제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

자료 일치성을 위한 동기화 프로토콜의 정형 검증

전승수, 권기현
경기대학교 전자계산학과
{dabins, khkwon}@kyonggi.ac.kr

Formal Verification of Synchronization Protocol for Data Coherence

Seungsu Chun, Gihwon Kwon
Dept. of Computer Engineering, Kyonggi University

요 약

본 논문에서는 유·무선 네트워크 환경에서의 자료 일치성 유지를 위한 동기화 프로토콜(SyncML1.1)을 정형 검증한다. 프로토콜 검증에 관한 기존 연구에서는 약한 일관성을 갖는 동기화 프로코콜은 다루어지지 않았으며 정리 증명 혹은 모델 체킹에 대한 선택적 접근으로인해 증명 영역이 제한적이였다. 본 연구에서는 프로토콜의 의미를 정의하고 행위 및 시스템 구조, 메시지 패싱에 대한 개별적 모델을 단계적으로 설계하는 한편, 동기화 프로콜에서의 안전성 및 궁극성 속성을 시제 논리 및 고차 논리를 통해 정의하고 이를 혼합된 검증 방법을 통해 강하게 증명했다. 자료 일치성 및 상호 배제, 궁극성 속성은 시퀀트 계산법에서의 역방향 전략으로 증명되며, 모델 체킹 기술을 하나의 증명 규칙으로 사용했다. 본 연구는 두가지 부분에서 특징과 의미을 갖는다. 첫째, 정리증명과 모델 체킹이 혼합된 검증 방법을 사용했다. 둘째, 객체 제약 언어 및 정형 명세 언어의 사용을 통해 모델의 생성을 보다 단계적이고 정형적으로 유도했으며 이를 통해 검증의 신뢰성 및 품질을 높였다.

1. 서론

개인 및 기업 정보가 네트워크를 통해 분산 관리되고 주요 거래가 전자상거래 시스템을 통해 이루어지면서 프로토콜의 신뢰성 및 안전성에 대한 정형 검증의 중요성이 더욱 강조되고 있다[1]. [1]본 논문에서는 유·무선 네트워크 환경에서의 자료 일치성 유지를 위한 동기화 프로토콜(SyncML1.1)을 정형 검증한다. SyncML 은 다양한 기종 및 네트워크, 응용 프로그램 사이에서의 자료 일치성을 보장하는 표준 프로토콜이다[2]. 자료 일치성(Coherence)은 하나의 정보를 표현하기 위하여 사용되는 여러 개의 자료가 동일한 값을 갖는 것이며 이러한 일치성이 유지되는 현상을 자료 일관성(Consistency)이라 한다. 이러한 자료 일치성은 병렬 및 병행 다중 프로세서의 설계에 있어 매우 중요하게 고려되어지며 일관성을 위한 엔진은 병렬 및 분산 컴퓨팅 시스템의 핵심 컨퍼넌트가 된다[3]. 기존의 연구들은 하드웨어를 기반으로하는 공유 및 분산 메모리 모델 혹은 분산 및 병렬 컴퓨팅 시스템에서의 순차적 일관성[3]을 검증하는데 집중된 반면, 약한 일관성을 갖는 동기화 프로토콜은 다루어지지 못했다. 프로토콜에 대한 정형 검증의 일반적 문제는 동기 및 비동기성을 동시에 갖는 프로토콜의 특성과 무한 영역에 대한 매개 변수의 처리 및 증명, 특히 높은 복잡도를 갖는 행위의 추상화 모델 유도가 어렵다는 것이다[4]. 또한 정리 증명[5] 혹은 모델 체킹[6]에 대한 선택적 증명 방법으로 인해 증명 영역이 매우 제한적이였다. 최근 이러한 문제를 해결하기 위해 정리 증명과 모델 체킹이 혼합된 새로운 검증 방법이 시도되고 있다[7]. 본 논문에서는 비동기성을 갖는 프로토콜의 무한 영역을 아키텍처로 모델링하고 시스템 정의를 통해 동기화 관계의 대칭성을 보인다. 메시지 패싱 및 행위 모델, 영역 명세를 통해 추상화 모델을 단계적으로 유도했다. 우리는 얻어진 모델에 대한 혼합된 검증 방법으로 동기화 프로토콜의 전체 영역을 증명했다. 또한 추상화 모델의 유도를 일반화함으로써 검증의 신뢰성 및 품질을 높였다.

[1] 이 연구는 전자통신 연구원(ETRI, 계약 3010-2002-0090)의 지원으로 수행되었음.

2. 시스템 모델

우리는 SyncML 프로토콜과 동기화 시스템의 전체 영역을 정형적으로 모델링한다. 이를 위해 SyncML 프로토콜 명세서(1.1)를 바탕으로 아키텍쳐 및 메시지 패싱, 행위에 대한 계층적 모델을 단계적으로 명세한다. 모델 및 속성 명세를 위해 고차 논리[5] 및 CTL[6], 객체 제약 언어[8] 등을 사용한다.

2.1 아키텍쳐

일반적인 자료 일치성 프로토콜은 상호 연결된 모든 캐쉬 및 메모리의 일치성 유지를 위해 모든 프로세서를 대상으로 중계(Broadcast) 및 스누핑(Snooping)을 사용하고 순차적 혹은 프로세서 일관성 모델을 갖는다[4]. 반면, 동기화 프로토콜는 약한 일관성[9] 모델을 갖으며 메시지 패싱을 통해 일치성을 유지한다.

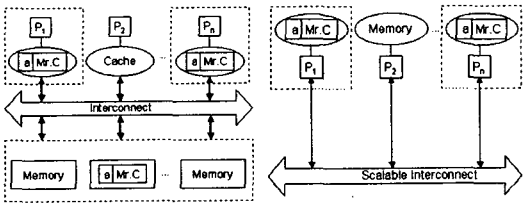

그림 1 UMA 와 NUMA 에서의 자료 일치

이러한 일관성의 차이는 그림 1 과 같이 분산 및 병렬 컴퓨팅 시스템의 성능 및 확장성을 위한 메모리 모델에 따라 달라진다[10]. 예를 들어, 공유와 분산 메모리 모델은 각각 UMA(Uniform Memory Access)와 NUMA (non-uniform MA) 혹은 COMA(Cache Only Memory Architectures)로 명세된다. SyncML 시스템의 경우 모든 프로세서가 메모리갖고 독립적 처리를 수행하기 때문에 기본적으로 NUMA 구조를 갖는다. 하지만 서버와 같은 특정 컴퓨터를 공유함으로 그림 2 와 같은 CC-NUMA DSM (Cache Coherence NUMA Distributed Shared Memory) 아키텍쳐를 갖게 된다.

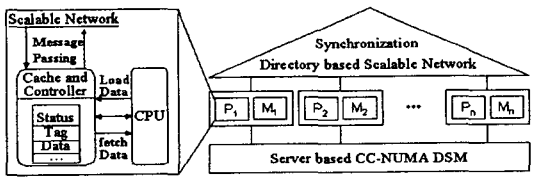

그림 2 SyncML 시스템

SyncML 시스템은 동기화를 통해 자료 일치성을 유지하며 이를 위한 프로세서간의 메시지 패싱이 진행된다. 또한 서버와 같은 특정 공유 공간을 갖으며 공유 공간은 디렉토리 혹은 주소로 분류된 메모리 구조를 갖는다. 그림 2 와 같이 SyncML 시스템은 무한한 프로세서와 메모리, 자료를 갖으며 각각의 프로세서가 비동기성을 갖기 때문에 동기화 프로토콜 역시 비동기적으로 수행된다.

2.2 SyncML 동기화 시스템

우리는 SyncML 명세서와 위에 제시된 메모리 아키텍쳐 및 일반적 프로토콜 정의를 바탕으로 SyncML 시스템 SM 을 다음과 같이 정의한다.

$SM = \{SPU, SPU_0, Dir, Sync\}$

여기서 SPU 는 동기화 프로세서의 집합이며, $SPU_0 \subseteq SPU$ 는 동기화를 시도한 초기 프로세서 단위의 집합, Dir 는 디렉토리 함수로서, SPU 의 상태를 배정한다. $Sync$ 는 $SPU \times SPU$ 간의 동기화 시도이다.
프로세서 단위를 spu 라 할 때,

$\forall spu \in SPU \bullet \exists spu' \in SPU \bullet (spu, spu') \in Sync$ 이다.

또한, $spu \in SPU$ 는 메모리 Mem, 동기화를 위한 정보 $Cache$, 동기화 컨트롤러 $Cont$ 를 갖는다. 따라서 아래와 같이 정의된다.

$spu = \{Mem, Cache, Cont\}$

정리 1. '프로세서 단위 spu 와 프로세서 단위 spu'은 동기화 관계를 갖는다.'라 하고, $spu \leftrightarrow spu'$라고 하자.

$spu \leftrightarrow spu'$ iff $(spu, spu') \in Sync \wedge (spu', spu) \in Sync$

즉, 동기화 관계를 만족하는 식 $SyncCase$ 는

$$SyncCase = \bigvee Sync \left(\bigwedge spu \leftrightarrow \bigwedge spu' \right),$$

이다. spu 의 인자는 Mem, $Cache$, $Cont$ 이며 각 인자에 대한 정의는 다음과 같다.

$Mem = \{Data, direc\}$
$Data$: Set of data
$direc$: Set of directory in memory
$Cont$ ::= On | Off

Mem 는 자료의 집합과 디렉토리의 집합을 갖으며, spu 와 같이 무한하다. $Cont$ 은 동기화의 상태이다.

정리 2. 'Mem 의 디렉토리 집합 dMem 과 Mem'의 디렉토리 dMem'이 같다.'라고 하고, equal(dMem, dMem')하자. 따라서, 디렉토리와 동기화 관계 식은

$\forall d Mem \forall d Mem' \forall Sync(\text{equal}(d Mem, d Mem') \rightarrow Sync(spu, spu'))$ 이다.

2.3 메시지 패싱 모델

$Sync$ 관계에서 진행되는 것이 메시지 패싱이다. 메시지 패싱 방법은 7 가지의 동기화 타입 $SyncType$ 에 따라 달라진다.

$SyncType$ = TwoWay | Slow | OneWay from Server
| One-Way Client Only | Refresh from Server
| Refresh Client Only Server | Alerted Sync

동기화를 위한 모든 정보는 캐쉬 $Cache$ 에 저장되며 메시지 패싱을 통해 상호 통신한다. 즉, 동기화의 조건과 요소가 $Cache$ 에 명시된다.

$Cache = \{Status, Tag, data\}$

이 논문은 동기화 타입에서 우선적으로 권고하는 Two way Sync 방식만을 다룬다. 특징은 $Sync$ 에 참여한 프로세서 단위는 서로의 갱신 정보를 Package 로 묶어 패싱되며 #1 부터 #6 까지의 우선 순위를 갖는다[1].

$Sender\_spu\_message = \{\#1, \#3, \#5\}$
$Receiver\_spu\_message = \{\#2, \#4, \#6\}$

정리 3. 'spu 이 $Sync$'라고 하자, 만족되는 식은

$AG(spu\#n \rightarrow AX[\#n+2 \wedge A[\neg\#n+4 \ U \ \#n+2]]), n = 1 \text{ or } 2$

2.4 행위 모델

일반적인 동기화 절차에서 *spu* 는 요청, 진입, 반환의 순으로 진행된다. 여기서는 이를 IMS(Invalid, Modified, Shared) 상태로 정의한다. Two way 방식은 양방향 동기화이며 캐쉬를 통해 정보와 자료를 패싱한다. *spu* 의 인자는 다음과 같이 정의된다.

Cache = {*Status*, *Tag*, *data*}
Status ::= Modified | Shared | Invalid
Tag = {*Com*, *dMem*}, *data*:Set of data ::= Exist | Empty
Shared : Valid and readable copy and visible Sync
Invalid : Sync started, Modified : data-fetch finished

정리 4. 각 *spu* 는 비동기적이고 *Sender* 와 *Receiver* 는 동기적이라고 하자. 순차적 상태 전이의 반복은 다음과 같이 정의된다.

Sender_spu_States =
Shared —#2→ Invalid —#4→ Modified —#6→ Shared
Receiver_spu_States =
Shared —#1→ Invalid —#3→ Modified —#5→ Shared

우리는 OCL(Object Constraint Language)을 통해 객체를 제약한다. OCL 은 UML 에 대한 정형성을 지원하는 한편, 영역 명세 및 시스템의 수행성을 지원한다[11][12]. *SM* 의 각 요소는 다음과 같이 정의된다.

Class = { *SenderClass*, *ReceiverClass*}
Event = {On, #1, #2, #3, #4, #5, #6, Off}
Object ={*User*, *Sender*, *Receiver*}

SM 에서 *Sender* 와 *Receiver* 는 동기적이며 Two way 방식에서의 동기화 및 일관성의 주체는 *Sender* 이다.

표 1 OCL 을 통한 *SenderClass.SM* 의 영역 명세

Cont, *data*,*Com*, d*Mem*, SyncCase, Shared,Invalid,Modified:Boolean	
On()	Pre : *Cont* = F and <F,T,T,T,F,T,F,F> Post : *Cont* = T and <T,T,T,T,F,T,F,F>
#2()	Pre : Shared=T and Invalid=F SyncCase=F and <T,T,T,T,F,T,F,F> Post : Shared=F and Invalid=T and SyncCase=T and <T,T,T,T,T,F,T,F>
#4()	Pre : Invalid=T and Modified=F and <T,T,T,T,T,F,T,F> Post : Invalid=F and Modified=T and <T,T,T,T,T,F,F,T>
#6()	Pre : Modified=T and Shared=F and SyncCase=T and <T,T,T,T,T,F,F,T> Post : Modified=F and Shared=T and SyncCase=F and <T,T,T,T,F,T,F,F>
Off()	Pre : *Cont* = T and <T,T,T,T,F,T,F,F> Post : *Cont*=F and <F,T,T,T,F,T,F,F>

2.5 추상화 모델 *M'*

SyncML 은 비동기적이며 메시지 패싱은 동기적이다. *Cont* 가 *Sender.Class.SM* 의 동기화에 추가적으로 관여하며, 다음과 같은 추상화 모델이 유도된다.

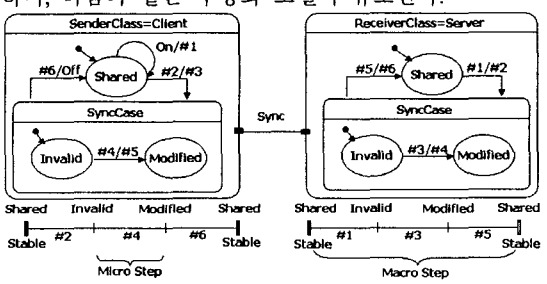

그림 3 상태도를 통한 행위 명세와 Step 의 의미

3. 속성

프로토콜을 명세하기 위해서는 높은 표현력의 정형언어가 필요하다. 우리는 CTL (Computation Tree Logic)과 일차 논리(First Order Logic)를 통해 속성을 명세하며 자료 일치성 속성을 증명하기 위한 안전성과 궁극성[13]을 포괄적으로 검증한다. 동기화를 통한 자료 일치성은 중계 및 스누핑(Snooping)방법을 통한 강한 일관성 혹은 순차적 일관성과는 다르게 약한 일관성을 갖는다. 동기화의 기본 속성은 상호 배제[14] 및 자료 일치성, 메시지 패싱에 대한 궁극성이 있으며, 따라서 각 속성은 상호 의존성을 갖는다. Sync 시스템에서의 각 속성은 다음과 같이 정의된다.

- 상호 배제
  AG ϕ_1 = AG $\forall sd$:*Sender* $\forall$ *rv*:*Receiver* $\forall st$:*Status*.
  (*sd* =SyncCase(*st*) $\Rightarrow$!*rv*=Shared(*st*))
- 자료 일치성
  AG ϕ_2 = AG $\forall sd$:*Sender* $\forall rv$:*Receiver* $\forall st$:*Status*.
  (*rv*= Shared(*st*) $\Rightarrow$ *sd*= Shared(*st*))
- 강한 궁극성
  AG ϕ_3 = AG AF $\forall sd$:*Sender* $\forall rv$:*Receiver* $\forall st$:*Status*.
  (*sd*=Shared(*st*) $\wedge$ *rv*=Shared(*st*))

따라서, Sync 시스템이 만족해야할 속성은
M;Σ;Γ |- AG ϕ = AG($\phi_1 \wedge \phi_2 \wedge \phi_3$) 이다.

4. 정리 증명과 모델 체킹이 혼합된 검증

우리는 전체 증명 절차에서 유한 상태 모델을 하나의 해석 영역으로 추가하고 시퀀트 계산법을 통해 증명을 진행한다. 또한 모델 체킹 기술은 하나의 추론 규칙[]으로 적용되며 역방향 전략으로 증명을 진행한다. '가정 Γ이면 결론Δ이다.'가 모델 M 과 가장 불변식 Σ 에서 만족된다면, 다음과 같이 정의된다.

M; Σ; Γ |- Δ iff M|$_\Sigma$ |= $\bigwedge\Gamma \rightarrow \bigwedge\Delta$.

M;Σ;Γ |- AG ϕ 의 증명;

$$\frac{\text{|- Subgoal.1} \quad \text{|- Subgoal.2} \quad \text{|- Subgoal.3}}{\text{M;Σ;Γ |- AG } \phi_1 \quad \text{M;Σ;Γ |- AG } \phi_2 \quad \text{M;Σ;Γ |- AG } \phi_3}\ \wedge_R$$

M=(S, →, I) ;Σ;Γ |- Δ, AG ϕ = AG ($\phi_1 \wedge \phi_2 \wedge \phi_3$)

Subgoal.1; **M'*:Abstract Model

$$\frac{\text{Subgoal.1.1} \quad \text{Subgoal.1.2} \quad \text{Subgoal.1.3}}{\text{M;Σ;Γ |-Δ, ψ} \quad \text{M'=(S,→S);Σ,ψ;|-}\phi_1 \quad \text{M'; Σ; ψ |- AXψ}}\ \text{Induct AG}$$

M;Σ;Γ |- AG ϕ_1

Subgoal.1.1; M;Σ;Γ |- Δ, ψ
ψ:*sd*=SyncCase(*st*)⇒!*rv*=Shared(*st*),
Initial State :*sd*=Shared(*st*)∧*rv*=Shared(*st*) |-ψ

Subgoal.1.2; M'=(S,→S);Σ,ψ;|- ϕ_1
*SR:Symmetry Reductions,*COI:Cone of Influence Reduction
ϕ_1=$\forall sd$:*Sender* $\forall rv$=*Receiver* $\forall st$:*Status*.
(*sd* =SyncCase(*st*) ⇒ !*rv*=Shared(*st*))

$$\frac{\dfrac{M'|_{(rv)}\ ;\cdot\ ;\cdot\ |\text{- !}rv\text{=Shared}(st)}{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\cdot;\cdot\ |\text{- !}rv\text{=Shared}(st)}\ \text{COI}}{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\cdot;\cdot\ sd\text{ = SyncCase}(st)|\text{- !}rv\text{=Shared}(st)}\ W_L$$

$$\frac{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\cdot;\cdot\ sd\text{ = SyncCase}(st)\ |\text{- !}rv\text{=Shared}(st)}{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\cdot;\cdot|\text{- (}sd\text{ = SyncCase}(st)\Rightarrow\text{ !}rv\text{=Shared}(st))}\ \wedge_L$$

$$\frac{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\cdot;\cdot|\text{- (}sd\text{ = SyncCase}(st)\Rightarrow\text{ !}rv\text{=Shared}(st))}{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\psi;\cdot|\text{- (}sd\text{ = SyncCase}(st)\Rightarrow\text{ !}rv\text{=Shared}(st))}\ \Rightarrow_R$$

$$\frac{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\psi;\cdot|\text{- (}sd\text{ = SyncCase}(st)\Rightarrow\text{ !}rv\text{=Shared}(st))}{M'\text{=(S,}\rightarrow\text{S);}\Sigma,\psi;\cdot|\text{-}\phi_1}\ \text{S\_goal.1.1}\ \forall_R AG(SR)$$

Subgoal.1.3; ψ : *sd*=SyncCase(*st*) $\Rightarrow$!*rv*=Shared(*st*)
*SMV(Symbolic Model Verifier) *MC(Model Check)

$$\frac{\dfrac{M\text{·}C\ (M', \bigwedge\Sigma, \bigwedge\psi \rightarrow \bigvee AX\ \psi) = true}{M';\Sigma;\ \psi \vdash AX\ \psi}\ MC(SMV)}{M';\Sigma;\ \psi \vdash \forall sd.\forall rv.AX\ \psi}\ \forall_R AX(SR)$$

Subgoal.2; ϕ_2 : $\forall$*sd:Sender*$\forall$*rv:Receiver*$\forall$*st:Status.*(*rv*=Shared(*st*) $\Rightarrow$ *sd*= Shared(*st*)), ϕ_2' : *rv* = Shared(*st*) $\Rightarrow$ *sd* = Shared(*st*)

$$\frac{\dfrac{M\text{·}C(M', \bigwedge\Sigma, \bigwedge\phi_2' \rightarrow \bigvee AG\ \phi_2') = true}{M';\Sigma;\ \phi_2' \vdash AG\ \phi_2'}\ MC(SMV)}{M';\Sigma;\ \phi_2 \vdash AG\ \phi_2}\ \forall_R AG(SR)$$

Subgoal.3; ϕ_3 = AF $\forall$*sd:Sender*$\forall$*rv:Receiver*$\forall$*st:Status.*
(*sd*=Shared(*st*) $\wedge$*rv*=Shared(*st*))
ϕ_3' : AF *sd*=Shared(*st*) $\wedge$*rv*=Shared(*st*)

$$\frac{\dfrac{M\text{·}C\ (M', \bigwedge\Sigma, \bigwedge\phi_3' \rightarrow \bigvee AG\ \phi_3') = true}{M';\Sigma;\ \phi_3' \vdash AG\ \phi_3'}\ MC(SMV)}{M';\Sigma;\ \phi_3 \vdash AG\ \phi_3}\ \forall_R AG(SR)$$

*AGR:Assume-Guarantee Reasoning
$M;\Sigma;\Gamma \vdash AG\ \phi_1 \wedge \vdash AG\ \phi_2 \wedge \vdash AG\ \phi_3 \Rightarrow_{AGR} \vdash AG\ \phi$

다른 동기화 방법은 다음과 같이 증명된다. 동기화 방법에 대해 Two Way = M_1, Slow = M_2, Server Only One Way = M_3, Server Only Refresh = M_4, Client Only One Way = M_5, Client Refresh = M_6 라고 하자.
$M_1 \equiv_{COI} M_2$: Bisimulation, $M_1 \leftrightarrow_{AGR} M_3$: Symmetry
$M_3 \Rightarrow_{Abstract} M_5$:Simulation 관계이다. 따라서 M_2, M_3, M_5는 증명된다. 한편, 얻어진 모델에 대한 속성은 모델 체킹을 통해 증명될 수 있다.

5. 관련 연구 및 비교

자료 일치성에 관한 대부분의 연구는 하드웨어 혹은 유선 네트워크 기반의 순차적 일관성에 관한 것이였다[15][16]. 또한 동기화 프로토콜에 대한 의미 및 정형적 모델은 제시되지 못했다. 특히 선택적 정형 검증 기술이 적용되어 모델링 및 검증 영역이 제한됐다[17][18][19]. 본 연구에서는 동기화 프로토콜 모델과 속성을 단계적으로 유도했다. 또한 혼합된 검증 방법을 통해 얻어진 속성을 강하고 포괄적으로 검증했다. 특히, 유한 상태 모델의 생성을 위해 xUML[19]에서 사용되는 객체 제약 언어와 같은 추가적 정형 언어의 사용을 통해 신뢰성있는 모델 생성 방법을 제시했다.

6. 결론

본 연구에서는 자료 일치성 및 상호 배제, 궁극성 속성을 강하게 증명했으며, 모델 체킹을 하나의 증명 규칙으로 사용했다. 본 연구의 특징은 첫째, 정리증명과 모델 체킹이 혼합된 검증 방법을 사용했다. 둘째, 객체 제약 언어 및 정형 명세 언어의 사용을 통해 모델의 생성을 보다 단계적이고 정형적으로 유도했으며 이를 통해 검증의 신뢰성 및 품질을 높였다. 향후 연구 과제는 정의 및 모델에 대한 정형성을 강화시키는 것이며 제시된 방법과 속성을 구현된 시스템에 적용하는 것이다.

참고문헌

[1]B. Aziz, et. al., "Implementing Protocol Verification for E-Commerce," International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, 6-12 Aug, 2001.
[2]http://www.syncml.org
[3]L. Lamport, "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs," IEEE trans. On Computer Vol.28, No.9, pp.690-691, Sept. 1979.
[4]Daniel J. Sorin, et al., "Specifying and Verifying a Broadcast and a Multicast Snooping Cache Coherence Protocol," Vol.13, No.6, June 2002
[5]Z. Manna, et. al., "STeP: The Stanford temporal prover," Technical Report STAN-CS-TR-94-1518, Computer Science Department, Stanford University, 1994.
[6]E.M. Clarke, et. al., "Automatic verification of finite state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and* Systems, Vol.8, No.2, pp. 244-263, 1986.
[7]Sergey Berezin, "Model Checking and Theorem Proving: a Unified Framework," Ph.D. Thesis, CMU, 2002.
[8]M. Richters et. al, "On Formalizing the UML Object Constraint Language OCL," Proc. 17th International Conference on Conceptual Modeling(ER'98),*LNCS*, 1998.
[9]M. Dubois, et. al., "Synchronization, Coherence, and Event Ordering in Multiprocessors," IEEE Computer, Vol.21, No.2, pp.9-21, February 1988.
[10]F. Pong, et. al., "Verification techniques for cache coherence protocols," ACM Computing Surveys (CSUR) Vol.29, Issue 1, March 1997.
[11]S. Chandra et. al., "Experience with a language for writing coherence protocols," In Proceedings of the 1st USENIX Conference, Oct. 1997.
[12]J. Whittle, et. al., "Generating Statechart Designs from Scenarios," ACM Press., in 22nd International Conference on Software Engineering (ICSE'00), 2000.
[13]M. Plakal, et. al., "Lamport Clocks: Verifying A Directory Cache-Coherence Protocol," appear in the 10th Annual ACM Symposium on Parallel Algorithms and Architectures(SPAA), June 1998.
[14] L. Lamport, "The Mutual Exclusion Problem," Parts I,II, Journal of the ACM, Vol.33, No.2, April 1986, pp.313-348.
[15]E.M. Clarke, et. al., "Verify Security Protocols with Brutus," ACM Transactions on Software Engineering and Methodology, Vol.9, No.4, pp.443-487, October 2000.
[16]E.M. Clarke, et. al., "Verification of Futurebus+ Cache Coherence Protocol," Proc. 1 lth Intl. Symp. on Computer. Hardware Description. Lang. and their Application, 1993.
[17]J.M. Wing, et. al., "Model Checking Software Systems: A Case Study," Proceedings of 3rd ACM SIGSOFT Symposium, October 1995.
[18]L. Mummert, "Using Belief to Reason About Cache Coherence," In Proceedings of the Symposium on Principles of Distributed Computing, May 1994.
[19]H. Garavel, et. al., "System Design of a CC-NUMA Multiprocessor Architecture using Formal Specification, Model-Checking, Co-Simulation, and Test Generation," Springer(STTT), vol.3, No.3, July 2001.
[20]D. Harel, et. al., "Executable Object Modeling with Statecharts," IEEE Computer, Vol.3, No.7, pp.31-42, 1997.