PACSR : 확률적 ACSR

성순용

부산외국어대학교

PACSR : A Probabilistic ACSR

Soonyong Seong

Pusan University of Foreign Studies

E-mail : syseong@pufs.ac.kr

요 약

실시간 시스템의 기술 및 분석에 사용되는 정형적 방법 중에 시간 개념을 도입하고 시간 연산자를 추가한 프로세스 대수학들이 많이 발표되어지고 있다. 그 중에서도 ACSR은 동기적으로 발생하는 timed action과 비동기적으로 발생하는 event로 시스템을 기술하는 기법이다. timed action은 공유 자원을 사용하면서 시간을 소모하는 과정을 나타내고, event는 프로세스 간의 동기화 과정을 기술한다. 실시간 시스템을 보다 효율적으로 표현하기 위하여 본 논문에서는 확률 개념이 추가된 PACSR을 정의한다. 또한 확장된 PACSR을 이용하여 자원할당시스템의 기술이 보다 효과적으로 이루어질 수 있음을 보이고자 한다.

ABSTRACT

There has been significant progress in the development of timed process algebra for the specification and analysis of real-time systems in recent years. ACSR is a timed process algebra, which supports synchronous timed actions and asynchronous instantaneous events. Timed actions model the usage of shared resources and the passage of time, whereas events allow synchronization between processes. To be able to specify real-time systems more effectively, this paper suggests the notion of probabilities. This paper also illustrates extended PACSR with a typical resource allocation system and its deadlock specification and analysis.

키워드

프로세스 대수학, PACSR, 공유자원, 프로세스 동기화

Ⅰ. 서 론

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

기존 ACSR에서는 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. 본 논문에서는 이와 같은 확률 개념이 추가된 PACSR을 정의하고, 확장된 PACSR을 이용하여 자원할당시스템의 기술이 보다 효과적으로 이루어질 수 있음을 보이고자 한다.

Ⅱ. ACSR

ACSR은 시간과 자원을 소모하며 발생하는 timed action과 동기화를 위해 필요한, 순간적으로 발생하는 event, 이 두 가지 형태의 동작을 기본으로 하여 프로세스를 기술하고 있다.

timed action은 순차적으로 재사용 가능한 자원과 우선순위를 나타내는 숫자의 순서쌍의 집합으로 한 단위 시간을 소모하는 동작을 표기한다. 이때 자원집합 R의 각 자원은 한 단위씩으로 구성되며, 그 결과 어느 시점에 있어서 최대 한 번만 표기 가능하다. timed action A에 대해 $\rho(A)$는 A가 사용하는 자원의 집합을, $\pi_r(A)$는 A의 자원 r에 대한 우선순위 값을 나타낸다. {}는 자원을 전혀 사용하지 않으면서 한 단위 시간을 소모함을 표시한다.

event는 실행하는 데 전혀 시간을 소모하지 않으며 (a, p)와 같은 쌍으로 표기한다. 이때 a는 event의 라벨을, p는 그 실행 우선순위를 나타낸다. event e에 대해 $l(e)$는 e의 라벨을, $\pi(e)$는 우선순위를 나타낸다. event 발생시 값의 전송도 가능하다[5]. 예를 들어 (a, p)?x는 input event로서 전송받은 값을 x에 주는 것을 표기하고, (a, p)!x는 ouput event로서 x 값을 전송함을 표기한다. 그리고 특수 라벨 τ를 사용하여 같은 라벨을 갖는 input event와 output event가 동기화되어 동시에 실행될 때 그 결과 event를 표시한다.

이 두 가지 실행 동작으로부터 프로세스의 기술이 가능해진다. ACSR의 프로세스 P는 다음과 같은 문법으로 설명 가능하다.

- NIL : 아무런 실행 동작도 취하지 않는, 교착상태의 프로세스를 나타낸다.
- A : P : 한 단위의 시간을 소모하는 timed action A를 실행하고 다음 프로세스 P로 진행한다.
- e . P : event e를 실행하고 P로 진행한다.
- be → P : 부울식 be가 참일 때 P를 실행한다.
- P1 + P2 : P1과 P2 중 하나를 선택 실행한다.
- P1 || P2 : 두 프로세스의 실행이 동시에 병렬적으로 진행된다. 즉 P1 || P2 에서 event는 시간을 소모하지 않으니까 각각 순서적으로 발생하든지, 또는 같은 라벨을 갖는 input, output event라면 동기적으로 동시 발생 가능하며, timed action은 서로 같은 자원을 사용하지 않는 한, 각각 한 단위 시간을 소모하며 동시에 병렬 실행된다.
- $[P]_I$: P가 집합 I에 있는 자원을 독점적으로 사용한다.
- P\F : P 실행에 있어 집합 F에 포함된 라벨의 event 실행이 제한된다. 즉 $a \in F$라면 (a, 1)?, (a, 1)!과 같은 event의 실행은 불가능하고, 두 event가 동기화된 $(\tau, 2)$ event만 실행 가능해져 프로세스 동기화를 제어할 때 유용하게 이용될 수 있다.
- C(x) : C(x) ≡ P와 같이 프로세스를 재귀적으로 정의할 때 사용된다.

ACSR을 이용한 시스템 기술의 예로서 다음과 같이 C 단위 만큼의 시간 동안 CPU를 사용하는 JOB을 살펴보자.

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

JOB은 자원 CPU가 가능해질 때까지 기다리다 ({} : JOB), CPU를 할당받으면 그 실행시간 S가 C가 될 때까지 계속 CPU를 사용하고 멈추게 된다.

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

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

1) α, β가 모두 timed action 이고
$(\rho(\beta) \subseteq \rho(\alpha)) \wedge (\forall r \in \rho(\alpha), \pi_r(\alpha) \leq \pi_r(\beta))$
$\wedge (\exists r \in \rho(\beta), \pi_r(\alpha) < \pi_r(\beta))$
2) α, β가 모두 event이고
$\pi(\alpha) < \pi(\beta) \wedge l(\alpha) \approx l(\beta)$
3) α는 timed action, β는 event이고
$l(\beta) = \tau \wedge \pi(\beta) > 0$

즉 timed action β가 timed action α보다 α의 모든 자원에 대해 우선순위가 작지 않고 최소한 하나 이상의 자원에 있어서는 우선순위가 클 때, 또는 같은 라벨을 갖는 event α, β에서 β가 우선순위가 클 때, β가 α를 선점하게 하며, 동기화 event (τ, x)는 모든 timed action을 선점하여 실행되도록 정의한 것이다.

지금까지 ACSR을 이용하여 프로세스를 기술하는 문법에 대해 살펴보았다. 이와 같이 기술된 프로세스는 간단한 대수학 법칙을 사용하여 보다 표준화된 형태의 표현으로 전이가 가능하며 이와 같은 전이를 통하여 시스템 분석이 가능해진다.

III. PACSR 정의

앞에서도 언급한 바와 같이 기존의 ACSR에서는 P + Q와 같은 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 자원 할당 시스템에서는 자원의 요구나 반납 비율이 시스템 상태에 따라 영향을 받게 되므로 이러한 문제를 제어하기 위한 확률적 선택 기능을 ACSR에 추가하는 것이 필요

하게 되었다. 이와 같은 확률 개념이 추가된 ACSR을 PACSR(Probabilistic ACSR)이라 명명하기로 한다.

PACSR에서 P + Q 연산은 다음과 같이 다시 정의된다.

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

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

여러 개의 프로세스 중에서 선택해야 하는 경우도 비슷하게 처리 가능하다.

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

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 timed action {(CPU, 1)}이 실행되고, CPU 자원이 사용가능하지 않으면 timed action {}이 실행된다.

2) JOB = {} : JOB + {(CPU1, 1)} : P1
+ {(CPU2, 1)} : P2

CPU1과 CPU2가 모두 사용가능하면 1/2확률로 두 timed action {(CPU1, 1)}과 {(CPU2, 1)} 중에서 선택하여 실행하고, 둘 중의 하나의 자원 CPU1만 사용가능하면 {(CPU1, 1)}을 실행하며, 두 자원이 모두 사용가능하지 않으면 {} : JOB가 실행된다.

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

event (a, 1)?가 event (a, 1)!와 동기화 되면 event (τ, 2)가 발생하여 프로세스 P2로 가고, 동기화가 불가능한 상태에서 자원 CPU가 사용가능하면 timed action {(CPU, 1)}이 실행되며, 앞의 두 동작이 모두 불가능한 경우에는 timed action {}이 실행된다.

4) JOB = 1/2 • {} : JOB + 1/2 • {(CPU, 1)} : P

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

이와 같이 정의된 PACSR을 사용하면, 앞에서 예를 든 C 단위 시간만큼의 CPU를 사용하는 JOB 문제를 일반화된 CPU 사용시간을 갖는 JOB'으로 확장하여 기술할 수 있다.

JOB' = {} : JOB' + {(CPU, 1)} : P
P = p_p • {(CPU, 1)} : P + (1 - p_p) • 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 timed action {(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.

IV. 자원할당시스템 분석

앞에서 정의한 PACSR을 이용하여 일반적인 자원 할당 시스템을 표기하고 이때 발생하는 교착상태를 분석하고자 한다. 앞으로 분석할 대상 시스템을 다음과 같이 정의한다.

시스템에는 n 개의 동종 프로세스와 서로 다른 r 종류의 재사용 가능 자원이 한 단위씩 존재한다. 시스템 전체에서 프로세스들의 자원 요구 및 반환 연산이 반복 수행되는데, 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 요구 및 반환 연산은 현재 보유한 자원의 수가 i 개일 때 그 평균 수행 비율이 각각 $\lambda(i)$, $\mu(i)$인 Poisson 분포의 빈도로 발생한다.

$$\lambda(i) = \begin{cases} \lambda & 0 \le i < r \\ 0 & i \ge r \end{cases}$$

$$\mu(i) = \begin{cases} \mu & 1 \le i \le r \\ 0 & i > r \end{cases}$$

자원의 요구 연산 시 해당 자원이 다른 프로세스에 이미 할당된 상태이면 그 자원이 반환될 때까지 기다려야 하는데, 이와 같이 대기하는 상태에 있는 프로세스를 보류 프로세스라 하고 보류 프로세스가 아닌 프로세스를 실행 프로세스라 한다. 실행 프로세스만이 새로운 자원을 하나 더 요구하거나 자신이 보유한 자원 중의 하나를 반환할 수 있다. 보류 프로세스는 기다리는 자원이 자신에게 할

당되어야만 실행 프로세스로 전환된다. 요구한 자원이 현재 할당된 상태가 아니면 바로 할당받는다.

이 시스템을 PACSR로 표기하면 아래와 같다. 문제를 단순화하기 위해 두 종류의 자원 R_1, R_2만 존재하는 경우를 다루었다. r 개의 자원을 갖는 시스템으로의 확장은 같은 방식으로 기술 가능하다.

전체 시스템은 다음과 같이 자원 R_1과 R_2를 경합하는 n 개의 프로세스 P_i로 구성된다.

$$SYSTEM = [P_1 \,||\, \dots \,||\, P_i \,||\, \dots \,||\, P_n]_{\{R1,R2\}}$$

$$P_i = P_{i\{\}}$$
$$P_{i\{\}} = (1-\lambda) \bullet \{\} : P_{i\{\}} + \lambda/2 \bullet P_{i\{\}}^{(1)} + \lambda/2 \bullet P_{i\{\}}^{(2)}$$
$$P_{i\{\}}^{(m)} = \{(R_m,1)\} : P_{i\{m\}} + \{\} : P_{i\{\}}^{(m)}$$
$$P_{i\{m\}} = (1-\lambda-\mu) \bullet \{(R_m,1)\} : P_{i\{m\}} + \lambda \bullet P_{i\{m\}}^{(n)} + \mu \bullet \{\} : P_{i\{\}}$$
$$P_{i\{m\}}^{(n)} = \{(R_m,1), (R_n,1)\} : P_{i\{m,n\}} + \{(R_m,0)\} : P_{i\{m\}}^{(n)}$$
$$P_{i\{m,n\}} = (1-\mu) \bullet \{(R_m,1), (R_n,1)\} : P_{i\{m,n\}} + \mu/2 \bullet \{(R_n,1)\} : P_{i\{n\}} + \mu/2 \bullet \{(R_m,1)\} : P_{i\{m\}}$$

SYSTEM 기술에서 집합 A에 대해 P_{iA}는 $m \in A$인 경우 프로세스 P_i가 자원 R_m을 보유하고 있는 상태를 나타낸다. $P_{iA}^{(m)}$은 프로세스 P_i가 $m \notin A$인 자원 R_m을 요구하고 있는 상태이다.

우선 각 프로세스의 초기상태는 아무 자원도 보유하지 않은 상태 $P_{i\{\}}$에서 시작한다. $P_{i\{\}}$는 현재 자원을 전혀 보유하고 있지 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 1/2의 확률로 선택하게 된다. 위 기술에서 m=0이면 n=1이고, m=1이면 n=0이다.

$P_{i\{\}}^{(m)}$은 자원 R_m을 사용 가능하여 얻게 되면 timed action $\{(R_m,1)\}$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, 보유한 자원 R_m을 반환할 수도 있다. 그 결과 λ의 확률로 $P_{i\{m\}}^{(n)}$으로 진행하고, μ의 확률로 $P_{i\{\}}$을 수행하며, 나머지 1-λ-μ의 확률로 보유한 자원을 갖고 timed action $\{(R_m,1)\}$을 한 단위시간만큼 수행한 후 같은 과정을 반복한다.

$P_{i\{m\}}^{(n)}$은 자원 R_n도 얻어 timed action $\{(R_m,1), (R_n,1)\}$을 수행할 수도 있고, R_n을 얻을 때까지 R_m을 보유한 상태로 대기 상태에 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 timed action을 한 단위시간만큼 수행한다.

지금까지 전형적인 자원 할당 시스템을 PACSR을 이용하여 기술해 보았다. 이를 분석하여 각 프로세스 P_i가 $P_{iA}^{(m)}$ 상태에 머무는 비율을 계산하면 자원할당을 위해 대기하는 시간 비율을 계산할 수 있다. 또한 P_{iA}의 집합 A의 크기를 이 상태의 시간만큼 누적하여 평균하면 각 프로세스가 보유한 자원의 평균이 계산된다. 우리의 관심 분야인 교착상태에 대해서도 쉽게 분석 가능하다. SYSTEM이 진행하다가 임의의 서로 다른 두 프로세스 P_i와 P_j에 대해 $P_{i\{0\}}^{(1)} \,||\, P_{j\{1\}}^{(0)}$ 인 상태가 발생하면 교착상태가 된다. SYSTEM에서 가능한 서로 다른 상태들을 찾아내어 상태들 간의 전이 비율을 계산할 수도 있다[6]. 이렇게 계산된 전이 비율을 이용하면 안정상태에서 각 상태의 확률, 교착상태로 전이되는 평균비율, 교착상태가 발생하는 평균 간격, 요구연산이 보류되거나 교착상태를 유발할 확률 등을 도출해낼 수 있다.

V. 결 론

실시간 시스템을 기술하고 분석하는 도구로서 시간 개념을 도입한 프로세스 대수학이 많이 이용되어지고 있는 바, 그 중에서도 ACSR은 프로세스 동기화로 인한 시간 지연 뿐만 아니라 공유 자원 결합에 따른 시간 지연도 함께 표현할 수 있는 보다 구체적이고 현실성 있는 계산 모델이다.

본 논문에서는 ACSR에 확률 개념을 도입한 PACSR을 소개하고, 이를 이용하여 일반적인 자원 할당 시스템을 표기하여 이때 발생하는 교착상태를 비롯한 제반 기술 및 분석이 효율적으로 이루어질 수 있음을 확인하였다. 앞으로 보다 많은 분야에 PACSR을 적용, 분석 가능하도록 표현력 및 개념 확장이 계속 시도되어져야 할 것이다.

참고문헌

[1] R. Milner, Communication and Concurrency, Prentice Hall, 1989

[2] C.A.R. Hoare, "Communicating sequential Processes," CACM, 21(8), pp 666-676, Aug. 1978

[3] J.C.M. Baeten and J.A. Bergstra, "Real Time Process Algebra," Formal Aspects of Computing, 3(2), pp 142-188, 1991

[4] I. Lee, P. Bremond-Gregoire, and R. Gerber, " A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems", Proceedings of the IEEE, pp 158-171, Jan. 1994

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

[6] 성순용, "확률적 ACSR", 외대논총, 2003