멀티미디어 객체 동기화를 위한 확장된 프로세스 대수

(An Extended Process Algebra for Synchronization of Multimedia Objects)

김 기 병 † 김 형 주 ††
(Ki-Byoung Kim) (Hyoung-Joo Kim)

요 약 기존의 멀티미디어 동기화 메카니즘은 각 미디어 별 데이타를 정적 객체(passive object)로 모델링한 후 이들 간의 동기화를 표현하였다. 그러나 멀티미디어 데이타를 객체로 표현하게 되면 각 객체는 동적인 성질을 가지는 경우가 많다. 본 논문에서는 기존의 멀티미디어 동기화 메카니즘을 확장하여 동적 객체(active object)를 지원할 수 있도록 확장하였고 이 경우 생길 수 있는 동적 객체의 동기화를 제시하였다. 또한 이를 프로세스 대수를 이용하여 표현하였는데, 이를 위해 동적인 프로세스를 표현하기 위한 Milner의 π-calculus에 기반하여, 이를 시간의 추이에 의한 동기화를 표현할 수 있도록 시간적 요소를 추가하여 확장하였고, 확장이 기존의 π-calculus에 포함됨을 보였다. 또한 기존의 Petri-net 방법에 비해 확장된 π-calculus의 장점으로 일반화의 용이및 행위에 대한 등가성(bisimulation)을 제시하였다.

Abstract In this paper, we propose a synchronization scheme for dynamic multimedia objects which were modeled as passive objects formerly. We represent multimedia synchronizations using an extended process algebra of Milner's π-calculus which is originally proposed for representing dynamic processes. We add timing concept into π-calculus for expressing synchronization over the time. We show the useful properties of our scheme to Petri-net such as generalization, bisimulation on behavior, and prove that our scheme is conservative extension of π-calculus.

1. 서 론

컴퓨터 하드웨어와 주변기기의 급속한 발전은 컴퓨터를 이용한 정보처리의 한계를 넓혀가고 있으며 기존의 문자 위주의 정보처리 방식은 점차로 기존의 미디어와 더불어 오디오 및 비디오와 같은 새로운 미디어를 지원하는 고도의 정보 서비스를 실현 가능하게 하고 있다. 멀티미디어를 이용한 정보 처리에서는 서로 상이한 미디어들은 조작이나 전송 시에 서로 다른 특성을 가지게 되므로 그 처리는 미디어 별로 독자적으로 수행된다. 멀티미디어 데이타를 조작하고자 할 때 특별히 요구되는 기능중에는 동기화(synchronization)기능이 있다.

† 종신회원 : 서울대학교 컴퓨터공학과
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
논문접수 : 1994년 5월 3일
심사완료 : 1995년 3월 27일

이는 복수의 미디어 사이의 시간적 또는 공간적 상호관계를 데이타의 조작 전후에 있어서 잘 유지시키기 위한 기능이며 현재까지 많은 연구 대상이 되고 있다. [1][2][3][4][7][8][9][14][18]

각각의 미디어 데이타는 독립적으로 존재하지만 멀티미디어 데이타를 표현하기 위해서 표현 단계에서는 서로 다른 미디어의 데이타들에 공존하거나 통합되기도 하며 또한 이러한 데이타들은 동기화를 위해 상호 작용을 하게 된다. [18]

기존의 멀티미디어 객체에 대한 동기화 연구는 임의의 네트워크 및 분산환경에서의 동기화 문제와 소프트웨어 시스템에서의 동기화 표현을 위한 연구로 나눌 수 있다. 특히 DBMS의 질의는 멀티미디어 동기화를 질의에 이용할 수 있으므로 유용한 응용 중의 하나이며, 동기화의 명세에 관한 연구는 멀티미디어의 응용분야가 넓어지면서 그 중요성이 점차로 부각되고 있다. [3][14]

[15]

이를 명세(specification)하기 위한 연구로는 주로 그래프 방법에 기반한 Petri-net, TPN(Timed Petri-net), OCPN(Object Composition Petri-net)등의 연구가 있다.[9]

그래프 표현 방법에서는 일단 객체가 한번 정해지면 수행 도중에는 객체의 상태의 변화가 없는 정적 객체(passive object)[1]를 가정하여 제안되었다. 그러나 객체지향 개념을 이용한 멀티미디어 객체들은 동적인 성질을 가지는 경우가 많다. 그러므로 다양한 미디어를 표현하는 객체는 객체의 특성에 따라 정적 객체(passive object)와 동적 객체(active object)로 분류할 수 있다. 비디오를 통한 비디오 스트림이나 오디오를 통한 오디오 스트림은 구성 요소가 소리와 화상 신호 등이 복합되어 질 수 있으나, 그 형태가 한번 정해지면 바뀌지 않는 정적인 객체로 볼 수 있으며, 멀티미디어 객체 관리자(object manager)나 멀티미디어 DBMS, 저작 도구 등을 이용하여 저작된 객체들은 여러가지 단일 미디어들이 논리적으로 통합된 멀티미디어 객체이며 그 내부나 상태는 능동적으로 변화될 수 있다.

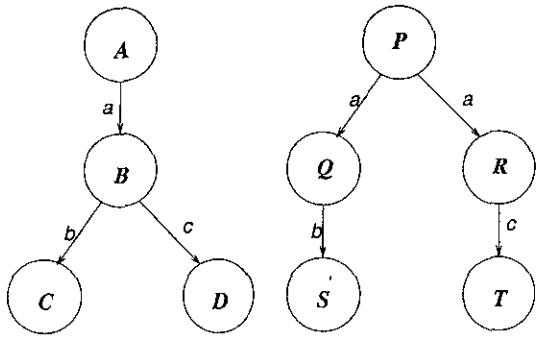

그림 1 그래프의 등가성(equivalence) 문제

멀티미디어 DBMS에서는 질의 처리를 위해 이들 객체들의 동일성(equivalence)을 확인할 수 있어야 한다. 그러나 그래프 방법에서는 객체의 행위를 정확히 표시할 수 없는 경우가 있다. 그림 1에서의 scenario I과 scenario II는 그 결과만 보면 동일한 것으로 간주할 수 있으나, 객체의 행위를 모델링한다면, 이는 다른 것으로 볼 수 있다. 즉 객체를 검색하기 위해 객체의 동일성(equivalence)이 중시된다면 이는 구별되는 것이 바람직하다.

프로세스 대수는 원래 프로세스 간의 병행성이나 비결정성을 표현하기 위해 제안된 이론으로서, 이의 예로는 CCS, CSP등을 들 수 있는데, 프로세스 대수를 적절하게 적용하기 위한 응용에 대한 연구는 이 분야에서 점차 관심이 커지고 있는 부분이다.

본 논문에서는 멀티미디어 응용에서 사용되어지는 객체가 정적인 경우와 함께 동적인 경우에 대해서 동기화를 고려해보았으며, 이를 표현하기 위해서 프로세스 대수를 이용하였다 이를 위하여 Milner의 π-calculus를 이용하였는데, 이 과정에서 멀티미디어 객체의 동기화에 필수적인 시간적 요소를 표현하기 위해 π-calculus를 확장하였고, 이 확장이 기존의 π-calculus에 포함됨을 보였다. 또한 확장된 π-calculus를 이용한 동기화의 구현이 용이함을 보였다. 또한 Petri-net등의 그래프 방법에 비해 객체의 행위(behavior)의 표현이 더욱 적절함을 보였다. 이러한 특징을 이용하면 확장된 π-calculus를 이용하여 멀티미디어 데이타를 표현할 경우, 객체의 등가성(bisimulation)을 이용하여 멀티미디어 질의에 사용될 수 있음을 알아보았다.

2. 관련 연구

멀티미디어 데이타를 소프트웨어 시스템에서 어떻게 표현할 것인가는 멀티미디어 데이타의 모델링에 관한 연구를 통하여 행해져 왔고, 모델링에 관련된 연구는 다양한 미디어의 데이타를 통합하거나 동기화하기 위한 연구와 밀접하게 관련되어 연구되어 왔다. [4]에서는 13가지 멀티미디어 객체의 동기화를 제안하였고 [9]에서는 이 중 역관계에 의한 중복을 제거한 7가지 동기화를 제안하고 구현하였다. 여기서 제안한 동기화는 그림 2와 같다.

[18]에서는 미디어 통합에 따르는 범주를 3가지로 구분하고 동기화를 8가지로 구분하였으며 제한적 중단(restricted blocking) 개념을 처음으로 소개하였다 또한 여기에서는 다이나믹 에어전트에 의한 동기화의 가능성을 소개하였는데, 객체 간의 동기화의 종류를 pre-defined order, prioritized order, combined order의 3가지로 제시하였다.

ISO/IEC JTC1 SC2/WG12(Multimedia Hyper-

1) 여기서는 객체가 일단 생성된 후 변화가 없을 경우 정적 객체(passive object), 변화가 있을 경우 동적 객체(active object)라는 용어를 사용하였다

media Expert Group)에서는 멀티미디어 객체의 동기화를 다루기 위해 객체지향 모델을 도입하였다. 가장 최근의 문서 [1]에 따르면 객체의 동기화를 크게 시간적 동기화와 조건적 동기화로 구분하였고, 이 중 시간적 동기화는 [9]에서와 동일하게 제시하였다. [1][5][6]

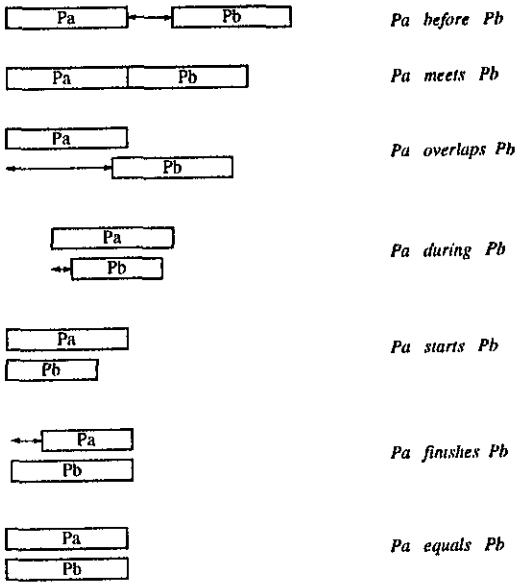

그림 2 객체의 동기화 방법

현재 멀티미디어 동기화의 주된 수학적 모델로 사용되고 있는 Petri-net은 원래 CCS, CSP등과 함께 병행 프로세스의 표현을 위해 제시된 것이다. 그러므로 Petri-net과 CCS와의 비교 연구는 여러 문헌에서 찾을 수 있다. Petri-net의 특징은 객체간의 종속관계를 표현하기에 좋다는 것이나, 이에 비해 CCS등 algebra 접근 방식의 특징은 행위(behavior)를 모델링하기 좋은 것으로 알려져 있다. [10][11][14][15][16]

또한 TCCS[13]에서는 프로세스 대수인 CCS에 시간적 요소를 추가하는 연구와 그 성질을 제시하였다.

[15]에서는 Petri-net을 이용한 소프트웨어 명세는 인과 관계를 표현하기에 적당하며, 객체의 일반화나 객체의 행위(behavior)를 명세하는데는 대수(algebra)가 더욱 유용함을 보였다. 대수를 이용할 경우, 객체의 인과 관계 및 행위에 따른 등가성을 표현할 수 있음을 보였다.

3. 프로세스 대수

π-calculus는 [10][11]에서 처음 소개되었으며, 상태가 동적으로 변화하는 객체 및 이들 간의 통신을 표현하기 위한 메카니즘이다. π-calculus는 Milner의 CCS에 기반한 대수이며, CCS는 Hoare의 CSP를 근간으로 하고 있다. π-calculus에서는 객체가 가지고 있는 모든 구성 요소에 대해 객체 간의 전송을 지원한다. 앞에서 열거한 기존의 동기화 메카니즘은 객체 간의 동기화 메카니즘을 제시하고 있으므로 객체의 상태가 능동적으로 변화하고 이를 이용하여 동기화를 수행하는 본 논문에서는 동적인 객체를 표현하고 이들의 동기화 메카니즘을 표현하기 위하여 π-calculus를 이용하여 다이나믹 에이전트를 지원하는 동기화 메카니즘을 제시하였다.

3.1 π-calculus 소개

π-calculus는 동적으로 변화하는 프로세스 간의 통신을 표현하기 위한 대수로서 제안되었다.[10][11] 프로세스 대수는 주로 분산 환경의 병행 프로세스를 표현하기 위하여 연구되었으며 CCS,CSP,ACP,Meiji 등은 성공적인 모델들의 예이다. 이러한 모델상에서 병행적으로 수행되는 프로세스에 대한 등가성(bisimulation)과 같은 여러 가지의 유용한 수학적 성질들이 제시되고 보여왔으며 이러한 성질들은 여러 분야에서 응용의 가치가 높은 것으로 평가받고 있다. [12][14][15]

π-calculus는 프로세스 대수인 CCS를 확장한 것으로서 CCS의 모든 대수적 속성을 유지하지만 링크나 에이전트 및 속성을 나타내는 이름의 구별을 없앰으로써 보다 간결하고 명료한 표현이 가능하며, 모든 이름이 객체간의 전송의 대상이 될 수 있다. [10]

π-calculus는 다음과 같다.

K는 이름의 무한집합, u, v, w, x, y, z는 여기에 속하는 이름이라 가정하자. 또한 K는 에이전트의 식별자의 집합, $A, B, C\ldots$은 에이전트 식별자라고 가정하자. π-calculus 는 프로세스의 집합 P에 대해서 다음 여섯가지 연산을 정의한다.

정의 1: π-calculus[10][11]

1. *Summation:* $\sum_{i\in I} P_i$ 단 인덱스 집합 I는 유한 집합 P_i 중 하나의 객체만이 배타적으로 수행된다. 합이 존재하지 않을 경우 inaction이라 하며 0으로 표시한다. 이원 연산자로서 합은 P_1+P_2와 같이 나타

낼 수 있다.

2. Prefix: $\overline{y}x.P, y(x).P$ or $\tau.P$.
   $\overline{y}x$는 negative prefix라 한다. $\overline{y}$는 객체의 출력 포트이다. 즉 $\overline{y}x.P$는 이름 x를 포트 y에 출력한 후 P를 수행한다. $y(x)$는 positive prefix라 하며 y는 입력 포트이다. 즉 $y(x).P$는 입력 포트 y로 부터 x로 이름을 얻어 $P\{x/y\}$를 수행한다.
3. Composition: $P_1|P_2$
   두개의 객체 P_1과 P_2가 병행적으로 수행된다 이때 두개의 객체가 동일 포트를 이용하여 교신한다면 이들은 동기화되어질 수 있다.
4. Restriction: $(x)P$.
   x를 포트로 하는 즉 $\overline{x}$와 x를 이용하는 행위가 금지된다.
5. Match: $[x=y]P$.
   x와 y가 일치하면 P를 수행한다. 만일 일치하지 않으면 이것은 0이 된다.
6. 에이전트의 정의: $A(y_1, y_2, \cdots, y_n)$
   임의의 에이전트 식별자에 대해 유일한 정의식 $A(x_1, x_2, \cdots, x_n) \stackrel{def}{=} P$가 존재하며 $x_1, x_2, \cdots, x_n$은 P에서 유효한 유일한 이름이다.
7. Equivalence의 정의[2)]
   P: agent의 집합
   $S \subset P \times P$인 binary relation S가 $(P, Q) \in S$에 대해 다음 두 조건을 만족할 때, 이를 bisimulation이라 한다.
   - 만약 $P \xrightarrow{\alpha} P'$이면, $Q \xrightarrow{\hat{\alpha}} Q'$이며 $(P', Q') \in S$를 만족하는 Q'이 존재한다.
   - 만약 $Q \xrightarrow{\alpha} Q'$이면, $P \xrightarrow{\hat{\alpha}} P'$이며 $(P', Q') \in S$를 만족하는 P'이 존재한다.

Bisimulation의 정의를 이용하면 그림 1의 두 시나리오는 서로 다름을 알 수 있다.[15]

π-calculus 예 - 1

$P \equiv \overline{a}b.P', R \equiv a(x).R'$이라 하자. P와 R을 병렬로 수행시킨다면 (parallel composition) $\overline{a}b.P' \mid a(x).R'$이 되며, 이를 프로세스 그래프로 나타내면 다음 그림과 같다. 이 결과,
링크 a를 통하여 데이타 5가 P에서 R로 전달된다

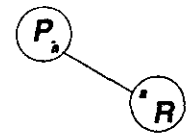

그림 예 - 1

π-calculus 예 - 2

$P \equiv \overline{b}a.\overline{b}5.P', Q \equiv b(y).b(z).\overline{y}z.Q' R \equiv a(x).R'$이라 하자. P는 b를 통하여 a및 5를 전달하며, Q는 b를 통하여 두개의 y, z를 받은 후 받은 값 y를 통하여 값 z를 전송하게 된다. 다시 R은 y를 통하여 z를 전송한다.

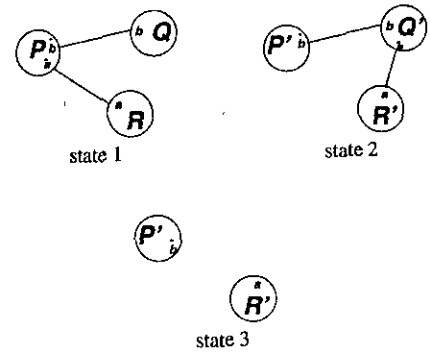

그림 예 - 2

공리 1: [10][11][12]
1. Identity: $a\tau = \tau a = a$, 단 τ는 empty action

3.2 확장 π-calculus

이제 시간적 요소를 고려하기 위해 다음을 정의하자.

정의 2: 시간 요소
1. Temporal prefix: $\delta.P$
   δ는 임의의 숫자이다. 이는 프로세스 수행 전의 δ 시간의 지연을 나타내며, δ가 음수일 경우에는 프로세스 수행 후의 $-\delta$시간의 지연을 나타낸다. 이때 상태 전이에 의해 P는 empty action τ를 수행한다.

정의 3,4: Relabeling 함수 및 Action
State transition function $\stackrel{def}{=} f \cdot P \xrightarrow{act} P'$,

2) 편의상 bisimulation의 정의는 원래 논문의 것을 간략화하였음.

단 P,P'은 대수의 에이전트이며, act는 에이전트의 전이에 의해 배출하는 이름이다. 이 때 배출되는 문자 act를 행위(action)로 정의하자 또한, 연속되는 행위는 문자 act가 나열된 스트링의 형태로 표현된다.

정의된 에이전트는 relabeling 함수에 의해 차례로 이름을 배출하게 되고, 이름을 연결한 스트링이 에이전트의 동작이 된다. 즉, 위의 정의에 의해 에이전트의 동작은 relabeling 함수에 의해 배출되는 스트링에 대응된다.

그러므로 확장 π-calculus의 동기화에 대한 정확성을 보이기 위해서는 이 때 배출되는 스트링이 목적하는 행위의 결과와 일치함을 보이면 된다

π-calculus에는 통신을 위한 메카니즘이 포함되어 있다. 이 중 같은 포트에 대한 입력과 출력은 π-calculus에 존재하는 유일한 상보적(complementary) 동작이다. 즉, 이는 동기화에 사용될 수 있으며, 이를 이용한 동기화의 표현을 용이하게 하기 위해 입력에 대한 출력이나 출력에 대한 입력을 상보적인 표현으로 나타낼 수 있도록 다음과 같이 약속하자.(정의 4의 새로운 의미는 없으며, 단지 앞의 정의에서 표현된 입력 및 출력에 대한 기호 치환의 의미이다.)

정의 5: 상보적 동작(complementary action)

$P=\bar{x}y.A$이고 $Q=(x)y.A$라면 두 에이전트 P,Q는 포트 x를 이용하여 데이타를 전달하게되며 이 때 동기화가 일어난다. 이 경우

$$P \xrightarrow{let} \hat{w}.A, Q \xrightarrow{let} \check{w}.A$$

를 상보적 동작이라 정의하자. 이 때 $\check{w}$가 포트 x에 출력 연산이라면 $\hat{w}$는 포트 x의 입력 연산이 되며, 반대로 $\hat{w}$가 포트 x의 입력 연산이라면 $\hat{w}$는 동일 포트에 대한 출력 연산이 된다 또한 하나의 에이전트는 입력과 출력, 다른 에이전트는 출력과 입력의 형태일 경우도 상보적 동작의 한 예이다.

4. 확장 π-calculus의 동기화 표현

여기서는 먼저 [9]에서 제시한 7가지 동기화 메카니즘이 확장된 π-calculus와 relabeling 함수에 의해 생성되는 행위 스트링을 이용하여 올바르게 생성됨을 보인다.

4.1 정적 동기화 표현

[4]에서 제시한 13가지 동기화 방법은 [9]에서 7가지 동기화 방법으로 제시되었으며 정적인 객체의 동기화를 위하여 충분함을 보였다. 여기서는 π-calculus에 의한 프로세스 대수가 정적인 객체 간의 동기화를 지원하는 것을 보이기 위해 [9]의 7가지 동기화 방법을 π-calculus를 이용하여 표현할 수 있음을 보였다

정리 1:

[9]의 7가지 동기화는 relabeling 함수에 의해 생성되는 스트링과 일치한다.

증명

o P_α,P_β는 임의의 두개의 에이전트, f는 relabeling 함수라 하자.

1. P_α *before* P_β:

$Act \xrightarrow{let} P_\alpha.P_\beta.P_\alpha \xrightarrow{let} P,P_\beta \xrightarrow{let} \delta.P'$ 이라 하자.

$f:P_\alpha.P_\beta \xrightarrow{P} P_\beta$ (정의3,4)

시간 δ 경과 전, $f.P_\beta \xrightarrow{\tau} P_\beta$ (정의2,3,4)

시간 δ 경과, $f:P_\beta \xrightarrow{P'} 0$ (정의2,3,4)□

2. P_α *meets* P_β.

$Act \xrightarrow{let} P_\alpha.P_\beta.$

$f:P_\alpha.P_\beta \xrightarrow{P_\alpha} P_\beta$ (정의3,4)

$f.P_\beta \xrightarrow{P_\beta} 0$ (정의4)□

3. P_α *overlap* P_β:

$Act \xrightarrow{let} P_\alpha \mid P_\beta, P_\alpha \xrightarrow{let} \delta_1.P, P_\beta \xrightarrow{let} \delta_2.P'$라 하자.

(단, $\delta_1<0,-\delta_1>\delta_2$)

$f:P_\alpha \mid P_\beta \xrightarrow{P_\alpha} 0 \mid P_\beta$ (정의3,4)

시간 δ_2 경과 전, $f\ 0 \mid P_\beta \xrightarrow{\tau} 0 \mid P_\beta$ (정의2,3,4)

시간 δ_2 경과 및 $-\delta_1$ 경과 전, $f:0 \mid P_\beta \xrightarrow{P'} 0 \mid 0'$ (정의2,3,4)

시간 $-\delta_1$ 경과 후, $f:0 \mid 0' \xrightarrow{\tau} 0'$ (정의2,3,4)□

4. P_α *during* P_β:

$Act \xrightarrow{let} P_\alpha \mid P_\beta, P_\alpha \xrightarrow{let} \delta.P.\bar{x}y, P_\beta \xrightarrow{let} P'.x(y)$라 하자.

$f:P_\alpha \mid P_\beta \xrightarrow{P'} P_\alpha \mid x(y)$ (정의3,4)

시간 δ 경과, $f:P.\bar{x}y \mid x(y) \xrightarrow{P} \bar{x}y \mid x(y)$ (정의2,3,4)

$\bar{x}y \mid x(y) \xrightarrow{\tau} 0 \mid 0'$ (정의3,4)□

5. P_α *starts* P_β:

$Act \xrightarrow{let} P_\alpha \mid P_\beta$ 리 하자.

$P_\alpha \mid P_\beta \xrightarrow{P_\alpha,P_\beta} 0$ (정의3,4)□

6. P_α *finishes* P_β:

$Act \overset{let}{=\!=} P_\alpha \mid P_\beta, P_\alpha \overset{let}{=\!=} \delta.P, \hat{w}, P_\beta \overset{let}{=\!=} P', \check{w}$ 라 하자.

시간 δ 경과 전, $f:P_\alpha \mid P_\beta \xrightarrow{P'} P_\alpha \mid \check{w}$ (정의3,4)

시간 δ 경과, $f:P_\alpha \mid \check{w} \xrightarrow{P} \hat{w} \mid \check{w}$ (정의3,4)

$f:\hat{w} \mid \check{w} \xrightarrow{\tau} 0$ (정의5)□

7. P_α *equals* P_β:

$Act \overset{let}{=\!=} P_\alpha \mid P_\beta, P_\alpha \overset{let}{=\!=} P.\hat{w}, P_\beta \overset{let}{=\!=} P'.\check{w}$ 라 하자.

$f \cdot P_\alpha \mid P_\beta \xrightarrow{P,P'} \hat{w} \mid \check{w}$ (정의3,4)

$f:\hat{w} \mid \check{w} \xrightarrow{\tau} 0$ (정의5)□

4.2 동적 동기화 표현

기존의 동기화 메카니즘에서 표현하는 동기화는 정적인 객체를 가정하고 있으며, 동기화 전,후에 있어서 객체는 동기화와 무관하게 동작만이 수행된다. 이는 특히 녹화된 테이프나 비디오로 부터 상영되는 경우 또는 비트 이미지의 스트림으로 구성된 경우에 적합하나, 저작도구로부터 저작된 멀티미디어 객체의 경우는 사용자의 정의에 따라 능동적인 객체의 구성이 가능하다. 예를 들어 오디오와 동화상 간의 동기화는 각 객체의 끝단에서 일어나지만, [20]에서와 같은 입술 동기화(lip synchronization)[3]의 경우에는 객체의 수행 중에 동기화의 필요성이 대두된다. 오디오 데이타나 비디오 데이타들은 필요에 따라 수시로 동기화를 수행할 수 있다. 객체들은 능동적으로 움직이며 이들은 서로 간의 조건에 따라 동기화를 수시로 수행할 수 있다. 그러므로 객체 동기화 메카니즘에서 동적인 객체를 가정할 경우 그림 3의 동기화 메카니즘이 필요하다. 여기서 확장된 π-calculus가 동적 객체를 위한 동기화를 표현할 수 있음을 보인다.

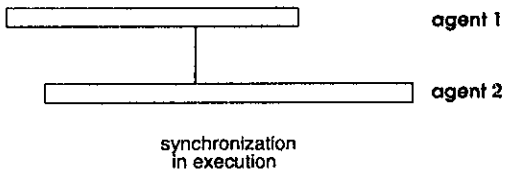

그림 3 능동적 에이전트의 동기화

정리 2:

동적 동기화(그림 3)는 확장된 π-calculus에 의해 생성되는 스트링과 일치한다.

o P_α, P_β는 임의의 두개의 에이전트, f는 relabeling 함수라 하자.

- P_α *sync-in* P_β:

$Act \overset{let}{=\!=} P_\alpha \mid P_\beta, P_\alpha \overset{let}{=\!=} P.\hat{w}.P', P_\beta \overset{let}{=\!=} Q.\check{w}.Q'$라 하자.

$f.P_\alpha \mid P_\beta \xrightarrow{P,Q} \hat{w}.P' \mid \check{w}.Q'$ (정의3,4)

$f.\hat{w}.P' \mid \check{w}.Q' \xrightarrow{\tau} P' \mid Q'$ (정의5)

$f:P' \mid Q' \xrightarrow{P',Q'} 0$ (정의5)□

다음과 같은 예가 동적인 에이전트를 이용하여 나타내어질 수 있다.

o 예 1: 동적인 에이전트의 반복 (그림 4)

- $S_0 = \check{u}.S_0$
- $S_1 = \check{w}.\hat{u}.S_1$
- $P_r = \hat{w}.\delta.P_r$

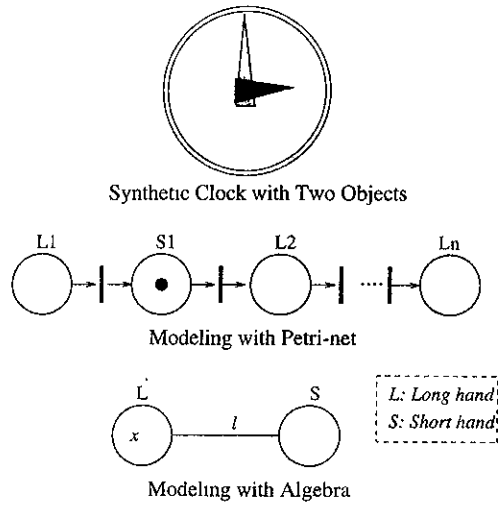

그림 4 에이젼트의 동기화 예

여기에서는 움직이는 시계의 바늘을 표현하는 두개의 객체(에이전트)를 가정하였다. 이 때 두개의 객체는 모양이 변화하나 일정한 속성을 유지하는 동적 객체이며 이들의 진행 중에 동기화가 수시로 행해진다. 여기서 P_r는 δ 시간 간격으로 동기화 요구를 발생하는 객체이며, S_1은 긴 바늘, S_0는 짧은 바늘에 해당하는 객체로 각각

3) 이는 lip sync에 사용되는 오디오와 이미지가 합성(synthesize)될 경우임. [20] 참조

P_r, S_i과 동기화 된다. S_i은 P_r로 부터 일정한 펄스로 긴 바늘을 구동하며, 1회전 시마다 동기화 포트를 통하여 미리 정한 동기 데이타를 전송하게 되고 이를 수신한 S_0는 S_0를 위한 적절한 동작을 취하게 된다. 여기에서는 객체의 수행 중에 동기화가 자주 일어나는 것을 알 수 있다

정리 3.
정의 2에 의한 π-calculus의 확장은 원 π-calculus의 성질을 보존한다.

증명
P는 순수한 π-calculus의 임의의 에이전트, f는 relabeling 함수,

$P' \xrightarrow{let} \delta.P$라 하자.

o P는 f에 의해

$f.P \xrightarrow{P} 0$

이 때 생성되는 행위 스트링은 "P"이다.

o 이제 P에 확장된 시간 요소가 추가된 P'에 대하여 f에 의해

$f:P' \xrightarrow{\tau} P'$, 시간 δ 경과 전

$f:P' \xrightarrow{P} 0$, 시간 δ 경과

이 때 생성되는 행위 스트링은 "$\tau.\tau.\cdots.\tau.P$"

(시간 δ를 경과하기 전까지, 임의의 횟수만큼의 relabeling 함수가 수행될 때마다 τ를 생성한다.)

$\tau.(\tau.\cdots,\tau.P) = \tau.\cdots,\tau.P$ (공리1)

$\tau.\cdots.\tau.P = \cdots = \tau.P = P$ (공리1)

그러므로, 임의의 에이전트 P에 대해 행위 스트링은 확장된 시간 요소에 관계없이 일치한다 □

5. 비교

기존의 동기화 방법을 이용하여 우리는 정해진 객체를 동기화하며 늘어놓을 수가 있다. 즉, 슬라이드나 비디오 스트림과 같은 그 내용이 능동적으로 바뀌지 않는 객체의 동기화에 적합함을 알 수 있다.

프로세스 대수를 이용하여 제안된 동기화 메카니즘에서는 객체가 동기화의 수행 시점을 보다 자유롭게 결정할 수 있으므로, 동기화와 객체가 밀접하게 연관된 기존의 방법에 비해 동기화의 자율성이 부여된다. 확장 π-calculus에서는 객체의 동기화가 객체의 행위의 일부로 표현되므로 객체의 시간적인 동작 속성을 객체 자신이 가질 수 있음을 알 수 있다. 또한 객체들 간의 능동적인 협력(메시지의 교환)이 가능하므로 객체가 능동적으로 행동하면서 객체 동기화를 수행할 수 있게 된다. 프로세스 대수는 객체의 내용과 함께 변화되는 상태를 표현할 수 있으므로 이를 이용하면, 같은 구조를 가지기만 하면 그 상태에 관계없이 객체의 재사용이 용이하다.

그러나 OCPN[9]과 같은 기존의 동기화 메카니즘에서는 모든 객체는 정적이므로 객체의 내용 표현 만이 가능하다 즉 Petri-net을 확장한 OCPN을 이용한 동기화 방식은 동기화는 객체의 행위(behavior)와는 별개로 취급되었으며 미리 정해진 시점에서의 동기화만 가능하므로 객체의 행위 자체가 동기화와 결부되어있는 경우를 표현하는 것은 어렵다. [17]에서는 inter-object 동기화와 intra-object 동기화를 도입하여 이를 어느 정도 해결하였으나, 여전히 내포적인(recursive) Petri-net의 모양을 가지며 내포된 Petri-net에 대해서는 일반화의 문제가 여전히 있음을 알 수 있다.

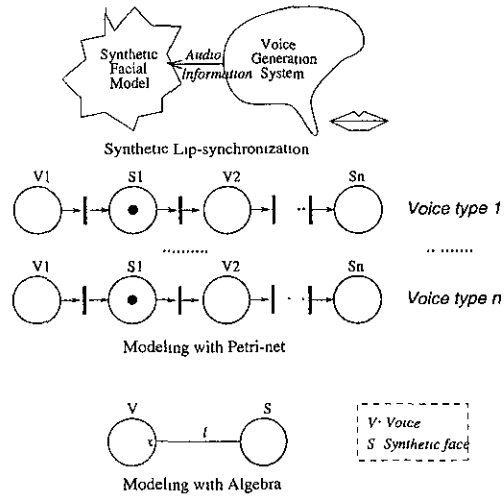

그림 5 OCPN의 동기화 및 대수를 이용한 일반화

확장 π-calculus에서는 그림 4의 경우, 예 1을 표현하기 위하여 세개의 에이전트 정의만으로 이를 표현 할 수 있다. 또한 객체가 동일한 구조를 가질 경우 relabeling에 의해 일반화될 수 있다. 이 결과 그림 5에서 보는 바와 같이 복잡한 형태의 동기화를 일반화된 객체를 이용하여 간단하게 표현할 수 있음을 알 수 있다. 그림 5를 Petri-net을 이용해서 표현할 경우 더 이상의 일반화는 어렵다

멀티미디어 객체들간의 동기화를 수행하는 것은 일반

적으로 DBMS의 측면에서 보면 객체 관리자이다. 이들은 MHEG 커널과 같이 분산 환경에서 다수가 존재할 수 있다. 이럴 경우 객체 간의 교신은 더욱 중요한 문제로 제기된다. 프로세스 대수는 처음 병행 프로세스 간의 상호 동작을 위해 제안되어졌으므로 통신을 지원하는 동기화는 분산 환경이나 네트워크 환경으로 쉽게 확장될 수 있으므로 동적인 객체를 지원하는 멀티미디어의 표현(specification)에 적합함을 알 수 있다.

모델의 추상화에 대한 장점은 여러 분야에서 이미 널리 입증되어 있다. 에이전트의 동작을 프로세스 대수를 이용하여 추상화하는 것에 대한 장점은 에이전트의 동작 방식을 쉽게 구분할 수 있다는 점에서도 찾을 수 있다 CSP, CCS등에 기반한 π-calculus와 이를 확장한 본 논문의 확장 프로세스 대수는 bisimulation을 이용하여 임의의 두 에이전트의 결과및 종속 관계 외에도 행위(behavior)가 동등함을 판단할 수 있으며, 에이전트 행동의 최적화나 DBMS의 질의에 유용하게 사용될 수 있다

5.1 응용 시 고려 사항

확장 π-calculus는 에이전트에 기반한 객체 기반 프로세스 대수이다. 즉 π-calculus를 이용하여 객체를 나타내는 것은 정의 1.6을 이용하여 가능하다 이러한 객체들은 객체를 관리하는 시스템의 속성에 따라 객체지향 시스템으로 확장될 수 있다. π-calculus에서는 단지 객체 간의 동기화 및 통신만을 지원하며 객체의 속성이나 상속등은 객체 지향 DBMS와 같은 다른 응용 프로그램과 결부되어질 수 있다. 이 경우 객체 지향 DBMS를 이용하게 된다면 각 객체들은 객체지향적 속성을 가지게 된다.

π-calculus에서는 사용되는 이름이 통신 포트이거나 속성일 경우를 구별하지 않고 사용하였다. 그러나 이의 구현 시에는 이의 구별이 필요하게 되므로 이에 따른 주의가 필요하다. 또한 여기에서는 통신과 동기화를 위한 에이전트의 개념만이 존재하나, 실제 구현 시에는 에이전트를 제어할 수 있는 메소드가 대부분의 경우에 필요하다 그러나 π-calculus가 에이전트및 이의 속성을 지원하므로 이의 확장은 용이하다. 즉 π-calculus에서 지원하는 에이전트의 개념이 매우 간결하므로 이를 객체지향 시스템에서 쉽게 수용할 수 있다. 순수한 π-calculus의 경우는 일본 통산성 ETL(Electronic Technical Laboratory)에서 소프트웨어 명세시스템으로서 개발한 예가 있다. [15]

6. 결 론

여기에서는 동적 성질을 가지는 객체들의 동기화를 표현하기 위하여, 분산,병행 프로세스들의 동작을 표현하기 위해 제안된 프로세스 대수를 이용하였으며, 이를 이용하여 멀티미디어 객체의 동기화를 표현하는 것이 가능함을 보였다

본 논문에서는 프로세스 대수의 유용성을 보이기 위해 π-calculus를 확장하고 이를 이용하여 정적 동기화 메카니즘과 동적 동기화를 표현하여 보았고, 에이전트 간의 동기화를 이용한 협력이 가능함을 보였으며, 본 논문에서 제시한 π-calculus의 확장이 기존의 π-calculus를 보존하는 확장(conservative extension)임을 보였다.

또한 기존의 방법 중 OCPN과 같은 확장 Petri-net에 의한 동기화를 이용할 경우 표현이 비효율적인 예를 보이고, 이를 본 논문에서의 확장 π-calculus를 이용하여 표현하였다.

이를 통하여 기존의 동기화 메카니즘이 가지고 있는 제한적인 객체의 허용에 비해, 프로세스 대수가 일반적인 객체에 대한 동기화를 가능하게 하며, 기존의 방법이 가지고 있는 동기화를 위한 데이타의 중복이나 비효율성의 가능성을 줄일 수 있고, 보다 정확한 동일성(equivalence) 표현이 가능하므로 DBMS의 질의에 유용하게 사용될 수 있음을 제시하였다. 또한 각 에이전트의 통신의 지원으로 MHEG kernel이나 DBMS의 멀티미디어 관리자 같은 멀티미디어 관리 시스템을 효율적으로 지원할 수 있는 방법임을 보였다.

참 고 문 헌

[1] ISO/IEC Draft International Standard 13522-1, Information Technology - Coding of Multimedia and Hypermedia Information -, Sep. 1994

[2] Blair and D. Hutchison and D. Shepard, Multimedia Systems, In The 3rd IFIP Conference on High Speed Networking, Mar. 1991.

[3] Jean-Pierre Courtiat and R. C. de Oliveira and L. F. Rust da Costa Carmo, Towards a new multimedia synchronization mechanism and its formal specification, In Proceedings of ACM Multimedia '94, 133-148, San Francisco, CA, Oct 1994.

[4] L. Hamblin,Instants and Intervals, In Proc. 1st Conf. Int. Soc. for the Study of Time, 324-331, Springer Verlag, 1972.

[5] ISO/IEC JTC1/SC21/WG12, MHEG Working Document S version 3, Nov. 1990.

[6] ISO/IEC JTC1/SC29/WG12, Information Technology Coded Representation of Multimedia and

Hypermedia Information Objects: MHEG Working Document S version 5, Jan. 1992.
[7] L. Li and A. Karmouch and N.D. Georganas, Multimedia Teleorchestra with Independent Sources: Part I - temporal modeling of collaborative multimedia scenarios, Multimedia Systems, 1(4), Feb. 1994.
[8] L. Li and A. Karmouch and N.D. Georganas, Multimedia Teleorchestra with Independent Sources: Part II - temporal modeling of collaborative multimedia scenarios, Multimedia Systems, 1(4), Feb. 1994.
[9] Thomas D.C. Little and Arif Ghafoor, Synchronization and Storage Models for Multimedia Objects, IEEE Journal on Selected Areas in Communications, 8(3):413-427, Apr. 1990.
[10] Robin Milner and David Walker, A Calculus of Mobile Processes I, Information and Computation, 1-40, 1992.
[11] Robin Milner and David Walker, A Calculus of Mobile Processes II, Information and Computation, 41-77, 1992.
[12] Moller, F and C. Tofts, A Temporal Calculus of Communicating Systems, In Proceedings of CONCUR '90, Lecture Notes in Computer Science 458, Aug. 1990.
[13] Faron Moller, Process Algebra as a Tool for Real time Analysis, Technical Report by ESPRIT BRA 3006, Springer Verlag, 1991.
[14] K. Ohmaki and Y. Sato and Ki-Byoung Kim, Applications of Process Algebraic Approaches to Practical Systems - Experiences and Perspectives -, *Technical Report of IEICE*, CST94-3(4):17-24, 1994.
[15] K. Ohmaki and Y. Sato and I. Ogata and K. Futatsugi, Algebraic Approaches for Nets Using Formulas to Describe Practical Software Systems, IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, E76-A(10):1580-1590, Oct. 1993.
[16] James L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-hall inc., Englewood Cliffs, NJ., 1990.
[17] Maveed U Qazi and Miae Woo and Arif Ghafoor, A Synchronization and Communication Model for Distributed Multimedia Objects, *In Proceedings* of ACM Multimedia '93, Aug. 1993.
[18] Ralf Steinmetz, Synchronization properties in Multimedia systems, IEEE Journal on Select. Areas in Comm., 8(3):401-412, Apr. 1990.
[19] Costantino Thanos, Multimedia Office Filing:The MULTOS Approach, Van Nostrand Reinhold Company, 1990.
[20] Keith Waters and Tom Levergood, An Automatic Lip-Synchronization Algorithm for Synthetic *Faces*, *In Proceedings of ACM Multimedia* '94, 149-156, Oct. 1994.

김 기 병
1990년 2월: 서울대학교 계산통계학과 졸업. 1992년 2월: 서울대학교 계산통계학과 전산과학전공 석사. 1992년 3월-현재: 서울대학교 컴퓨터공학과 박사과정. 관심분야 데이타 베이스, 객체지향데이타베이스, 멀티미디어 데이타베이스

김 형 주
제 22권 1호 참조