論文
3-7-8

동적 시간 페트리네트 (Dynamic Timed Petri Nets)

김 영 찬* 김 탁 곤**
(Young-Chan Kim) (Tag-Gon Kim)

요 약

시간 페트리네트는 동시발생적인(concurrent) 시스템을 명세하기 위해 폭 넓게 사용되는 형식론 중의 하나이다. 하지만 시간 페트리네트는 그 형식론 내에서 명확하게 시스템의 역학(dynamics)을 표현하지 못하는 단점을 가지고 있다. 이와 같은 불완전한 형식론은 설계자로 하여금 형식론 자체뿐만 아니라 특정 문제에 의존적인 시뮬레이션 알고리즘까지 고려하게 함으로서 모델링을 매우 어렵게 한다. 이 논문에서는 시스템의 역학을 표현할 수 있는 동적 시간 페트리네트 형식론과 추상적인 시뮬레이터를 제안한다.

ABSTRACT

Timed Petri nets are one of the most important formalisms used to specify concurrent systems, However, timed Petri nets cannot represent the dynamics of systems within the formalism explicitly. Such incomplete formalisms make the modeling very difficult because modelers should know not only the formalism itself but also its problem-specific simulation algorithm. In this paper we present a formalism which can express the dynamics of systems and an abstract simulator.

1. 서론

페트리네트 (Petri Nets) [1,2]는 컴퓨터 하드웨어나 소프트웨어, 통신 프로토콜 등과 같은 다양한 시스템의 모델링에 폭 넓게 사용되어 왔다. 특히 페트리네트는 동시 발생적인 (concurrent) 시스템을 명세하고 성능을 평가하기 위해 사용되는 중요한 형식론의 하나로 인식되고 있다.

초기의 페트리네트는 시간이나 성능(performance)에 대한 고려 없이 시스템의 논리적 행위(behavior)만을 표현하기 위해서 사용되었다. 하지만, 다양한 시스템의 동적인 행위를 분석하기 위해 페트리네트 형식론을 확장하는 시도가 있어 왔다. 최초의 시간 페트리네트 (Timed Petri Nets)는 Ramchandani [3]에 의해 소개되었다.

Ramchandani는 일반적인 페트리네트의 트랜지션에 지연시간을 추가하는 방식을 사용하였다. 이 후로 많은 연구자들이 시간 페트리네트에 관한 연구를 발표하였다 [3-6].

하지만 이들의 시간 페트리네트는 형식론 내에서 시스템의 역학(dynamics)을 명시적으로 표현할 수 없었다. 즉, 시간 페트리네트 형식론을 가지고 시스템의 동적인 부분까지 정확하게 분석하기 위해서 설계자가 시간 페트리네트 형식론뿐만 아니라 특정한 시뮬레이션 알고리즘까지 고려하여야 하는 문제점이 있었다. 이와 같은 불완전한 형식론은 모델링을 매우 어렵게 하기 때문에 모델링 형식론과 그것의 시뮬레이션 알고리즘을 명확하게 확립하는 것이 중요

* 정회원 : 한밭대학교 정보통신-컴퓨터공학부 조교수
** 정회원 : 한국과학기술원 전자전산학부 교수

논문접수 : 2002. 6. 17.
심사완료 : 2002. 7. 5.

하다. 또한 모델링 형식론과 시뮬레이션 알고리즘을 혼합하기보다 명확하게 분리하면 다음과 같은 장점들이 있게 된다. 첫째, 모델들은 모델링 형식론만을 사용해서 개발될 수 있다. 둘째, 미리 개발된 모델의 재사용성이 증가한다. 셋째, 여러 가지 다양한 시뮬레이터 중에서 가장 좋은 것을 선택할 수 있다.

한편 전체 상태 집합, 확장된 상태전이 함수, 동적입사행렬들의 도입으로 동적 표현력(dynamic expressibility)을 가지도록 확장한 시간 페트리네트 형식론이 [11]에서 제안되었다. 하지만 이 형식론은 페트리네트의 동시성의 일부를 표현하지 못하는 문제점을 가지고 있다. 예를 들면, 두개의 트랜지션이 동시에 enable되고 충돌(conflict)이 일어나지 않은 상태에서 한 트랜지션의 fire가 다른 트랜지션이 enable되어 지연되는 시간을 처음부터 다시 시작하는 잘못된 영향을 미치게 된다.

본 논문에서는 시스템의 동적 행위를 표현할 수 있는 동적 시간 페트리네트 형식론을 제시하고, 또한 이 형식론과 관련되지만, 형식론과 혼합되지 않고 분리된 시뮬레이터를 제시하고자 한다. 이러한 연구는 이산사건 시스템을 모델링하기 편리한 DEVS 형식론[10]의 동적 표현능력과 같은 좋은 점에 많은 영향을 받았다.

이 논문의 구조는 다음과 같다. 2절에서는 이 논문을 이해하기 위해 필요한 기본적인 정의와 결과들을 설명한다. 3절에서는 동적 시간 페트리네트 형식론을 제시한다. 4절에서는 동적 페트리네트의 시뮬레이션 알고리즘과 추상 시뮬레이터를 제시한다. 5절에서는 예제를 통해 동적 페트리네트의 명세와 시뮬레이션의 궤적을 보여준다. 6절에서 이 논문의 결론을 제시한다.

2. 관련 연구

본 논문에서 N은 음수가 아닌 정수를, R은 음수가 아닌 실수를, n는 페트리네트의 플레이스의 개수를($n=|P|$), m는 트랜지션의 개수를($m=|T|$)를 나타내기 위해서 사용된다.

2.1 시간 페트리네트 (Timed Petri Nets)

페트리네트 이론은 동시 발생적인 시스템을 명세하기 위해 사용되는 잘 알려진 형식론중의 하나이다. 이 절에서 페트리네트 이론의 기본적인 개념을 간단히 살펴본다. 자세한 내용은 참고문헌을 참조하라.

페트리네트

이 논문에서는 [2]에서 정의된 페트리네트 형식론에 약간의 수정을 한 것을 채용한다.

페트리네트는 다음의 구조를 갖는다.

$PN=\langle P, T, I, O\rangle$

P 플레이스(place)의 집합
T 트랜지션(transition)의 집합
$I: T\rightarrow P^{\infty}$ 입력함수
$O: T\rightarrow P^{\infty}$ 출력함수

I와 O는 트랜지션 T로 부터 장소의 가방(bag 혹은 multiset) P^{∞}로의 매핑(mapping)이다.

마킹(marking) $\mu: P\rightarrow N$은 플레이스로부터 자연수로의 매핑이다. 마킹은 플레이스에 토큰(token)의 할당하는 것을 의미한다. 일반적으로 μ는 n-벡터로 표현된다.

$$\mu=(\mu_1, \mu_2, \cdots, \mu_n)$$

여기서, $|P|=n$이고 $\mu_i=\mu(p_i)$는 플레이스 p_i에 있는 토큰의 개수를 나타낸다. 마크된 페트리네트(marked Petri net)는 페트리네트 PN과 마킹 μ의 튜플 즉 (PN, μ)로 정의된다.

페트리네트의 실행은 그 페트리네트가 가지는 토큰의 수와 분포에 의해 제어된다. 페트리네트는 트랜지션을 fire하여 실행된다. 트랜지션의 fire는 그 트랜지션의 입력 플레이스에서 토큰을 제거하고 출력 플레이스에 토큰을 추가하는 과정을 거친다. 임의의 트랜지션 t_j가 fire할 수 있으려면 t_j는 반드시 enable되어 있어야 한다. t_j는 다음의 조건을 만족할 때 enable되었다고 정의된다:

$$\mu(p_i) \geq \#(p_i, I(t_j)) \quad \text{for all } p_i \in P$$

위의 조건을 만족하지 않으면 t_j는 disable되었다고 말한다.

마킹 μ에서 enable된 모든 트랜지션의 집합을 $E(\mu)$로 표현한다.

$$TPN_1=(P, T, I, O, \mu, \theta)$$

$$\begin{aligned} P &= \{p_1, p_2, p_3, p_4, p_5, p_6\} \\ T &= \{t_1, t_2, t_3, t_4, t_5\} \\ \mu &= (2,1,2,0,1,0) \\ \theta &= (1,3,2,0,1) \end{aligned}$$

$$\begin{aligned} I(t_1) &= \{p_2{:}1, p_3{:}1, p_6{:}1\} & O(t_1) &= \{p_1{:}1\} \\ I(t_2) &= \{p_6{:}1\} & O(t_2) &= \{p_5{:}1\} \\ I(t_3) &= \{p_2{:}1, p_3{:}1, p_5{:}1\} & O(t_3) &= \{p_4{:}2\} \\ I(t_4) &= \{p_4{:}1\} & O(t_4) &= \{p_3{:}1\} \\ I(t_5) &= \{p_1{:}2\} & O(t_5) &= \{p_6{:}1\} \end{aligned}$$

[그림 1] 시간 페트리네트: TPN_1

[Fig. 1] A timed Petri net: TPN_1

페트리네트의 상태공간

마크된 페트리네트 (PN, μ)이 주어졌을 때, 그 네트의 상태는 마킹함수 μ에 의해 정의된다. n-플레이스 페트리네트의 상태공간은 모든 가능한 마킹의 집합이다. 상태 변화는 트랜지션의 firing의 의해 기술된다. 상태전이함수 δ는 다음과 같이 정의된다: 만약 $\delta(\mu, t_j)=\mu'$이라면,

$$\mu'(p_i)=\mu(p_i)-\#(p_i, I(t_j))+\#(p_i, O(t_j)) \quad \text{for each } p_i \in P$$

초기 마킹 μ로 부터 하나 혹은 여러 개의 트랜지션을 fire하여 도달할 수 있는(reachable) 모든 마킹 집합을 $R(\mu)$로 나타낸다.

시간 페트리네트 (Timed Petri Nets)

시간 페트리네트는 다음과 같이 정의된다.

$$TPN=\langle PN, \mu, \theta \rangle$$

여기서 $\langle PN, \mu \rangle$는 마크된 페트리네트이고, θ: $T \rightarrow R$는 트랜지션의 집합으로부터 음수가 아닌 실수로의 매핑이다.

$$\theta(t_j)=\theta_j$$

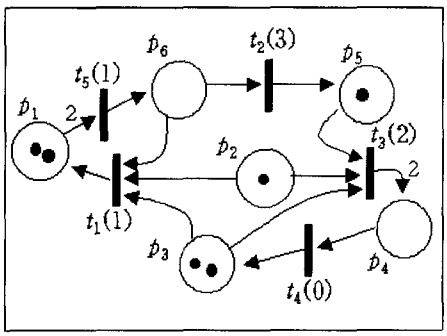
여기서 θ_j는 트랜지션 t_j가 가지는 지연시간이다. 시간 페트리네트에서 θ_j는 다음의 의미를 가진다.

트랜지션 t_j가 enable될 때, 그것은 즉시 fire하지 않고 θ_j에 의해 주어진 지연시간 만큼 지연된다. 이 지연동안 토큰은 t_j의 입력 플레이스에 남아 있게 된다.

이 논문에서 사용하는 이러한 개념은 [3]의 개념과 동일하다.

[그림 1]은 시간 페트리네트 TPN_1의 명세 예를 보여주고 있다.

[그림 2]는 [그림 1]과 동등한 명세방법인 그림 형태로 표현된 TPN_1을 보여주고 있다.



[그림 2] TPN_1의 그림형태

[Fig. 2] Graphical representation of TPN_1

이 시간 페트리네트는 6개의 플레이스와 4개의 트랜지션을 가진다. 트랜지션 위의 레이블 $t_j(d)$는 트랜지션 t_j가 $\theta_j=d$을 가지는 것을 나타낸다. 예를 들면, $t_5(1)$는 트랜지션 t_5는 지연시간 θ_5으로 1을 가진다는 것을 나타낸다. 모서리 (u, v) 위의 정수 k는 그 모서리의 무게(weight) 즉 u에서 v로 k개의 병렬 모서리가 존재하는 것을 나타낸다.

예를 들면, (p_1, t_5)는 무게로 2를 가지는 것은 p_1에서 t_5로 두 개의 모서리가 있는 것을 나타낸다.

시간 페트리네트의 상태공간

시간 페트리네트의 역학(dynamics)은 비 시간적 페트리네트의 역학보다 좀 더 복잡하다. 시간 페트리네트의 상태는 마킹함수 μ와 시간 구조 θ로부터 유도되는 정보로 구성된다. $n=|P|$이고 $m=|T|$이라고 할 때, 상태공간은 $N^m \times R^m$이 된다. 상태전이함수를 정의하기 위해서는 다음의 정의가 필요하다.

1. 남은시간함수 τ는 다음과 같이 정의된다:

   $\tau: R(\mu) \times T \rightarrow R$

   이 함수는 어떤 마킹에서 enable된 각 트랜지션의 fire까지 남은 시간을 의미한다.
2. DTPN의 상태집합 S는 다음과 같다.

   $S=\{(\mu', \tau') \mid (\mu', \tau') \in R(\mu, \tau)\}$

   즉 초기상태(μ, τ)에서 하나 혹은 여러 개의 트랜지션을 fire하여 도달할 수 있는(reachable) 모든 상태 집합$R(\mu, \tau)$이다.

   상태전이함수 $\delta: S \times T \rightarrow S$는 다음과 같이 정의된다: $\delta((\mu, \tau), t_j)=(\mu', \tau')$이라고 가정하면, 마킹의 전이는 일반적인 페트리네트와 동일하고,

   $$\mu'(p_i)=\mu(p_i)-\#(p_i, I(t_j))+\#(p_i, O(t_j)) \quad \text{for each } p_i \in P$$

   남은시간함수의 전이는 다음과 같이 정의된다:

   $$\tau'(t_j)=\begin{cases}\theta_j & \text{if } t_j \in E(\mu') \\ \infty & \text{otherwise}\end{cases} \quad \text{for each } t_j \in T$$

   다시 말해 현재 상태에서 남은시간함수 값이 가장 작은 트랜지션이 선택되어 fire하면 새로운 마킹 μ'이 구해지고, μ'에서 enable된 각 트랜지션의 남은시간함수 값은 $\tau'(t_j)=\theta_j$이 되고, disable된 각 트랜지션의 남은시간함수 값은 $\tau'(t_j)=\infty$가 된다.

   이러한 남은시간함수의 전이 방식은 현재 상태에서 어느 정도 머무르고 있는지, 앞으로 얼마나 시간이 경과해야 상태전이를 할 것인가와 같은 시스템의 동적 속성을 명시할 수 없는 단점이 있다.

3. 동적 시간 페트리네트

본 논문에서 저자들은 동적 표현능력을 가지는 동적 시간 페트리네트 (Dynamic Timed Petri Nets)의 형식론을 다음과 같이 제안한다:

$DTPN=\langle P, T, I, O, \mu, \theta, \text{select} \rangle$

P	플레이스(place)의 집합
T	트랜지션(transition)의 집합
$I: T \rightarrow P^\infty$	입력함수
$O: T \rightarrow P^\infty$	출력함수
$\mu: P \rightarrow N$	마킹함수(marking function)
$\theta: T \rightarrow R$	지연함수(duration function)
$\text{select}: 2^T \rightarrow T$	선택함수

위의 정의는 다음의 제한조건 하에서 정의된다:

1. 남은시간함수 τ는 다음과 같이 정의된다:

   $\tau: R(\mu) \times T \rightarrow R$

   이 함수는 현재의 마킹에서 enable된 각 트랜지션의 fire까지 남은 시간을 의미한다.
2. DTPN의 상태집합S는 다음과 같다.

   $S=\{(\mu', \tau') \mid (\mu', \tau') \in R(\mu, \tau)\}$

   즉 초기상태(μ, τ)에서 하나 혹은 여러 개의 트랜지션을 fire하여 도달할 수 있는(reachable) 모든 상태 집합$R(\mu, \tau)$이다.
3. DTPN의 전역 상태집합Q는 다음과 같다.

   $$Q=\{(\mu', \tau', e) \mid (\mu', \tau') \in R(\mu, \tau), \; 0 \le e \le \tau_{\min}(\mu', \tau')\}$$

   여기서 e는 현 상태 (μ', τ')에서 머문 시간을 나타내고, $\tau_{\min}(\mu', \tau')$는 현 상태 (μ', τ')에서 enable된 트랜지션들의 남은 시간함수 값 중 가장 작은 값이다.

   $\tau_{\min}(\mu', \tau')=\min\{\tau'(t_j) \mid t_j \in E(\mu')\}$

여기서 P, T, I, O, μ, θ는 시간 페트리네트의 구성요소와 같다. 사실 선택함수 select가 추가된 점

외에는 시간 페트리네트의 구조와 동일하다. 하지만, 형식론의 구조는 동일할지라도 제한조건이 시스템 동적 속성을 표현할 수 있도록 확장되어 있는 것을 알 수 있다.

남은시간함수 τ는 마킹에 따라 트랜지션에 음수가 아닌 실수를 연관시킨다. 이것은 현재의 마킹에서 enable된 각 트랜지션의 fire까지 남은 시간을 의미한다. 남은시간함수 τ는 보통의 시간 페트리네트의 남은시간함수와 매우 유사해 보이지만 동적표현 능력을 갖기 위해 다음과 같은 다른 의미를 가진다. 만약 t_j가 마킹 μ에서 막 enable되었다면 $\tau(\mu, t_j)$는 θ_j를 가지고, 만약 t_j가 막 enable된 후 e시간이 흘렀다면 $\tau(\mu, t_j)$는 $\theta_j - e$를, 만약 t_j가 막 disable되었다면, $\tau(\mu, t_j)$는 ∞를 가진다.

선택함수 select는 두개 이상의 트랜지션들이 같은 남은시간함수 값을 가질 때 하나를 선택하기 위해 사용된다. 일반적으로 페트리네트 학계에서는 enable된 여러 트랜지션들 중에서 무작위(random)로 하나를 선택하여 fire하는 시뮬레이션 규칙을 사용하는 것이 관례이다. 이 관례를 따른다면 select는 무작위 선택함수가 된다. 따라서 대부분의 다른 페트리네트 형식론에서는 명시적으로 select함수를 명세하지 않는다. 하지만 저자들은 필요하다면 정확하게 시뮬레이션 궤적(trajectory)을 제어할 수 있는 점이 동적 시뮬레이션에서 매우 중요하고 생각한다. 따라서 저자들은 동적 시간 페트리네트 형식론에 select함수를 명시하도록 하였다.

이와 같이 제시된 동적 시간 페트리네트의 형식론은 시스템의 동적 속성 (dynamic characteristics)을 명시할 수 있게 한다. 즉 전역상태 $(\mu, \tau, e) \in Q$을 통해 현 상태에서 머무르고 있는 시간e를 알 수 있고 또한 $\tau_{\min}(\mu', \tau') - e$시간이 경과하면 상태전이가 일어나는 것을 알 수 있게 되었다.

동적 시간 페트리네트의 상태공간

시간 페트리네트의 상태는 마킹함수 μ와 남은시간함수 τ로 구성된다. $n = |P|$이고 $m = |T|$이라고 할 때, 상태공간은 $N^m \times R^m$이 된다.

상태전이함수 $\delta: Q \times T \to S$는 다음과 같이 정의된다: $\delta((\mu, \tau, e), t_f) = (\mu', \tau')$이라고 가정하면, 마킹의 전이는 일반적인 페트리네트와 동일하고,

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_f)) + \#(p_i, O(t_f)) \quad \text{for each } p_i \in P$$

남은시간함수는 다음과 같이 정의된다:

$$\tau'(t_j) = \begin{cases} \theta_j & \text{if } t_j \in E(\mu') \wedge (t_j = t_f \vee \tau(t_j) = \infty) \\ \tau(t_j) - \tau(t_f) & \text{if } t_j \in (E(\mu') - \{t_f\}) \wedge \tau(t_j) \neq \infty \\ \infty & \text{otherwise} \end{cases} \quad \text{for each } t_j \in T$$

정성적으로 설명을 하면, 남은시간함수는 신구 마킹에서의 각 트랜지션의 enable/disable에 따라서 다음과 같이 정의된다.

- 현재 fire된 트랜지션 t_f가 다음의 새로운 마킹 μ'에서 다시 enable되면 t_f의 남은 시간 $\tau'(t_f)$는 θ_f로 초기화되며,
- μ에서는 disable되었지만 μ'에서 enable된 각 트랜지션 t_j의 남은 시간 $\tau'(t_j)$는 θ_j을 갖게 되며,
- μ와 μ'에서 계속 enable된 각 트랜지션 t_j의 남은 시간 $\tau'(t_j)$는 $\tau(t_j) - \tau(t_f)$을 갖게 되며,
- 나머지 경우 즉 μ'에서 disable된 각 트랜지션 t_j의 남은 시간 $\tau'(t_j)$는 ∞가 된다.

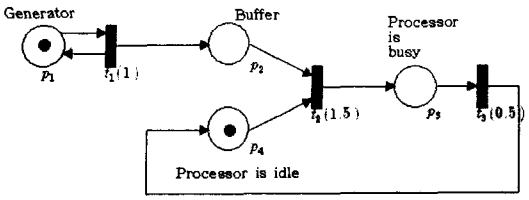
$DTPN_1 = (P, T, I, O, \mu, \theta, \text{select})$

$P = \{p_1, p_2, p_3, p_4\}$
$T = \{t_1, t_2, t_3\}$
$\mu = (1, 0, 0, 1)$
$\theta = (1, 0.5, 1.5)$
select = select_{typ}

$I(t_1) = \{p_1:1\}$ $O(t_1) = \{p_1:1, p_2:1\}$
$I(t_2) = \{p_2:1, p_4:1\}$ $O(t_2) = \{p_3:1\}$
$I(t_3) = \{p_3:1\}$ $O(t_3) = \{p_4:1\}$

[그림 3] 프로세서-버퍼 시스템에 대응하는 DTPN 모델 : $DTPN_1$

[Fig. 3] A DTPN model of processor-buffer system, $DTPN_1$



[그림 4] $DTPN_1$ 그림형태

[Fig. 4] Graphical representation of $DTPN_1$

[그림 3]과 [그림 4]는 프로세서-버퍼 시스템에 대응하는 동적 시간 페트리네트의 예를 보여 주고 있다. 그 동적 시간 페트리네트는 4개의 플레이스와 3개의 트랜지션을 가진다.

시간 페트리네트처럼 트랜지션 위의 레이블 $t_j(d)$는 트랜지션 t_j가 $\theta_j = d$을 가지는 것을 나타낸다. 예를 들면, $t_2(1.5)$는 트랜지션 t_2는 지연시간 θ_2으로 1.5을 가진다는 것을 나타낸다.

4. 추상 시뮬레이터

DTPN 역학(dynamics)은 여기에 제시되는 추상 시뮬레이터를 사용하여 해석되어 질수 있다. 추상 시뮬레이터는 세 개의 상태변수를 가진다.

(1) μ DTPN의 마킹함수에 대응하는 변수

(2) τ DTPN의 남은시간함수에 대응하는 변수

(3) t_L 현 시뮬레이션 시간에 대해 마지막으로 트랜지션이 fire 했을 때의 시간을 저장하는 변수

추상 시뮬레이터는 위의 상태변수로부터 유도될 수 있는 두개의 변수를 사용한다.

(4) t_N 현재 enable된 트랜지션들 중에서 다음에 fire할 수 가장 빠른 시간을 저장하기 위한 변수

(5) t_f 현 상태에서 enable된 트랜지션 중에서 fire하도록 선택된 트랜지션을 저장하기 위한 변수

위의 변수를 사용하여 추상 시뮬레이터는 enable된 트랜지션이 존재하는 동안, 다음의 프로세스를 반복 실행한다.

단계 1: 현 상태(μ, τ)에서 enable된 트랜지션의 집합$E(\mu)$를 구한다. 다음에 fire할 수 가장 빠른 시간 t_N을 다음과 같이 구한다:

$$t_N = t_L + \tau_{\min}(\mu, \tau)$$

단계 2: 먼저 스케줄된 시간이 $t_N - t_L$인 트랜지션들을 구한다.

$$T_{t_N - t_L} = \{t_j \mid t_j \in E(\mu) \wedge \tau(t_j) = t_N - t_L\}$$

그 다음 select를 적용하여 다음에 fire할 트랜지션 t_f을 선택한다:

$$t_f = \text{select}(T_{t_N - t_L})$$

```
t_L = t_N
variable: μ, τ, t_L, t_N, t_f;
while E(μ) ≠ ∅
  t_N := t_L + τ_min(μ, τ);
  t_f := select({t_j | t_j ∈ E(μ) ∧ τ(t_j) = t_N − t_L});
  (μ, τ) := δ(μ, τ, t_f);
  t_L := t_N;
end while
```

[그림 5] 동적 시간 페트리네트의 추상 시뮬레이터

[Fig. 5] Abstract simulator of dynamic timed Petri nets

단계 3: 선택된 t_f을 가지고 다음 상태를 결정한다.

$$(\mu', \tau') = \delta(\mu, \tau, t_f)$$

단계 4: 현 시뮬레이션 시간을 사용하여 마지막으로 트랜지션이 fire된 시간을 나타내는 변수 t_L를 변경한다.

[그림 5]에서 추상 시뮬레이터를 Pascal과 유사한 언어를 사용하여 구현한 것을 보여주고 있다.

만약 전역 시뮬레이션 시간 t가 주어지면, 추상 시뮬레이터는 마지막으로 트랜지션이 fire된 후 경과된 시간 e를 다음과 같이 계산할 수 있고,

$$e = t - t_L$$

다음에 스케줄된 트랜지션의 fire까지 남은 시간 r은 다음과 같이 계산할 수 있고,

$$r = t_N - t = \tau_{\min}(\mu, \tau) - e$$

또한 전역 상태 Q는 다음과 같이 계산할 수 있다.

$$Q = (\mu, \tau, e)$$

5. 예제

[그림 4]의 $DTPN_1$를 사용하여 상태전이를 살펴보면 다음과 같다.

(1) 초기상태 (μ, τ)는 초기 마킹 (1,0,0,1)에서 enable된 트랜지션 집합이 $\{t_1\}$이므로 남은 시간 함수 값으로 t_1만 θ_1을 가지고 나머지 트랜지션들은 ∞을 갖는다. 따라서, 초기상태 (μ, τ)는 다음과 같다.

$$(\mu, \tau) = ((1,0,0,1), (1, \infty, \infty))$$

(2) 시뮬레이션 시간이 1 만큼 진행된 후, t_1이 fire하게 된다.

(3) 새로운 마킹은 (1,1,0,1)이 되고, 이 때의 enable된 트랜지션집합은 $\{t_1, t_2\}$이 된다. t_1이 fire된 후 다시 enable되므로 $\tau_1 = \theta_1 = 1$이 되고, t_2는 새롭게 enable되므로 $\tau_2 = \theta_2 = 1.5$가 된다. t_3는 enable되지 않으므로 $\tau_3 = \infty$가 된다. 따라서 새로운 남은 시간함수는 (1,1.5,∞)로 된다. 따라서 새로운 상태 (μ, τ)는 다음과 같다:

$$(\mu, \tau) = ((1,1,0,1), (1, 1.5, \infty))$$

(4) t_1는 fire까지 남은 시간이 1이고 t_2는 fire까지 남은 시간이 1.5이기 때문에 1 만큼 시뮬레이션 시간이 지난 후에 t_1이 fire하게 된다.

(5) 새로운 마킹은 (1,2,0,1)이 되고, enable된 트랜지션 집합은 $\{t_1, t_2\}$가 된다. t_1이 fire된 후 다시 enable 되므로 $\tau_1 = \theta_1 = 1$이 되고, t_2는 새로운 상태에서도 enable 되어 있으므로 $\tau_2 = 1.5 - 1 = 0.5$가 된다. t_3는 enable되지 않으므로 $\tau_3 = \infty$가 된다.

$$(\mu, \tau) = ((1,2,0,1), (1, 0.5, \infty))$$

(4) t_1는 fire까지 남은 시간이 1이고 t_2는 fire까지 남은 시간이 0.5이기 때문에 0.5시간 단위만큼 시뮬레이션 시간이 지난 후에 t_2이 fire하게 된다.

(5) 새로운 마킹은 (1,1,1,0)이 되고, enable된 트랜지션 집합은 $\{t_1, t_3\}$가 된다. 남은 시간 함수 τ에 대해 설명하면 다음과 같다.

t_1이 현 상태에서도 다시 enable되므로 $\tau_1 = 1 - 0.5 = 0.5$이 되고, t_2는 새로운 상태에서 disable되므로 $\tau_2 = \infty$가 된다. t_3는 enable되므로 $\tau_3 = \theta_3 = 0.5$가 된다.

$$(\mu, \tau) = ((1,1,1,0), (0.5, \infty, 0.5))$$

(6) t_1과 t_3의 fire까지 남은 시간이 0.5이다. 동시에 두개 이상의 이벤트가 일어날 수 없기 때문에 select를 사용하여 t_1과 t_3 둘 중에

하나를 선택하여야 한다. 만약 t_1이 리턴(return) 되었다면,

$$\text{select}(\{t_1, t_3\}) = t_1$$

0.5시간 단위만큼 시뮬레이션 시간이 지난 후에 t_1이 fire하게 된다.

(7) 새로운 마킹은 ((1,2,1,0),(1,∞,0)가 되고, enable된 트랜지션 집합은 $\{t_1, t_3\}$가 된다. 남은 시간 함수 τ에 대해 설명하면 다음과 같다. t_1이 fire된 후 다시 enable되므로 $\tau_1 = \theta_1 = 1$이 되고, t_2는 새로운 상태에서도 disable되어 있으므로 $\tau_2 = \infty$가 된다. t_3는 계속 enable되어 있으므로

$\tau_3 = 0.5 - 0.5 = 0$ 가 된다.

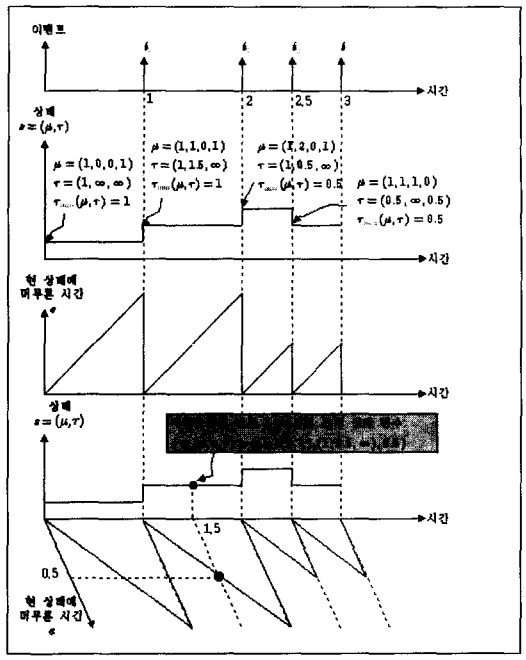
$$(\mu, \tau) = ((1,2,1,0),(1,\infty,0))$$

(8) t_1는 fire까지 남은 시간이 1이고 t_3는 fire까지 남은 시간이 0이기 때문에 0시간 단위만큼 시뮬레이션 시간이 지난 후에 t_3이 fire하게 된다. (시뮬레이션 시간은 변함이 없지만 시뮬레이션 루프가 한번 실행되는 상태임)

(9) 새로운 마킹은 (1,2,0,1)이 되고, enable된 트랜지션 집합은 $\{t_1, t_2\}$가 된다. t_1이 현 상태에서도 다시 enable되므로 $\tau_1 = 1 - 0 = 1$이 되고, t_2는 새로운 상태에서 enable되므로 $\tau_2 = \theta_2 = 1.5$가 된다. t_3는 disable되므로 $\tau_3 = \infty$가 된다.

$$(\mu, \tau) = ((1,2,0,1),(1,1.5,\infty))$$

(10) 위의 과정과 유사하게 시뮬레이션은 계속 진행된다. 다만, 시뮬레이션의 종료는 더 이상 enable된 트랜지션이 없거나 주어진 시뮬레이션 시간이 도달되었을 경우이다. 시뮬레이션 종료시간을 받아들이는 것은 시뮬레이터 구현의 관련된 내용이고 형식론에서는 중요하지 않으므로 설명을 생략한다.



[그림 6] $DTPN_1$의 전체 상태 궤적 (global state trajectory)의 일부

[Fig. 6] Global state trajectory of $DTPN_1$

위의 시뮬레이션에 대응하는 상태 궤적도가 [그림 6]에 보여지고 있다.

6. 결론

기존의 시간 페트리네트는 형식론 내에서 시스템의 역학(dynamics)을 명시적으로 표현할 수 없었다. 이런 문제를 해결하기 위해 본 논문에서는 시스템의 역학을 잘 표현할 수 있는 동적 시간 페트리네트 형식론을 제시하고, 또한 이 형식론과 관련되지만, 형식론과 혼합되지 않고 분리된 시뮬레이터를 제시하였다. 또한 예를 통해 제안한 형식론과 추상적 시뮬레이터가 잘 동작함을 보였다.

※ 참고문헌

[1] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceeding of the IEEE*, Vol 77, No. 4, pp. 541-580, Apr. 1989.

[2] J. L. Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[3] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*, Phd Thesis, MIT, 1973

[4] P. M. Merlin and D. J. Farber, "Recoverability of Communication Protocols - Implications of a Theoretical Study," *IEEE Tran. on Communcations*, Vol COM-24, pp. 1671-1680, Nov. 1976.

[5] J. Sifakis, "Structural Properties of Petri Nets," in *Mathematical Foundations of Computer Science*, 1978, J. Winkowski, Ed. pp. 474-483. Springer-Verlag, 1978.

[6] W. M. Zuberek, "Timed Petri Nets and Preliminary Performance Evaluation," in *Proceedings of the 7th Annual Symposium on Architecture*, 1980, 00. 88-96.

[7] T. Agerwala, "Putting Petri Nets to Work," *IEEE Computer Magazine*, Vol 12, No. 12, pp. 85-94, 1979.

[8] B. Prabhakaran and S. V. Raghavan, "Synchronization Models for Multimedia Presentation with User Participation," *Multimedia Systems*, Vol 2. No. 2, pp. 53-62, Aug. 1994

[9] J. Carlier, Ph. Chretienne, and C. Girault, "Modelling Scheduling Problems with Timed Petri Nets,", in *Advances in Petri Nets 1984*, G. Rozenberg, Ed., Vol. 62-82, Springer-Verlag, 1985.

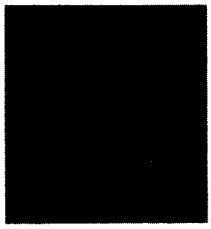
[10] B. P. Zeigler, Theory of *Modelling and Simulation*, John Wiley & Sons, New York, 1976.

[11] 손진곤, 황종선, 백두권, "동적 표현능력을 갖는 Timed Petri Net 형식론", 한국정보과학회 논문지, Vol 19, No. 1, 1992.

김 영 찬

1985: 아주대학교 공과대학 전자공학과 학사
1987: 한국과학기술원 전기 및 전자공학과 석사
1995: 한국과학기술원 전기 및 전자공학과 박사
1985 - 1990: 삼성전자 종합연구소 연구원
1995 - 1995: 한국과학기술원 위촉연구원
1997 - 1998: 시스템공학연구소 및 전자통신연구원 연구원
1998 - 현재: 한밭대학교 정보통신-컴퓨터공학부 조교수
관심분야: 시스템 검증, 형식론(Petri Net, DEVS등), 데이타베이스

김 탁 곤



1988 : 아리조나대학교 전자/컴퓨터공학과 공학박사
1980 - 1983: 국립수산대학(현: 부경대학교) 통신공학과 전임강사
1987 - 1989: 아리조나 환경연구소 연구 엔지니어
1989 - 1991: 캔사스 대학교, 전자전산학과, 조교수
1991 - 현재: 한국과학기술원, 전자전산학과, 조교수/부교수/교수
2001 - 현재: 미국 시뮬레이션 기술사
2000 - 현재: SIMULATION: Trans of SCS 편집위원장
모델링/시뮬레이션 방법론 및 환경 개발, 시뮬레이션에 기반한 시스템 분석등에 관하여 국제적으로 활발한 연구 활동을 하고 있으며 국제학술지/국제학술발표대회에 100 편 이상의 논문을 개재 하였음. 시뮬레이션 모델링에 관련된 다수의 국제학술지 편집을 담당하고 있다. 저서(공저자: B.P. Zeigler, H. Praehofer)로 2000 년 미국 Academic Press에서 발간한 모델링 시뮬레이션 전문 교재인 Theory of Modeling and Simulation(2nd edition) 이 있음. 한국시뮬레이션 학회 초대 편집위원장을 역임하였고 현재 학술부회장임. IEEE(국제 전기전자 학회) 및 SCS(국제 시뮬레이션 학회)의 Senior Member 이며 ACM(미국전산학회) 및 Eta Kappa Nu 의 Regular Member 임.