자원 할당 시스템에서 오류가 발생하는 상황을 확률적 ACSR로 기술해 보았다.

이를 분석하여 각 프로세스 Pi가 PiA(m) 상태에 머무는 비율을 계산하면 자원할당을 위해 대기하는 시간 비율을 계산할 수 있다. 또한 PiA의 집합 A의 크기를 이 상태의 시간만큼 누적하여 평균하면 각 프로세스가 보유한 자원의 평균이 계산된다. SYSTEM이 진행하다가 임의의 서로 다른 두 프로세스 Pi와 Pj에 대해 Pi{0}(1) ‖ Pj{1}(0) 인 상태가 발생하면 교착상태가 되므로 교착상태 발생에 대한 분석도 가능해진다. 이때 pm과 qm의 값을 변화시켜 주면 오류가 발생하는 비율이 미치는 영향에 대해서도 분석가능하다.

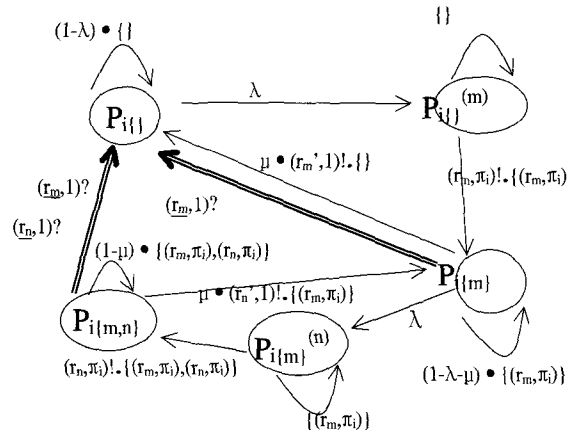

그림 3. SYSTEM의 상태 전이
Fig. 3 Transition of SYSTEM states

Ⅳ. 결 론

실시간 시스템을 기술하고 분석하는 도구로서 시간과 확률 개념을 도입한 프로세스 대수학이 많이 이용되어지고 있는 바, 그 중에서도 ACSR은 프로세스 동기화로 인한 시간 지연뿐만 아니라 공유 자원 경합에 따른 시간 지연도 함께 표현할 수 있는 보다 구체적이고 현실성 있는 계산 모델이다. 그리고 ACSR에 확률적 선택연산 개념을 도입한 것이 확률적 ACSR이다.

본 논문에서는 확률적 ACSR을 이용하여 일반적인 자원 할당 시스템에서 오류가 발생하고 이로부터 복구하는 상황을 표기하여 이때 발생하는 교착상태를 비롯한 제반 기술 및 분석이 효율적으로 이루어질 수 있음을 보였다.

참고문헌

[1] Holger Hermanns, Ulrich Herzog, Joost-Pieter Katoen, "Process algebra for performance evaluation", Theoretical Computer Science, vol. 274, no. 1-2, pp 43-87, March 2002

[2] Mario Bravetti, "Expressing Priorities and External Probabilities in Process Algebra via Mixed Open/C;osed Systems", Electronic Notes in Theoretical Computer Science, vol. 194, no. 2, pp 31-57, January 2008

[3] Pedro R. D'Argenio, Joost-Pieter Katoen, "A theory of stochastic systems, Part II: Process algebra", Information and Computation, vol. 203, no. 1, pp 39-74, November 2005,

[4] Insup Lee, Hanêne Ben-Abdallah, Jin-Young Choi, "A Process Algebraic Method for Real- Time Systems", Formal Methods for Real- Time Computing, John Wiley&Sons Ltd, 1996

[5] Insup Lee, Anna Philippou, Oleg Sokolsky, "Resources in process algebra", Journal of Logic and Algebraic Programming, vol. 72, pp. 98-122, May/June 2007

[6] Jin-Young Choi, Insup Lee, Hong-Liang Xie, "The Specification and Schedulability Analysis of Real-Time Systems using ACSR," Proc. 16th IEEE Real-Time Systems Symposium, 1995

저자소개

성순용(Soonyong Seong)

1983 서울대학교 자연과학대학 계산통계학과 (이학사)
1985 서울대학교 대학원 계산통계학과(이학석사)
1992 서울대학교 대학원 계산통계학과(이학박사)
1989 - 현재 부산외국어대학교 컴퓨터공학과 교수
※관심분야 : 운영체제, 성능평가